UNIVERSIDAD PRIVADA DEL NORTE

ESCUELA DE POSTGRADO Y ESTUDIOS CONTINUOS

IMPACTO EN LA INTEGRIDAD DE DATOS UTILIZANDO EL MODELO DE DATOS DE RELACIÓN FUNCIONAL PROPUESTO SOBRE LA ONTOLOGÍA DE UN MODELO DE NEGOCIO PARA AUTOMATIZAR UN DISEÑO DE BASE DE DATOS.

Tesis para optar el grado de **MAESTRO** en:

INGENIERÍA DE SISTEMAS CON MENCIÓN EN GERENCIA DE SISTEMAS DE INFORMACIÓN

Autor:

Victor Piero Milcar Tumba Burgos

Asesor:

Dr. Alberto Mendoza de los Santos

Trujillo, Perú

2021

One way to learn the mind of the Creator is to study His creation. We must pay God the compliment of studying His work of art and this should apply to all realms of human thought. A refusal to use our intelligence honestly is an act of contempt for Him who gave us that intelligence...

Ernest Thomas Sinton Walton, Awarded the Nobel Prize in Physics in 1951

ÍNDICE DE CONTENIDO

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

ÍNDICE DE CUADROS

Resumen

Los resultados de esta investigación permitieron definir, en base al *Modelo de Datos de Relación Funcional* propuesto, un proceso algorítmico que consolide un esfuerzo para construir un proceso automático que obtenga *Dependencias de Relación Funcional* al aplicar *Restricciones Lógicas* (las que están relacionadas con el concepto matemático de *Relación Funcional*) sobre Estructuras Lógicas de Datos, para lograr el diseño correcto de una base de datos y verificar que se cumpla con la *Integridad de Datos*.

El objetivo de esta investigación es dar solución al problema que ocurre en cualquier organización de cualquier parte del mundo al diseñar bases de datos de tipo relacional. Por lo tanto, lo relevante en la investigación es verificar que el modelo propuesto funcione aplicándolo sobre el modelo de negocio de una empresa. Posteriormente se creó un algoritmo que pueda automatizar la ejecución del modelo, de manera tal que, aunque el modelo sea exitoso, no sea haga de forma manual, y por lo tanto el diseño no tome demasiado tiempo.

Finalmente, esta investigación concluye que se puede determinar que la ejecución del Modelo de Datos de Relación Funcional sobre un Modelo de Negocio permite conservar la Integridad de Datos de la información organizacional, logrando la automatización de un Diseño de Base de Datos a partir de Estructuras Lógicas de Datos.

Palabras Claves: Modelo de Datos de Relación Funcional, Dependencias de Relación Funcional, Restricciones Lógicas, Estructuras Lógicas de Datos, Ontología del Modelo de Negocio, Restricciones de Integridad de Datos, Integridad de Datos, Modelo de Negocio, Lógica de Negocio, Reglas de Negocio.

Abstract

The results of this research allowed defining, based on the proposed Functional Relationship Data Model, an algorithmic process that consolidates an effort to build an automatic process that obtains Functional Relationship Dependencies by applying Logical Constraints (those that are related to the mathematical concept Functional Relationship) on Logical Data Structures, to achieve the correct design of a database and verify that it complies with Data Integrity.

The objective of this research is to solve the problem that occurs in any organization anywhere in the world when designing relational databases. Therefore, what is relevant in the research is to verify that the proposed model works by applying it to the business model of a company. Subsequently, an algorithm was created that can automate the execution of the model, in such a way that, although the model is successful, it is not done manually, and therefore the design does not take too long.

Finally, this research concludes that it can be determined that the execution of the Functional Relationship Data Model on a Business Model allows to preserve the Data Integrity of the organizational information, achieving the automation of a Database Design from Logical Structures of data.

Keywords: Functional Relation Data Model, Functional Relation Dependencies, Logical Constraints, Logical Data Structures, Business Model Ontology, Data Integrity Rules, Data Integrity, Business Model, Business Logic, Business Rules.

Dedicatoria y Agradecimientos

Quiero dedicar esta investigación, agradeciéndole todo lo logrado, en primer lugar y sobre todas las cosas, a Dios, el Señor, el Camino, la Verdad y la Vida.

También quiero dedicar esta investigación sobremanera a mi esposa Milagros, a mi hermosa hija Lucía y a mi providencial hijo Gael, por ser ellos el motor propulsor de todo lo logrado.

Así como también quiero dedicar esta investigación a mis familiares cercanos, los que siempre me han dado su apoyo.

Y no quisiera dejar de agradecer, de manera especial, al profesor Alberto Mendoza de los Santos por la asesoría y el apoyo.

I. Introducción

1. Realidad Problemática

Actualmente, los modelos de negocio, que operan en las organizaciones, luchan por permanecer a flote. De manera que siempre están cambiando su información empresarial de forma vertigionosa, haciendo que su lógica de negocio cree nuevas reglas de negocio constantemente, y que por lo tanto las cambiantes relaciones entre conceptos de negocio impliquen también modificaciones a las formas en las que los datos de las organizaciones se almacenan y se consultan. Evidentemente, como los cambios ocurren con frecuencia, el modelado para las bases de datos relacionales se ha vuelto un gran desafío, sobretodo debido al tiempo y los recursos que se requieren. Es por ello que, desafortunadamente, cuando se usa una base de datos relacional, los cambios en la estructura lógica de datos podrían ser una tarea muy compleja, incluso si solamente se requiere agregar o eliminar una columna en una tabla.

En todo el mundo, desde los gerentes de tecnología hasta los desarrolladores de primera línea, todos se están dando cuenta de que los diseños de las bases de datos relacionales que se crean simplemente no están hechos para los desafiantes cambios que requiere la dinámica de los grandes volúmenes de datos. Es la razón por la que en el mercado de tecnología constantemente hay propuestas de nuevos productos de bases de datos de tipo *NoSQL* que pretenden optimizar el almacenamiento y posterior tratamiento de los datos.

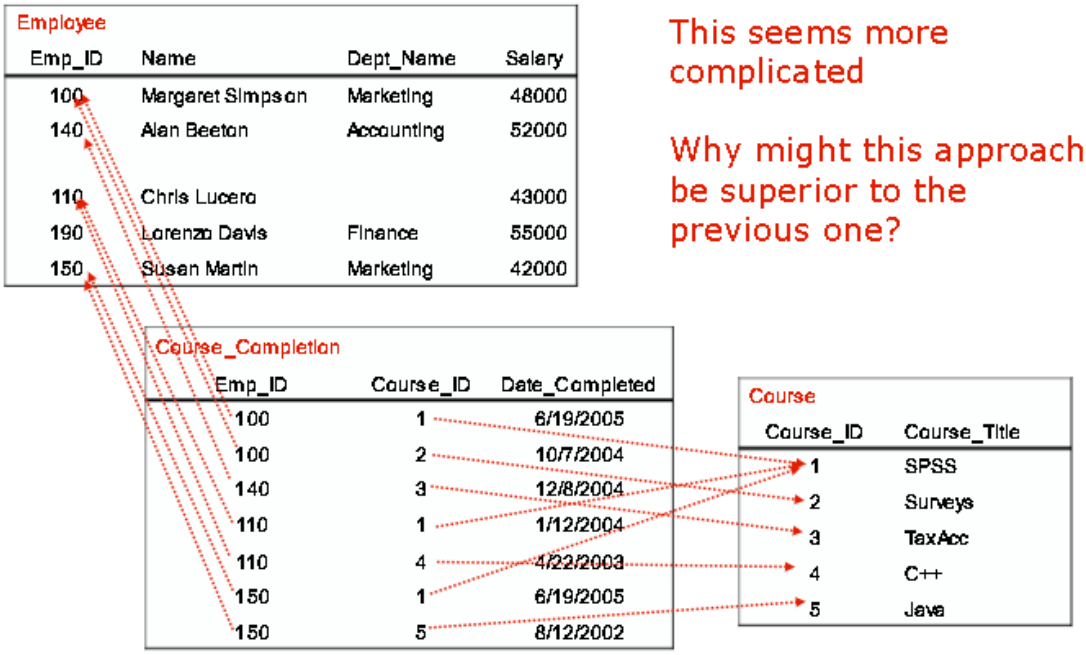
En relación a los problemas que surgen del diseño de bases de datos relacionales, el ex director de Información del Gobierno Federal de EE.UU., Vivek Kundra, en la *Open Government and Innovations Conference* de 2009, dijo que pensar en los datos mientras se contempla la idea de una base de datos relacional, estructurada, es una noción que está muerta, ya que parte de la información más valiosa va a vivir en vídeos, en blogs y en audios, y será inherentemente desestructurada (Beizer, 2009). Siendo la conclusión de Kundra que las bases de datos relacionales no permiten un diseño que maneje el cambio dinámico de los datos.

De cierto es que la opinión de Kundra con respecto a la existencia del modelo de datos relacional es muy pesimista, más allá de encontrar una forma de mejorarlo, y se da en el sentido de que para él sencillamente no funciona y no debería utilizarse, y que además se necesita crear un nuevo modelo que corresponda a datos no estructurados.

A día de hoy, las decisiones concernientes a un modelo de negocio ya se toman en minutos, ni siquiera en días, y los datos para respaldar esas decisiones se deben entregar en el formato correcto, con la latencia más reducida posible, y con la mayor eficiencia que el diseño pueda

proporcionar. Además, la variedad de datos aumenta, no sólo en volumen sino también en complejidad, a medida que las organizaciones luchan por mantener sus modelos de negocio; y por otro lado las nuevas aplicaciones que se diseñan, las migraciones entre un origen de datos y otro, las adquisiciones de nuevos sistemas que administran las bases de datos, y la reutilización de datos, son las razones comunes que conducen a la disparidad de datos estructurados en el transcurso del tiempo, sin mencionar todos los datos no estructurados que no se tienen en cuenta en el modelado de datos relacional.

Gráfico 1 : Estructura de Datos Relacional.



Fuente: (Monroe, 2008)

Para comprender por qué todo aquello del diseño es muchas veces problemático, es mejor observar más de cerca cómo se modelan los datos en una base de datos relacional. Como se sabe, las bases de datos relacionales organizan los datos en tablas, que tienen filas y columnas, de la misma manera como se haría en las hojas de cálculo de Microsoft Excel. Cada fila representa una entrada única (instancia); cada columna describe atributos únicos (dominio de datos). Entonces, en cada tabla se elige una columna como clave principal, que sirve para identificar de forma exclusiva y única cada fila de la tabla. En consecuencia, después de crear una tabla, se debe asegurar que su diseño cumpla con todas las restricciones de integridad que correspondan a la integridad de la entidad, a la integridad

referencial, a la integridad de dominio y a la integridad definida por el usuario que es dueño (sea parcialmente o totalmente) del modelo de negocio, para que de esta manera no existan problemas de anomalías de datos. Se necesita también que el diseño esté correctamente normalizado para que no existan problemas de redundancia de datos. Tales restricciones son las que mantienen la consistencia de los datos y aseguran el almacenamiento correcto y las consultas eficientes.

Por consiguiente, para lograr este proceso de diseño (esquema de datos), sobre un modelo de datos relacional, se involucra a un equipo especializado, ya que es un proceso muy importante por ser la piedra angular de datos que utiliza todo sistema de información, y el resultado final a menudo se representa en un Diagrama de Entidad-Relación. Es por ello que, en realidad, cuando se analiza la problemática, más allá del valor intrínseco del mismo modelo de datos relacional, lo que existe es un problema crítico al momento de utilizar este modelo para diseñar una base de datos; es decir, hay un desconocimiento de base con respecto al modelo o una falta de pericia técnica en el momento en el que los diseñadores de datos crean sus diseños, ya que en la mayoría de los casos se hace en base a intuiciones que derivan del modelo negocio. Y es que los esquemas de datos (diseños) siempre crean la forma consistente y perenne en la que los sistemas de información describen y consultan los datos, y ninguna organización tendría éxito si no fuera cuidadosa en la forma en que modela sus datos.

Es evidente que un factor importante es que las bases de datos relacionales requieren hacer el modelado de datos relacional por adelantado, y por consiguiente no manejan muy bien los cambios en los esquemas cuando son necesarios. De manera tal que la integridad de datos debe ser reinvidicada en cada modificación que se le aplica al diseño de base de datos relacional. Por lo tanto, el modelado de datos relacional siempre se convierte en algo engorroso para las organizaciones, trayendo consigo los siguientes problemas:

- **La complejidad en la creación de un nuevo diseño.** El proceso de modelado de datos relacional para un nuevo diseño puede llevar meses, y algunas veces años, dependiendo de la complejidad del diseño de la base de datos que se genere.

- **Errores en la creación de un nuevo diseño.** Frente a toda la complejidad de los esquemas relacionales (diseños), todo el modelado de datos relacional debe hacerse antes de cargar cualquier estructura lógica de datos o construir algún sistema de información, y esto conlleva a la existencia muchos errores dentro del diseño, debido a que éste no puede plasmar con precisión la lógica de negocio que intenta modelar.

- **La complejidad en la modificación de un diseño ya creado.** Los proyectos de modelado de datos relacional, que requieren restructurar el diseño ya construido de una

base de datos, se convierten en procesos que necesitan de mucho esfuerzo, ya que, por ejemplo, un pequeño cambio en una tabla puede causar una cascada de cambios, y no solo en el diseño, sino también en todas las aplicaciones del sistema de información que se sostengan sobre la base de datos. Sin contar las veces en las que hay que hacer cambios por cada *prueba y error* que se presenta en el proyecto.

- **Errores al modificar un diseño ya creado.** Como ya es sabido, debido a que en la creación de un diseño relacional no se aplican indicadores matemáticos que permitan medir la exactitud del mismo diseño, no es posible verificar en qué medida una estructura lógica de datos, que es definida por la lógica del negocio, se corresponde con la ontología del modelo de negocio. En consecuencia, lo mismo sucede con los diseños que requieren ser modificados; es la razón por la que las áreas de soporte tecnológico constantemente están recibiendo tiquetes de incidencias que conllevan a corregir errores de diseño que afectan al modelo de negocio.

El presente proyecto de investigación pretende resolver la problemática en cuestión, proponiendo el Modelo de Datos de Relación Funcional, el cual realiza un proceso inverso al realizado cuando se utiliza el Modelo de Datos Relacional; es decir, se empieza con las estructuras lógicas de datos correctamente definidas para luego pasar a analizar las dependencias entre los conjuntos de datos de la estructura, a través de restricciones lógicas que se corresponden con indicadores matemáticos. Todo lo cual dará como resultado un diseño de base de datos exacto y medible porque obedece a las reglas y la lógica del negocio. En consecuencia, dichas Estructuras Lógicas de Datos, que se utilizarán como muestra en la investigación para la aplicación del modelo propuesto, serán proporcionadas por una empresa de servicios de hospedajes de la ciudad de Trujillo, y corresponderán a datos del año 2014.

A. Macro

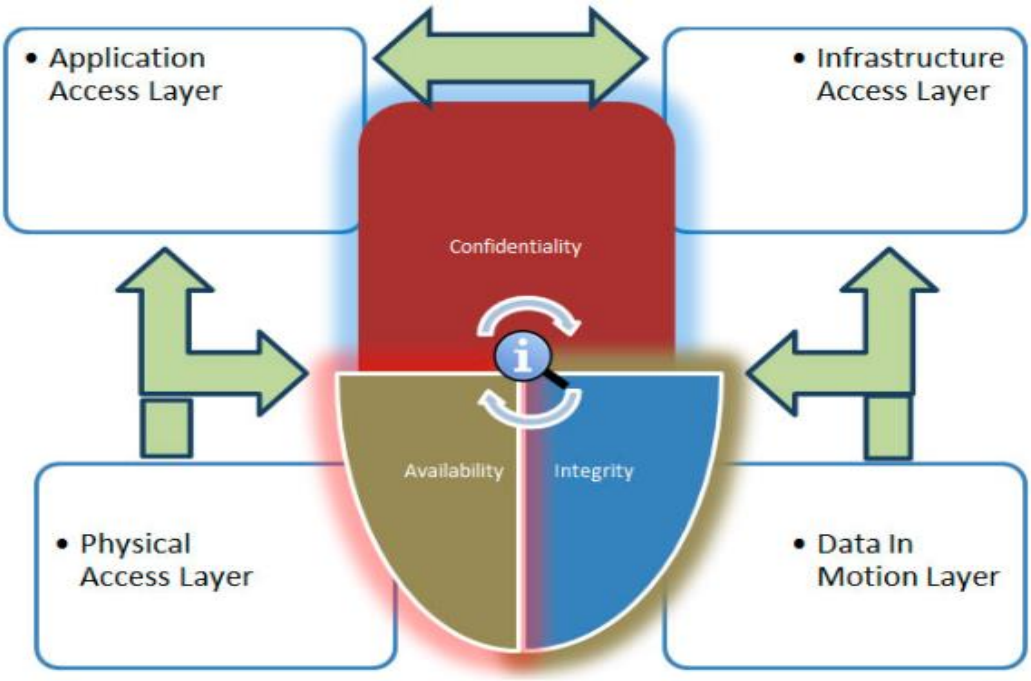
Las organizaciones más competitivas de cualquier mercado y de cualquier industria están obligadas a mantener actualizados sus sistemas de información, y en condiciones estables, para que sus modelos de negocio puedan mentenerse a flote satisfactoriamente. Sin embargo, cuando hablamos de mantener a salvo la información organizacional tenemos que remitirnos a los principios mismos de la Seguridad de la Información: Confidencialidad, Disponibilidad e Integridad.

Por tal motivo, en el presente proyecto de investigación nos centraremos en la problemática de la redundancia y las anomalías de los datos organizacionales al crear un

diseño con el modelo de datos relacional; de manera que tenemos que centrarnos concretamente en el principio de la Integridad de la Seguridad de la Información.

Como se sabe, la integridad se refiere a la protección de la información contra modificaciones o destrucciones no autorizadas (Computing Services Information Security Office, 2011). Y aquí, cuando mencionamos el término “no autorizadas”, nos referimos a que dichas modificaciones o destrucciones no están conciliadas con el modelo de negocio de la organización, siendo ésta la propietaria de la información institucional. Por lo tanto, garantizar la integridad es en realidad garantizar que la información que se utilice por los sistemas de información sea precisa, completa y sin corrupción.

Gráfico 2 : Requerimientos de la Seguridad de la Información.



Fuente: (Infosec Resources, 2018)

En consecuencia, los problemas que surgen del diseño de bases de datos relacionales, como las anomalías y la redundancia de datos, van a generar siempre vulneraciones a la integridad de datos, y por lo tanto a la Integridad de la Seguridad de la Información de la organización.

La integridad de datos, al ser un conjunto de principios relacionados con la forma de determinar qué condiciones son válidas para una base de datos, describe solo aquellas reglas que son inherentemente parte del modelo de datos relacional, especificando las restricciones de integridad.

Por lo tanto, la integridad de datos está muy relacionada con la idea de la manipulación de datos, en el sentido de que la integridad se refiere a la idea de determinar cuáles son los cambios válidos y los cambios no válidos que correspondan a los datos. Siendo que la manipulación de datos se refiere al proceso de cómo se accede a los datos y cómo se modifican en la base de datos (Beynon-Davies, Database Systems, Third Edition, 2004).

B. Delimitación de Variables

En el contexto de la construcción de las bases de datos de tipo organizacional, **un diseño de base de datos, que esté sometido a las Restricciones de Integridad de Datos, las cuales se sostienen en las Reglas de Negocio y que sirven como pilar fundamental de la Seguridad de la Información de la organización**, requiere ineludiblemente que la persona que hará el diseño tenga un conocimiento muy exhaustivo de los procesos claves de la organización y de las Reglas de Negocio vigentes circunscritas a la Lógica de Negocio. Sin embargo, para que un diseño de base de datos tenga la exactitud necesaria que permita hacerlo medible válidamente, de tal forma que haya una conciliación entre las Restricciones de Integridad de Datos y las Reglas de Negocio, se necesita que **el Modelo de Datos de Relación Funcional, que se propone en esta investigación, sea aplicado sobre las Estructuras Lógicas de Datos que se corresponden con el Modelo de Datos Conceptual de la Ontología del Modelo de Negocio, teniendo en cuenta que lo que se deberá aplicar son Restricciones Lógicas bajo el concepto matemático de Relación Funcional**. En consecuencia, dichas Estructuras Lógicas de Datos, que se utilizarán como muestra en la investigación para la aplicación del modelo propuesto, serán proporcionadas por una empresa de servicios de hospedajes de la ciudad de Trujillo, y corresponderán a datos del año 2014.

2. Pregunta de Investigación

A. Formulación del Problema

¿Cuál es la eficacia del Modelo de Datos de Relación Funcional propuesto sobre la Ontología de un Modelo de Negocio para lograr la automatización de un Diseño de Base de Datos que conserve su Integridad de Datos?

3. Objetivos de la Investigación

A. Objetivo General

Evaluar la eficacia del Modelo de Datos de Relación Funcional propuesto sobre la Ontología de un Modelo de Negocio para lograr la automatización de un Diseño de Base de Datos que conserve su Integridad de Datos.

B. Objetivos Específicos

(1) **[OE1]** Introducir conceptos preliminares y elaborar la propuesta del Modelo de Datos de Relación Funcional.

(2) **[OE2]** Diseñar una Base de Datos aplicando el Modelo de Datos de Relación Funcional propuesto sobre la Ontología de un Modelo de Negocio de la empresa de servicios de hospedaje.

(3) **[OE3]** Evaluar la Integridad de Datos del Diseño de Base de Datos creado para la empresa de servicios de hospedaje, verificando que esté sujeta a las Reglas de Negocio.

(4) **[OE4]** Elaborar un proceso algorítmico que defina la creación automática de un Diseño de Base de Datos utilizando el Modelo de Datos de Relación Funcional propuesto.

4. Justificación de la Investigación

La presente investigación se justifica teóricamente en cuanto a que permite estudiar propositivamente el **Modelo de Datos de Relación Funcional**, bajo un enfoque que aplica el concepto matemático de Relación Funcional, a través de Restricciones Lógicas, sobre las Estructuras Lógicas de Datos que obedecen al Modelo de Datos Conceptual y que se corresponde con la Ontología del Modelo de Negocio de la organización. Todo lo cual

posteriormente conduciría a la definición de un proceso algorítmico que permita lograr con eficacia la creación de un Diseño de Base de Datos que esté bajo las Restricciones de Integridad de Datos conciliadas con las Reglas de Negocio.

La justificación práctica de la presente investigación se sostiene en que este proyecto de investigación permitiría definir, en base a la proposición del Modelo de Datos de Relación Funcional, un proceso algorítmico que establezca la automatización de la obtención de las Dependencias de Relación Funcional, las que permitirán a su vez lograr el Diseño de Base de Datos, aplicando Restricciones Lógicas correspondientes al concepto matemático de Relación Funcional.

La aplicación del Modelo propuesto comprenderá desde las Estructuras Lógicas de Datos, como una especificación de la abstracción de la Ontología del Modelo del Negocio (que representa, dentro de su dominio de aplicación, la Semántica de Conceptos), y que corresponden a los Procesos Claves de Negocio de la organización (formando éstos la parte central del Modelo del Negocio). Dichas Estructuras Lógicas de Datos, que se utilizarán como muestra en la investigación para la aplicación del modelo propuesto, serán proporcionadas por una empresa de servicios de hospedajes de la ciudad de Trujillo, y corresponderán a datos del año 2014.

Posteriormente, se procederá con la obtención de un Diseño de Base de Datos, el cual deberá intrínsecamente estar bajo las Restricciones de Integridad de Datos que obedecen a las Reglas de Negocio. De manera tal que el diseño obtenido pueda ser, además de una estructura medible, una proyección verfificable de la Lógica de Negocio (entendiéndose ésta como un conjunto de reglas potencialemente variables). En consecuencia, como se indicó anteriormente, tal proceso algorítmico deberá ser la integración de la aplicación del Modelo de Datos de Relación Funcional sobre volúmenes de instancias de datos.

5. Alcance de la Investigación

La presente investigación se circunscribirá dentro de la propuesta del Modelo de Datos de Relación Funcional en contraparte al Modelo de Datos Relacional, desde una perspectiva que, no habiendo sido abordada antes, obedece a la definición matemática de Relación Funcional. Tal modelo propuesto se deberá aplicar sobre las Estructuras Lógicas de Datos que se corresponden con la Ontología referente al Modelo de Negocio de una organización. En consecuencia, dichas Estructuras Lógicas de Datos, que se utilizarán como muestra en la investigación para la aplicación del modelo propuesto, serán proporcionadas por una empresa de servicios de hospedajes de la ciudad de Trujillo, y corresponderán a datos del año 2014.

El propósito central de la investigación es conseguir la creación de un Diseño de Base de Datos que se desglose fielmente del Modelo de Datos Conceptual que respalda los conceptos propios de los procesos organizacionales inmersos, utilizando como insumos de entrada Estructuras Lógicas de Datos correspondientes a la lógica conceptual de la organización. De manera tal que, cuando el Diseño de Base Datos de Relación Funcional sea creado, se responda a las Restricciones de Integridad de Datos que se corresponden al conocimiento de las Reglas de Negocio que mantiene la organización.

En definitiva, debido a su naturaleza de enfoque confrontacional, **el presente estudio es correlacional, cuyo propósito es examinar el problema de investigación confrontando con los antecedentes**, del cual se tienen ciertas dudas técnicas y de conceptos, y que, de lograrse satisfactoriamente los objetivos de la investigación, nos proporcionará utilidad académica para estudios futuros que encaminen a una aplicación empresarial del modelo propuesto.

II. Marco Teórico

1. Antecedentes

Desde los años ochenta del siglo XX, se ha venido definiendo la teoría de dependencias, para las bases de datos relacionales, como una restricción de integridad formalizada que establece que siempre que existan tuplas en la base de datos, que se cumplen en ciertos atributos, también debe estar presente una tupla adicional que coincida con aquellas tuplas de una manera específica. Por lo tanto, se muestra que el problema de inferencia para las dependencias es indecidible; es decir, **no puede haber un algoritmo para determinar si una dependencia dada es una consecuencia lógica de un conjunto finito de dependencias**. El resultado de indecidibilidad es válido tanto si las bases de datos se consideran necesariamente finitas como si no lo hacen (Gurevich & Lewis, 1982). El objetivo de la teoría de dependencias es formalizar las restricciones sobre los datos que comprenden una base de datos relacional.

En general, una dependencia es una declaración, en el sentido de que cuando ciertas tuplas están presentes en la base de datos, también lo están otras tuplas distintas. Tales declaraciones pueden usarse, por ejemplo, para capturar la idea de que los atributos están relacionados funcionalmente o son independientes de alguna manera. En parte, está el deseo de equilibrar dos fuerzas opuestas: por un lado, las dependencias deben ser de una forma lo suficientemente general como para expresar propiedades interesantes, pero, por otro lado, la forma no debe ser tan general como para que las interrogantes naturales sobre dependencias se vuelven indecidibles o computacionalmente intratables.

Una pregunta importante sobre cualquier clase de dependencias es su problema de inferencia: dado un conjunto finito de dependencias D y una dependencia única Dn, entonces poder determinar si Dn es verdadero en cada base de datos en la que cada miembro de D también es verdadero. Una solución al problema de inferencia conlleva la capacidad de determinar si dos conjuntos de dependencias son equivalentes, si un conjunto de dependencias es redundante, etc.

Y es que las dependencias funcionales juegan un papel importante en la teoría relacional y el diseño de bases de datos relacionales. Las investigaciones actuales se basan en el hecho de que pueden existir dependencias funcionales en un conjunto de datos que son independientes del modelo relacional del conjunto de datos. Y es útil notar que el descubrimiento de dependencias funcionales de la base de datos se ha convertido recientemente en un problema de investigación frecuente. En consecuencia, a fines de los años noventa del siglo XX surgió

TANE (Huhtala, Kärkkäinen, Porkka, & Toivonen, 1999), que se propone como un algoritmo eficiente para encontrar dependencias funcionales de grandes volúmenes de datos. TANE se basa en el particionamiento de conjuntos de filas con respecto a sus valores de atributos, el cual permite que las pruebas de validez de las dependencias funcionales sean más rápidas y eficientes, y que además las filas erróneas o excepcionales puedan ser identificadas fácilmente. Los experimentos mostraron a TANE como un algoritmo rápido; el método de TANE consiste en representar conjuntos de atributos mediante equivalencias, sobre el conjunto de atributos de la estructura lógica de datos.

En consecuencia, para poder afrontar los problemas derivados del diseño de base de datos relacionales, también han surgido investigaciones que tratan de definir y consolidar un método algorítmico para procesar el diseño de forma automática o semiautomática. Por ejemplo, el algoritmo FD_Mine (Yao, J.Hamilton, & J.Butz, 2002) se diseñó para el descubrimiento de dependencias funcionales de bases de datos, proponiéndose como un nuevo algoritmo. FD_Mine aprovecha la extensa teoría de las dependencias funcionales para reducir tanto el tamaño del conjunto de datos como el número de dependencias funcionales que se verificarán mediante el uso de equivalencias descubiertas.

Descubrir las dependencias funcionales de las bases de datos existentes es una técnica importante que se requiere consistentemente para las herramientas de diseño y administración de bases de datos. Siendo que diversas técnicas han sido investigadas durante muchos años, este problema se ha abordado recientemente desde un punto de vista alineado a la minería de datos, de una manera novedosa y más eficiente, siguiendo los principios de los algoritmos con niveles de inteligencia. En el paper de investigación que define el algoritmo FUN (Novelli & Cicchetti, 2001), se propone una nueva caracterización de dependencias funcionales mínimas que proporciona un marco formal más simple que las propuestas anteriores. El algoritmo, definido para hacer cumplir el enfoque propuesto, ha sido implementado y probado. Se dice que es más eficiente que el algoritmo TANE.

Y, sin embargo, en los últimos años se propuso el algoritmo DFD (Abedjan, Schulze, & Naumann, 2014), el cual indica que el descubrimiento de dependencias funcionales en un conjunto de datos es de gran importancia para el rediseño de la base de datos, la detección de anomalías y las aplicaciones de limpieza de datos. Sin embargo, como la naturaleza del problema es exponencial en el número de atributos, ninguno de los enfoques existentes puede aplicarse en grandes conjuntos de datos. Por tal razón se presenta un nuevo algoritmo (DFD) para descubrir todas las dependencias funcionales en un conjunto de datos siguiendo una estrategia transversal de profundidad de la red de atributos que combina una poda agresiva y una verificación de resultados eficiente. Se indica que el enfoque es capaz de escalar mucho

más allá de los algoritmos existentes hasta 7.5 millones de tuplas, y es hasta tres órdenes de magnitud más rápido que los enfoques existentes en conjuntos de datos más pequeños.

En consecuencia, en la investigación del presente informe, teniendo en cuenta las falencias de las investigaciones que proponen algoritmos para el descubrimiento de dependencias funcionales para el diseño de bases de datos, que se pretende correcto, se propone un nuevo Modelo de Datos que se corresponde al concepto matemático de la Relación Funcional, cuyas bases de diseño se sostienen sobre Dependencias de Relación Funcional.

2. Bases Teóricas

A. Modelo de Negocio (Eriksson & Penker, 2000).

Un modelo de negocio es una abstracción de cómo funciona un negocio. Sus detalles difieren según la perspectiva de la persona que crea el modelo, cada uno de los cuales naturalmente tendrá un punto de vista ligeramente diferente de los objetivos y visiones del negocio, incluida su eficiencia y los diversos elementos que actúan en concierto dentro del negocio. Esto es normal y el modelo de negocio no resolverá por completo estas diferencias. Lo que hará el modelo comercial es proporcionar una visión simplificada de la estructura comercial que actuará como base para la comunicación, las mejoras o las innovaciones, y definirá los requisitos del sistema de información que son necesarios para respaldar el negocio. No es necesario que un modelo de negocio capture una imagen absoluta del negocio o describa cada detalle del negocio.

El modelo de negocio es el punto focal en torno al cual se llevan a cabo los negocios o alrededor del cual se mejoran las operaciones comerciales. Los modelos en evolución también ayudan a los desarrolladores de software a estructurar y enfocar su pensamiento. Trabajar con los modelos aumenta su comprensión del negocio y, con suerte, su conocimiento de las nuevas oportunidades para mejorar el negocio.

La palabra negocio en este contexto se usa como un término amplio. Las empresas que pueden modelarse con las técnicas presentadas no tienen necesariamente que generar ganancias (por ejemplo, una organización de ayuda para las personas sin hogar o para las víctimas de la guerra también es una empresa que debe organizarse de la manera más efectiva posible).

Cualquier tipo de operación continua que tenga o use recursos y tenga uno o más objetivos puede denominarse empresa. El propietario de la empresa establece los objetivos y asigna recursos para que la empresa funcione. El modelador de negocios crea

la estructura, diseña los procesos y asigna los recursos para lograr los objetivos. El desarrollador del sistema luego adapta, diseña o desarrolla sistemas de información apropiados que apoyan el funcionamiento del negocio.

Funciones del Modelo de Negocio

El modelo de negocio funciona como el plan para conducir un negocio. Actúa como la base para la toma de decisiones y afecta las decisiones sobre la priorización de objetivos, la obtención de los recursos adecuados o la negociación con subcontratistas. También sirve como una descripción actualizada de cómo se realiza el negocio, y permite cambios y mejoras en el proceso, como reducir costos, mejorar la calidad o acortar el tiempo de comercialización. El modelo puede anticipar y pronosticar los cambios que son necesarios para mantener una ventaja sobre la competencia. El modelo no puede proporcionar todas las respuestas, pero en cuanto al jugador de ajedrez, es una estrategia o plan básico a seguir. Una ventaja de modelar en un lenguaje como UML es que representa visualmente funciones y relaciones que generalmente son difíciles de visualizar con claridad.

B. **Reglas de Negocio** (Carlos Coronel, 2011, págs. 32-34)**.**

Cuando los diseñadores de bases de datos seleccionan o determinan las entidades, los atributos y las relaciones que se utilizarán para construir un modelo de datos, pueden comenzar por comprender a fondo qué tipos de datos hay en una organización, cómo se usan y en qué períodos de tiempo se usan. Pero tales datos e información, por sí mismos, no producen la comprensión requerida del negocio total. Desde el punto de vista de la base de datos, la recopilación de datos se vuelve significativa solo cuando refleja reglas de negocio definidas adecuadamente. **Una regla de negocio es una descripción breve, precisa e inequívoca de una política, procedimiento o principio dentro de una organización específica.** En cierto sentido, las reglas de negocio tienen un nombre incorrecto: se aplican a cualquier organización, grande o pequeña, una empresa, una unidad gubernamental, un grupo religioso o un laboratorio de investigación, que almacena y utiliza datos para generar información.

Las reglas de negocio, derivadas de una descripción detallada de las operaciones de una organización, ayudan a crear y aplicar acciones dentro del entorno de esa organización. Las reglas de negocio deben presentarse por escrito y actualizarse para reflejar cualquier cambio en el entorno operativo de la organización.

Las reglas de negocio escritas correctamente se utilizan para definir entidades, atributos, relaciones y restricciones. **Cada vez que vea declaraciones de relaciones como "un agente puede servir a muchos clientes, y cada cliente puede ser atendido por un solo agente", verá reglas de negocio en funcionamiento**.

Para ser efectivas, las reglas de negocio deben ser fáciles de entender y difundir ampliamente, para asegurar que cada persona en la organización comparta una interpretación común.

Las reglas de negocio describen, en un lenguaje simple, las características principales y distintivas de los datos tal como los ve la empresa.

Ejemplos de reglas de negocio son los siguientes:

- Un cliente puede generar muchas facturas.
- Una factura es generada por un solo cliente.
- No se puede programar una sesión de capacitación para menos de 10 empleados o para más de 30 empleados.

Tenga en cuenta que esas reglas de negocio establecen entidades, relaciones y restricciones. Por ejemplo, las dos primeras reglas de negocio establecen dos entidades (CLIENTE e FACTURA) y una relación 1: M entre esas dos entidades. La tercera regla de negocio establece una restricción (no menos de 10 personas y no más de 30 personas), dos entidades (EMPLEADO y FORMACIÓN), y una relación entre EMPLEADO y FORMACIÓN.

Descubriendo las Reglas de Negocio

Las principales fuentes de reglas de negocio son los gerentes de la compañía, los encargados de formular políticas, los gerentes de departamento y la documentación escrita, como los procedimientos, estándares y manuales de operaciones de la compañía. Una fuente más rápida y directa de reglas de neogocio son las entrevistas directas con los usuarios finales. Desafortunadamente, debido a que las percepciones difieren, los usuarios finales son a veces una fuente menos confiable cuando se trata de especificar reglas de negocio. Por ejemplo, un mecánico del departamento de mantenimiento podría creer que cualquier mecánico puede iniciar un procedimiento de mantenimiento, cuando

en realidad solo los mecánicos con autorización de inspección pueden realizar dicha tarea.

Tal distinción puede parecer trivial, pero puede tener importantes consecuencias legales. Aunque los usuarios finales son contribuyentes cruciales para el desarrollo de reglas de negocio, vale la pena verificar las percepciones del usuario final.

Con demasiada frecuencia, las entrevistas con varias personas que realizan el mismo trabajo arrojan percepciones muy diferentes de cuáles son los componentes del trabajo. Si bien tal descubrimiento puede apuntar a "problemas de gestión", ese diagnóstico general no ayuda al diseñador de la base de datos. El trabajo del diseñador de la base de datos es conciliar tales diferencias y verificar los resultados de la conciliación para garantizar que las reglas de negocio sean apropiadas y precisas.

El proceso de identificación y documentación de reglas de negocio es esencial para el diseño de bases de datos por varias razones:

- Ayudan a estandarizar la vista de datos de la empresa.
- Pueden ser una herramienta de comunicación entre usuarios y diseñadores.
- Permiten al diseñador comprender la naturaleza, el rol y el alcance de los datos.
- Permiten al diseñador comprender los procesos de negocio.
- Permiten al diseñador desarrollar reglas y restricciones de participación en las relaciones apropiadas y crear un modelo de datos preciso.

Traducción de Reglas de Negocio a Componentes del Modelo de Datos

Las reglas de negocio preparan el escenario para la identificación adecuada de entidades, atributos, relaciones y restricciones. En el mundo real, los nombres se usan para identificar objetos. Si el entorno empresarial desea realizar un seguimiento de los objetos, habrá reglas de negocio específicas para ellos. Como regla general, un sustantivo en una regla de negocio se traducirá en una entidad en el modelo, y un sustantivo de asociación verbal (activo o pasivo) se traducirá en una relación entre las entidades. Por ejemplo, la regla de negocios "un cliente puede generar muchas facturas" contiene dos sustantivos (cliente y facturas) y un verbo (generar) que asocia los sustantivos. De esta regla de negocios, podría deducir que:

- El cliente y la factura son objetos de interés para el medio ambiente y deben estar representados por sus respectivas entidades.
- Existe una relación de "generación" entre el cliente y la factura.

Para identificar correctamente el tipo de relación, debe considerar que las relaciones son bidireccionales; es decir, van en ambos sentidos. Por ejemplo, la regla de negocios "un cliente puede generar muchas facturas" se complementa con la regla de negocios "una factura es generada por un solo cliente". En ese caso, la relación es de uno a muchos (1: M). El cliente es el lado "1" y la factura es el lado "muchos".

Como regla general, para identificar correctamente el tipo de relación, debe hacer dos preguntas:

- ¿Cuántas instancias de B están relacionadas con una instancia de A?
- ¿Cuántas instancias de A están relacionadas con una instancia de B?

Por ejemplo, puede evaluar la relación entre el alumno y la clase haciendo dos preguntas:

- ¿En cuántas clases puede inscribirse un estudiante? Respuesta: muchas clases.
- ¿Cuántos estudiantes pueden inscribirse en una clase? Respuesta: muchos estudiantes.

Por lo tanto, la relación entre el alumno y la clase es de muchos a muchos (M: N).

Convenciones de nombres

Durante la traducción de reglas de negocio a componentes del modelo de datos, identifica entidades, atributos, relaciones y restricciones. Este proceso de identificación incluye nombrar el objeto de una manera que lo haga único y distinguible de otros objetos en el dominio del problema. Por lo tanto, es importante que preste especial atención a cómo nombra los objetos que está descubriendo.

Los nombres de las entidades deben ser descriptivos de los objetos en el entorno empresarial y utilizar una terminología que sea familiar para los usuarios. El nombre de un atributo también debe ser descriptivo de los datos representados por ese atributo. También es una buena práctica prefijar el nombre de un atributo con el nombre de la entidad (o una abreviatura del nombre de la entidad) en el que se produce. Por ejemplo,

en la entidad CLIENTE, el límite de crédito del cliente puede llamarse CUS_CREDIT_LIMIT. El CUS indica que el atributo es descriptivo de la entidad CLIENTE, mientras que CREDIT_LIMIT facilita el reconocimiento de los datos que estarán contenidos en el atributo. Esto será cada vez más importante en capítulos posteriores cuando discutamos la necesidad de usar atributos comunes para especificar relaciones entre entidades. El uso de una convención de nomenclatura adecuada mejorará la capacidad del modelo de datos para facilitar la comunicación entre el diseñador, el programador de aplicaciones y el usuario final. De hecho, una convención de nomenclatura adecuada puede contribuir en gran medida a que su modelo se auto documente.

C. **Lógica de Negocio** (Osterwalder, 2004, págs. 19-22)**.**

La investigación del modelo de negocio es un dominio de investigación bastante joven y todavía tiene que demostrar su relevancia. Pero como se mencionó anteriormente, existen relativamente pocos conceptos y herramientas para ayudar a los gerentes a **capturar, comprender, comunicar, diseñar, analizar y cambiar la lógica de negocio** de su empresa. En mi opinión y en la opinión de muchos otros investigadores en este dominio, el concepto de modelo de negocio puede llenar parte de esta brecha y eventualmente puede ganar una posición importante en la gestión bajo incertidumbre.

En las siguientes secciones, describiré algunas de las funciones que puede desempeñar el concepto de modelo de negocio (es decir, el uso de una especificación de una conceptualización de modelos de negocio) en la gestión empresarial y, en particular, en lo que respecta a cuestiones de comercio electrónico. Identifiqué cinco categorías de funciones, que son comprender y compartir, analizar, administrar, prospectos y patentar modelos de negocios. Además, un enfoque ontológico de los modelos de negocio es indispensable para crear herramientas basadas en software que ayuden a cumplir estas cinco funciones.

Describo estas categorías para dar una perspectiva de lo que podría hacerse con la ayuda del concepto de modelo de negocio, particularmente sobre la base de la ontología del modelo de negocio. El alcance de esta disertación, sin embargo, es el diseño de una ontología de modelo de negocio. Los posibles roles del concepto de modelo de negocio no se especificarán más allá de esta perspectiva, excepto por las propuestas de investigación adicional.

Comprender y compartir la Lógica de Negocio

La primera área en la que los modelos de negocios pueden contribuir es comprender y compartir la lógica de negocios de una empresa. Concretamente, los modelos de negocios ayudan a capturar, visualizar, comprender, comunicar y compartir la lógica de negocios.

- **Capturar.** Como se explicó anteriormente, el modelo de negocio de una empresa es una representación simplificada de su lógica de negocio. Sin embargo, como tales modelos de negocios existen solo como conceptos abstractos o modelos mentales en la cabeza de las personas que razonan sobre ellos. Sin embargo, la experiencia muestra que, en muchos casos, estas personas no siempre son capaces de comunicar este modelo de negocio de una manera clara (Linder y Cantrell 2000). Además, debido a que las personas tienen diferentes modelos mentales, no entenderán automáticamente lo mismo en un modelo de negocio. Por lo tanto, se hace necesario un marco genérico (es decir, una ontología) para describir los modelos de negocio. Dicho marco puede entenderse como un lenguaje común entre las partes interesadas para sacar las ideas de sus cabezas y formularlas de una manera que todos entiendan.

- **Visualizar.** La capacidad del ser humano para procesar con éxito información compleja es bastante limitada. Como se puede mostrar teórica y empíricamente, el procesamiento de la información a través del sistema visual puede aumentar sustancialmente el grado en que la complejidad se puede manejar con éxito (Rode 2000). El uso de una ontología para capturar modelos de negocios significa que con poco esfuerzo adicional se pueden presentar gráficamente (Gordijn y Akkermans 2003).

- **Entender.** Hoy en día, los modelos de negocio son cada vez más complejos, particularmente aquellos con un fuerte componente de TIC y comercio electrónico. La relación entre los diferentes elementos de un modelo de negocio y los factores decisivos de éxito no siempre son observables de inmediato. Por lo tanto, el proceso de modelar los sistemas sociales y, en este caso, los modelos de negocios ayudan a identificar y comprender los elementos relevantes en un dominio específico y las relaciones entre ellos (Morecroft 1994; Ushold y King 1995). Además, la representación visual de un modelo de negocio puede mejorar dramáticamente la comprensión.

- **Comunicar y compartir.** Ya he señalado que el concepto de modelo de negocio ayuda a capturar, comprender y visualizar la lógica empresarial de la empresa. Ser

capaz de comunicarse y compartir esta comprensión con otras partes interesadas es simplemente una consecuencia lógica de lo anterior. Formalizar los modelos de negocios y expresarlos de una manera más tangible claramente ayuda a los gerentes a comunicarse y compartir su comprensión de un negocio entre otras partes interesadas (Fensel 2001).

Analizar la Lógica de Negocio

La segunda área en la que puede contribuir el concepto de modelo de negocio es analizar la lógica de negocios de una empresa. Concretamente, pueden mejorar la medición, la observación y la comparación de la lógica comercial de una empresa.

- **Medida.** Después de haber capturado el modelo de negocio en un primer paso, puede ser más fácil identificar las medidas relevantes a seguir para mejorar la gestión. De manera similar al Enfoque del Cuadro de Mando Integral (Kaplan y Norton 1992), un modelo de negocios muestra qué áreas monitorear en un negocio en particular. Esto es aún más relevante ya que en el negocio electrónico los indicadores a seguir siguen siendo un tema de debate.

- **Observar.** La lógica empresarial de una empresa cambia constantemente debido a presiones internas y externas, como se muestra en la sección 2.3.4. Por lo tanto, un enfoque estructurado de los modelos de negocio es importante para comprender qué problemas particulares han cambiado con el tiempo.

- **Comparar.** Al igual que observar el modelo comercial de una empresa a lo largo del tiempo, un enfoque estructurado permite a las empresas comparar su modelo comercial con el de sus competidores. Esto se basa en el razonamiento de que las cosas solo son comparables si se confiscan y se entienden de la misma manera. Además, comparar el modelo de negocio de uno con el de una empresa en una industria completamente diferente puede ayudar a obtener nuevos conocimientos y fomentar la innovación del modelo de negocio. En relación con el comercio electrónico y las industrias dinámicas, esto puede ayudar a los titulares a comprender cuán agresivos son los nuevos competidores y las nuevas empresas.

Gestionar la Lógica de Negocio

La tercera área de contribución de los modelos de negocio es mejorar la gestión de la lógica de negocios de la empresa.

El concepto de modelo de negocio ayuda a mejorar el diseño, la planificación, el cambio y la implementación de modelos de negocio. Además, con un enfoque de modelo de negocio, las empresas pueden reaccionar más rápido a los cambios en el entorno empresarial. Finalmente, el concepto de modelo de negocio mejora la alineación de la estrategia, la organización empresarial y la tecnología.

- **Diseño.** Diseñar un modelo de negocio coherente donde todos los elementos se refuercen mutuamente o al menos estén optimizados no es una tarea fácil. Hoy en día, los modelos de negocio son bastante complejos y su éxito a menudo se basa en la interacción de una serie de elementos aparentemente menores. Este es aún más el caso, ya que el comercio electrónico aumenta la gama de modelos de negocios imaginables. Tener a mano una ontología de modelo de negocio que describa los componentes esenciales y sus relaciones facilitará a los gerentes el diseño de un modelo de negocio sostenible.

- **Planificar, cambiar e implementar.** Cuando una empresa decide adoptar un nuevo modelo de negocio o cambiar uno existente, capturar y visualizar este modelo mejorará la planificación, el cambio y la implementación. Es mucho más fácil ir de un punto a otro, cuando uno puede entender exactamente, decir y mostrar qué elementos cambiarán. En este sentido, Linder y Cantrell (Linder y Cantrell 2000) hablan de los llamados modelos de cambio que son la lógica central de cómo una empresa cambiará con el tiempo para seguir siendo rentable en un entorno dinámico.

- **Reaccionar.** Una vez que los gerentes han capturado, mapeado y entendido el modelo de negocio, se han creado las bases para mejorar la velocidad y la idoneidad de la reacción a las presiones externas. Según Petrovic y Kittl (2001), los diseñadores de modelos de negocio pueden modificar fácilmente ciertos elementos de un modelo de negocio existente. Esto es sin duda esencial en un panorama competitivo incierto y rápidamente cambiante.

- **Alinear.** Ya he argumentado anteriormente que el concepto de modelo de negocio puede servir como federador entre el triángulo de la estrategia comercial, la organización empresarial y la tecnología. En otras palabras, el modelo de negocio forma una especie de puente conceptual que facilita la alineación de estos tres. Chesbrough y Rosenbloom (2000), por ejemplo, ven los modelos de negocio como una construcción mediadora entre la tecnología y el valor económico.

- **Mejora la toma de decisiones.** Habiendo afirmado que el concepto de modelo de negocio mejora la comprensión y la comunicación de la lógica comercial de la empresa, deduzco que los tomadores de decisiones toman decisiones más informadas y, por lo tanto, mejores. Además de esto, los modelos de negocio son una nueva unidad de análisis (Stähler 2002) que se pueden observar y comparar, ayudan a definir medidas y, por lo tanto, también deben mejorar las decisiones.

Perspectiva de la Lógica de Negocio

Una cuarta área de contribución de los modelos de negocio se refiere a los posibles futuros de una empresa. Creo que el concepto de modelo de negocios puede ayudar a fomentar la innovación y aumentar la preparación para el futuro a través de carteras y simulaciones de modelos de negocios.

- **Innovar.** Similar al argumento de mejorar el cambio y aumentar las capacidades de reacción en la empresa, creo que un enfoque de modelo de negocio conceptual y modular puede fomentar la innovación. De hecho, especificar un conjunto de elementos de modelo de negocio y bloques de construcción, así como sus relaciones entre sí, es como darle a un diseñador de modelos de negocio una caja de piedras de Lego. Puede jugar con estas piedras y crear modelos de negocio completamente nuevos, limitados solo por su imaginación y las piezas suministradas. Amit y Zott (2001) perciben explícitamente el modelo de negocio como un lugar de innovación.

- **Portafolio de modelos de negocios.** Con base en la ley de exceso de diversidad de Allen en teoría evolutiva (Allen 2001), se puede argumentar que podría ser interesante para una compañía mantener una cartera de modelos de negocio para estar preparada para el futuro. La idea detrás de la ley de Allen es que una estrategia evolutiva sostenible y exitosa requiere una cantidad de diversidad interna superior a la del medio ambiente. Allen sugiere que los agentes deben tener un stock de estrategias potenciales para ser enfrentados ante la imprevisibilidad en el cambio ambiental (Andriani 2001). En el caso de una empresa, esto significaría tener un stock de modelos de negocio para hacer frente al cambio.

- **Simular y probar.** Simular y probar modelos de negocio es obviamente el sueño de todo gerente. Aunque la simulación nunca podrá predecir el futuro, es una forma de hacer experimentos sin riesgos, sin poner en peligro a una organización (Sterman 2000). Al simular y probar posibles modelos de negocio, los gerentes estarán mejor preparados para el futuro. Del mismo modo, en el dominio del comercio electrónico,

Richards y Morrison (2001) comparan este tipo de herramienta de simulación con una especie de simulador de vuelo que permite construir mejores estrategias de comercio electrónico.

Patentar la Lógica de Negocio

Cada vez más, los empresarios y las empresas del comercio electrónico buscan patentar procesos de comercio electrónico e incluso tener en cuenta aspectos de su modelo de negocio. Por lo tanto, el modelado de negocios puede tener un papel importante en este dominio legal. Por ejemplo, Priceline basó gran parte de su estrategia comercial en una patente cuya tecnología iguala las ofertas de compradores con vendedores interesados en la red (Angwin 2000).

En consecuencia, patentar métodos de comercio electrónico también ha comenzado a crear una serie de batallas legales. Uno famoso es el caso entre el minorista en línea Amazon.com y el brazo en línea del librero Barnes & Noble (B&N). Amazon.com, que recibió una patente por su sistema de pedidos de "un clic", atacó a B&N por infracción de patente, supuestamente causada por su sistema de pago de "carril rápido" en el sitio web de B&N (Lesavich 2001). Queda por ver en qué dirección se patentan los modelos comerciales y los procesos comerciales.

D. Ontología del Modelo de Negocio (Osterwalder, 2004, págs. 42-47).

Una ontología permite describir con precisión el modelo de negocio de una empresa. Para lograr esto, se debe identificar cuatro áreas principales que constituyen los problemas esenciales del modelo de negocio de una empresa.

Influenciado por el enfoque del Cuadro de Mando Integral (Kaplan y Norton 1992) y, más en general, por la literatura de gestión empresarial (Markides 1999), se sugiere adoptar un marco que enfatice en las siguientes cuatro áreas que un modelo comercial debe abordar:

- **PRODUCTO:** En qué negocio se encuentra la empresa, los productos y las propuestas de valor que se ofrecen al mercado.
- **INTERFAZ DE CLIENTE:** Quiénes son los clientes objetivos de la empresa, cómo les ofrece productos y servicios y cómo construye una relación sólida con ellos.

- **GESTIÓN DE LA INFRAESTRUCTURA:** Cómo la empresa realiza eficientemente los problemas de infraestructura o logística, con quién y con qué tipo de empresa de red.

- **ASPECTOS FINANCIEROS:** ¿Cuál es el modelo de ingresos, la estructura de costos y la sostenibilidad del modelo de negocio?

Estas cuatro áreas se pueden comparar con las cuatro perspectivas del enfoque del Cuadro de Mando Integral de Norton y Kaplan (Kaplan y Norton 1992). El Balanced Scorecard es un concepto de gestión desarrollado a principios de los 90 que ayuda a los gerentes a medir y monitorear indicadores distintos a los puramente financieros. Los autores comparan su herramienta ahora bastante conocida con la cabina de un avión donde el piloto vuela el avión reaccionando a la información que obtienen de sus herramientas de tablero. Evidentemente, esta información tiene que cubrir todos los aspectos relevantes de volar un avión. Lo mismo se aplica a las empresas donde los gerentes deben monitorear las áreas esenciales de un negocio para liderarlo. Norton y Kaplan identifican cuatro perspectivas de la empresa sobre las cuales los ejecutivos deben estar atentos para realizar negocios exitosos. Desde la perspectiva del cliente, la empresa se pregunta cómo es vista por sus clientes. En la perspectiva interna, la empresa reflexiona sobre en qué debe sobresalir. En la perspectiva de innovación y aprendizaje, la empresa analiza cómo puede continuar mejorando y creando valor. Finalmente, en la perspectiva financiera, una empresa se pregunta cómo se ve a los accionistas. **Estas perspectivas parecen bastante adecuadas como punto de partida para una ontología de modelo de negocio**, más aún porque Norton y Kaplan proponen que pueden servir para la estrategia de mapeo en algunos de sus trabajos posteriores (Kaplan y Norton 2000).

Markides (Markides 1999) sigue un camino similar al proporcionar una receta muy simple para la estrategia comercial. Recomienda mirar el "quién", el "qué" y el "cómo" de un negocio. Esto significa que la primera pregunta que los ejecutivos deben hacerse es a quién deben dirigirse como clientes. La segunda pregunta es sobre qué productos o servicios debe ofrecer una empresa.

La última pregunta es sobre cómo estos servicios se pueden ofrecer mejor a los clientes. Estas tres trayectorias intuitivas son comparables a las perspectivas mencionadas anteriormente si se agrega el aspecto financiero a la receta de Markides.

La ontología del modelo de negocio es un conjunto de elementos y sus relaciones que tienen como objetivo describir la lógica de una empresa para ganar dinero. Cada elemento del modelo de negocio puede descomponerse en un conjunto de subelementos definidos.

Esta descomposición permite estudiar modelos de negocio en diferentes niveles de granularidad con más o menos detalle y de acuerdo con necesidades específicas.

E. Modelo de Base de Datos (Neeraj Sharma, 2010, págs. 67-68).

Un modelo de base de datos se utiliza para representar datos sobre datos, también conocidos como metadatos.

Un modelo de base de datos es una colección integrada de conceptos para descripción de datos, relaciones de datos, semántica de datos y restricciones de datos. Por lo general, se utiliza un modelo de base de datos para una descripción del esquema de la base de datos.

Los tipos de modelos de bases de datos son:

- **Modelo de Datos Externo.** Este modelo se usa para ver una representación de cada usuario, también llamado Universo del Discurso. El modelo principal es el Modelo Lógico basado en registros, cuyas vertientes son: el **Modelo Relacional**, el **Modelo de Red** y el **Modelo Jerárquico**.

- **Modelo de Datos Conceptual.** Este modelo se utiliza para una vista general de los datos y es independiente de un Sistema de Gestión de Base de Datos específico. El Modelo Lógico basado en objetos representa este modelo, y cuyas vertientes son: el **Modelo Entidad-Relación** y el **Modelo Orientado a Objetos**.

- **Modelo de Datos Interno.** Este modelo se utiliza para una traducción del modelo conceptual a un Sistema de Gestión de Base de Datos específico. El Modelo de Datos Físico representa este modelo.

El modelo más utilizado hoy en día es el modelo de base de datos relacional.

Los componentes de un modelo de base de datos son:

- **Componente Estructural:** Esto significa un conjunto de reglas comunes para desarrollar una base de datos.

- **Componente de Manipulación:** Se debe definir las operaciones aplicadas en la base de datos (para buscar o actualizar datos en una base de datos o para modificar la estructura de la base de datos).

- **Componente de Integridad de Datos:** Es un conjunto de reglas de integridad que garantiza la exactitud de los datos.

F. Modelo de Datos (Ullman & Widom, 2008, págs. 17-18).

La noción de un "modelo de datos" es una de las más fundamentales en el estudio de los sistemas de bases de datos. En este breve resumen del concepto, definimos una terminología básica y mencionamos los modelos de datos más importantes.

Un modelo de datos es una notación para describir datos o información. La descripción generalmente consta de tres partes:

- **Estructura de los datos.** Puede estar familiarizado con herramientas en lenguajes de programación como Cor Java para describir la estructura de los datos utilizados por un programa: matrices y estructuras ("estructuras") u objetos, por ejemplo. Las estructuras de datos utilizadas para implementar datos en la computadora a veces se denominan, en las discusiones sobre los sistemas de bases de datos, como un modelo de datos físicos, aunque en realidad están muy lejos de las puertas y electrones que realmente sirven como la implementación física de los datos. En el mundo de las bases de datos, los modelos de datos están en un nivel algo más alto que las estructuras de datos, y a veces se los conoce como modelos conceptuales para enfatizar la diferencia de nivel. Veremos ejemplos en breve.

- **Operaciones sobre los datos.** En los lenguajes de programación, las operaciones en los datos son generalmente cualquier cosa que se pueda programar. En los modelos de datos de bases de datos, generalmente hay un conjunto limitado de operaciones que se pueden realizar. En general, se nos permite realizar un conjunto limitado de consultas (operaciones que recuperan información) y modificaciones (operaciones que cambian la base de datos). Esta limitación no es una debilidad, sino una fortaleza. Al limitar las operaciones, es posible que los programadores describan las operaciones de la base de datos a un nivel muy alto, pero que el sistema de administración de la base de datos implemente las operaciones de manera eficiente. En comparación, generalmente es imposible optimizar programas en lenguajes convencionales como C, en la medida en que un algoritmo ineficiente (p. Ej., Bubbleort) se reemplaza por uno más eficiente (p. Ej., Quicksort).

- **Restricciones en los datos.** Los modelos de datos de bases de datos generalmente tienen una forma de describir las limitaciones de lo que pueden ser los datos. Estas

restricciones pueden variar desde lo simple (por ejemplo, "un día de la semana es un número entero entre 1 y 7" o "una película tiene como máximo un título") hasta algunas limitaciones muy complejas.

Modelos de datos importantes

Hoy, los dos modelos de datos de importancia preeminente para los sistemas de bases de datos son:

- El modelo relacional, incluidas las extensiones objeto-relacionales.
- El modelo de datos semiestructurados, que incluye XML y estándares relacionados.

G. Modelo Conceptual de Datos (Neeraj Sharma, 2010, págs. 67-77).

En el proceso de creación de un modelo de datos, primero se debe crear un modelo conceptual de los datos. **El modelo de datos conceptual es una imagen mental de un objeto físico familiar y no es específico de una base de datos.** En un nivel superior, describen las cosas de las que una organización desea recopilar datos y las relaciones entre estos objetos.

El objetivo de un diseño de base de datos conceptual es construir un modelo de datos conceptual. Para hacer eso, es importante seguir algunos pasos:

- **Esbozar un Diagrama de Entidad-Relación.** Primero se debe crear conjuntos de entidades para identificar atributos y establecer los conjuntos de relaciones adecuados.
- **Definir Restricciones de Integridad.** Se debe identificar y documentar las restricciones de integridad, como los datos requeridos, la integridad referencial, las restricciones del dominio de atributos, las restricciones empresariales y la integridad de la entidad.
- **Revisar el Modelo Final.** Esto requiere que se elimine las relaciones “M: N”, se elimine las relaciones recursivas, se elimine los supertipos, se elimine las relaciones con los atributos y se vuelva a examinar las relaciones “1:1”, que normalmente no son necesarias.

Modelo de entidad-relación

El modelo Entidad-Relación ha sido principalmente exitoso como una herramienta para diseñar bases de datos relacionales. Un diagrama de relación de entidad (DER) a menudo se usa para representar los requisitos de datos independientemente del tipo de base de datos utilizada, o incluso si se utiliza una base de datos. Un DER es una representación de datos estructurados.

Los conceptos utilizados en un modelo de entidad-relación son:

- **Conjunto de entidades.**
- **Atributo.**
- **Conjunto de relaciones.**
- **Restricción.**
- **Dominio.**
- **Extensión.**
- **Intensión.**

Entidades y conjuntos de entidades

Un conjunto de entidades es un conjunto de entidades del mismo tipo que comparten las mismas propiedades. Un sustantivo se utiliza para representar un conjunto de entidades.

Una entidad es una instancia de un conjunto de entidades. Una entidad es un elemento autodeterminable y distinguible dentro de un conjunto de entidades. Por ejemplo, una entidad puede ser:

- **Concreta** (PROFESOR o ESTUDIANTE)
- **Insustancial** (GRADO)
- **Una ocurrencia** (EXAMEN)

Dependiendo del contexto, un nombre dado como TEACHER podría usarse como una entidad, o un conjunto de entidades. Por ejemplo, si necesita diferentes tipos de personas, como PROFESOR o ESTUDIANTE, creará un conjunto de entidades llamado PERSONA con las entidades PROFESOR y ESTUDIANTE.

Si el contexto es un entorno universitario, use el PROFESOR como una entidad establecida con las entidades de PROFESOR PRINCIPAL, PROFESOR ASOCIADO, PROFESOR ASISTENTE, etc.

En un modelo lógico, las entidades se denominan tuplas y los conjuntos de entidades se denominan relaciones. Por ejemplo, puede crear una relación llamada PERSONA con dos atributos: NOMBRE y DIRECCIÓN. En este caso, escribe: PERSON = (NOMBRE, DIRECCIÓN).

Atributos

Un atributo es un elemento de datos que describe una propiedad de un conjunto de entidades. Los atributos determinan, explican o categorizan un conjunto de entidades.

Los atributos tienen valores, que pueden ser de diferentes tipos de datos, como números, cadenas de caracteres, fechas, imágenes, sonidos, etc. Cada atributo solo puede tener un valor para cada instancia del conjunto de entidades.

En un modelo físico, un atributo es una columna con nombre en una tabla y tiene un dominio. Cada tabla tiene una lista de atributos (o columnas).

Los tipos de atributos son:

- **Atributo simple (atómico):** este tipo de atributo tiene un solo componente. Por ejemplo, el atributo Gender tiene un solo componente con dos valores.
- **Atributo compuesto:** un atributo compuesto consta de muchos componentes. Por ejemplo, el atributo Name tiene los componentes apellido y nombre.
- **Atributo de valor único:** este tipo de atributo tiene un valor para una entidad. Por ejemplo, el atributo Título tiene un valor único para cada maestro.
- **Atributo de valores múltiples:** un atributo de valores múltiples tiene muchos valores para una entidad. Por ejemplo, una persona tiene muchos números de teléfono. Cada atributo solo puede tener un valor para cada instancia del conjunto

de entidades. Cuando encuentre un atributo de valores múltiples, debe transferirlo a otro conjunto de entidades.

- **Atributo derivado:** un atributo derivado tiene su valor calculado a partir de otro atributo o atributos. Por ejemplo, la suma de todos los estudiantes en una facultad. Un atributo derivado no es parte de una tabla de una base de datos, pero se muestra por claridad o se incluye con fines de diseño, aunque no agrega información semántica; También proporciona pistas para los programadores de aplicaciones.

- **Atributos inestables:** este tipo de atributo tiene valores que siempre cambian. Por ejemplo, el año de estudio de un estudiante.

- **Atributos estables:** los atributos estables cambiarán en alguna ocasión, si alguna vez.

- **Atributos obligatorios:** los atributos obligatorios deben tener un valor. Por ejemplo, en la mayoría de las empresas que rastrean información personal, se requiere Nombre.

- **Atributos opcionales:** los atributos opcionales pueden tener un valor o dejarse nulos.

- **Identificador único:** este tipo de atributo distingue una entidad de otra. Por ejemplo, en un salón de clases, puede distinguir entre un estudiante y otro usando una identificación de estudiante.

En el caso del modelo lógico, utiliza un enfoque especial para un identificador único. El concepto equivalente para un identificador único dentro del Modelo Lógico es una clave. Una clave es un campo o un conjunto de campos que tiene o tienen un valor único para cada registro en la relación. Necesita una clave para asegurarse de no cumplir con las redundancias dentro de una relación. Existen varios tipos de claves, cada una con características ligeramente diferentes:

- **Clave candidata:** una clave candidata es un atributo o conjunto de atributos que identifica de forma exclusiva un registro en una relación.

- **Clave primaria:** una clave primaria es una de las claves candidatas de una relación. Toda relación debe tener una clave primaria. Una clave primaria debe ser al menos:

- ✓ **Estable.** El valor de una clave primaria no debe cambiar o quedar nulo a lo largo de la vida de la entidad. Por ejemplo, considere un registro de estudiante; usar el campo de edad como clave primaria no sería apropiado porque el valor de la clave primaria no debe cambiar con el tiempo.

- ✓ **Mínimo.** La clave primaria debe estar compuesta por el número mínimo de campos para garantizar que las ocurrencias sean únicas.

- **Clave alternativa:** una clave alternativa es cualquier clave candidata que no se elija como clave principal. Puede convertirse en la clave principal si la clave primaria seleccionada no es apropiada.

- **Clave sustituta:** una clave sustituta actúa como clave principal pero no existe en el mundo real como un atributo real de un conjunto de entidades. Debido a que la clave sustituta no dice nada sobre el conjunto de entidades y proporciona datos incorrectos que aumentan artificialmente el tamaño de la base de datos, no es una buena práctica usarla. Con una herramienta como *InfoSphere Data Architect*, puede agregar claves sustitutas.

- **Claves simples:** estas teclas tienen un solo atributo.

- **Claves compuestas:** estas teclas tienen múltiples atributos.

- **Claves foráneas:** estas claves existen generalmente cuando hay dos o más relaciones. Un atributo de una relación tiene que existir en la otra relación. Existen conjuntos de relaciones entre los dos atributos. Se encuentra una situación especial cuando se encuentra con un conjunto de relaciones unarias. En esta situación, debe usar una clave externa dentro de la misma relación.

Conjuntos de relaciones

Un conjunto de relaciones es un conjunto de relaciones entre dos o más conjuntos de entidades, y se representan regularmente utilizando un verbo.

Una relación es una instancia de un conjunto de relaciones y establece una asociación entre entidades que están relacionadas. Estas relaciones representan algo importante en el modelo.

Siempre existe un conjunto de relaciones entre dos conjuntos de entidades (o un conjunto de entidades relacionado con sí mismo). Debe leer un conjunto de relaciones en doble sentido, de un conjunto de entidades a otro.

Un conjunto de relaciones puede definirse más formalmente como una relación matemática en conjuntos de entidades de la siguiente manera.

Vamos a dejar que los siguientes conceptos estén representados por las siguientes variables:

- **Conjunto de entidades:** E
- **Entidad:** e
- **Conjunto de relaciones:** R
- **Relación:** r

Dado un conjunto de entidades ***E1, E2 ... Ek***, una relación R define una regla de correspondencia entre estos conjuntos de entidades. Una instancia ***R (E1, E2, ..., EK)*** de la relación ***R*** significa que las entidades ***E1, E2 ... Ek*** están en una relación **R** en esta instancia.

Restricciones

Cada negocio tiene restricciones sobre qué valores de atributo y qué relaciones están permitidas. En el modelo de datos conceptuales, las restricciones se utilizan para manejar estas restricciones.

Una restricción es un requisito que los conjuntos de entidades deben satisfacer en una relación. Las restricciones pueden referirse a un solo atributo de un conjunto de entidades, o a conjuntos de relaciones entre entidades. Por ejemplo, "cada PROFESOR debe trabajar en un solo DEPARTAMENTO".

Los tipos de restricciones son:

- **Cardinalidades:** Se basan en la cantidad de conjuntos de relaciones posibles para cada conjunto de entidades. Las diferentes restricciones de cardinalidad se enumeran a continuación. Dada una relación binaria R entre E (entidad izquierda

en las figuras a continuación) y F (entidad derecha en las figuras a continuación), se dice que R es:

- ✓ **uno a uno (1: 1):** Si tanto E como F tienen una participación de valor único como se muestra en la figura a continuación.
- ✓ **uno a muchos (1: M):** Si E tiene una participación individual y F tiene una participación de valores múltiples como se muestra en la figura a continuación.
- ✓ **muchos a muchos (M: M):** Si tanto E como F tienen una participación de valores múltiples

Las relaciones de muchos a muchos no son compatibles con el modelo relacional y deben resolverse dividiendo el conjunto de relaciones “M: M” original en dos conjuntos de relaciones “1: M”.

Usualmente, los identificadores únicos de los dos conjuntos de entidades participan en la construcción del identificador único del tercer conjunto de entidades.

- ➢ **Cardinalidades de participación (opcionalidad):** Este tipo de restricción especifica si la existencia de un conjunto de entidades depende de estar relacionado con otro conjunto de entidades a través del conjunto de relaciones. Las cardinalidades de participación pueden ser:
  - ✓ **Total u obligatorio:** cada conjunto de entidades debe participar en una relación y no puede existir sin esa participación; La participación es obligatoria.
  - ✓ **Parcial u opcional:** cada conjunto de entidades puede participar en una relación; La participación no es obligatoria.
- ➢ **Subconjuntos y supersets:** Cuando un grupo de instancias tiene propiedades especiales como atributos o conjuntos de relaciones que existen solo para ese grupo, tiene sentido subdividir un conjunto de entidades en subconjuntos. El conjunto de entidades es un superconjunto llamado “padre”. Cada grupo es un subconjunto llamado “hijo”. Un subconjunto consiste en todos los atributos del superconjunto y se hace cargo de todos los conjuntos de relaciones del superconjunto. Un subconjunto existe solo junto con otros subconjuntos y puede tener subconjuntos propios. Un subconjunto agrega regularmente sus propios

atributos o conjuntos de relaciones al superconjunto principal. Usando subconjuntos y superconjuntos puedes crear jerarquías.

- **Jerarquía:** Una jerarquía representa un conjunto ordenado de elementos. Por ejemplo, en una escuela, puede haber una jerarquía.

- **Conjunto de relaciones unarias:** En este tipo de restricción, el mismo conjunto de entidades participa muchas veces en los mismos conjuntos de relaciones. También se conoce como el conjunto de relaciones recursivas.

- **Datos históricos:** En algunos casos, debe conservar los datos a lo largo de las bases de datos. Por ejemplo, cuando los valores de los atributos están cambiando o cuando los conjuntos de relaciones están cambiando. Puede usar datos históricos como una restricción relacionada con el tiempo; por ejemplo, la restricción puede especificar que la fecha de finalización sea siempre posterior a la fecha de inicio de un atributo dado. En el modelo físico, puede usar una restricción ***CHECK*** para este propósito. Las restricciones a considerar incluyen que la fecha de inicio puede cambiarse a una fecha anterior, la actividad ya ha comenzado o la fecha de inicio y finalización no es la misma fecha.

  Ejemplo: Un estudiante debe graduarse después de comenzar sus estudios, no antes. En este caso, la fecha de finalización debe ser mayor que la fecha de inicio.

  Además, cuando es necesario modificar la información existente, debe mantener los valores anteriores. Esto es particularmente importante en situaciones en las que la información es sensible, como un cambio de calificación del estudiante.

Dominio

Un dominio es un conjunto de valores posibles para uno o más atributos. El usuario puede definir la definición del dominio. Los modelos de dominio permiten al usuario definir tipos de datos específicos para el negocio.

La cláusula ***CHECK*** en el estándar ***SQL*** permite restringir dominios. La cláusula ***CHECK*** permite que el diseñador de esquemas especifique un predicado que debe ser satisfecho por cualquier valor asignado a una variable cuyo tipo está en el dominio.

Extensión

Las bases de datos cambian con el tiempo. Los datos en una base de datos en un punto particular en el tiempo se llaman extensiones de la base de datos. La extensión se refiere al conjunto actual de tuplas en una relación, es una instancia de los conjuntos de registros y los conjuntos de relaciones entre ellos.

Intensión

Intensión o esquema es el modelo lógico de una base de datos y está representado por conjuntos de entidades. Una instancia describe y restringe la estructura de las tuplas que puede contener. Una instancia está representada por conjuntos de entidades y es el modelo de una base de datos. Las operaciones de manipulación de datos en tuplas se permiten solo si observan las intensidades expresadas de las relaciones afectadas.

H. **Relación** (Codd, 1990, págs. 1-2)**.**

La palabra *relación* se usa en inglés y otros idiomas naturales sin preocuparse por una comunicación precisa. Incluso en los diccionarios que intentan ser precisos, las definiciones son bastante flojas, poco económicas y ambiguas. El *Oxford English Dictionary* dedica una página entera de letra pequeña a la palabra "relación". Una pequeña parte de la descripción es la siguiente: "Esa característica o atributo de las cosas que implica considerarlas en comparación o contraste entre sí; la forma particular en que se piensa una cosa en relación con otra; cualquier conexión, correspondencia o asociación, que puede concebirse como algo que existe naturalmente entre las cosas ".

Por otro lado, los matemáticos se preocupan por la comunicación precisa, un nivel muy alto de abstracción y la economía del esfuerzo que se deriva de hacer que las definiciones y los teoremas sean lo más generales posible. Una preocupación especial es evitar la necesidad de un tratamiento especial de casos especiales, excepto cuando sea absolutamente necesario. La definición generalmente aceptada de una relación en matemáticas es la siguiente: "Dados los conjuntos *S1, S2, …, Sn* (no necesariamente distintos), R es una relación en estos n conjuntos si es un conjunto de n-tuplas, el primer componente se extrae de *S1*, el segundo componente de *S2*, y así sucesivamente ".

De manera más concisa, R es un subconjunto del producto cartesiano *S1 × S2 x, . . .,x Sn*. Se dice que la relación R es de grado n. Cada uno de los conjuntos *S1, S2, …, Sn* en el que se definen una o más relaciones se denomina dominio.

Es importante tener en cuenta que una relación matemática es un conjunto con propiedades especiales. Primero, todos sus elementos son tuplas, todas del mismo tipo. En segundo lugar, es un conjunto desordenado. Esto es justo lo que se necesita para las bases de datos comerciales, ya que es probable que muchas de las relaciones en dichas bases de datos tengan miles de tuplas, a veces millones. En varias bases de datos desarrolladas recientemente, hay dos mil millones de tuplas. En tales circunstancias, los usuarios no deben cargar con la numeración o el orden de las tuplas.

El modelo relacional se ocupa de las tuplas por su contenido de información, no por medios ajenos a las tuplas, como números de tuplas, identificadores de tuplas o direcciones de almacenamiento. El modelo también evita sobrecargar a los usuarios al tener que recordar qué tuplas están al lado de cada una, en cualquier sentido de "proximidad".

Como una consecuencia de adoptar relaciones como la percepción del usuario de la forma en que se organizan los datos, los programas de aplicación se vuelven independientes de cualquier orden de tuplas en el almacenamiento que pueda estar vigente en algún momento. Esto permite cambiar el orden almacenado de las tuplas cuando sea necesario sin afectar negativamente la corrección de los programas de aplicación. Los cambios en el orden almacenado pueden tener que hacerse por una variedad de razones. Por ejemplo, el patrón de tráfico en la base de datos puede cambiar y, en consecuencia, el orden adoptado anteriormente puede no ser el más adecuado para obtener un buen rendimiento.

Una relación matemática tiene una propiedad que algunas personas consideran contra-intuitiva, y que no parece ser coherente con la definición en el Oxford English Dictionary. Esta propiedad es que una relación unaria (grado uno) puede ajustarse a esta definición. Por lo tanto, una relación matemática de grado mayor que uno interrelaciona dos o más objetos, mientras que una relación matemática de grado uno no. En algunos casos, la intuición puede ser una guía pobre. En cualquier caso, ya sea que el concepto de una relación unaria sea contra-intuitivo o no, a los matemáticos y a las personas orientadas a la informática no les gusta tratarlo de manera diferente de las relaciones de mayor grado.

Al aplicar las computadoras de manera efectiva (ya sea en ciencia, ingeniería, educación o comercio) existe, o debería existir, una preocupación similar por la comunicación precisa, un alto nivel de abstracción y generalidad. Sin embargo, si no se tiene cuidado, el grado de generalidad a veces se puede alcanzar más allá de lo que se necesita en la práctica, y esto puede tener consecuencias costosas.

Una relación R en el modelo relacional es muy similar a su contraparte en matemáticas. Cuando se concibe como una tabla, R tiene las siguientes propiedades:

- Cada fila representa una tupla de R;
- El orden de las filas es irrelevante;
- Todas las filas son distintas entre sí en contenido.

I. Estructuras Lógicas de Datos (cl500).

Se utilizan varias estructuras lógicas de datos para expresar las relaciones entre elementos de datos individuales o registros en una base de datos. Las estructuras lógicas de datos comunes son jerárquicas, de red y relacionales, predominando las relacionales.

Estructura jerarquica

En una estructura jerárquica a veces denominada estructura de árbol, los datos almacenados se vuelven cada vez más detallados a medida que se ramifican más y más en el árbol. Cada segmento, o nodo, puede subdividirse en dos o más nodos subordinados, que pueden subdividirse en dos o más nodos adicionales. Sin embargo, cada nodo puede emanar de un solo "padre". Para el usuario, cada registro se asemeja a un organigrama en el que los segmentos o nodos se ajustan a una jerarquía o árbol bien definidos. Solo hay un segmento, o raíz, en la parte superior. Los programas de aplicaciones procesan bases de datos jerárquicas un registro a la vez, como con las estructuras de archivos convencionales. Las estructuras de bases de datos jerárquicas se usan comúnmente con grandes sistemas informáticos de mainframe.

Estructura de red

La estructura de red es similar a la estructura jerárquica con la excepción de que en la estructura de red, un nodo puede tener más de un padre. La compensación entre la simplicidad del diseño de una estructura jerárquica y la eficiencia de almacenamiento de una estructura de red es una consideración muy importante en la implementación de la base de datos. Las estructuras de red se usan más comúnmente con los sistemas mainframe y minicomputadora, rara vez con microcomputadoras.

Estructura relacional

La estructura relacional organiza los datos en tablas bidimensionales. Estas tablas ofrecen una gran flexibilidad y un alto grado de seguridad de datos. La estructura relacional usa relativamente poca memoria o almacenamiento secundario. Desafortunadamente, el proceso de creación de estas tablas es bastante elaborado. Otra desventaja de esta estructura es que generalmente requiere más tiempo para acceder a la información que cualquiera de las otras dos estructuras. Esto se debe a que se debe buscar mucha más información para responder a las preguntas planteadas al sistema. Además, algunas implementaciones usan una cantidad fija de almacenamiento para cada campo, lo que resulta en una utilización de almacenamiento ineficiente. A pesar de estas desventajas, la estructura relacional ha ganado una rápida aceptación y actualmente es la más popular de las tres estructuras. Esta estructura se usa casi exclusivamente con sistemas de microcomputadoras y se aplica cada vez más a sistemas de minicomputadoras y mainframes. Muchos expertos predicen que eventualmente reemplazará las otras estructuras por completo.

Las estructuras de los datos dentro de una base de datos se pueden ver de dos maneras, física y lógicamente.

La estructura de datos físicos se refiere a la disposición física de los datos en el dispositivo de almacenamiento secundario, generalmente el disco. Por lo general, esta estructura física es una preocupación de los especialistas que diseñan DBMS. Los analistas, programadores y usuarios generalmente están menos interesados en la estructura física que en la estructura lógica.

La estructura de datos lógica se refiere a cómo los datos "parecen" estar organizados y los significados de los elementos de datos en relación entre sí. Esta estructura, o modelo, generalmente se define en términos de un esquema, una visión conceptual general de las relaciones lógicas entre los elementos de datos en la base de datos. Incluye los nombres de los elementos principales, sus atributos y las relaciones lógicas entre ellos. La siguiente figura muestra un esquema que podría aplicarse a la base de datos de una tienda de autopartes. Aunque está muy simplificado, ilustra que existen relaciones entre los números de parte, los tipos de automóviles, los modelos y los fabricantes. Las líneas de conexión indican enlaces y las flechas el tipo de relación.

Por ejemplo, en este esquema, el fabricante está vinculado al número de parte, y la relación en uno a muchos (una flecha de dos puntas implica muchas y una flecha de una punta). Es decir, un fabricante produce muchos números de parte, pero cada número de

parte tiene solo un fabricante. También hay enlaces indirectos indicados. Un número de parte, por ejemplo, está vinculado al tipo de vehículo a través del modelo.

J. **Modelo de Datos Relacional** (Ullman & Widom, 2008, págs. 21-26)**.**

El modelo relacional nos brinda una única forma de representar datos: como una tabla bidimensional llamada relación. El modelo relacional se basa en tablas. La porción de estructura del modelo relacional podría parecerse a una matriz de estructuras en C, donde los encabezados de columna son los nombres de campo, y cada una de las filas representa los valores de una estructura en la matriz.

Sin embargo, debe enfatizarse que esta implementación física es solo una de las formas posibles en que la tabla podría implementarse en estructuras de datos físicos. De hecho, no es la forma normal de representar relaciones, y una gran parte del estudio de los sistemas de bases de datos aborda las formas correctas de implementar tales tablas. Gran parte de la distinción proviene de la escala de las relaciones: normalmente no se implementan como estructuras de memoria principal, y su implementación física adecuada debe tener en cuenta la necesidad de acceder a relaciones de gran tamaño que residen en el disco.

Atributos

Las columnas de una relación se nombran por atributos. Por lo general, un atributo describe el significado de las entradas en la columna a continuación. En general, seguiremos la convención de que los nombres de las relaciones comienzan con una letra mayúscula y los nombres de los atributos comienzan con una letra minúscula. Sin embargo, hablaremos de las relaciones en abstracto, donde los nombres de los atributos no importan. En ese caso, utilizaremos letras mayúsculas simples para las relaciones y los atributos, por ejemplo, R (A, B, C) para una relación genérica con tres atributos.

Esquemas

El nombre de una relación y el conjunto de atributos para una relación se llama esquema para esa relación. Mostramos el esquema de la relación con el nombre de la relación seguido de una lista entre paréntesis de sus atributos. Los atributos en un esquema de

relación son un conjunto, no una lista. Sin embargo, para hablar de relaciones, a menudo debemos especificar un orden "estándar" para los atributos. Por lo tanto, cada vez que introducimos un esquema de relación con una lista de atributos, como se indicó anteriormente, tomaremos este orden como el orden estándar cuando visualicemos la relación o cualquiera de sus filas. En el modelo relacional, una base de datos consta de una o más relaciones. El conjunto de esquemas para las relaciones de una base de datos se denomina esquema de base de datos relacional, o simplemente un esquema de base de datos.

Tuplas

Las filas de una relación, distintas de la fila de encabezado que contiene los nombres de los atributos, se denominan tuplas. Una tupla tiene un componente para cada atributo de la relación. Cuando deseamos escribir una tupla de forma aislada, no como parte de una relación, normalmente usamos comas para separar componentes, y usamos paréntesis para rodear la tupla. Tenga en cuenta que cuando una tupla aparece aisladamente, los atributos no aparecen, por lo que debe darse alguna indicación de la relación a la que pertenece la tupla. Siempre usaremos el orden en que se enumeraron los atributos en el esquema de relación.

Dominios

El modelo relacional requiere que cada componente de cada tupla sea atómico; es decir, debe ser de algún tipo elemental como entero o cadena. No está permitido que un valor sea una estructura de registro, conjunto, lista, matriz o cualquier otro tipo que razonablemente pueda dividir sus valores en componentes más pequeños. Se supone además que asociado con cada atributo de una relación hay un dominio, es decir, un tipo elemental particular. Los componentes de cualquier tupla de la relación deben tener, en cada componente, un valor que pertenezca a tc: el dominio de la columna correspondiente. Es posible incluir el dominio, o tipo de datos, para cada atributo en un esquema de relación. Lo haremos agregando dos puntos y un tipo después de los atributos.

Representaciones equivalentes de una relación

Las relaciones son conjuntos de tuplas, no listas de tuplas. Por lo tanto, el orden en que se presentan las tuplas de una relación es irrelevante.

Además, podemos reordenar los atributos de la relación como elijamos, sin cambiar la relación. Sin embargo, cuando reordenamos el esquema de relación, debemos tener cuidado de recordar que los atributos son encabezados de columna. Por lo tanto, cuando cambiamos el orden de los atributos, también cambiamos el orden de sus columnas. Cuando las columnas se mueven, los componentes de las tuplas también cambian su orden. El resultado es que cada tupla tiene sus componentes permutados de la misma manera que se permutan los atributos.

Instancias de relación

Una relación sobre películas no es estática; más bien, las relaciones cambian con el tiempo. Esperamos insertar tuplas para nuevas películas, tal como aparecen. También esperamos cambios en las tuplas existentes si obtenemos información revisada o corregida sobre una película, y tal vez la eliminación de tuplas para películas que se expulsan de la base de datos por alguna razón.

Es menos común para el esquema de una relación con el cambio. Sin embargo, hay situaciones en las que podríamos querer agregar o eliminar atributos. Los cambios de esquema, aunque son posibles en los sistemas de bases de datos comerciales, pueden ser muy costosos, porque cada uno de quizás millones de tuplas necesita reescribirse para agregar o eliminar componentes. Además, si agregamos un atributo, puede ser difícil o incluso imposible generar valores apropiados para el nuevo componente en las tuplas existentes.

Llamaremos a un conjunto de tuplas para una relación dada una instancia de esa relación. Sin embargo, un sistema de base de datos convencional mantiene solo una versión de cualquier relación: el conjunto de tuplas que están en la relación "ahora". Esta instancia de la relación se llama la instancia actual.

Claves de relaciones

Existen muchas restricciones en las relaciones que el modelo relacional nos permite colocar en los esquemas de la base de datos. Sin embargo, un tipo de restricción es tan fundamental que lo introduciremos aquí: restricciones clave. Un conjunto de atributos

forma una clave para una relación si no permitimos que dos tuplas en una instancia de relación tengan los mismos valores en todos los atributos de la clave. Recuerde que la declaración de que un conjunto de atributos forma una clave para una relación es una declaración sobre todas las instancias posibles de la relación, no una declaración sobre una sola instancia.

En las corporaciones estadounidenses, es normal que cada empleado tenga un número de Seguro Social. Si la base de datos tiene un atributo que es el número de Seguro Social, entonces este atributo también puede servir como clave para los empleados. Tenga en cuenta que no hay nada de malo en que haya varias opciones de clave, como lo sería para los empleados que tienen tanto la identificación de empleado como los números de Seguro Social. La idea de crear un atributo cuyo propósito es servir como clave está bastante extendida. Además de las identificaciones de los empleados, encontramos identificaciones de estudiantes para distinguir a los estudiantes en una universidad. Encontramos números de licencia de conducir y números de registro de automóviles para distinguir conductores y automóviles, respectivamente. Sin duda, puede encontrar más ejemplos de atributos creados con el objetivo principal de servir como claves.

K. **Dependencias Funcionales** (Ullman & Widom, 2008, págs. 67-73)**.**

Existe una teoría de diseño para las relaciones que nos permite examinar un diseño cuidadosamente y realizar mejoras basadas en unos pocos principios simples. La teoría comienza por hacernos establecer las restricciones que se aplican a la relación. La restricción más común es la "dependencia funcional", una declaración de un tipo que generaliza la idea de una clave para una relación. Existen herramientas simples para mejorar nuestros diseños mediante el proceso de "descomposición" de las relaciones: la sustitución de una relación por varias, cuyos conjuntos de atributos incluyen todos los atributos del original.

Definición de dependencia funcional

Una dependencia funcional de una relación R es una declaración de la forma "Si dos tuplas de R coinciden en todos los atributos A1,A2,...,An (es decir, las tuplas tienen los mismos valores en sus respectivos componentes para cada uno de estos atributos), también deben estar de acuerdo en toda otra lista de atributos B1,B2,...,Bm. Escribimos esta dependencia funcional formalmente como:

A1A2...An → B1B2...Bm

Y decimos que "A1,A2,...,An determinan funcionalmente B1,B2,...,Bm ".

Claves de relaciones

Decimos que un conjunto de uno o más atributos {A1,A2,...,An} es una clave para una relación R si:

- Esos atributos determinan funcionalmente todos los demás atributos de la relación. Es decir, es imposible que dos tuplas distintas de R se pongan de acuerdo sobre todos los A1,A2,...,An.
- Ningún subconjunto apropiado de {A1,A2,...,An} determina funcionalmente todos los demás atributos de R; es decir, una clave debe ser mínima. Cuando una clave consta de un solo atributo A, a menudo decimos que A (en lugar de {A}) es una clave.

A veces una relación tiene más de una clave. Si es así, es común designar una de las claves como clave principal. En los sistemas de bases de datos comerciales, la elección de la clave primaria puede influir en algunos problemas de implementación, como la forma en que se almacena la relación en el disco. Sin embargo, la teoría de las dependencias funcionales no otorga un papel especial a las "claves primarias".

Superclaves

Un conjunto de atributos que contiene una clave se denomina superclave, abreviatura de "superconjunto de una clave". Por lo tanto, cada clave es una superclave. Sin embargo, algunas superclaves no son claves (mínimas). Tenga en cuenta que cada súper clave cumple la primera condición de una clave: determina funcionalmente todos los demás atributos de la relación. Sin embargo, una súper clave no necesita satisfacer la segunda condición: mínima.

L. Reglas sobre dependencias funcionales.

Aprenderemos a razonar sobre los Dependencias Funcionales. Es decir, supongamos que un conjunto de dependencias funcionales satisface una relación. A menudo, podemos

deducir que la relación debe satisfacer ciertas otras dependencias funcionales. Esta capacidad de descubrir dependencias funcionales adicionales es esencial cuando discutimos el diseño de un correcto esquema de relación.

Razonamiento sobre dependencias funcionales

Comencemos con un ejemplo motivador que nos mostrará cómo podemos inferir una dependencia funcional de otras dependencias funcionales dadas.

Ejemplo: si se nos dice que una relación R (A, B, C) satisface las dependencias funcionales A → B y B → C, entonces podemos deducir que R también satisface la dependencia funcional A → C. Entonces, para probar que A → C, debemos considerar dos tuplas de R que estén de acuerdo con A y demostrar que también están de acuerdo con C.

Tenemos que las tuplas que se cumplen en el atributo A son (a, b1, c1) y (a, b2, c2). Dado que R satisface A → B, y estas tuplas se cumplen sobre A, entonces también deben cumplirse sobre B. Es decir, b1 = b2, y las tuplas son realmente (a, b, c1) y (a, b, c2), donde b es a la vez b1 y b2. De manera similar, dado que R satisface a B → C, y las tuplas se cumplen sobre B, entonces también deben cumplirse sobre C. Por lo tanto, c1 = c2; es decir, las tuplas se cumplen sobre C. Estamos demostrando que dos tuplas de R que se cumplen sobre A también se cumplen sobre C, y por lo tanto tenemos la dependencia funcional A → C.

Las dependencias funcionales a menudo se pueden presentar de varias maneras diferentes, sin cambiar el conjunto de instancias legales de la relación. Decimos:

- Dos conjuntos de dependencias funcionales S y T son equivalentes si el conjunto de instancias de relación que satisfacen S es exactamente el mismo que el conjunto de instancias de relación que satisfacen T.

- En términos más generales, un conjunto de dependencias S funcionales se deriva de un conjunto de dependencias funcionales T si cada instancia de relación que satisface todas las dependencias funcionales en T también satisface todas las dependencias funcionales en S.

Observe entonces que dos conjuntos de dependencias funcionales S y T son equivalentes si y solo si S se sigue de T, y T se sigue de S. En esta sección veremos varias reglas útiles sobre las dependencias funcionales. En general, estas reglas nos permiten

reemplazar un conjunto de dependencias funcionales por un conjunto equivalente, o agregar a un conjunto de otras dependencias funcionales que se siguen del conjunto original. Un ejemplo es la regla transitiva que nos permite seguir cadenas de dependencias funcionales.

M. Diseño de Base de Datos (Date, 2013, págs. 133-136).

La introducción de Codd del modelo relacional allanó el camino para la investigación de numerosos aspectos de la gestión de bases de datos. Uno de esos aspectos fue, y sigue siendo, el diseño de la base de datos; de hecho, la teoría del diseño es ahora un campo extremadamente rico, con una extensa literatura propia. Esa teoría se refiere a preguntas como las siguientes: ¿Qué es exactamente lo que constituye, o caracteriza, un diseño de base de datos "bueno"? ¿Y cómo se puede lograr un diseño tan "bueno"? Por supuesto, aquí solo podemos arañar la superficie de tales asuntos; pero, al igual que con las transacciones (el tema del capítulo anterior), un cierto grado de familiaridad con el diseño de la base de datos, o más bien con la teoría del diseño de la base de datos, es esencial para una comprensión básica de lo que se trata la tecnología de bases de datos en general.

Entonces, la teoría del diseño es ciertamente parte de la teoría relacional en general; sin embargo, no es parte del modelo relacional como tal. Más bien, es una teoría separada que se construye sobre ese modelo. De hecho, al modelo relacional como tal no le importa qué tan bien, o qué tan mal, esté diseñada la base de datos: siempre que la base de datos sea al menos relacional, lo que significa que cumple con el Principio de información, entonces todos los conceptos del el modelo relacional todavía se aplica, y en particular todos los operadores del álgebra relacional todavía funcionan. Por lo tanto, si la base de datos está mal diseñada, es el usuario el que sufre, no el modelo relacional como tal.

De hecho, gran parte de la teoría del diseño es simplemente sentido común glorificado, en cierto modo. Bueno, eso no es realmente justo; sería mejor decir que es sentido común formalizado. La formalización es importante, porque si algo puede formalizarse, puede mecanizarse; en otras palabras, podemos hacer que la máquina haga el trabajo. Pero una mayor discusión sobre ese punto en particular aquí nos llevaría demasiado lejos.

La selección descuidada de un esquema de base de datos relacional puede generar redundancia y anomalías relacionadas. Si la repetición de esta información es redundante, entonces también introduce el potencial para varios tipos de errores.

El problema del diseño de esquemas de buena relación en las siguientes etapas:

- Primero exploramos con más detalle los problemas que surgen cuando nuestro esquema está mal diseñado.
- Luego, presentamos la idea de "descomposición", rompiendo un esquema de relación (conjunto de atributos) en dos esquemas más pequeños.
- A continuación, presentamos la "Forma Normal de Boyce-Codd" o "FNBC", una condición en un esquema de relación que elimina estos problemas.
- Estos puntos están unidos cuando explicamos cómo asegurar la condición FNBC descomponiendo esquemas de relación.

Anomalías

Los problemas de redundancia que ocurren cuando intentamos atiborrar demasiado en una sola relación se llaman anomalías. Los principales tipos de anomalías que encontramos son:

- **Redundancia.** La información puede repetirse innecesariamente en varias tuplas.
- **Anomalías de Inserción.** Ocurre cuando se cambia la información en una tupla pero se deja la misma información sin cambios en otra.
- **Anomalías de Eliminación.** Si un conjunto de valores se surpime, podemos perder otra información como efecto secundario.

N. Normalización (Neeraj Sharma, 2010, págs. 96-108).

La normalización es un procedimiento en el diseño de bases de datos relacionales que tiene como objetivo convertir los esquemas relacionales en una forma más deseable. El objetivo es eliminar la redundancia en las relaciones y los problemas que se derivan de ella, como las anomalías de inserción, eliminación y actualización.

Las formas normales progresan hacia la obtención de un diseño óptimo. La normalización es un proceso paso a paso, donde cada paso transforma los esquemas relacionales en una forma normal superior.

Cada forma normal contiene todas las formas normales anteriores y alguna optimización adicional sobre ellos.

Primera Forma Normal (1FN)

Se considera que una relación está en *Primera Forma Normal* si todos sus atributos tienen dominios que son indivisibles o atómicos. La idea de los valores atómicos para el atributo asegura que no haya "grupos repetidos". Esto se debe a que un sistema de gestión de bases de datos relacionales es capaz de almacenar un único valor solo en la intersección de una fila y una columna.

Los grupos repetidos son cuando intentamos almacenar múltiples valores en la intersección de una fila y una columna y una tabla que contendrá dicho valor no es estrictamente relacional. Según la definición extendida de C. J. Date, "una tabla está en 1FN si y solo si cumple las siguientes cinco condiciones:

- No hay ordenamiento de arriba a abajo para las filas.
- No hay ordenamiento de izquierda a derecha para las columnas.
- No hay filas duplicadas.
- Cada intersección de fila y columna contiene exactamente un valor del dominio aplicable (y nada más).
- Todas las columnas son regulares [es decir las filas no tienen componentes ocultos como ID de fila, ID de objeto o marcas de tiempo ocultas]. "

La intersección de cada fila y columna en cada tabla ahora tiene un valor atómico que no puede desglosarse más y, por lo tanto, la descomposición ha producido una relación en 1FN.

Segunda Forma Normal (2FN)

Una relación está en *Segunda Forma Normal* cuando está en *Primera Forma Normal* (1NF) y no existe algún atributo no clave que dependa de una parte de la clave candidata, sino que más bien deberá depender de la clave candidata completa.

De la definición anterior se deduce que una relación que tiene un solo atributo como su clave candidata siempre está en 2FN.

Cada atributo que no es clave ahora depende de la clave candidata completa y no de una parte de ella. Por lo tanto, la descomposición ahora ha producido una relación en 2FN.

Tercera Forma Normal (3FN)

Una relación está en *Tercera Forma Normal* si está en *Segunda Forma Normal* y no existe ningún atributo no clave que dependa transitivamente de la clave candidata.

Es decir, cada atributo depende directamente de la clave primaria y no a través de una relación transitiva donde un atributo Z puede depender de un atributo no clave Y e Y a su vez depende de la clave primaria X.

La transitividad, como se sabe, significa que cuando X → Y e Y → Z, entonces X → Z.

De la relación 3FN se deduce que los atributos no clave son mutuamente independientes. Cada atributo no clave es mutuamente independiente y depende solo de la clave candidata completa. Por lo tanto, la descomposición anterior ahora ha producido una relación en 3FN.

Forma Normal de Boyce-Codd (FNBC)

La *Forma Normal de Boyce-Codd* es una versión más estricta de la *Tercera Forma Normal* que se aplica a las relaciones donde puede haber claves candidatas superpuestas.

Se dice que una relación está en la Forma Normal de Boyce-Codd si está en la *Tercera Forma Normal* y cada FD no trivial dado para esta relación tiene una clave candidata como determinante. Es decir, por cada X → Y, X es una clave candidata.

Cuarta Forma Normal (4FN)

La *Cuarta Forma Normal* (4FN) se puede entender en términos de dependencias de valores múltiples. Se dice que la *Cuarta Forma Normal* (4FN) para un esquema relacional

existe cuando las dependencias de valores múltiples no relacionadas que existen no son más de una en la relación dada.

Para entender esto claramente, primero debemos definir una dependencia de valores múltiples.

Dependencias de valores múltiples

Decimos que un atributo A determina de manera múltiple otro atributo B cuando para un valor dado de A, existen múltiples valores de B que existen.

Dependencia de valores múltiples (MVD) se denota como, A →→ B. Esto significa que A determina múltiples B, B es multi-dependiente de A o A flecha doble B.

Una relación está en 4FN cuando no hay dos o más MVD en una relación dada de modo que los atributos de valores múltiples sean mutuamente independientes. Más formalmente, una relación R (A, B, C) está en 4FN si hay MVD en la relación dada de modo que A →→ B y A →→ C, entonces B y C no son mutuamente independientes, es decir, están relacionados el uno al otro.

Las relaciones entonces se descomponen en 4FN cuando ya que no hay más de una MVD mutuamente independiente en la relación descompuesta.

3. Definición de Términos

- **Modelo de Datos Relacional:** Modelo de datos en el que las relaciones son tablas que representan información. Las columnas están encabezadas por atributos; cada atributo tiene un dominio asociado o tipo de datos. Las filas se llaman tuplas, y una tupla tiene un componente para cada atributo de la relación. Este modelo obedece al concepto matemático de *Relación*.

- **Modelo de Datos de Relación Funcional:** Este modelo, basado en el Modelo de Datos Relacional, obedece al concepto matemático de *Relación Funcional*.

- **Dependencias de Relación Funcional:** Dependencias entre conjuntos de atributos presentes en una estructura lógica de datos, las cuales obedecen al concepto matemático de *Relación Funcional*.

- **Restricciones Lógicas Funcionales:** Conjunto de restricciones lineales que, por medio operadores lógicos, expresan relacionales funcionales entre conjuntos de datos.

- **Estructuras Lógicas de Datos:** Estructura de datos que contiene elementos comerciales que proporcionan una base para el diseño de una base de datos. Estas estructuras establecen de forma lógica el funcionamiento de los procesos de negocio, así como el análisis del impacto que ocurren sobre éstos.

- **Ontología del Modelo de Negocio:** Es una especificación explícita de una conceptualización. Se puede entender como una descripción formal de los conceptos y relaciones en un dominio específico.

- **Restricciones de Integridad de Datos:** Son las reglas que se pueden aplicar a las columnas de una tabla, de forma que se pueda establecer diferentes tipos de integridad de datos.

- **Integridad de Datos:** La integridad de datos es el mantenimiento y la garantía de la precisión y la consistencia de los datos durante todo su ciclo de vida, y es un aspecto crítico para el diseño, la implementación y el uso de cualquier sistema que almacene, procese y recupere datos. La integridad de los datos es la contraparte a la corrupción de datos. La intención general de cualquier técnica de integridad de datos es la misma: garantizar que los datos se registren exactamente como se pretendía al momento en que fueron modelados; y, en una posterior recuperación de datos, garantizar que los datos sean los mismos que cuando se registraron originalmente. En resumen, la integridad de los datos tiene como objetivo evitar cambios no contemplados en la información.

- **Modelo de Negocio:** Representa los aspectos centrales de un negocio, incluido el propósito, el proceso comercial, los clientes, el valor agregado de la oferta, las estrategias, la infraestructura, las estructuras organizativas, el abastecimiento, las prácticas comerciales y los procesos operativos y políticas que están incluidos en la cultura organizacional.

- **Lógica de Negocio:** Conforma las reglas de negocio que circunscriben los flujos de los procesos de negocio. La lógica de negocio permite que los objetivos y las estrategias de la organización, que han sido definidas en los procesos de negocio, sean coherentes.

- **Reglas de Negocio:** Una regla de negocio define o restringe algún aspecto del negocio, resolviéndose siempre como verdadero o falso. Las reglas de negocios están

destinadas a establecer la estructura del negocio o controlar el comportamiento del negocio, ya que describen las operaciones y restricciones que se aplican en el contexto de una organización. Las reglas de negocio se pueden aplicar a las personas, a los procesos, al comportamiento corporativo en sí mismo, a los sistemas informáticos de una organización, etc., y se implementan para ayudar a la organización a alcanzar sus objetivos. Las reglas de negocio son directivas, diferenciándose de la lógica de negocio, que es solamente estratégica.

- **Esquemas de Relación:** Es el nombre de relación, junto con los atributos de esa relación y sus tipos. Una colección de esquemas de relación forma un esquema de base de datos. Los datos particulares para una relación o colección de relaciones (tuplas) se denominan una instancia de ese esquema de relación o esquema de base de datos.

III. Hipótesis

1. Declaración de la Hipótesis

La aplicación del Modelo de Datos de Relación Funcional propuesto sobre la Ontología de un Modelo de Negocio es eficaz para lograr la automatización de un Diseño de Base de Datos que conserve su Integridad de Datos.

2. Variables

A. Variable Independiente

Modelo de Datos de Relación Funcional sobre la Ontología de un Modelo de Negocio.

B. Variable Dependiente

Diseño de Base de Datos que conserve su Integridad de Datos.

3. Operacionalización de Variables

Tabla 1 : Matriz de Operacionalización de Variables.

Nº	Variable	Tipo de Variable	Operacionalización	Categorías o Dimensiones	Definición	Indicador	Nivel de Medición	Unidad de Medida	Índice	Valor
1	Modelo de Datos de Relación Funcional sobre la Ontología de un Modelo de Negocio.	Cuantitativa	Según el profesor Andy Pavlo, de Carnegie Mellon University, en términos generales, en un modelode datos relacional, una dependencia funcional es una forma de restricción. Fundamentalmente, se puede afirmar que para que exista una dependencia funcional el valor de una variable depende del valor de otra variable. En la presente investigación, se propone el modelo de datos funcionalmente relacional, que obedece íntegramente al concepto matemático de relación funcional.	Dependencias funcionalmente relacionales del Modelo de Datos Funcionalmente Relacional	Según la sexta edición del libro Database Systems Concepts, del autor Abraham Silberschatz, en un modelo de datos relacional, una dependencia funcional nos permite expresar restricciones que identifican de forma única los valores de ciertos atributos. Sin embargo, el modelo de datos de relación funcional, de la presente investigación, nos permite expresar las restricciones obedeciendo al concepto matemático de relación funcional	Conceptos de Relación Funcional	Discreto	Función Sobreyectiva	Restricción Lógica Sobreyectiva	Es una función sobreyectiva
										No es una función sobreyectiva
								Función Biyectiva	Restricción Lógica Biyectiva	Es una función biyectiva
										No es una función biyectiva
								Relación Multivaluada	Restricción Lógica Multivaluada	Es una relación multivaluada
										No es una relación multivaluada
2	Diseño de Base de Datos que conserve su Integridad de Datos.	Cuantitativa	Conceptualmente, Las restricciones de integridad son un conjunto de reglas de validación de datos que se puede especificar para restringir los valores de datos que se pueden almacenar. Las restricciones de integridad ayudan a preservar la validez y la consistencia de los datos. Se aplican las restricciones de integridad cuando los valores asociados con una variable	Reglas de Integridad en el Diseño de Base de Datos	En el libro Managing Database Objects in SQL, 2008, se indica que las restricciones de integridad se utilizan para garantizar la precisión y la coherencia de los datos en una base de datos relacional. La integridad de los datos se maneja en una base de datos relacional a través del concepto de integridad referencial. Muchos tipos de restricciones de integridad juegan un papel	Inserción de Datos en un diseño de base de datos relacional	Discreto	Restricción de la Integridad de la Entidad	Valores únicos de la entidad	Se cumple la Restricción de la Integridad de la Entidad
										No se cumple la Restricción de la Integridad de la Entidad
								Restricción de la Integridad de la Referencia	Valores referenciales entre conjunto de atributos	Se cumple la Restricción de la Integridad Referencial

			se agregan, actualizan o eliminan.		en la integridad referencial.					No se cumple la Restricción de la Integridad Referencial
								Restricción de la Integridad del Dominio	Valores válidos para los atributos	Se cumple la Restricción de la Integridad del Dominio
										No se cumple la Restricción de la Integridad del Dominio
				Normalización dentro de la base de datos relacionales	Según la sexta edición del libro Database Systems Concepts, del autor Abraham Silberschatz, una base de datos modela un conjunto de entidades y relaciones en el mundo real. Generalmente hay una variedad de restricciones (reglas) sobre los datos que se encuentran en el mundo real. Una instancia de una relación que satisface todas estas restricciones del mundo real se llama una instancia legal de la relación; una instancia legal de una base de datos es aquella en la que todas las instancias relacionadas son instancias legales.	Formas Normales en la Normalización de Base de Datos Relacionales	Discreto	Primera Forma Normal	Resultado de cumplimiento	Se cumple la Primera Forma Normal
										No se cumple la Primera Forma Normal
								Segunda Forma Normal	Resultado de cumplimiento	Se cumple la Segunda Forma Normal
										No se cumple la Segunda Forma Normal
								Tercera Forma Normal	Resultado de cumplimiento	Se cumple la Tercera Forma Normal
										No se cumple la Tercera Forma Normal

								Forma Normal de Boyce-Codd	Resultado de cumplimiento	Se cumple la Forma Normal de Boyce-Codd
										No se cumple la Forma Normal de Boyce-Codd
								Cuarta Forma Normal	Resultado de cumplimiento	Se cumple la Cuarta Forma Normal
										No se cumple la Cuarta Forma Normal
								Quinta Forma Normal	Resultado de cumplimiento	Se cumple la Quinta Forma Normal
										No se cumple la Quinta Forma Normal

Fuente: Elaboración propia.

IV. Descripción de Métodos y Análisis

1. Tipo de Investigación

Según la profundización en el objeto de estudio (mejor detallado en el **Nivel de Investigación**), se está considerando el tipo de investigación **correlacional-aplicado**, ya que lo que se busca en términos generales es analizar e investigar aspectos concretos del **Modelo de Datos de Relación Funcional**, el cual aún no ha sido analizado en profundidad, debido a que es una propuesta original en este proyecto de investigación. En concreto, se enfoca en un análisis correlacional entre el Modelo de Datos de Relación Funcional y el Modelo de Datos Relacional, el cual será útil para que investigaciones posteriores puedan abordar, mediante un análisis en profundidad, la temática en cuestión, a fin de que los resultados puedan ser aplicados de forma más diversa.

Este tipo de investigación no parte de teorías demasiado detalladas, sino que trata de tomar una vía alterna a fin de encontrar patrones significativos en los datos que deben ser analizados para que, a partir de estos resultados, se pueda crear las primeras definiciones completas que se desglosen de las ocurrencias en el objeto de estudio.

2. Nivel de Investigación

Alinénadonos al grado de profundidad en el objeto de estudio del presente proyecto, el nivel de investigación es **correlacional**, debido a que, en contraparte al **Modelo de Datos de Relacional**, se trata de exponer el **Modelo de Datos de Relación Funcional**, el cual no ha sido abordado antes, ya que apela al concepto matemático de **Relación Funcional** para definir la organización de elementos de datos, así como la interrelación entre éstos. Y precisamente dicho modelo es la propuesta de investigación original que sedimenta el **Diseño de Investigación** del proyecto.

3. Diseño de Investigación

Apelando al tipo y al nivel de investigación, podemos indicar que el diseño de investigación es **Experimental**, y de tipo **cuasiexperimental**, ya que se ha considerado la manipulación de la variable independiente "**Modelo de Datos de Relación Funcional sobre la Ontología de un Modelo de Negocio**" para observar el efecto o el resultado que provoca sobre la variable

dependiente "**Diseño de Base de Datos que conserve su Integridad de Datos**", y así poder finalmente verificar si se responde positivamente a la hipótesis del proyecto de investigación.

4. Método de Investigación

El método de la presente investigación obedece a un enfoque de **modelación**. Por consiguiente, para lograr efectivamente el estudio, se realizó el siguiente procedimiento:

- Recopilación de la información del conocimiento de los procesos claves de la empresa de servicios de hospedaje para definir correctamente el modelo de negocio.
- Revisión documentaria de los procesos claves de la empresa de servicios de hospedaje para evaluar y definir las reglas de negocio que se ejecutan, en base a las restricciones de la lógica de negocio.
- Especificación de los conceptos del negocio circunscriptos al dominio de aplicación de la empresa de servicios de hospedaje, que viene siendo alimentado por las actividades claves de los procesos empresariales.
- Formalización de la representación del conocimiento del negocio, el cual es abstraído por la Ontología de Negocio, utilizando el lenguaje formal de las lógicas descriptivas, las cuales nos permitirán definir la semántica conceptual dentro del dominio de aplicación.
- Proposición de un modelo de datos conceptual que nos permita entender la relación semántica existente entre los conceptos del domino de aplicación.
- Recopilación de la información empresarial, correspondiente a los conceptos que están circunscritos en el modelo de datos conceptual, para ser volcada en estructuras lógicas de datos.
- Proposición del Modelo de Datos de Relación Funcional, que a su vez utiliza Dependencias de Relación Funcional para inferir Restricciones Lógicas entre atributos (conceptos). Las Restricciones Lógicas involucradas serán de tipos sobreyectiva, biyectiva y multivaluada; siendo los dos primeros la proyección del concepto de función relacional, y la última del concepto de relación multivaluada.
- Aplicación del Modelo de Datos de Relación Funcional sobre las Estructuras Lógicas de Datos, las cuales se estructuran en base a la información de los procesos claves de la empresa de servicios de hospedaje. Dichas estructuras pertenecen a información del año 2014.

5. Unidad de Análisis

Una empresa de servicios de hospedaje de la ciudad de Trujillo con un modelo de negocio e información histórica.

6. Población

Información histórica, correspondiente al año 2014, del proceso de ventas de ocupaciones de habitaciones de una empresa de servicios de hospedaje de la ciudad de Trujillo.

7. Muestra

Información histórica, correspondiente al año 2014, de un subproceso de ventas de ocupaciones de habitaciones de una empresa de servicios de hospedaje de la ciudad de Trujillo.

8. Técnicas

- **Revisión documental:** Para la propuesta del Modelo de Datos de Relación Funcional se tuvo que revisar sesudamente la literatura académica al respecto. De manera tal que esto ayudó a que se pudieran plantear conceptos preliminares de investigación, los cuales se pueden observar en los resultados del primer objetivo específico OE1.

- **Análisis de información:** Se analizó la información histórica de la empresa de servicios de hospedaje, a fin de evaluar sus procesos principales de su modelo de negocio, y así poder aplicar sobre ellos el Modelo de Datos de Relación Funcional. Esto se puede verificar a partir de los resultados del segundo objetivo específico OE2.

9. Instrumentos

Dentro de los instrumentos que se considerarán, están los siguientes:

- **Revisión bibliográfica:** Se realizó revisiones bibliográficas con respecto a conceptos como modelo de negocio, reglas de negocio, ontología de modelo de negocio, modelo de datos conceptuales, estructuras lógicas de datos y modelo de datos relacional. Y una vez revisados y planteados estos conceptos, se procedió a proponer el Modelo de Datos de Relación Funcional, así como sus fundamentos. Esto se puede verificar en los resultados del primer objetivo específico OE1.

- **Esquemas conceptuales:** Se realizaron esquemas para poder estructurar cada concepto de investigación. De manera tal que estos esquemas desembocaron en los gráficos, tablas y cuadros que se muestran en los resultados de la investigación.

- **Creación de estructuras lógicas de datos:** Para poder aplicar el Modelo de Datos de Relación Funcional sobre la información histórica de los procesos de la empresa de servicios de hospedaje, se procedió a crear estructuras lógicas de datos en base a dicha información, y que se pueden observar en los resultados del segundo objetivo específico OE2.

- **Fichas de Resumen:** Las fichas que se utilizaron se asemejaron a hojas de un bloc de notas, y que se utilizó para plantear los lemas, teoremas y pruebas y demás matemáticos y lógicos que se proponen para la definición del proceso algorítmico propuesto en los resultados del primer objetivo específico OE1.

- **Matriz de Categorías:** Estas matrices sirvieron para la creación de las tablas y cuadros que se muestran en los resultados de la investigación y los anexos.

10. Indicadores

Se considerarán los siguientes indicadores:

- **Conceptos de Relación Funcional:** Estos conceptos se aplicarán sobre las estructuras lógicas de datos que se crearán. De manera tal que se pueda determinar cuáles son las Dependencias de Relación Funcional válidas y no válidas que existen entre los conjuntos de datos de dicha estructura. Las restricciones lógicas que permiten determinar dichas dependencias obedecen a valores de función sobreyectiva, de función biyectiva y de relación multivaluada. Esto se puede observar en los resultados del segundo objetivo específico OE2.

- **Inserción de Datos en un Diseño de Base de Datos:** La inserción de los datos se hará para verificar si se cumplen con las restricciones de integridad de datos que se pretende

impactar. Dichas restricciones son de Entidad, de Referencia y de Dominio. También se realizarán eliminaciones de datos, en base a las inserciones, que corroborarán la integridad de los datos en el diseño. Esto se puede observar en los resultados del tercer objetivo específico OE3.

- **Formas Normales sobre Base de Datos de Relación Funcional:** Las formas normales que se proponen para el Modelo de Datos de Relación Funcional corresponden a los teroremas propuestos en la presentación de los conceptos de investigación. La aplicación de estos teroremas ayudarán a verificar el cumplimiento hasta la Quinta Forma de Normalización de una base de datos relacional. Esto se puede observar en los resultados del tercer objetivo específico OE3.

V. Resultados

1. [OE1] Introducir Conceptos Preliminares de Investigación

A. Modelo de Negocio

Es evidente que cuando uno se refiere a un ***modelo de negocio*** sabe indefectiblemente que en la actualidad sigue siendo esencial para toda organización exitosa, sea una empresa nueva o ya establecida. Y se hace necesario hacer un esfuerzo de simplificación que permita definir el término en un plano pragmático y organizacional.

La palabra ***modelo*** muchas veces nos hace imaginarnos, como en una clase universitaria de Ciencias, muchas pizarras cubiertas con fórmulas matemáticas misteriosas. Sin embargo, los modelos de negocio son todo menos misteriosos, ya que, en el fondo, son argumentos discursivos que ayudan a explicar de forma inteligible cómo funcionan las empresas en su ciclo de vida. Un buen modelo de negocio, antes que nada, siempre deberá responder a dos preguntas cruciales: (1) ***quién es el cliente*** y (2) ***qué valora el cliente***. Pero también deberá responder a las preguntas fundamentales que cada líder empresarial debe hacer: (1) ***cómo podemos generar rentabilidad al poner en funcionamiento un negocio*** y (2) ***cuál es la lógica económica inherente que nos va a permitir, técnicamente, explicar la manera en la que podremos ofrecer valor a los clientes manteniendo un costo adecuado y equilibrado en el mercado***.

Por lo tanto, un ***modelo de negocio exitoso*** representa un mejor ***modelo*** que los otros existentes como alternativas, ya que podrá ofrecer más ***valor*** a un grupo definido de clientes en el mercado. En otras palabras, un modelo de negocio exitoso puede reemplazar por completo la antigua forma de hacer las cosas, convirtiéndose en la regla o en el paradigma para la próxima generación de líderes empresariales de un mismo sector.

Crear un modelo de negocio es, en términos análogos, muy parecido a escribir un nuevo argumento para una historia, ya que en cierta medida todas las historias nuevas son innovaciones de las viejas, por lo que son reconstrucciones o adecuaciones de los temas universales que son inherentes a toda la experiencia humana. Del mismo modo, un nuevo modelo de negocio es una variación de la ***cadena de valor universal*** que es base de todas las empresas que pertenecen al mismo sector del negocio.

Para hacer tangible un modelo de negocio, la herramienta por excelencia que ayuda a definir ontológicamente la gestión estratégica de un modelo de negocio nuevo o existente es ***Business Model Canvas***, ya que permite que las empresas puedan alinear sus

esfuerzos a través de la abstracción conceptual de las descripciones formales concernientes al negocio. En la tesis (Osterwalder, 2004) que realizó la persona que creó el concepto de dicha herramienta, el profesor Alexander Osterwalder, nos define el término ***modelo de negocio*** como una representación de cómo una empresa, que compra y vende bienes y servicios, puede ganar dinero. Esta representación sería una abstracción de la ***lógica de negocio*** de una empresa, por lo que se da bajo una comprensión abstracta de la forma en la que una empresa puede ganar dinero a través de ***lo que ofrece***, ***a quién lo ofrece*** y ***cómo puede lograrlo***.

B. Reglas de Negocio

Como se puede notar, el ***modelo de negocio*** debe denotar explícitamente ***qué ofrece al cliente como valor agregado único***; sin embargo, cuando nos referimos a las ***reglas de negocio*** de una empresa, tratamos de respondernos una interrogante más, que es acerca de ***qué le está permitido hacer a la empresa para darle al cliente lo que le ofrece***, en el sentido de cercar unos límites concretos circunscritos a lo que le es permitido a ella, y de esta manera enmarcarnos en el ***conjunto de restricciones*** que se condicen con la ***lógica de negocio*** dentro del mismo ***modelo de negocio***.

Los profesores Hans-Erik Eriksson y Magnus Penker (Eriksson & Penker, 2000) definen una ***regla de negocio*** como una enunciación que puede controlar o afectar tanto la ejecución de un ***proceso de negocio*** como la ***estructura de los recursos*** en el negocio. Es decir, la enunciación denota concretamente una condición que debe mantenerse *sine qua non*, o, en todo caso, indica una condición que controla qué actividad específica debe seguir a continuación dentro de un mismo proceso, especificando la forma en que debe ejecutarse tal proceso, y de esta manera poder expresar un objetivo comercial propio del modelo de negocio de la empresa, detallando así las condiciones de una relación entre conceptos o restringiendo el comportamiento de un recurso dentro un proceso. De manera que las ***reglas*** controlan la ***dirección del negocio***, ya que se definen para satisfacer criterios y requisitos externos (regulaciones, restricciones impuestas debido a la competencia con otras empresas, etc.), y en definitiva para lograr de maneras segura y clara los objetivos de la empresa.

Tabla 2 : Esquema Conceptual entre Modelo de Negocio y Regla de Negocio.

Ámbito del Negocio	Interrogante	Explicación del Enfoque
Modelo de Negocio	¿Qué ofrece?	Qué productos o servicios con valor agregado se ofrecerán al mercado con un precio adecuado
	¿A quién ofrece?	Cuáles serán los clientes a los cuales se ofrecerá los productos o servicios
	¿Cómo puede ofrecer?	A través de qué plan estratégico de negocios podremos identificar claramente las metas y objetivos de la empresa
Reglas de Negocio	¿Qué está permitido hacer para ofrecer?	Cuáles serán las restricciones de negocio que servirán para controlar la ejecución de los procesos y la estructura de recursos de la empresa

Fuente: Elaboración propia.

C. Ontología de un Modelo de Negocio

a. Relación entre Ontología y Modelo de Negocio

Para fines de la presente investigación se necesitará realizar una adecuada trazabilidad entre la formalización de una ***ontología del modelo de negocio*** y un modelo de datos conceptual, el cual, en el transcurso, nos permitirá definir correctamente una estructura de datos que sirva como materia prima para el diseño de una base de datos empresarial. De manera que estableciendo un ***modelo ontológico*** que corresponda a un modelo de negocio es posible encontrar una forma rigurosa de verificar la validez y la vigencia del mismo ***modelo de negocio*** y de las ***reglas de negocio***, y que usualmente cambian en el transcurso del ***ciclo de vida*** del negocio. Es decir, teniendo un modelo ontológico se podrá convalidar con cuidado y precisión los componentes correspondientes al modelo de negocio, las relaciones entre los conceptos del negocio y las reglas que encaminan al mismo modelo de negocio. De manera que si los conceptos o las reglas de negocio cambian, entonces también lo hará el ***modelo ontológico***.

Según las precisiones del profesor Osterwalder (Osterwalder, 2004, pág. 14), poder establecer un ***modelo de negocio*** coherente donde todos los elementos se apoyen unos a otros no es una tarea fácil. Hoy en día, los modelos de negocio son bastante complejos y su éxito a menudo se basa en la interacción correcta de una serie de elementos aparentemente menores (***ontología del negocio***). De manera que tener a disposición una ontología del modelo de negocio que describa los componentes esenciales, las reglas de negocio y las relaciones entre conceptos, sin duda facilitará

a los gerentes el diseño correcto de un modelo de negocio que además sea verificable y sostenible en base a un conjunto específico de datos y limitaciones.

b. Relación entre Ontología y Lógicas Descriptivas

En el trabajo de investigación de Veronica Gacitua-Decar y Claus Pahl (Gacitua-Decar & Pahl, 2010), se indica que tener una solución basada en ontologías puede obtener los siguientes beneficios: (1) extender una mejora a través de un marco estructurado de clasificación, (2) alinear con modelos de dominio basados en ontología y marcos de integración, (3) usar ontologías como notación de modelado semántico, (4) explotar el punto de vista lógico de las ontologías para inferir conocimiento adicional de hechos afirmados, (5) usar ontologías como un marco de razonamiento formal para soportar actividades de la arquitectura del conocimiento, tales como el descubrimiento y la coincidencia de conceptos.

En la presente investigación, nos centraremos sobretodo en el tercer beneficio de la lista anterior, usando la formalización de las ontologías como notación del modelado semántico de los conceptos del negocio, ya que uno de nuestros objetivos de investigación es saber si la formalización ontológica del modelo del negocio obedece a las restricciones de integridad de datos de las reglas de negocio dentro del contexto del mismo modelo de datos conceptual, el cual dará lugar posteriormente a un diseño de base de datos empresarial que además sea verificable y automatizable.

Para entender un poco mejor qué son las lógicas descriptivas, habrá que revisar que en el manual de ***lógicas descriptivas*** de Franz Baader (Baader, 2003) se indica claramente que la relación entre las ***lógicas descriptivas*** y las ***bases de datos*** es bastante fuerte, ya que a menudo existe la necesidad de construir sistemas en los que estén presentes tanto un **DL-KRS (*Description Logics-Knowledge Representation System*)** como un **DBMS (*DataBase Management System*)**. El DBMS tiene que ver sobretodo con la persistencia de los datos, así como con el manejo de grandes volúmenes de datos, mientras que un DL-KRS maneja el conocimiento intensivo, usualmente manteniendo la base de conocimiento en nuestro mapa mental, incluyendo, por ejemplo, afirmaciones sobre individuos que corresponden a los datos).

Por lo tanto, las lógicas descriptivas proporcionan un marco formal para los modelos ontológicos dentro del contexto de un dominio de aplicación, y que se ha demostrado que es bastante cercano a los lenguajes utilizados en el modelado de datos

semánticos, como el modelo de entidad-relación, ya que las lógicas descriptivas están equipadas con herramientas de razonamiento que pueden aportar a la fase de modelado conceptual ventajas significativas, en comparación con los lenguajes tradicionales, cuya función se limita al modelado. Por ejemplo, al usar la consistencia del concepto como parte de la realidad del mundo, se puede verificar en el momento de realizar el diseño si una entidad puede tener al menos una instancia, evitando así claramente todas las dificultades que surgen al descubrir tal situación cuando se está poblando la base de datos.

Un dominio de aplicación es una circunscripción de la realidad, siendo el punto de partida para el análisis y la creación de un modelo de dominio. De manera que un dominio de aplicación puede ser una organización, un departamento dentro de una organización o simplemente un lugar de trabajo. Un dominio de aplicación normalmente incluye un lenguaje específico de dominio, ya que determina en gran medida cómo se orientará y ejecutará un proyecto. Esto significa que las personas en este dominio usan términos y conceptos específicos y piensan en su dominio de una manera específica.

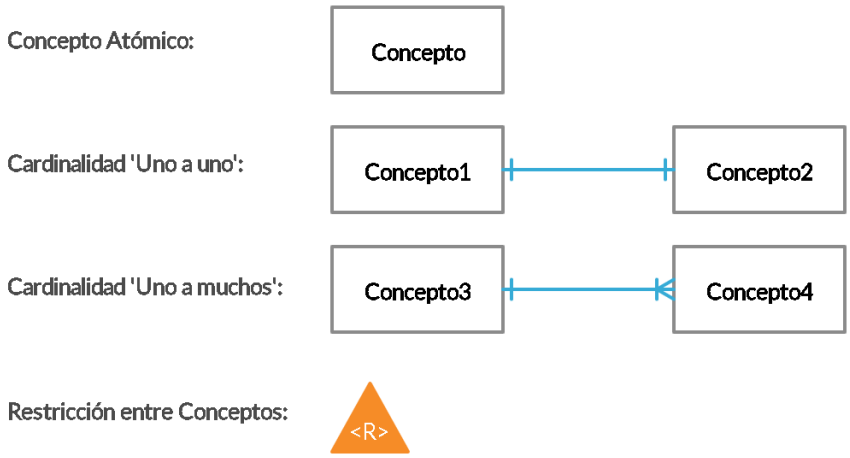
D. Modelo de Datos Conceptual

Para el profesor Toby J. Teorey (Teorey, 2011), el ***modelo de datos conceptual***, como herramienta de comunicación entre el diseñador y el usuario final, es más exitoso durante el análisis de los requisitos y las fases de diseño lógico, ya que su éxito se debe al hecho de que el modelo de datos conceptual es fácil de entender para personas que no tienen un conocimiento técnico del tema. Otra razón de su efectividad es que es un enfoque de arriba hacia abajo que utiliza el concepto de abstracción, propio de un ***modelo ontológico***. El número de entidades en una base de datos es típicamente mucho menor que el número de elementos de datos individuales porque los elementos de datos generalmente representan los atributos. Por lo tanto, usar entidades como una abstracción para elementos de datos y enfocarse en las relaciones entre entidades reduce en gran medida el número de objetos bajo consideración y simplifica el análisis. Aunque todavía es necesario representar elementos de datos por atributos de entidades en el nivel conceptual, sus dependencias normalmente se limitan a los otros atributos dentro de la entidad o, en algunos casos, a aquellos atributos asociados con otras entidades con una relación directa con su entidad.

Las principales dependencias entre atributos que se producen en los modelos de datos son las dependencias entre las claves de entidad, los identificadores únicos de diferentes

entidades que se capturan en el proceso de modelado conceptual de datos. Los casos especiales, como las dependencias entre elementos de datos de entidades no relacionadas, se pueden manejar cuando se identifican en el análisis de datos subsiguiente. El enfoque de diseño de base de datos lógico definido aquí utiliza tanto el modelo de datos conceptuales como el modelo de datos relacional en etapas sucesivas. Se beneficia de la simplicidad y facilidad de uso del modelo de datos conceptual y la estructura y el formalismo asociado del modelo de datos relacional. Para facilitar este enfoque, es necesario construir un marco para transformar la variedad de construcciones de modelos de datos conceptuales en tablas que ya están normalizadas o que pueden normalizarse con un mínimo de transformación.

Gráfico 3 : Símbolos del Modelo de Relaciones entre Conceptos Propuesto.



Fuente: Elaboración propia.

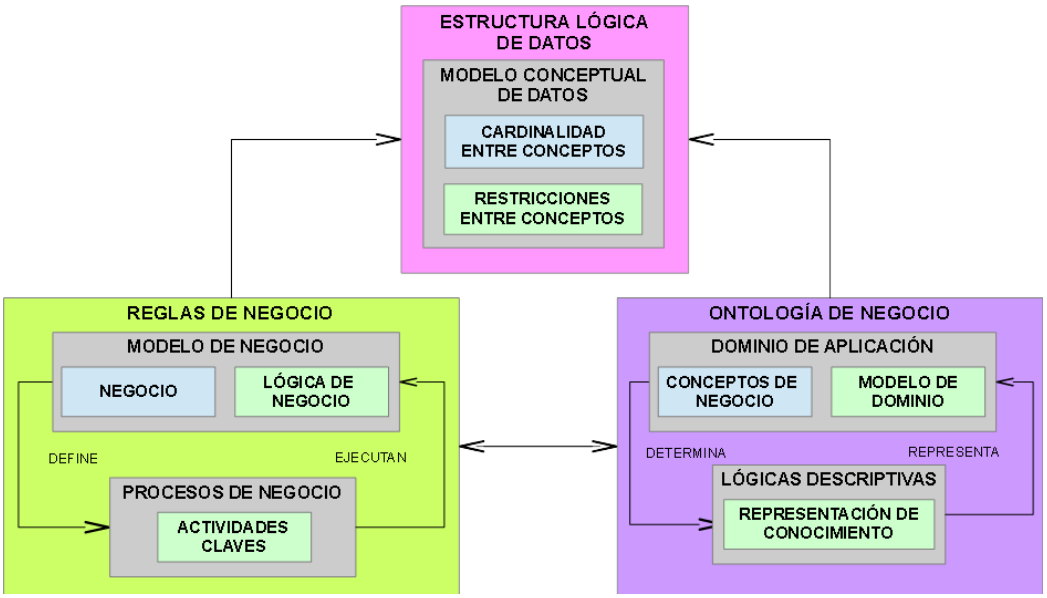
Para efectos de la presente investigación, se diseñó un modelo que nos permitirá diagramar nuestro ***modelo de datos conceptual*** correspondiente al modelo de negocio de la muestra, la que estamos manejando como nuestro dominio ontológico.

De manera que se ha procedido a crear un nuevo modelo conceptual que nos permita visualizar las relaciones entre conceptos atómicos que juegan un rol importante en los procesos de negocio del estudio. Así que, para tal fin, denominaremos a la herramienta de este modelo como el ***Diagrama de Relaciones entre Conceptos***.

E. Estructuras Lógicas de Datos

Por lo general, se utilizan varias estructuras lógicas de datos para expresar las relaciones entre elementos de datos individuales o registros en una base de datos. Las estructuras lógicas de datos comunes son las jerárquicas, las de red y las relacionales, pero para esta investigación se pondrá un énfasis especial en esta última.

Gráfico 4 : Esquema Conceptual de la Estructura Lógica de Datos.



Fuente: Elaboración propia.

La estructura relacional organiza los datos en tablas bidimensionales. Estas tablas ofrecen una gran flexibilidad y un alto grado de seguridad de datos.

La estructura relacional usa relativamente poca memoria o reducido almacenamiento secundario en dispositivos que no son accesibles constantemente por un sistema informático, pero que sirven para resguardo.

Desafortunadamente, ***el proceso de creación de tablas es bastante elaborado***. Otra desventaja de la estructura relacional es que generalmente requiere más tiempo para acceder a la información que cualquiera de las otras dos estructuras (de red y de jerarquía).

Aquella sutiación se debe a que sus procesos internos buscan mucha más información para responder a las consultas requeridas por un sistema de información. Además, algunas implementaciones usan una cantidad fija de almacenamiento para cada campo, lo que resulta en una utilización de almacenamiento ineficiente.

A pesar de aquellas desventajas, la estructura relacional sigue ganando aceptación y actualmente es la más popular de las tres estructuras.

F. Modelo de Datos Relacional

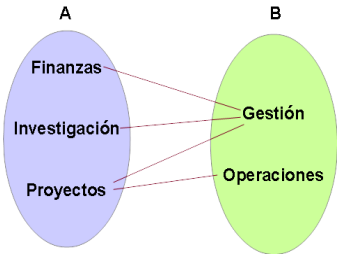
a. Relación Matemática y Modelo de Datos Relacional

En un ***modelo de datos relacional*** se establece que todos los datos deberán ser modelados como ***relaciones***, haciendo referencia estrictamente al concepto matemático de ***relación***, el cual es definido por las conexiones entre conjuntos que se contempla en la ***Teoría de Conjuntos***.

De manera que cuando se habla del ***Álgebra Relacional***, dentro del ***modelo de datos relacional***, se hace una referencia directa al ***concepto matemático del Álgebra de Conjuntos***.

Partiendo de la base anterior, podemos indicar que, dentro de un ***modelo de datos relacional***, las ***relaciones*** en sí mismas son ***mapeos de valores atómicos y unidimensionales***, tal cual como las ***conexiones entre elementos que pertenecen a conjuntos***, y que definen el ***concepto matemático de relación***.

Gráfico 5 : Relación Matemática entre dos Conjuntos.



Fuente: Elaboración propia.

Por ejemplo, supongamos que tenemos dos conjuntos. El conjunto **A = {Finanzas, Investigación, Proyectos}** y el conjunto **B = {Operaciones, Gestión}**. Definamos entonces una ***relación matemática*** entre los conjuntos **A** y **B**, de la forma **R: A ↔ B**, considerando que **R ⊂ A x B**, siendo **A x B** el producto cartesiano de **A** y **B**, tal que los pares ordenados de la forma matemática **(a,b) ∈ R** se establece porque **a ∈ A** y **b ∈ B**. Por lo tanto, podríamos arbitrariamente constituir una **relación matemática** entre los dos conjuntos, como se muestra en el gráfico ***Relación Matemática entre dos Conjuntos***.

Según la gráfica anterior, los pares ordenados de la relación matemática **aRb ≡ (a,b) ∈ R**, serán los siguientes:

aRb = {(Finanzas,Gestión),(Investigación,Gestión),(Proyectos,Gestión), (Proyectos,Operaciones)}.

Sin embargo, para definir la ***relación matemática*** anterior como una ***relación*** dentro de un ***modelo de datos relacional***, se necesitará de la siguiente estructura de datos:

Tabla 3 : Estructura de Datos de Forma Relacional.

	A	B
t1	Finanzas	Gestión
t2	Investigación	Gestión
t3	Proyectos	Gestión
t4	Proyectos	Operaciones

Fuente: Elaboración propia.

Donde **R** es un ***esquema de relación*** sostenido sobre el ***conjunto de atributos*** **U = [A,B]**, tal que es definido como **R (U)**, y donde **A** y **B** son dos atributos dentro del conjunto de atributos de **U**, tal que **A ∪ B ⊆ U**.

También se puede observar que la ***cantidad de tuplas*** **{t1, t2, t3, t4} ⊂ A x B** está definida por la ***cardinalidad*** de **R = {t1[AB],t2[AB],t3[AB],t4[AB]}**, cuyo valor, **|R|**, es igual a **4**.

De lo anterior, se puede notar que es perfectamente posible hacer una trazabilidad desde el ***concepto matemático de relación*** hacia el ***concepto de relación del modelo de datos relacional***, y viceversa.

Es decir, podemos homologar ambos conceptos de la siguiente manera:

Tabla 4 : Conciliación entre la Relación Matemática y Modelo de Datos Relacional.

Relación Matemática	Modelo de Datos Relacional
Conjuntos **A**,**B**	Atributos **A**,**B**
Los conjuntos **A** y **B** contienen elementos, que son dominio o codominio	Los atributos **A** y **B** contienen tuplas, las que son valores de un dominio específico
Relación **R: A ↔ B**	Relación **R = {t1[AB],…,tn[AB]}**
R ⊂ A x B	**R ⊂ A x B**
R se define por el mapeo de elementos entre conjuntos	**R** se define por las tuplas de una relación con atributos

Fuente: Elaboración propia.

Como se desprende del modelo de datos relacional, la relación **R = {t1[AB],t2[AB],t3[AB],t4[AB]}** connota en sí misma un conjunto implícito de relaciones matemáticas en **R: A ↔ B**.

b. Definiciones Formales dentro del Modelo de Datos Relacional

Como ya se evidenció, dentro de un **modelo de datos relacional**, una **relación** es una ***estructura de datos*** que se utiliza para ***modelar datos***. De manera que, basándonos en el trabajo de David Lewis (Lewis, 2016), si queremos hacer una definición más formal, podríamos proceder definiendo que una **Relación R** que se establece sobre un conjunto de **Atributos {A1,…,An}** y de **Dominios {D1,…,Dn}**, se constituye formalmente por dos partes diferenciadas: **La Cabecera** y **el Cuerpo**.

- **La Cabecera.**
  - La **Cabecera** está constituida por un conjunto fijo y no ordenado de **atributos**.

- Cada Atributo se describe por pares de la forma **<nombre-atributo : nombre-dominio>**, conformándose un conjunto de atributos de la forma **{<A1 : D1>, <A2 : D2>,…, <An : Dn>}**, donde **n** es la cantidad de Atributos.
- A cada atributo **Ai** le corresponde exactamente uno de los correspondientes dominios **Di**.
- Un **dominio** es un conjunto de valores nominales que son atómicos y unidimensionales y que pertenecen a un mismo tipo.
- Los **nombres de atributos** son distintos, ya que no se pueden repetir.
- Los **dominios {D1,…,Dn}** no necesariamente deberán ser distintos, ya que se pueden repetir.

➢ **El Cuerpo**

- El Cuerpo está constituido por un conjunto **tuplas** no ordenadas.
- Cada **tupla** es un conjunto único de pares del tipo **<nombre-atributo : valor-atributo>**.
- En cada **tupla i**, hay exactamente un único par **<Aj : Vij>** para cada atributo **Aj** de la cabecera, de modo que la **tupla i-ésima** de una relación con **atributos {A1,…,An}** se represente de la forma **{<A1 : Vi1>, <A2 : Vi2>, … , <An : Vin>}**.
- Para cualquier par **<Aj : Vij>** dado, **Vij** es un valor del domino **Dj** que se asocia con el atributo **Aj**.

c. Definición de Dato dentro del Modelo de Datos Relacional

En 1923 C. K. Ogden y I. A. Richards publicaron un libro (Ogden & Richards, 1946) donde describen el famoso ***Triángulo Semiótico***, afirmando que la ***representación mental*** de un ***concepto*** que se quiere transmitir a través de un ***signo***, está compuesta por ***significado***, ***significante*** y ***referente***, teniendo en cuenta que un concepto, como definición general, es una idea abstracta o genérica de instancias

particulares del mundo cognoscible, pudiendo aplicarse a la idea formada de lo que debería ser una cosa como parte del conocimiento.

El ***referente*** juega un papel crucial dentro de la ***referencia***, como parte de las funciones del lenguaje. La referencia permite la ***cohesión textual*** (referencias funcionales), y es muy importante al momento de estudiar los conceptos dentro de un contexto discursivo (que para efectos de la presente investigación se circunscribirá dentro de un ámbito empresarial) para transmitir correctamente un pensamiento.

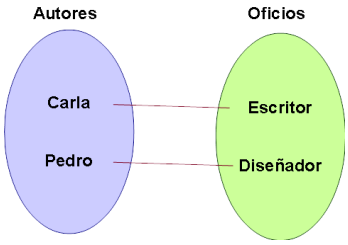
Cuando en un contexto discursivo dos conceptos, o más, apunten a un mismo referente, entonces se establece una ***correferencia*** entre conceptos, debido a que dichos conceptos correferenciales no dependen uno del otro para saber cuál es el referente al que se hace indicación. Por otro lado, cuando dos conceptos, o más, sí dependen uno del otro (referencia funcional) para que tengan sentido propio dentro del contexto, entonces hablamos de ***cadena referencial***.

- **Correferencia.**

Planteemos el siguiente contexto discursivo para explicar la correferencia:

- ***Carla*** escribió el último libro.
- La ***escritora*** es muy prolífica.
- Además, ***Pedro*** diseñó la tapa del libro.
- El ***diseñador*** es muy demandado.

Gráfico 6 : Correferencia en el Contexto Discursivo.



Fuente: Elaboración propia.

Analizando la referencia funcional de las relaciones entre conceptos, podemos graficar la correferencia en el gráfico ***Correferencia en el Contexto Discursivo***.

De la imagen anterior, podemos afirmar que los referentes ***escritora*** y ***diseñador***, pertenecientes al concepto ***Oficios***, no dependen de los referentes ***Carla*** y ***Pedro***, respectivamente, que a su vez éstos pertenecen al concepto ***Autores***.

La no dependencia ocurre debido a que ambos conceptos no necesitan estar juntos para que sus referentes tengan un significado en el contexto discursivo. Por lo tanto, la relación funcional entre los dos conceptos (Autores y Oficios) obedece a la definición de ***Función Biyectiva***.

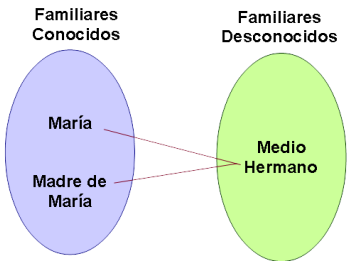
- **Cadena referencial.**

Planteemos el siguiente contexto discursivo para explicar la cadena referencial:

- ***María*** y ***su madre*** conocieron, cada una, a su ***medio hermano***.
- ***María*** lo abrazó, diciéndole que él es ***su medio hermano***.
- ***Su madre*** lo abrazó, diciéndole que él es ***su medio hermano***.

Analizando la referencia funcional de las relaciones entre conceptos, podemos graficar lo siguiente:

Gráfico 7 : Cadena Referencial en el Contexto Discursivo.



Fuente: Elaboración propia.

De la imagen anterior, podemos afirmar que el referente ***medio hermano***, perteneciente al concepto ***Familiares Desconocidos***, depende de los referentes ***María*** y ***Madre de María***, que a su vez pertenecen al concepto ***Familiares Conocidos***. La dependencia ocurre debido a que ambos conceptos necesitan estar juntos para que sus referentes tengan un significado en el contexto discursivo. Por lo tanto, la relación funcional entre los dos conceptos (Familiares Conocidos y Familiares Desconocidos) obedece a la definición de ***Función Sobreyectiva***.

Ahora bien, el modelo de datos relacional de datos es un modelo abstracto que organiza elementos de datos y determina cómo se relacionan entre sí, obedeciendo al concepto de relación, y teniendo en cuenta las propiedades de las entidades (conceptos) del mundo real.

Para el profesor Paul Beynon-Davies (Beynon-Davies, 2004), tanto los datos como la información están incorporados en el concepto de un signo y que, por lo tanto, los signos y los sistemas de signos son la materia fundamental de los sistemas de bases de datos. Es así que un signo es todo lo que tenga significado, ya que, en cierto sentido, todas las acciones humanas tienen un significado.

El mundo en el que se encuentran los seres humanos resuena con los sistemas de signos. Un sistema de signos es cualquier colección organizada de signos. El lenguaje cotidiano es probablemente el ejemplo más aceptado y complejo de un sistema de signos. Sin embargo, los signos existen en la mayoría de las otras formas de actividad humana, ya que los signos son críticos para el proceso de comunicación y comprensión humana. Y esta comprensión humana también se da en el ámbito de los datos y la información que se almacenan en las bases de datos. Así que, para efectos de un mejor entendimiento de lo afirmado en líneas anteriores, dentro del ámbito del modelo de datos relacional, se vio necesario hacer una analogía que permita efectuar una trazabilidad entre la ***Representación Mental*** del ***Signo Lingüístico*** y la ***Representación de los Datos*** en el ***Modelo de Datos Relacional***.

Tabla 5 : Conciliación Conceptual entre Signo Lingüístico y el Modelo de Datos Relacional.

Signo Lingüístico	Modelo de Datos Relacional
Signo. Es el instrumento que introduce en los sentidos humanos una representación ontológica (referencias funcionales entre referentes), y que pretende transmitir un pensamiento en	**Dato.** Es un conjunto de caracteres (alfabetos, números, símbolos, etc.) que se recopilan para ser analizados, siendo este conjunto la información

base al conocimiento (información racional).	desorganizada que representan condiciones, o ideas, u objetos.
Significado. Es el concepto o idea que es la mínima expresión asociada (1) a la forma sensible o perceptible (***significante***) del signo y (2) al objeto que representa (***referente***).	**Elemento de Datos.** Es la construcción atómica en cualquier ***modelo de datos***, ya que es la mínima expresión de datos que tiene un significado o una semántica precisos.
Significante. Es la palabra, o el tipo, o el símbolo, que se hace perceptible para el ser humano, dentro de la limitación de su conocimiento, y con que se hace referencia al ***significado***.	**Tipo de Datos.** Es el rango de operaciones válidas para un ***elemento de datos***, ya que define no solo el formato sino también el conjunto permitido de operaciones que podemos realizar.
Referente. Es el ***significado*** hecho tangible en una persona, o en una cosa, o en un objeto real; o puede ser algo más abstracto, como un conjunto de cualidades o sentimientos.	**Ítem de Datos.** Es una subunidad de información descriptiva o valor especificado dentro de un ***elemento de datos***, donde éste pertenece a un único dominio de datos.
Referencias. Son un conjunto definido de relaciones biunívocas que se establecen entre ciertas unidades o expresiones lingüísticas (signo) y una entidad del universo creado (referente) en el contexto discurso.	**Modelo de Datos.** Es un conjunto de principios generales para manejar las relaciones entre los datos, teniendo en cuenta (1) la forma en que se estructuran (datos) y (2) el dominio sobre el que se utilizan los datos.

Fuente: Elaboración propia.

G. Propuesta del Modelo de Datos de Relación Funcional

a. Modelo Propuesto

Como ya se sabe, el ***Modelo de Datos Relacional*** se fundamenta en el concepto de ***relación matemática***. Pero el ***Modelo de Datos de Relación Funcional***, propuesto en la presente investigación, se presenta como un enfoque extendido del Modelo de Datos Relacional, debido a que se sustenta en el concepto matemático de ***relación funcional matemática***.

Tal como Abraham Silberschatz lo indica en su libro (Silberschatz, Korth, & Sudarshan, 2011), SQL no proporciona una forma de especificar dependencias funcionales (restricciones lógicas que permiten deducir un diseño de base de datos), más allá de declarar superclaves utilizando las restricciones de clave primaria o única.

Allí también se indica que es posible, aunque un poco complicado, escribir aserciones que impongan una dependencia funcional, y que actualmente ningún sistema de base

de datos soporta las aserciones complejas que se requieren para aplicar una dependencia funcional, debido a que las aserciones serían difíciles de probar.

En la presente investigación, se propone un modelo que utiliza dependencias de relación funcional en lugar de dependencias funcionales, siendo que dichas dependencias propuestas podrán ser automatizables dentro de un proceso algorítmico. Estas dependencias de relación funcional se convertirán en las restricciones lógicas que permitirán definir con exactitud un diseño de base de datos (que en este caso será de relación funcional) para la descomposición de tablas. Dicha exactitud significa que el diseño podrá ser medido y monitoreado.

b. Relación Funcional

El concepto de ***relación funcional*** hace referencia a cada una de las relaciones que además es también una ***función***. Y, por definición, se sabe que una función es una relación binaria que no tiene dos elementos que tengan la misma primera coordenada; o, dicho de otra manera, en una función dos pares ordenados distintos siempre tienen primeras coordenadas diferentes (Haan & Koppelaars, 2007).

En la presente investigación consideraremos la ***función biyectiva*** (con el símbolo "↠") y la ***función sobreyectiva*** (con el símbolo "→"). Sin embargo, también consideraremos la ***relación multivaluada*** (con el símbolo "⊸"), en referencia a aquella función que no cumple con la definición tradicional de función, pero que sí es una relación. En consecuencia, utilizaremos estos tipos de funciones matemáticas, los cuales restringirán de manera lógica la forma en la que se estructuran los datos en un modelo de relación funcional, que precisamente es el modelo propuesto en la presente investigación.

Tabla 6 : Símbolos Funcionales Utilizados para las Restricciones Lógicas.

Tipo	Símbolo Matemático
Función Sobreyectiva	→
Función Biyectiva	↠
Relación Multivaluada	⊸

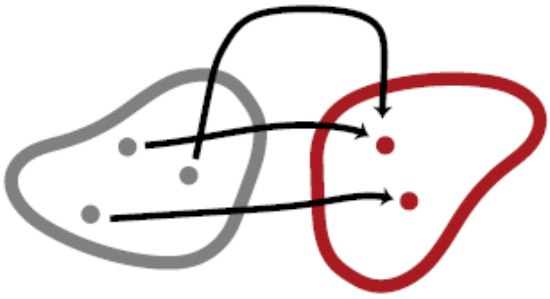
Fuente: Elaboración propia.

(1) Función Sobreyectiva

En un modelo de datos de relación funcional, una **función sobreyectiva** entre el atributo **A** y el atributo **B**, estableciéndose en la dependencia de relación funcional **A** → **B**, indica que **A** es un **atributo clave** y **B** es un **atributo no clave**, dentro de un mismo conjunto de atributos.

Por lo tanto, en una relación funcional sobreyectiva, la dependencia de relación funcional **A** → **B**, para el diseño de una base de datos de relación funcional, se establecería, por ejemplo, en la forma **R1(A, B)**, siendo **A** un **atributo de clave primaria**, y siendo **B** un **atributo simple** que es identificado de manera única por **A**. Tanto el atributo **A** como el atributo **B** están dentro de una misma tabla.

Gráfico 8 : Ejemplo de Función Sobreyectiva.



Fuente: (Faculty of Arts and the Taylor Institute of Teaching, 2017).

(2) Función Biyectiva

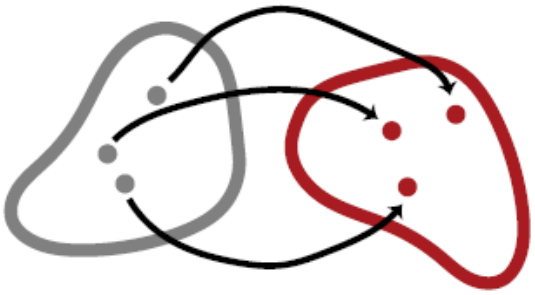
En un modelo de datos de relación funcional, una **función biyectiva** entre el atributo **A** y el atributo **B**, estableciéndose en las dependencias de relación funcional **A** ↠ **B** ∨ **B** ↠ **A**, indica que **A** es un atributo que siempre acompaña a **B**, independientemente de si los dos sean **atributos clave** o **atributos no clave** dentro de un mismo conjunto de atributos.

Por lo tanto, en una relación funcional biyectiva, la dependencia de relación funcional **A** ↠ **B**, para el diseño de una base de datos de relación funcional,

se establecería, por ejemplo, en la forma **R2(A, B, C)**, siendo **A** y **B** **atributos de clave primaria compuesta**, y siendo **C** un **atributo simple** que es identificado de manera única por **A** y **B**. Tanto el atributo **A** como el atributo **B** están dentro de una misma tabla.

Otra manera de establecerse una relación funcional biyectiva para el diseño de una base de datos de relación funcional es en la forma **R3(C, A, B)**, siendo **C** un **atributo de clave primaria**, y siendo **A** y **B** dos **atributos simples** que son identificados de manera única por **C**. De igual forma, tanto el atributo **A** como el atributo **B** están dentro de una misma tabla.

Gráfico 9 : Ejemplo de Función Biyectiva.



Fuente: (Faculty of Arts and the Taylor Institute of Teaching, 2017)**.**

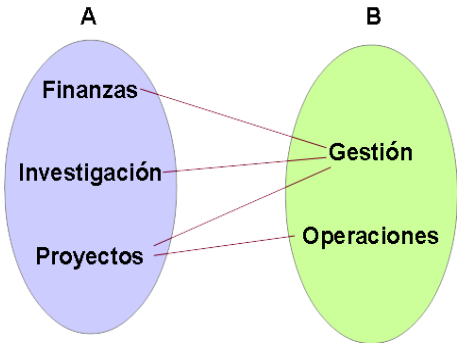
(3) Relación Multivaluada

En un modelo de datos de relación funcional una **relación multivaluada** entre el atributo **A** y el atributo **B**, estableciéndose en las dependencia relacionales **A** $\multimap$ **B** $\vee$ **B** $\multimap$ **A**, indica que no podrá existir una función sobreyectiva o biyectiva entre **A** (sea un **atributo clave** o un **atributo no clave**) y **B** (sea un **atributo clave** o un **atributo no clave**).

Por lo tanto, en una relación multivaluada, la dependencia relacional **A** $\multimap$ **B**, para el diseño de una base de datos de relación funcional, se establecería, por ejemplo, en las formas **R4(A, C)** y **R5(B, D)** siendo **A** un **atributo de**

clave primaria, y siendo **C** un **atributo simple** que es identificado de manera única por **A**; siendo también **B** un **atributo de clave primaria**, y siendo **D** un **atributo simple** que es identificado de manera única por **B**. Tanto el atributo **A** como el atributo **B** están en tabla distintas. En otras palabras, no podría ocurrir que los atributos **A** y **B** estén en una misma tabla en la forma **R6(A, B)** o **R7(B, A)**.

Gráfico 10 : Ejemplo de Relación Multivaluada.



Fuente: Elaboración propia.

H. Obtención de Relaciones Binarias de una Estructura Lógica de Datos

Teniendo en cuenta que, en un modelo de datos relacional, las columnas de cualquier estructura de datos son atributos, entonces, manteniendo esta misma definición para el modelo de datos de relación funcional, se deberá saber de qué manera los atributos conformarán relaciones binarias sobre las que se deberán aplicar las restricciones lógicas, cuyos resultados, a su vez, nos proporcionarán exactamente cuáles relaciones binarias pertenecen al concepto de dependencia de relación funcional, y cuáles no lo hacen.

a. Teorema para Obtener Relaciones Binarias entre Atributos

El conjunto de relaciones binarias de entre conjuntos de atributos, que se utilizará, estará descrito formalmente por el siguiente teorema:

Teorema 1: Dado un esquema de relación **R = {A1,…,An}**, donde **A1,…,An** es el conjunto de atributos de **R**, se tendrá entonces desde **1** hasta **n-1** niveles de relación que empezarán desde **1−aria** hasta **(n-1)−aria**, los cuales estarán dados por las combinaciones sin repetición del conjunto de atributos **A1,…,An**. Cada nivel de relación **r−aria**, donde **r** ∈ **{1,…,n-1}**, a su vez, compondrá relaciones binarias con cada elemento del conjunto de atributos **A1,…,An**. Estas relaciones binarias nos permitirán finalmente, después de la aplicación de ciertas restricciones lógicas, obtener dependencias de relación funcional válidas.

La cantidad de combinaciones sin repetición del conjunto de atributos **A1,…,An**, con cantidad **n** de atributos, para cada nivel de relación **r−aria**, estará dada por la siguiente función:

F: X $\rightarrow$ **Y**,

En la cual, cada par de elementos **n**, **r** ∈ **X** se asocia con cada elemento **F(n,r)** ∈ **Y**, tal que se define como sigue:

$$F(n,r) = \frac{n!}{r!(n-r)!}$$

Asimismo, la cantidad de relaciones binarias, que es dada entre el conjunto de combinaciones sin repetición para cada nivel de relación **r−aria** y cada elemento del conjunto de atributos **A1,…,An**, estará dada por la siguiente función:

G: X $\rightarrow$ **Y**,

En la cual, cada par de elementos **F**, **n** ∈ **X** se asocia con cada elemento **G(F,n)** ∈ **Y**, tal que se define como sigue:

$$G(F,n) = F.n$$

b. Ejemplificación del Teorema 1

Por ejemplo, si tenemos un esquema de relación **R = {a, b, c}**, cuyos atributos son **a, b** y **c**, y donde **n = |R| = 3**, entonces se establecerán relaciones para cada nivel de relación de tipo **r−aria**, empezando desde **1−aria** hasta **2−aria**, y la cantidad de combinaciones sin repetición de cada nivel de relación **r−aria** de **R** serán como sigue a continuación:

(1) En el nivel de relación 1−aria o unaria de R.

Las combinaciones sin repetición serán las siguientes:

1) **C1.1(** {a} **)**

2) **C1.2(** {b} **)**

3) **C1.3(** {c} **)**

Con lo cual, como se observó anteriormente, la cantidad de combinaciones sin repetición del nivel de relación **1−aria** de **R** es igual a **3**, tal como se demuestra a continuación:

$$F(n,r) = \frac{n!}{r!(n-r)!}$$

$$F(3,1) = \frac{3!}{1!(3-1)!}$$

$$F(3,1) = 3$$

Y las relaciones binarias, que se da entre el conjunto de combinaciones sin repetición para el nivel de relación **1−aria** y cada elemento del conjunto de atributos **a, b** y **c**, serán las siguientes:

1) **R1.1(** {a}, {a} **)**

2) **R1.2(** {a}, {b} **)**

3) **R1.3(** {a}, {c} **)**

4) **R1.4(** {b}, {a} **)**

5) **R1.5(** {b}, {b} **)**

6) **R1.6(** {b}, {c} **)**

7) **R1.7(** {c}, {a} **)**

8) **R1.8(** {c}, {b} **)**

9) **R1.9(** {c}, {c} **)**

Con lo cual, como se observó anteriormente, la cantidad de relaciones binarias para el nivel de relación **1−aria** de **R** es igual a **9**, tal como se demuestra a continuación:

$$G(F, n) = F.n$$

$$G(3, 3) = 3.3$$

$$G(3, 3) = 9$$

(2) En el nivel de relación 2−aria o binaria de R.

Las combinaciones sin repetición serán las siguientes:

1) **C2.1(** {a,b} **)**

2) **C2.2(** {a,c} **)**

3) **C2.3(** {b,c} **)**

Con lo cual, como se observó anteriormente, la cantidad de combinaciones sin repetición del nivel de relación **2−aria** de **R** es igual a **3**, tal como se demuestra a continuación:

$$F(n, r) = \frac{n!}{r!(n-r)!}$$

$$F(3,2) = \frac{3!}{2!(3-2)!}$$

$$F(3,2) = 3$$

Y las relaciones binarias, que se da entre el conjunto de combinaciones sin repetición para el nivel de relación **2−aria** y cada elemento del conjunto de atributos **a, b** y **c**, serán las siguientes:

1) **R2.1(** {a,b}, {a} **)**

2) **R2.2(** {a,b}, {b} **)**

3) **R2.3(** {a,b}, {c} **)**

4) **R2.4(** {a,c}, {a} **)**

5) **R2.5(** {a,c}, {b} **)**

6) **R2.6(** {a,c}, {c} **)**

7) **R2.7(** {b,c}, {a} **)**

8) **R2.8(** {b,c}, {b} **)**

9) **R2.9(** {b,c}, {c} **)**

Con lo cual, como se observó anteriormente, la cantidad de relaciones binarias para el nivel de relación **2−aria** de **R** es igual a **9**, tal como se demuestra a continuación:

$$G(F,n) = F.n$$

$$G(3,3) = 3.3$$

$$G(3,3) = 9$$

De forma tal que la lista completa de relaciones binarias de todos los niveles de relación, que en este caso solamente son **1−aria** de **R** y **2−aria** de **R**, será la siguiente:

(1) **R1.1(** {a}, {a} **)**

(2) **R1.2(** {a}, {b} **)**

(3) **R1.3(** {a}, {c} **)**

(4) **R1.4(** {b}, {a} **)**

(5) **R1.5(** {b}, {b} **)**

(6) **R1.6(** {b}, {c} **)**

(7) **R1.7(** {c}, {a} **)**

(8) **R1.8(** {c}, {b} **)**

(9) **R1.9(** {c}, {c} **)**

(10) **R2.1(** {a,b}, {a} **)**

(11) **R2.2(** {a,b}, {b} **)**

(12) **R2.3(** {a,b}, {c} **)**

(13) **R2.4(** {a,c}, {a} **)**

(14) **R2.5(** {a,c}, {b} **)**

(15) **R2.6(** {a,c}, {c} **)**

(16) **R2.7(** {b,c}, {a} **)**

(17) **R2.8(** {b,c}, {b} **)**

(18) **R2.9(** {b,c}, {c} **)**

En consecuencia, el resultado será que habrá 18 relaciones binarias a las cuales se les aplicarán las restricciones lógicas funcionales. Éstas permitirán determinar a qué tipo pertenece cada unas de las 18 relaciones binarias; es decir, al analizar cada

relación binaria se podrá conocer si es del tipo de función sobreyectiva o biyectiva, o si por el contrario obedece a una relación multivaluada, que se aleja del concepto tradicional de función.

I. Restricciones Lógicas del Modelo de Relación Funcional

Para obtener todas las dependencias de relación funcional válidas a partir de una estructura lógica de datos, se necesita aplicar restricciones lógicas a la cardinalidad de proyecciones de datos correspondientes a cada una de las relaciones entre las columnas, las que técnicamente son mapeos de elementos sin repetición entre los conjuntos de datos de dichas columnas.

Por lo tanto, cuando uno cuenta con una estructura lógica de datos, en un formato tabular (filas y columnas), puede utilizar estas columnas (atributos o conjuntos de atributos con dominios de valores) como conjuntos de datos a los que se le podrá aplicar restricciones lógicas que nos permitan conocer qué tipo de relación funcional existe entre dichos conjuntos de datos.

Las relaciones funcionales existentes obedecerán siempre al contexto de la lógica de negocio presente en la estructura lógica de datos. Es evidente que dichas relaciones funcionales deberán ser vistas siempre desde una perspectiva matemática.

Para aplicar las restricciones lógicas sobre un conjunto de datos utilizaremos el concepto de **Proyección** del **Álgebra Relacional**, denotado por el símbolo **Π**. De tal forma que si, por ejemplo, se requiere obtener la **proyección de datos** sin repetición de un **conjunto de atributos A** sobre la **relación R** (estructura lógica de datos), entonces denotaremos en la forma **ΠA (R)**.

Y para obtener la cantidad de elementos sin repetición correspondiente al **conjunto de atributos A**, entonces se aplicará la cardinalidad a su proyección de datos, denotándose en la forma **|ΠA (R)|**.

En consecuencia, para efectos de la presente investigación, se han definido las restricciones lógicas **sobreyectiva**, **biyectiva** y **multivaluada**, las que se serán definidas a continuación.

a. Restricción Lógica de una Relación Funcional Sobreyectiva

La restricción lógica de una relación funcional sobreyectiva nos permite saber si un **dominio de valores (datos)** dentro de un **conjunto de atributos A** mantiene una relación funcional sobreyectiva con respecto a otro **dominio de valores (datos)** dentro de un **conjunto de atributos B**, denotándose la restricción como sigue a continuación:

$$|\Pi AB\ (R)| = |\Pi A\ (R)| \wedge |\Pi AB\ (R)| > |\Pi B\ (R)| \Rightarrow A \rightarrow B$$

La restricción nos indica que si la cantidad de elementos de la **proyección de datos** sin repetición de **dos conjuntos de atributos A** y **B** sobre la **relación R** es igual a la cantidad de elementos de la **proyección de datos** sin repetición del **conjunto de atributos A** sobre la **relación R**, y además es mayor a la cantidad de elementos de la **proyección de datos** sin repetición del **conjunto de atributos B** sobre la **relación R**, entonces podemos afirmar que el **conjunto de atributos A** implica sobreyectivamente sobre el **conjunto de atributos B**.

Pudiendo, además, establecerse, como ejemplo, la siguiente forma de relación para el diseño lógico de una base de datos de relación funcional:

- **R1(A, B)**, donde **A** es el conjunto de atributos de clave primaria y **B** es un conjunto de atributos simple que es identificado de manera única por **A**, y en el que tanto **A** como **B** están dentro de una misma relación.

b. Restricción Lógica de una Relación Funcional Biyectiva

La restricción lógica de una relación funcional biyectiva nos permite saber si un **dominio de valores (datos)** dentro de un **conjunto de atributos A** mantiene una relación funcional biyectiva con respecto a otro **dominio de valores (datos)** dentro de un **conjunto de atributos B**, denotándose la restricción como sigue a continuación:

$$|\Pi AB\ (R)| = |\Pi A\ (R)| \wedge |\Pi AB\ (R)| = |\Pi B\ (R)| \Rightarrow A \twoheadrightarrow B \vee B \twoheadrightarrow A$$

La restricción nos indica que si la cantidad de elementos de la **proyección de datos** sin repetición de **dos conjuntos de atributos A** y **B** sobre la **relación R** es igual a la cantidad de elementos de la **proyección de datos** sin repetición del **conjunto de atributos A** sobre la **relación R**, y además es igual a la cantidad de elementos de

la **proyección de datos** sin repetición del **conjunto de atributos B** sobre la **relación R**, entonces podemos afirmar que el **conjunto de atributos A** implica biyectivamente sobre el **conjunto de atributos B**, y viceversa.

Pudiendo, además, establecerse, como ejemplo, las siguientes formas de relaciones para el diseño lógico de una base de datos de relación funcional:

- **R2(A, B, C)**, donde **A** y **B** son dos conjuntos de atributos de clave primaria y **C** es un conjunto de atributos simple que es identificado de manera única por **A** y **B**, y en el que tanto **A** como **B** están dentro de una misma relación.
- **R3(C, A, B)**, donde **C** es un conjunto de atributos de clave primaria y **A** y **B** es un conjunto de atributos simple que es identificado de manera única por **C**, y en el que tanto **A** como **B** están dentro de una misma relación.

c. Restricción Lógica de una Relación Funcional Multivaluada

La restricción lógica de una relación multivaluada nos permite saber si un **dominio de valores (datos)** dentro de un **conjunto de atributos A** mantiene una relación multivaluada con respecto a otro **dominio de valores (datos)** dentro de un **conjunto de atributos B**, denotándose la restricción como sigue a continuación:

$$|\Pi AB\ (R)| > |\Pi A\ (R)| \wedge |\Pi AB\ (R)| > |\Pi B\ (R)| \Rightarrow A \multimap B \vee B \multimap A$$

La restricción nos indica que si la cantidad de elementos de la **proyección de datos** sin repetición de **dos conjuntos de atributos A** y **B** sobre la **relación R** es mayor a la cantidad de elementos de la **proyección de datos** sin repetición del **conjunto de atributos A** sobre la **relación R**, y además es mayor a la cantidad de elementos de la **proyección de datos** sin repetición del **conjunto de atributos B** sobre la **relación R**, entonces podemos afirmar que el **conjunto de atributos A** no implica ni sobreyectivamente ni biyectivamente sobre el **conjunto de atributos B**, y viceversa.

Pudiendo, además, establecerse, como ejemplo, las siguientes formas de relaciones para el diseño lógico de una base de datos de relación funcional:

- **R4(A, C) y R5(B, D)** donde **A** y **B** son dos conjuntos de atributos de clave primaria, pero de diferentes relaciones, siendo y **C** un conjunto de atributos

simple que es identificado de manera única por **A**, y siendo **D** un conjunto de atributos simple que es identificado de manera única por **B**.

- **R6(A, C) y R7(D, B)** donde **A** y **D** son dos conjuntos de atributos de clave primaria, pero de diferentes relaciones, siendo y **C** un conjunto de atributos simple que es identificado de manera única por **A**, y siendo **B** un conjunto de atributos simple que es identificado de manera única por **D**.

- **R8(C, A) y R9(B, D)** donde **C** y **B** son dos conjuntos de atributos de clave primaria, pero de diferentes relaciones, siendo y **A** un conjunto de atributos simple que es identificado de manera única por **C**, y siendo **D** un conjunto de atributos simple que es identificado de manera única por **B**.

- **R10(C, A) y R11(D, B)** donde **C** y **D** son dos conjuntos de atributos de clave primaria, pero de diferentes relaciones, siendo y **A** un conjunto de atributos simple que es identificado de manera única por **C**, y siendo **B** un conjunto de atributos simple que es identificado de manera única por **D**.

J. Obtención de Dependencias de Relación Funcional

Cuando, en base a una estructura lógica de datos, se quiere obtener sus tuplas correspondientes, ineludiblemente se deberá hacer referencia al conjunto de elementos de cada fila en dicha estructura. Cada instancia de tupla (fila de la estructura) sostenida en los atributos, después de aplicarle las restricciones lógicas funcionales, será la que nos dé una información parcial acerca de qué tipo de relación funcional existen entre los atributos.

Para poder afirmar exactamente el tipo de relación existente entre dos atributos (o dos conjuntos de atributos) será necesario evaluar todas las tuplas que impliquen a dichos atributos. Por lo tanto, aplicando las restricciones lógicas sobre las proyecciones de tuplas sostenidas estructuralmente en las relaciones binarias implicadas, entonces se podrá obtener las dependencias de relación funcional válidas y no válidas.

Tabla 7 : Restricciones Lógicas Funcionales.

Tipo	Restricción Lógica
Relación Funcional Sobreyectiva	**$(\lvert \Pi AB\ (R) \rvert = \lvert \Pi A\ (R) \rvert) \wedge (\lvert \Pi AB\ (R) \rvert > \lvert \Pi B\ (R) \rvert) \Rightarrow A \rightarrow B$**

Relación Funcional Biyectiva	**$(\|\Pi AB\ (R)\| = \|\Pi A\ (R)\|) \wedge (\|\Pi AB\ (R)\| = \|\Pi B\ (R)\|) \Rightarrow A \twoheadrightarrow B \vee B \twoheadrightarrow A$**
Relación Multivaluada	**$(\|\Pi AB\ (R)\| > \|\Pi A\ (R)\|) \wedge (\|\Pi AB\ (R)\| > \|\Pi B\ (R)\|) \Rightarrow A \multimap B \vee B \multimap A$**

Fuente: Elaboración propia.

K. Reglas de Inferencia sobre Dependencias de Relación Funcional

Las reglas de inferencia que nos permitirán establecer el cumplimiento de las formas normales, para la normalización de una base de datos de relación funcional, que deberán asegurarnos que ***no habrá redundancia de los datos en el diseño***, que ***no existirán problemas al momento de actualizar los datos***, y que ***se mantendrá protegida la integridad de los datos***.

De manera que, para formalizar las inferencias, se establecerá los siguientes lemas:

Lema 1: Sea **R** un esquema de relación sostenido sobre el conjunto de atributos **U**, tal que es definido como **R(U)**, si tenemos que **X** e **Y** son dos subconjuntos de **U**, tal que **{X∪Y}** ⊂ **U**, entonces podemos establecer que a su vez **{x1**,**x2**,…,**xn-1**,**xn}** son **n** únicos subconjuntos de atributos de **X**, tal que **{x1∪x2**,…,**xn-1∪xn}** ⊆ **X**; y de la misma manera podemos establecer que **{y1**,**y2**,…,**ym-1**,**ym}** son **m** únicos subconjuntos de atributos de **Y**, tal que **{y1∪y2**,…,**ym-1∪ym}** ⊆ **Y**. Por lo tanto, estableceremos la siguiente dependencia de relación funcional:

1) **{x1.x2**,…,**xn-1.xn}** → **{y1.y2**,…,**ym-1.ym}**

⇔ **X → Y**

Lema 2: Sea **R** un esquema de relación sostenido sobre el conjunto de atributos **U**, tal que es definido como **R (U)**, si tenemos que **A, B, C, X**, **Y** y **Z** son tres subconjuntos de **U** distintos entre sí, tal que **{A∪B∪C∪X∪Y∪Z}** ⊂ **U**. Por lo tanto, estableceremos las siguientes dependencias de relación funcional:

1) **X → YZ**

2) **X → Y**

3) **X → Z**

4) **A → C**

5) **B → C**

Y en base a los lemas definidos anteriormente, se definirá a su vez los siguientes teoremas, que corresponden a las reglas de inferencia que nos permitirán eliminar la redundancia en las dependencias de relación funcional.

a. Dependencias Reflexivas

Teorema 1.1: Si de la formulación del **Lema 1** tenemos que existe una relación funcional sobreyectiva entre el subconjunto de atributos **{x1.x2**,…,**xn-1.xn}** y el subconjunto de atributos **{y1.y2**,…,**ym-1.ym}**, estableciéndose de la forma **{x1.x2**,…,**xn-1.xn}** → **{y1.y2**,…,**ym-1.ym}**, y además **{y1.y2**,…,**ym-1.ym}** está incluido en **{x1.x2**,…,**xn-1.xn}**, ya sea parcialmente o completamente, tal que **{y1.y2**,…,**ym-1.ym}** ⊆ **{x1.x2,…,xn-1.xn}**, entonces podemos inferir que, ya que existe reflexividad, la siguiente dependencia de relación funcional deberá ser suprimida:

1) **{x1.x2**,…,**xn-1.xn}** → **{y1.y2**,…,**ym-1.ym}**

⇔ **X → Y**

Prueba 1.1.1: Se deduce del **Teorema 1.1** que la dependencia de relación funcional **{x1.x2**,…,**xn-1.xn}** → **{y1.y2**,…,**ym-1.ym}**, que obedece a una relación funcional sobreyectiva, deberá ser suprimida ante la existencia de reflexividad en la dependencia, si se cumple cualquiera de los siguientes casos:

Caso 1:

1) **{x1.x2**,…,**xn-1.xn}** → **{y1.y2**,…,**ym-1.ym}**

2) **{y1.y2**,…,**ym-1.ym}** ⊆ **{x1.x2,…,xn-1.xn}**

3) **{x1.x2**,…,**xn-1.xn}** = **{y1.y2**,…,**ym-1.ym}**

Caso 2:

1) **{x1.x2,…,xn-1.xn} → {y1.y2,…,ym-1.ym}**

2) **{y1.y2,…,ym-1.ym} ⊆ {x1.x2,…,xn-1.xn}**

3) **{x1.x2,…,xn-1.xn} ≠ {y1.y2,…,ym-1.ym}**

b. Dependencias con Biyección Parcial

Teorema 1.2: Si de la formulación del **Lema 1** tenemos que **xp** es un subconjunto de atributos de **{x1.x2,…,xn-1.xn}**, tal que **xp** ⊂ **{x1.x2,…,xn-1.xn}**; y, de la misma manera, tenemos que **yr** es un subconjunto de atributos de **{y1.y2,…,ym-1.ym}**, tal que **yr** ⊆ **{y1.y2,…,ym-1.ym}**, entonces cuando exista una relación funcional biyectiva entre **xp** y **yr**, estableciéndose en la forma **xp** ↠ **yr** o, siendo equivalente, en la forma **yr** ↠ **xp**, podemos inferir que la siguiente dependencia de relación funcional deberá ser suprimida:

1) **{x1.x2,…,xn-1.xn} → {y1.y2,…,ym-1.ym}**

⇔ **X → Y**

Prueba 1.2.1: Se deduce del **Teorema 1.2** que la dependencia de relación funcional **{x1.x2,…,xn-1.xn}** → **{y1.y2,…,ym-1.ym}**, que obedece a una relación funcional sobreyectiva, deberá ser suprimida ante la existencia de una relación funcional biyectiva entre **xp** y **yr** en la dependencia, si se cumple lo siguiente:

1) **{x1.x2,…,xn-1.xn} → {y1.y2,…,ym-1.ym}**

2) **xp ⊂ {x1.x2,…,xn-1.xn}**

3) **yr ⊆ {y1.y2,…,ym-1.ym}**

4) **yr ↠ xp ∨ xp ↠ yr**

5) **{x1.x2,…,xn-1.xn} – xp ≠ Ø**

c. Eliminar Dependencias con Sobreyección en el Lado Izquierdo

Teorema 1.3: Si de la formulación del **Lema 1** tenemos que **xp** y **xq** son dos subconjuntos de atributos de **{x1.x2,…,xn-1.xn}**, tal que **{xp∪xq}** ⊂ **{x1.x2,…,xn-1.xn}**, de manera tal que **xq** es suprimido de **{x1.x2,…,xn-1.xn}** cuando exista una relación funcional sobreyectiva entre **xp** y **xq**, estableciéndose esto último en la forma **xp** → **xq**. Entonces podemos inferir que dependencia **{x1.x2,…,xn-1.xn}** → **{y1.y2,…,ym-1.ym}** tendrá la siguiente simplificación:

1) {x1.x2,…,xn-1.xn} – xq → {y1.y2,…,ym-1.ym}

⇔ **X – xq → Y**

Prueba 1.3.1: Se deduce del **Teorema 1.3** que la dependencia de relación funcional **{x1.x2,…,xn-1.xn} – xq → {y1.y2,…,ym-1.ym}**, que obedece a una relación funcional sobreyectiva, es la simplificación de la dependencia de relación funcional **{x1.x2,…,xn-1.xn}** → **{y1.y2,…,ym-1.ym}** si se cumple cualquiera de los siguientes casos:

Caso 1:

6) {x1.x2,…,xn-1.xn} → {y1.y2,…,ym-1.ym}

7) {xp∪xq} ⊂ {x1.x2,…,xn-1.xn}

8) xp → xq

9) {x1.x2,…,xn-1.xn} – xq → {y1.y2,…,ym-1.ym}

10) {x1.x2,…,xn-1.xn} – {xp.xq} = Ø

Caso 2:

1) {x1.x2,…,xn-1.xn} → {y1.y2,…,ym-1.ym}

2) {xp∪xq} ⊂ {x1.x2,…,xn-1.xn}

3) xp → xq

4) {x1.x2,…,xn-1.xn} – xq → {y1.y2,…,ym-1.ym}

5) {x1.x2,…,xn-1.xn} – {xp.xq} ≠ Ø

6) xp.xq ⊸ {x1.x2,…,xn-1.xn} – {xp.xq}

d. Dependencias con Sobreyección Parcial

Teorema 1.4: Si de la formulación del **Lema 1** tenemos que **xp** es un subconjunto de atributos de **{x1**.**x2**,…,**xn-1**.**xn}**, tal que **xp** ⊂ **{x1.x2,…,xn-1.xn}**. De manera tal que **{x1.x2,…,xn-1.xn} – xp** es suprimido de **{x1**.**x2**,…,**xn-1**.**xn}** cuando exista una relación funcional sobreyectiva entre **{y1**.**y2**,…,**ym-1**.**ym}** y **xp**, estableciéndose en **xp** → **{y1**.**y2**,…,**ym-1**.**ym}** o, no siendo equivalente, en **{y1**.**y2**,…,**ym-1**.**ym}** → **xp**; y además debe cumplirse la existencia de una relación funcional multivaluada entre **xp** y **{x1.x2,…,xn-1.xn} – xp**, estableciéndose en **xp –⊸ {x1**.**x2,…,xn-1.xn} – xp** o, siendo equivalente, en **{x1.x2,…,xn-1.xn} – xp –⊸ xp**. De manera tal que podemos inferir cualquiera de las posibles simplificaciones siguientes:

1) **xp** → **{y1**.**y2**,…,**ym-1**.**ym}**

⇔ **xp** → **Y**

2) **{y1**.**y2**,…,**ym-1**.**ym}** → **xp**

⇔ **Y** → **xp**

Prueba 1.4.1: Se deduce del **Teorema 1.4** que la dependencia de relación funcional **xp** → **{y1**.**y2**,…,**ym-1**.**ym}**, que obedece a una relación funcional sobreyectiva, es la simplificación de la dependencia de relación funcional **{x1**.**x2**,…,**xn-1**.**xn}** → **{y1**.**y2**,…,**ym-1**.**ym}**, si se cumple lo siguiente:

1) **{x1**.**x2**,…,**xn-1**.**xn}** → **{y1**.**y2**,…,**ym-1**.**ym}**

2) **xp** ⊂ **{x1.x2,…,xn-1.xn}**

3) **xp** → **{y1**.**y2**,…,**ym-1**.**ym}**

4) **xp –⊸ {x1.x2,…,xn-1.xn} – xp**

5) **{x1.x2,…,xn-1.xn} – xp –⊸ {y1**.**y2**,…,**ym-1**.**ym}**

Prueba 1.4.2: Se deduce del **Teorema 1.4** que la dependencia de relación funcional **{y1**.**y2**,…,**ym-1**.**ym}** → **xp**, que obedece a una relación funcional sobreyectiva, es la simplificación de la dependencia de relación funcional **{x1**.**x2**,…,**xn-1**.**xn}** → **{y1**.**y2**,…,**ym-1**.**ym}**, si se cumple lo siguiente:

1) **{x1**.**x2**,…,**xn-1**.**xn}** → **{y1**.**y2**,…,**ym-1**.**ym}**

2) **xp ⊂ {x1.x2,…,xn-1.xn}**

3) **{y1.y2**,…,**ym-1.ym} → xp**

4) **xp –⊸ {x1.x2,…,xn-1.xn} – xp**

5) **{x1.x2,…,xn-1.xn} – xp –⊸ {y1.y2**,…,**ym-1.ym}**

e. Dependencias con Sobreyección en el Lado Derecho

Teorema 2.1: Si de la formulación del **Lema 2** tenemos que el subconjunto de atributos **Z** depende del subconjunto de atributos **Y**, estableciéndose esto último de la forma **Y → Z**, entonces podemos inferir la siguiente simplificación:

1) **X → Y**

Prueba 2.1.1: Se deduce del **Teorema 2.1** que la dependencia de relación funcional **X → Y** es la simplificación de la dependencia de relación funcional **X → YZ**, si se cumple lo siguiente:

1) **X → YZ**

2) **X → Y**

3) **X → Z**

4) **Y → Z**

f. Descomposición de Dependencias de Relación Funcional

Teorema 2.2: Si de la formulación del **Lema 2** tenemos que el subconjunto de atributos **Z** depende del subconjunto de atributos **Y**, estableciéndose esto último de la forma **Y ↠ Z ∨ Y → Z**, entonces, de lo anterior, podemos inferir la siguiente descomposición:

1) **X → Y ∧ X → Z**

Prueba 2.2.1: Se deduce del **Teorema 2.2** que **X** → **Y** y **X** → **Z** es la descomposición de la dependencia de relación funcional **X** → **YZ**, si se cumple lo siguiente:

1) X → YZ

2) X → Y

3) X → Z

4) Y ↠ Z ∨ Y → Z

g. Composición de Dependencias de Relación Funcional

Teorema 2.3: Si de la formulación del **Lema 2** tenemos que existe una relación funcional biyectiva entre el subconjunto de atributos **Z** y el subconjunto de atributos **Y**, estableciéndose esto último de la forma **Y** ↠ **Z** ∨ **Y** → **Z**, entonces, de lo anterior, podemos inferir la siguiente composición:

1) X → YZ

Prueba 2.3.1: Se deduce del **Teorema 2.3** que **X** → **YZ** es la composición de las dependencias de relación funcional **X** → **Y** y **X** → **Z**, si se cumple lo siguiente:

1) X → YZ

2) X → Y

3) X → Z

4) Y ↠ Z ∨ Y → Z

Teorema 2.4: Si de la formulación del **Lema 2** tenemos que existe una relación funcional biyectiva entre el subconjunto de atributos **A** y el subconjunto de atributos **B**, estableciéndose esto último de la forma **A** ↠ **B** ∨ **A** → **B** ∨ **B** → **A**, entonces, de lo anterior, podemos inferir la siguiente composición:

1) AB → C

Prueba 2.4.1: Se deduce del **Teorema 2.4** que **AB → C** es la composición de las dependencias de relación funcional **A → C** y **B → C**, si se cumple lo siguiente:

1) **AB → C**

2) **A → C**

3) **B → C**

4) **A ↠ B ∨ A → B ∨ B → A**

h. Equivalencias de Normalización

Tabla 8 : Conciliación de la Normalización entre el Modelo Relacional y el Modelo de Relación Funcional.

Modelo Relacional	Modelo de Relación Funcional
Primer Forma Normal (1FN)	Tuplas de Estructura Lógica de Datos
Segunda Forma Normal (2FN)	Teorema 1.1, Teorema 1.2, Teorema 1.3, Teorema 1.4
Tercer Forma Normal (3FN)	Teorema 2.1
Forma Normal Boyce-Codd (BCFN)	Aplicación de Restricciones Lógicas Funcionales, Teorema 1.4
Cuarta Forma Normal (4FN)	Aplicación de Restricciones Lógicas Funcionales, Teorema 1.4
Quinta Forma Normal (5FN)	Aplicación de Restricciones Lógicas Funcionales, Teorema 1.4

Fuente: Elaboración propia.

2. [OE2] Aplicar el Modelo de Datos de Relación Funcional

A. Empresa de Servicio de Hospedaje

Para fines de la presente investigación, se escogió como muestra a una **empresa de servicios de hospedaje**, cuyo nombre real se mantendrá en reserva para efectos de confidencialidad de la personería jurídica. Dicha empresa opera dentro del rubro de servicios de Hotelería y Turismo.

La misión principal de la empresa es dar servicios de hospedamiento a clientes objetivos, así como servicios de comida y relajación dentro de sus instalaciones, que cubran las expectativas específicas de los clientes a los cuales se pretende proporcionar el valor agregado del servicio.

Para poder lograr este cometido, la empresa de servicios de hospedaje se encarga de invertir esfuerzos en mantener un constante análisis de los procesos de preventa y de postventa, sin descuidar los procesos claves correspondientes a las ventas de ocupaciones de habitaciones.

Para la poder aplicar se utilizarán como muestra en la investigación para la aplicación del modelo propuesto, **serán proporcionadas por una empresa de servicios de hospedajes de la ciudad de Trujillo, y corresponden a datos del año 2014**.

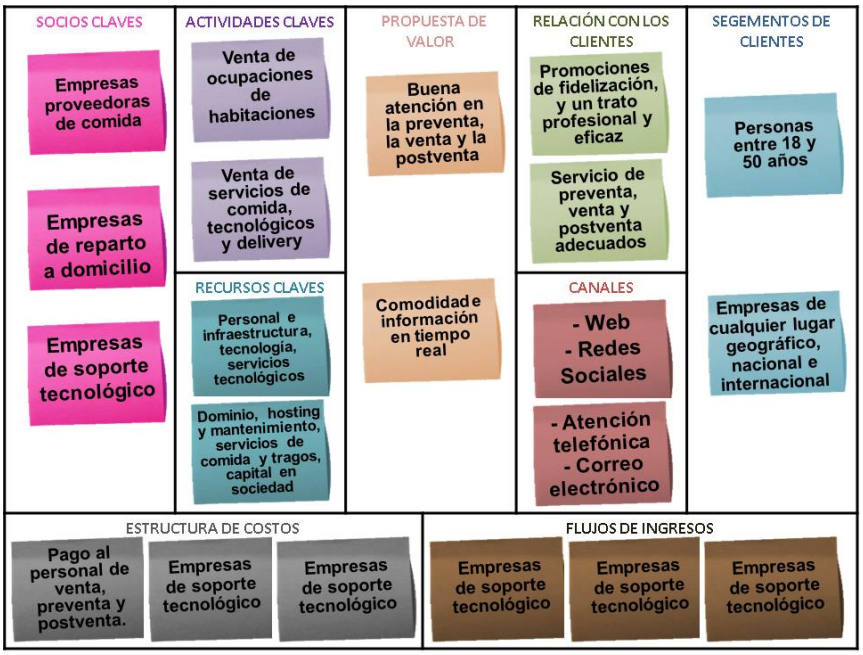
Según los requerimientos y el acuerdo con la empresa de servicios de hospedajes, toda información que se muestra deberá ser utilizada para uso exclusivo de la presente investigación, no pudiendo aquella ser copiada o reproducida de cualquier modo sin la coordinación con el autor y bajo permiso.

B. Enfoque Ontológico del Modelo Negocio

a. Modelo de Negocio

Para poder consolidar la gestión estratégica del modelo de negocio de la empresa de servicios de hospedaje, y con el fin de poder entender los procesos claves que se enmarcan dentro de los conceptos de negocio por los cuales se rige, hemos construido un panel del tipo ***Business Model Canvas***, el cual permitirá graficar la perspectiva del modelo, tal como se muestra en el gráfico siguiente.

Gráfico 11 : Abstracción Canvas del Modelo de Negocio.



Fuente: Elaboración propia.

En el panel mostrado se puede notar que los procesos que enmarcan las actividades claves del negocio son (1) la venta de ocupaciones de habitaciones y (2) la venta de servicios de comida y esparcimiento.

Sin embargo, en la presente investigación, nos centraremos sobretodo en el proceso de venta de ocupaciones de habitaciones, por ser el proceso central de la empresa de servicios de hospedaje.

b. Reglas de Negocio

Después de analizar los conceptos dentro de procesos del modelo de negocio de la empresa de servicios de hospedaje, se establecieron las siguientes reglas de negocio básicas:

(1) Toda PERSONA tiene un número único de DNI.

(2) Un ADULTO es una PERSONA que tiene una edad mayor o igual a 18.

(3) Todo MENOR es una PERSONA que tiene una edad menor a 18.

(4) Un EMPLEADO es un ADULTO que siempre tiene un código de EMPLEADO.

(5) Un CLIENTE es un ADULTO que siempre tiene un código de CLIENTE.

(6) Todo ADULTO es una PERSONA.

(7) Todo MENOR es una PERSONA.

(8) Todo EMPLEADO es un ADULTO.

(9) Todo CLIENTE es un ADULTO.

(10) Toda HABITACION tiene un único tipo de habitación y un único precio de habitación.

(11) En toda OCUPACION un único CLIENTE es un ocupante principal y una única HABITACION es reservada. Además, toda OCUPACION tiene una única fecha de entrada y puede tener ocupantes adicionales.

(12) En toda DESOCUPACION un único CLIENTE es un desocupante principal y una única HABITACION es desocupada. Además, toda DESOCUPACION se realiza sobre una única OCUPACION, tiene una única fecha de salida y puede tener ocupantes adicionales.

(13) En toda VENTA se tiene cuando menos una OCUPACION y un único EMPLEADO como vendedor.

(14) Todo SERVICIO es para toda OCUPACION vendida.

(15) Todo OCUPANTE es un ADULTO o un MENOR relacionado a toda ASIGNACION.

(16) En toda ASIGNACION todo OCUPANTE pertenece a una única OCUPACION o a una única DESOCUPACION.

c. Lógicas Descriptivas

Por otro lado, para poder formalizar ontológicamente la representación del conocimiento, restricto a las reglas de negocio, que compromete los procesos del modelo de negocio de la empresa de servicios de hospedaje, se utilizará la semántica formal que nos proporcionan las **lógicas descriptivas**, las cuales representarán

ontológicamente el conocimiento del dominio de aplicación, definiendo primero los conceptos más relevantes del mismo dominio para luego usar estos conceptos en la búsqueda de especificar relaciones, así como propiedades de objetos e individuos.

Las construcciones **TBox** (afirmaciones sobre conceptos) de las lógicas descriptivas se establecen en los siguientes axiomas que limitan el dominio correspondiente a los procesos del modelo de negocio de la empresa de servicios de hospedaje:

(1) PERSONA $\sqsubseteq$ =1 tieneNumeroDni.⊤

(2) ADULTO $\equiv$ PERSONA $\sqcap$ ≥18 tieneEdad.PERSONA

(3) MENOR $\equiv$ PERSONA $\sqcap$ <18 tieneEdad.PERSONA

(4) EMPLEADO $\equiv$ ADULTO $\sqcap$ ∀ tieneCodigoEmpleado.ADULTO

(5) CLIENTE $\equiv$ ADULTO $\sqcap$ ∀ tieneCodigoCliente.ADULTO

(6) ADULTO $\sqsubseteq$ PERSONA

(7) MENOR $\sqsubseteq$ PERSONA

(8) EMPLEADO $\sqsubseteq$ ADULTO

(9) CLIENTE $\sqsubseteq$ ADULTO

(10) HABITACION $\sqsubseteq$ =1 tieneTipoHabitacion
$\sqcap$ =1 tienePrecioHabitacion

(11) OCUPACION $\sqsubseteq$ =1 tieneOcupantePrincipal.CLIENTE
$\sqcap$ =1 tieneHabitacionReservada.HABITACION
$\sqcap$ =1 tieneFechaEntrada
$\sqcap$ ≥0 tieneOcupanteAdicional.OCUPANTE

(12) DESOCUPACION $\sqsubseteq$ =1 tieneDesocupantePrincipal.CLIENTE
$\sqcap$ =1 tieneHabitacionDesocupada.HABITACION
$\sqcap$ =1 tieneOcupacion.OCUPACION
$\sqcap$ =1 tieneFechaSalida
$\sqcap$ ≥0 tieneDesocupantesAdicionales.OCUPANTE

(13) VENTA $\sqsubseteq$ ∃ tieneOcupacion.OCUPACION
$\sqcap$ =1 tieneVendedor.EMPLEADO

(14) SERVICIO ⊑ ∀ esOcupacionVendida.OCUPACION

(15) OCUPANTE ⊑ (ADULTO ⊔ MENOR)
⊓ ∀ tieneAsignacion.ASIGNACION

(16) ASIGNACION ⊑ ∀ esOcupante.OCUPANTE
⊓ (=1 tieneOcupacion.OCUPACION ⊔
=1 tieneDesocupacion.DESOCUPACION)

d. Representación Formal del Conocimiento

De manera que la trazabilidad de las reglas de negocio de la empresa de servicios de hospedaje hacia una construcción de lógicas descriptivas nos dará la representación formal de la base de conocimiento del negocio de la empresa, tal como se muestra en el siguiente cuadro:

El cuadro siguiente se muestra la representación formal del conocimiento del negocio nos permite establecer las bases semánticas del negocio, que son loa conceptos de información de los procesos que realiza la empresa. Y de esta manera poder diagramar el Modelo de Datos Conceptual del Modelo de Negocio.

Tabla 9 : Representación Formal de la Base de Conocimiento del Negocio.

Representación Formal de la Base de Conocimiento del Negocio		
Nº	**Reglas de Negocio**	**Lógicas Descriptivas**
1	Toda PERSONA tiene un número único de DNI.	PERSONA ⊑ =1tieneNumeroDni.⊤
2	Un ADULTO es una PERSONA que tiene una edad mayor o igual a 18.	ADULTO ≡ PERSONA ⊓ ≥18tieneEdad.PERSONA
3	Todo MENOR es una PERSONA que tiene una edad menor a 18.	MENOR ≡ PERSONA ⊓ <18tieneEdad.PERSONA
4	Un EMPLEADO es un ADULTO que siempre tiene un código de EMPLEADO.	EMPLEADO ≡ ADULTO ⊓ ∀tieneCodigoEmpleado.ADULTO
5	Un CLIENTE es un ADULTO que siempre tiene un código de CLIENTE.	CLIENTE ≡ ADULTO ⊓ ∀tieneCodigoCliente.ADULTO

6	Todo ADULTO es una PERSONA.	ADULTO ⊑ PERSONA
7	Todo MENOR es una PERSONA.	MENOR ⊑ PERSONA
8	Todo EMPLEADO es un ADULTO.	EMPLEADO ⊑ ADULTO
9	Todo CLIENTE es un ADULTO.	CLIENTE ⊑ ADULTO
10	Toda HABITACION tiene un único tipo de habitación y un único precio de habitación.	HABITACION ⊑ =1tieneTipoHabitacion ⊓ =1tienePrecioHabitacion
11	En toda OCUPACION un único CLIENTE es un ocupante principal y una única HABITACION es reservada. Además, toda OCUPACION tiene una única fecha de entrada y puede tener ocupantes adicionales.	OCUPACION ⊑ =1tieneOcupantePrincipal.CLIENTE ⊓ =1tieneHabitacionReservada.HABITACION ⊓ =1 tieneFechaEntrada ⊓ ≥0tieneOcupanteAdicional.OCUPANTE
12	En toda DESOCUPACION un único CLIENTE es un desocupante principal y una única HABITACION es desocupada. Además, toda DESOCUPACION se realiza sobre una única OCUPACION, tiene una única fecha de salida y puede tener ocupantes adicionales.	DESOCUPACION ⊑ =1tieneDesocupantePrincipal.CLIENTE ⊓ =1tieneHabitacionDesocupada.HABITACION ⊓ =1 tieneOcupacion.OCUPACION ⊓ =1tieneFechaSalida ⊓ ≥0tieneDesocupantesAdicionales.OCUPANTE
13	En toda VENTA se tiene cuando menos una OCUPACION y un único EMPLEADO como vendedor.	VENTA ⊑ ∃tieneOcupacion.OCUPACION ⊓ =1tieneVendedor.EMPLEADO
14	Todo SERVICIO es para toda OCUPACION vendida.	SERVICIO ⊑ ∀esOcupacionVendida.OCUPACION
15	Todo OCUPANTE es un ADULTO o un MENOR relacionado a toda ASIGNACION.	OCUPANTE ⊑ (ADULTO ⊔ MENOR) ⊓ ∀tieneAsignacion.ASIGNACION
16	En toda ASIGNACION todo OCUPANTE pertenece a una única OCUPACION o a una única DESOCUPACION.	ASIGNACION ⊑ ∀esOcupante.OCUPANTE ⊓ (=1tieneOcupacion.OCUPACION ⊔ =1tieneDesocupacion.DESOCUPACION)

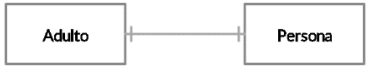
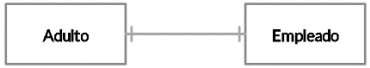
Fuente: Elaboración propia.

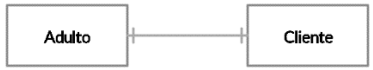
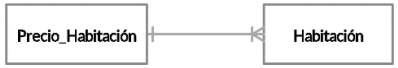
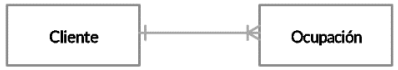
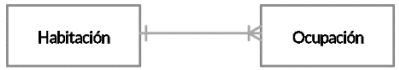
C. Modelo de Datos Conceptual del Modelo de Negocio

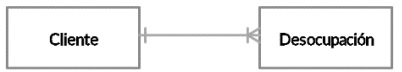
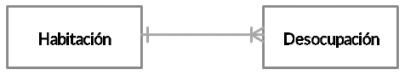
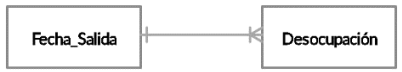
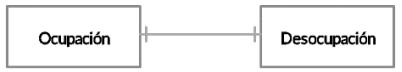
Para poder visualizar las relaciones entre los conceptos correspondientes a los procesos correspondientes a nuestra muestra de investigación, la empresa de servicios de hospedaje, hemos creado el siguiente diagrama, donde podemos visualizar no solamente los conceptos propios del dominio del negocio, sino también la cardinalidad y las restricciones de las relaciones entre los conceptos.

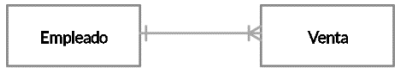
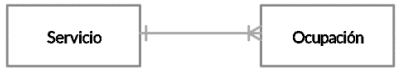
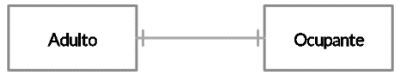
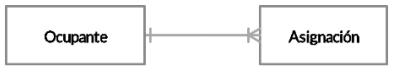
También se ha creado un cuadro de trazabilidad entre la formalización de la Ontología del Negocio y el Modelo de Datos Conceptual del Modelo de Negocio, tal como se muestra a continuación.

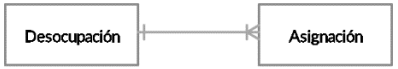
Tabla 10 : Trazabilidad (Ontología del Negocio - Modelo de Datos Conceptual).

Cuadro de Trazabilidad (Ontología del Negocio - Modelo de Datos Conceptual)			
Reglas de Negocio	**Lógicas Descriptivas**	**Cardinalidad entre Conceptos**	**Restricciones entre Conceptos**
Toda PERSONA tiene un número único de DNI.	PERSONA ⊑ =1tieneNumeroDni.⊤		Un número de DNI puede pertenecer a una única persona.
- Un ADULTO es una PERSONA que tiene una edad mayor o igual a 18. - Todo ADULTO es una PERSONA.	- ADULTO ≡ PERSONA ⊓ ≥18tieneEdad.PERSONA - ADULTO ⊑ PERSONA		Un adulto solamente puede ser una única persona.
			Una misma edad puede ser asignada a más de una persona adulta, pero un adulto solo puede tener una única edad.
- Todo MENOR es una PERSONA que tiene una edad menor a 18. - Todo MENOR es una PERSONA.	- MENOR ≡ PERSONA ⊓ <18tieneEdad.PERSONA - MENOR ⊑ PERSONA		Un menor solamente puede ser una única persona.
			Una misma edad puede ser asignada a más de una persona menor, pero un menor solo puede tener una única edad.
- Un EMPLEADO es un ADULTO que siempre tiene un código de EMPLEADO. - Todo EMPLEADO es un ADULTO.	- EMPLEADO ≡ ADULTO ⊓ ∀tieneCodigoEmpleado.ADULTO - EMPLEADO ⊑ ADULTO		Un empleado solamente puede ser un único adulto.

- Un CLIENTE es un ADULTO que siempre tiene un código de CLIENTE. - Todo CLIENTE es un ADULTO.	- CLIENTE ≡ ADULTO ⊓ ∀tieneCodigoCliente.ADULTO - CLIENTE ⊑ ADULTO		Un cliente solamente puede ser un único adulto.
Toda HABITACION tiene un único tipo de habitación y un único precio de habitación.	HABITACION ⊑ =1tieneTipoHabitacion ⊓ =1tienePrecioHabitacion		Un mismo tipo de habitación puede ser asignado a muchas habitaciones, pero una habitación solamente puede tener un único tipo de habitación.
			Un mismo precio de habitación puede ser asignado a muchas habitaciones, pero una habitación solamente puede tener un único tipo de habitación.
En toda OCUPACION un único CLIENTE es un ocupante principal y una única HABITACION es reservada. Además, toda OCUPACION tiene una única fecha de entrada y puede tener ocupantes adicionales.	OCUPACION ⊑ =1tieneOcupantePrincipal.CLIENTE ⊓ =1tieneHabitacionReservada.HABITACION ⊓ =1 tieneFechaEntrada ⊓ ≥0tieneOcupanteAdicional.OCUPANTE		Un mismo cliente puede tener muchas ocupaciones, pero una ocupación solamente puede tener a un único cliente como ocupante principal.
			Una misma habitación puede estar en muchas ocupaciones, pero una ocupación solamente puede tener una única habitación ocupada.

			Una misma fecha de entrada puede ser registrada para muchas ocupaciones, pero una ocupación solamente puede tener una única fecha de entrada registrada.
En toda DESOCUPACION un único CLIENTE es un desocupante principal y una única HABITACION es desocupada. Además, toda DESOCUPACION se realiza sobre una única OCUPACION, tiene una única fecha de salida y puede tener ocupantes adicionales.	DESOCUPACION ⊑ =1tieneDesocupantePrincipal.CLIENTE ⊓ =1tieneHabitacionDesocupada.HABITACION ⊓ =1 tieneOcupacion.OCUPACION ⊓ =1tieneFechaSalida ⊓ ≥0tieneDesocupantesAdicionales.OCUPANTE		Un mismo cliente puede tener muchas desocupaciones, pero una desocupación solamente puede tener a un único cliente como desocupante principal.
			Una misma habitación puede estar en muchas desocupaciones, pero una desocupación solamente puede tener una única habitación desocupada.
			Una misma fecha de salida puede ser registrada para muchas desocupaciones, pero una desocupación solamente puede tener una única fecha de salida registrada.
			Una desocupación solamente puede hacerse sobre una única ocupación.
En toda VENTA se tiene cuando menos una OCUPACION y un único	VENTA ⊑ ∃tieneOcupacion.OCUPACION ⊓ =1tieneVendedor.EMPLEADO		Una misma venta puede tener muchas ocupaciones, pero una ocupación solamente

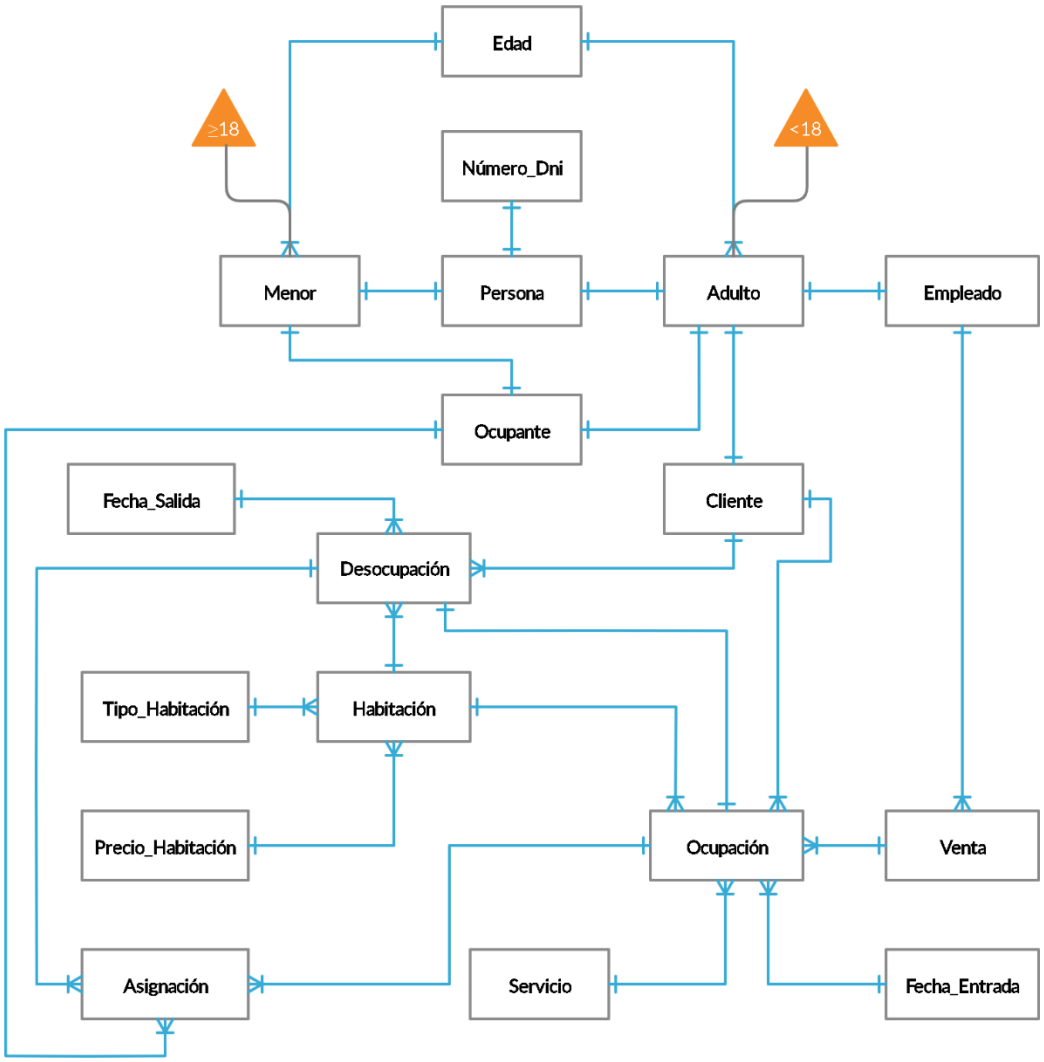
EMPLEADO como vendedor.			puede estar relacionada a una única venta.
			Un mismo empleado puede tener realizar muchas ventas, pero una venta solamente puede ser hecha por un empleado.
Todo SERVICIO es para toda OCUPACION vendida.	SERVICIO ⊑ ∀esOcupacionVendida.OCUPACION		Un mismo servicio puede ser asignado a muchas ocupaciones, pero una ocupación solamente puede tener un servicio.
- Todo OCUPANTE es un ADULTO o un MENOR relacionado a toda ASIGNACION. - En toda ASIGNACION todo OCUPANTE pertenece a una única OCUPACION o a una única DESOCUPACION.	- OCUPANTE ⊑ (ADULTO ⊔ MENOR) ⊓ ∀tieneAsignacion.ASIGNACION - ASIGNACION ⊑ ∀esOcupante.OCUPANTE ⊓ (=1tieneOcupacion.OCUPACION ⊔ =1tieneDesocupacion.DESOCUPACION)		Un ocupante solamente puede ser un único adulto.
			Un ocupante solamente puede ser un único menor.
			Un mismo ocupante puede estar en muchas asignaciones de ocupación, pero una asignación solamente puede contener un ocupante.
			Una misma ocupación puede tener muchas asignaciones de ocupantes, pero una asignación solamente

			puede ser para una única ocupación.
			Una misma desocupación puede tener muchas asignaciones de ocupantes involucradas, pero una asignación solamente puede ser para una única desocupación.

Fuente: Elaboración propia.

En base a las restricciones definidas al realizar la trazabilidad entre la Ontología del Negocio y el Modelo de Datos Conceptual, se elaboró el siguiente diagrama:

Gráfico 12 : Diagrama de Datos Conceptual.



Fuente: Elaboración propia.

D. Estructura Lógica de Datos del Modelo de Negocio

La empresa de servicios de hospedaje proporcionó la información relevante, que se encuentra en sus bases de datos organizacionales, ya sean de tipo físico o digitales, correspondiente a sus procesos operativos de venta. Dicha información estará volcada en estructuras lógicas de datos, las que representan los ***metadatos*** de la información organizacional, y se las presentará a continuación.

a. Estructura Lógica de Datos del Proceso de Ventas de Servicios

El **Proceso de Ventas de Servicios** corresponde a las tareas de ventas que cumple cada empleado de la empresa de servicios de hospedaje. Cada venta de servicio tiene la información de una o más ocupaciones de habitaciones, siendo también que puede haber uno o más ocupantes.

Tabla 11 : Estructura Lógica de Datos del Proceso de Ventas de Servicios.

Ventas				
CodigoVenta	**CodigoOcupacion**	**CodigoEmpleado**	**DniEmpleado**	**NombreEmpleado**
CV0000001	CO0000001	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000002	CO0000002	CE43987458	43987457	Víctor Manual Herrera Toledo
CV0000003	CO0000003	CE35958374	35958374	Carlos Antonio Correa Montoya
CV0000004	CO0000004	CE35958374	35958374	Carlos Antonio Correa Montoya
CV0000005	CO0000005	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000006	CO0000006	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000007	CO0000007	CE35958374	35958374	Carlos Antonio Correa Montoya

CV0000008	CO0000008	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000009	CO0000009	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000010	CO0000010	CE35958374	35958374	Carlos Antonio Correa Montoya
CV0000011	CO0000011	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000012	CO0000012	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000013	CO0000013	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000014	CO0000014	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000015	CO0000015	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000016	CO0000016	CE35958374	35958374	Carlos Antonio Correa Montoya
CV0000016	CO0000017	CE35958374	35958374	Carlos Antonio Correa Montoya
CV0000016	CO0000018	CE35958374	35958374	Carlos Antonio Correa Montoya
CV0000017	CO0000019	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000018	CO0000020	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000018	CO0000021	CE43987457	43987457	Víctor Manual Herrera Toledo
CV0000019	CO0000022	CE35958374	35958374	Carlos Antonio Correa Montoya
CV0000019	CO0000023	CE35958374	35958374	Carlos Antonio Correa Montoya
CV0000020	CO0000024	CE35958374	35958374	Carlos Antonio Correa Montoya

Fuente: Elaboración propia.

b. Estructura Lógica de Datos del Proceso de Ocupaciones de Habitaciones

El **Proceso de Ocupciones de Habitaciones** corresponde a las tareas de ocupación y desocupación que ejecuta la empresa de servicios de hospedaje. Cada ocupación es identificada por tener una fecha única de ocupación, una única habitación y un único cliente. En el caso de una desocupación, ésta se identifica por tener una fecha única de desocupación, una única habitación y un único cliente.

Tabla 12 : Estructura Lógica de Datos del Proceso de Ocupaciones de Habitaciones.

OCUPACIONES									
CodigoOcupacion	**CodigoCliente**	**DniCliente**	**NombreCliente**	**CodigoHabitacion**	**FechaEntrada**	**TipoHabitacion**	**PrecioNoche**	**CodigoServicio**	**NombreServicio**
CO0000001	CC41232354	41232354	Carlos Alberto Vereau Gutiérrez	CH0212	08-10-2014	Suite	100	CS0010	Servicio Normal
CO0000002	CC18264354	18264354	Sofía Marisol Pérez Cortez	CH0213	08-10-2014	Doble	50	CS0010	Servicio Normal
CO0000003	CC17269484	17269484	Lucía Madeleine Espinoza Mendoza	CH0214	08-10-2014	Doble	50	CS0020	Servicio Adicional
CO0000004	CC45295837	45295837	Nuria Geraldine Alcalde Ferreira	CH0215	08-10-2014	Individual	30	CS0020	Servicio Adicional
CO0000005	CC41232354	41232354	Carlos Alberto Vereau Gutiérrez	CH0213	09-10-2014	Doble	50	CS0030	Servicio Extendido
CO0000006	CC45295837	45295837	Nuria Geraldine Alcalde Ferreira	CH0213	10-10-2014	Doble	50	CS0020	Servicio Adicional
CO0000007	CC41232354	41232354	Carlos Alberto Vereau Gutiérrez	CH0212	10-10-2014	Suite	100	CS0030	Servicio Extendido
CO0000008	CC41232354	41232354	Carlos Alberto Vereau Gutiérrez	CH0212	11-10-2014	Suite	100	CS0030	Servicio Extendido
CO0000009	CC18264354	18264354	Sofía Marisol Pérez Cortez	CH0213	12-10-2014	Doble	50	CS0010	Servicio Normal
CO0000010	CC17269484	17269484	Lucía Madeleine Espinoza Mendoza	CH0214	12-10-2014	Doble	50	CS0010	Servicio Normal
CO0000011	CC41232354	41232354	Carlos Alberto Vereau Gutiérrez	CH0213	13-10-2014	Doble	50	CS0020	Servicio Adicional
CO0000012	CC18264354	18264354	Sofía Marisol Pérez Cortez	CH0215	14-10-2014	Individual	30	CS0020	Servicio Adicional
CO0000013	CC47938471	47938471	Carlos Alberto Vereau Gutiérrez	CH0216	15-10-2014	Individual	30	CS0030	Servicio Extendido
CO0000014	CC17269484	17269484	Lucía Madeleine Espinoza Mendoza	CH0216	16-10-2014	Individual	30	CS0030	Servicio Extendido
CO0000015	CC43121314	43121314	Miguel Ángel Acosta Burga	CH0212	16-10-2014	Suite	100	CS0030	Servicio Extendido

CO0000016	CC47938471	47938471	Carlos Alberto Vereau Gutiérrez	CH0214	16-10-2014	Doble	50	CS0010	Servicio Normal
CO0000017	CC41232354	41232354	Carlos Alberto Vereau Gutiérrez	CH0215	16-10-2014	Individual	30	CS0010	Servicio Normal
CO0000018	CC41232354	41232354	Carlos Alberto Vereau Gutiérrez	CH0213	16-10-2014	Doble	50	CS0020	Servicio Adicional
CO0000019	CC45295837	45295837	Nuria Geraldine Alcalde Ferreira	CH0217	16-10-2014	Doble	50	CS0030	Servicio Extendido
CO0000020	CC43121314	43121314	Miguel Ángel Acosta Burga	CH0214	17-10-2014	Doble	50	CS0030	Servicio Extendido
CO0000021	CC43121314	43121314	Miguel Ángel Acosta Burga	CH0213	17-10-2014	Doble	50	CS0030	Servicio Extendido
CO0000022	CC17269484	17269484	Lucía Madeleine Espinoza Mendoza	CH0215	17-10-2014	Individual	30	CS0020	Servicio Adicional
CO0000023	CC17269484	17269484	Lucía Madeleine Espinoza Mendoza	CH0212	17-10-2014	Suite	100	CS0020	Servicio Adicional
CO0000024	CC41232425	41232425	María del Pilar Mendoza Chávez	CH0216	17-10-2014	Individual	30	CS0030	Servicio Extendido

Fuente: Elaboración propia.

c. Estructura Lógica del Proceso de Asignaciones de Ocupantes

El **Proceso de Asignaciones de Ocupantes** corresponde a las tareas de asignación de ocupantes para cada ocupación que ejecuta la empresa de servicios de hospedaje. Una ocupación tendrá cuando menos una asignación de ocupante, que es el cliente, el cual deberá ser un adulto que posee un número único de DNI; también habrá más de una asignación de ocupante, tratándose de los acompañantes del cliente. Los acompañantes de los clientes siempre deberán poseer un número único de DNI, y no habrá restricción de edad para ellos.

Tabla 13 : Estructura Lógica de Datos del Proceso de Asignaciones de Ocupantes.

ASIGNACIONES				
CodigoOcupante	**CodigoOcupacion**	**DniCliente**	**NombreCliente**	**TipoOcupante**
COOA41232354	CO0000001	41232354	Carlos Alberto Vereau Gutiérrez	Principal
COOA18264354	CO0000002	18264354	Sofía Marisol Pérez Cortez	Principal
COOA17269484	CO0000003	17269484	Lucía Madeleine Espinoza Mendoza	Principal
COOA45295837	CO0000004	45295837	Nuria Geraldine Alcalde Ferreira	Principal
COOA41232354	CO0000005	41232354	Carlos Alberto Vereau Gutiérrez	Principal
COOA45295837	CO0000006	45295837	Nuria Geraldine Alcalde Ferreira	Principal
COOA41232354	CO0000007	41232354	Carlos Alberto Vereau Gutiérrez	Principal
COOA76394756	CO0000007	76394756	Paul Miller Vereau Espinoza	Adicional
COOA77864564	CO0000007	77864564	Caroline Stephany Vereau Espinoza	Adicional
COOA41232354	CO0000008	41232354	Carlos Alberto Vereau Gutiérrez	Principal
COOA76394756	CO0000008	76394756	Paul Miller Vereau Espinoza	Adicional
COOA77864564	CO0000008	77864564	Caroline Stephany Vereau Espinoza	Adicional
COOA18264354	CO0000009	18264354	Sofía Marisol Pérez Cortez	Principal
COOA17269484	CO0000010	17269484	Lucía Madeleine Espinoza Mendoza	Principal
COOA41232354	CO0000011	41232354	Carlos Alberto Vereau Gutiérrez	Principal
COOA76394756	CO0000011	76394756	Paul Miller Vereau Espinoza	Adicional
COOA77864564	CO0000011	77864564	Caroline Stephany Vereau Espinoza	Adicional
COOA18264354	CO0000012	18264354	Sofía Marisol Pérez Cortez	Principal
COOA47938471	CO0000013	47938471	Carlos Alberto Vereau Gutiérrez	Principal

COOA76394756	CO0000013	76394756	Paul Miller Vereau Espinoza	Adicional
COOA77864564	CO0000013	77864564	Caroline Stephany Vereau Espinoza	Adicional
COOA17269484	CO0000014	17269484	Lucía Madeleine Espinoza Mendoza	Principal
COOA43121314	CO0000015	43121314	Miguel Ángel Acosta Burga	Principal
COOA47938471	CO0000016	47938471	Carlos Alberto Vereau Gutiérrez	Principal
COOA41232354	CO0000017	41232354	Carlos Alberto Vereau Gutiérrez	Principal
COOA76394756	CO0000017	76394756	Paul Miller Vereau Espinoza	Adicional
COOA41232354	CO0000018	41232354	Carlos Alberto Vereau Gutiérrez	Principal
COOA77864564	CO0000018	77864564	Caroline Stephany Vereau Espinoza	Adicional
COOA45295837	CO0000019	45295837	Nuria Geraldine Alcalde Ferreira	Principal
COOA43121314	CO0000020	43121314	Miguel Ángel Acosta Burga	Principal
COOA43121314	CO0000021	43121314	Miguel Ángel Acosta Burga	Principal
COOA76545677	CO0000021	76545677	Sebastián Miguel Acosta Puelles	Adicional
COOA17269484	CO0000022	17269484	Lucía Madeleine Espinoza Mendoza	Principal
COOA17269484	CO0000023	17269484	Lucía Madeleine Espinoza Mendoza	Principal
COOA75456567	CO0000023	75456567	Micaela Daniela Salaverry Espinoza	Adicional
COOA41232425	CO0000024	41232425	María del Pilar Mendoza Chávez	Principal

Fuente: Elaboración propia.

E. Subestructura Lógica de Datos del Proceso de Ocupaciones de Habitaciones

Para efectos académicos, se extraerá una subestructura menor de la estructura lógica de datos del Proceso de Ocupaciones de Habitaciones, presentándola de la siguiente manera:

Tabla 14 : Subestructura Lógica de Datos del Proceso de Ocupaciones de Habitaciones.

CodigoCliente	NombreCliente	CodigoHabitacion	FechaEntrada	TipoHabitacion	PrecioNoche
CC41232354	Carlos Alberto Vereau Gutiérrez	CH0212	08-10-2014	Suite	100
CC18264354	Sofía Marisol Pérez Cortez	CH0213	08-10-2014	Doble	50
CC17269484	Lucía Madeleine Espinoza Mendoza	CH0214	08-10-2014	Doble	50
CC45295837	Nuria Geraldine Alcalde Ferreira	CH0215	08-10-2014	Individual	30
CC41232354	Carlos Alberto Vereau Gutiérrez	CH0213	09-10-2014	Doble	50
CC45295837	Nuria Geraldine Alcalde Ferreira	CH0213	10-10-2014	Doble	50
CC41232354	Carlos Alberto Vereau Gutiérrez	CH0212	10-10-2014	Suite	100
CC41232354	Carlos Alberto Vereau Gutiérrez	CH0212	11-10-2014	Suite	100
CC18264354	Sofía Marisol Pérez Cortez	CH0213	12-10-2014	Doble	50
CC17269484	Lucía Madeleine Espinoza Mendoza	CH0214	12-10-2014	Doble	50
CC41232354	Carlos Alberto Vereau Gutiérrez	CH0213	13-10-2014	Doble	50
CC18264354	Sofía Marisol Pérez Cortez	CH0215	14-10-2014	Individual	30
CC47938471	Carlos Alberto Vereau Gutiérrez	CH0216	15-10-2014	Individual	30
CC17269484	Lucía Madeleine Espinoza Mendoza	CH0216	16-10-2014	Individual	30
CC43121314	Miguel Ángel Acosta Burga	CH0212	16-10-2014	Suite	100
CC47938471	Carlos Alberto Vereau Gutiérrez	CH0214	16-10-2014	Doble	50
CC41232354	Carlos Alberto Vereau Gutiérrez	CH0215	16-10-2014	Individual	30

CC41232354	Carlos Alberto Vereau Gutiérrez	CH0213	16-10-2014	Doble	50
CC45295837	Nuria Geraldine Alcalde Ferreira	CH0217	16-10-2014	Doble	50
CC43121314	Miguel Ángel Acosta Burga	CH0214	17-10-2014	Doble	50
CC43121314	Miguel Ángel Acosta Burga	CH0213	17-10-2014	Doble	50
CC17269484	Lucía Madeleine Espinoza Mendoza	CH0215	17-10-2014	Individual	30
CC17269484	Lucía Madeleine Espinoza Mendoza	CH0212	17-10-2014	Suite	100
CC41232425	María del Pilar Mendoza Chávez	CH0216	17-10-2014	Individual	30

Fuente: Elaboración propia.

F. Obtención de un Diseño de Base de Datos

a. Proceso Algorítmico ODRF

Para poder aplicar el agoritmo ODRF sobre la Subestructura Lógica de Datos del Proceso de Ocupaciones de Habitaciones, nos ayudaremos de la herramienta Microsoft SQL Server (se recomienda la versión 2014 o posterior), que es un sistema de gestión de base de datos relacional.

Posteriormente insertaremos la estructura lógica de datos en una tabla llamada Tupla, tal como se muestra a continuación en el gráfico ***Inserción de Datos en Tabla Tuplas***.

La subestructura lógica de datos está compuesta por las columnas (atributos) CodigoCliente, Nombre Cliente, CodigoHabitación, FechaEntrada, TipoHabitacion y PrecioNoche.

Las filas están compuestas por la restricción que proporciona la instancia de relación que corresponde a las 24 tuplas que se muestran en el siguiente gráfico.

Gráfico 13 : Inserción de Datos en Tabla Tuplas.

	CodigoCliente	NombreCliente	CodigoHabitacion	FechaEntrada	TipoHabitacion	PrecioNoche
1	CC41232354	Carlos Alberto Vereau Gutiérrez	CH0212	2014-10-08	Suite	100.00
2	CC18264354	Sofía Marisol Pérez Cortez	CH0213	2014-10-08	Doble	50.00
3	CC17269484	Lucía Madeleine Espinoza Mendoza	CH0214	2014-10-08	Doble	50.00
4	CC45295837	Nuria Geraldine Alcalde Ferreira	CH0215	2014-10-08	Individual	30.00
5	CC41232354	Carlos Alberto Vereau Gutiérrez	CH0213	2014-10-09	Doble	50.00
6	CC45295837	Nuria Geraldine Alcalde Ferreira	CH0213	2014-10-10	Doble	50.00
7	CC41232354	Carlos Alberto Vereau Gutiérrez	CH0212	2014-10-10	Suite	100.00
8	CC41232354	Carlos Alberto Vereau Gutiérrez	CH0212	2014-10-11	Suite	100.00
9	CC18264354	Sofía Marisol Pérez Cortez	CH0213	2014-10-12	Doble	50.00
10	CC17269484	Lucía Madeleine Espinoza Mendoza	CH0214	2014-10-12	Doble	50.00
11	CC41232354	Carlos Alberto Vereau Gutiérrez	CH0213	2014-10-13	Doble	50.00
12	CC18264354	Sofía Marisol Pérez Cortez	CH0215	2014-10-14	Individual	30.00
13	CC47938471	Carlos Alberto Vereau Gutiérrez	CH0216	2014-10-15	Individual	30.00
14	CC17269484	Lucía Madeleine Espinoza Mendoza	CH0216	2014-10-16	Individual	30.00
15	CC43121314	Miguel Ángel Acosta Burga	CH0212	2014-10-16	Suite	100.00
16	CC47938471	Carlos Alberto Vereau Gutiérrez	CH0214	2014-10-16	Doble	50.00
17	CC41232354	Carlos Alberto Vereau Gutiérrez	CH0215	2014-10-16	Individual	30.00
18	CC41232354	Carlos Alberto Vereau Gutiérrez	CH0213	2014-10-16	Doble	50.00
19	CC45295837	Nuria Geraldine Alcalde Ferreira	CH0217	2014-10-16	Doble	50.00
20	CC43121314	Miguel Ángel Acosta Burga	CH0214	2014-10-17	Doble	50.00
21	CC43121314	Miguel Ángel Acosta Burga	CH0213	2014-10-17	Doble	50.00
22	CC17269484	Lucía Madeleine Espinoza Mendoza	CH0215	2014-10-17	Individual	30.00
23	CC17269484	Lucía Madeleine Espinoza Mendoza	CH0212	2014-10-17	Suite	100.00
24	CC41232425	María del Pilar Mendoza Chávez	CH0216	2014-10-17	Individual	30.00

Fuente: Elaboración propia.

1) Definición de Relaciones Binarias (n-aria)

Después de insertar la estructura lógica de datos en la tabla Tuplas, necesitaremos definir las relaciones binarias, de cuyas tuplas se obtendrá las Cardinalidades de Proyecciones, para que a éstas se les aplique las Restricciones Lógicas Funcionales.

En consecuencia, para definir las relaciones binarias, se aplicará el Teorema 1 al conjunto de atributos de la estructura lógica de datos.

El conjunto de atributos, según la muestra, es el siguiente:

- CodigoCliente.
- NombreCliente.
- CodigoHabitacion.
- FechaEntrada.
- TipoHabitacion.
- PrecioNoche.

Para visualizar la forma en la que se deberá definir las relaciones binarias, aplicando el Teorema 1 sobre el conjunto de atributos listados líneas arriba, remitirse al **Anexo 1**.

2) Obtención de Cardinalidades de Proyecciones de las Tuplas de las Relaciones Binarias (n-aria)

En cada relación binaria habrá dos conjuntos de atributos **A** y **B**, los cuales forman parte de una Relación **R**.

Para aplicar las restricciones lógicas sobre el conjunto de datos, sostenido en cada relación binaria, utilizaremos el concepto de **Proyección** del **Álgebra Relacional**, denotado por el símbolo **Π**, el cual será aplicado tanto al atributo **A** (**ΠA(R)**) como al atributo **B** (**ΠB (R)**), así como también al atributo **A** unido al atributo **B** (**ΠAB (R)**).

De tal forma que se obtendrá las respectivas cardinalidades de las proyecciones necesarias (**|ΠA(R)|**, **|ΠB(R)|** y **|ΠAB(R)|**) de cada relación binaria, sobre las que se aplicará las restricciones lógicas funcionales.

Para visualizar la forma en la que se deberá obtener las cardinalidades de las proyecciones de cada relación binaria, remitirse al **Anexo 2**.

3) Aplicación de Restricciones Lógicas sobre las Cardinalidades de Proyecciones de cada Relación Binaria

Al obtener las cardinalidades de las proyecciones necesarias (**|ΠA(R)|**, **|ΠB(R)|** y **|ΠAB(R)|**) de cada relación binaria sostenida sobre los atributos **A** y **B**, se deberá aplicar las restricciones lógicas funcionales sobre ellas.

Para visualizar la forma en la que se deberá aplicar las restricciones lógicas funcionales, remitirse al **Anexo 3**.

4) Lista de Dependencias de Relación Funcional (Válidas)

Después de que se aplique las restricciones lógicas funcionales, obtendremos las dependencias de relación funcional (válidas), las cuales se distinguirán con los símbolos → (Función Sobreyectiva) y ↠ (Función Biyectiva).

Para visualizar la lista completa de las dependencias de relación funcional, remitirse al **Anexo 4**.

5) Lista de Dependencias de Relación No Funcional (No Válidas)

Después de que se aplique las restricciones lógicas funcionales, obtendremos las dependencias de relación no funcional (no válidas), las cuales se distinguirán con el símbolo ⊸ (Relación Multivaluada).

Para visualizar la lista completa de las dependencias de relación no funcional, remitirse al **Anexo 5**.

b. Algoritmo RIDRF

Las reglas de inferencias aplicadas a la lista de dependencias de relación funcional que nos proporciona el algoritmo ODRF, nos permitirán eliminar y simplificar las dependencias, a fin de que solamente queden las dependencias necesarias para

obtener tanto un diseño de base de datos con tablas normalizadas como un diseño de base de datos con tablas no normalizadas.

Siendo que ambos diseños deberán mantener una integridad de datos correspondiente a las reglas de la lógica de negocio.

1) Aplicación de Reglas de Inferencia sobre Dependencias de relación funcional

El procedimiento que se utilizará para aplicar las reglas de inferencia, tal como se muestra en el **Anexo 6**, es el siguiente:

(1) A la lista de dependencias de relación funcional que nos proporciona el algoritmo ODRF, se le aplicará el Teorema 2.2, el cual aplicará la regla de inferencia correspondiente a la **Descomposición de Dependencias de Relación Funcional**. El resultado de la aplicación del Teorema 2.2, nos dará una nueva lista de dependencias de relación funcional.

(2) A partir del resultado de la aplicación del Teorema 2.2, se aplicará el Teorema 1.1, el cual aplicará la regla de inferencia que **halla las Dependencias de Relación Funcional Reflexivas**. Cada una de las dependencias halladas, deberán ser eliminadas. El resultado de la aplicación del Teorema 1.1, nos dará una nueva lista de dependencias de relación funcional.

(3) A partir del resultado de la aplicación del Teorema 1.1, se aplicará el Teorema 1.2, el cual aplicará la regla de inferencia que **halla las Dependencias con Biyección Parcial**. Cada una de las dependencias halladas, deberán ser eliminadas. El resultado de la aplicación del Teorema 1.2, nos dará una nueva lista de dependencias de relación funcional.

(4) A partir del resultado de la aplicación del Teorema 1.2, se aplicará el Teorema 1.3, el cual aplicará la regla de inferencia que **halla las Dependencias con Sobreyección en el Lado Izquierdo**. Cada una de las dependencias halladas, deberán ser eliminadas. El resultado de la aplicación del Teorema 1.3, nos dará una nueva lista de dependencias de relación funcional.

Dependencias Normalizadas: En esta parte el procedimiento se bifurca hacia la obtención de dependencias normalizadas:

(5) A partir del resultado de la aplicación del Teorema 1.3, se aplicará el Teorema 1.4, el cual aplicará la regla de inferencia que **halla las Dependencias con Sobreyección Parcial**. Cada una de las dependencias halladas, deberán ser eliminadas. El resultado de la aplicación del Teorema 1.4, nos dará una nueva lista de dependencias de relación funcional en proceso de normalización.

(6) A partir del resultado de la aplicación del Teorema 1.4, se aplicará el Teorema 2.3, que nos dará el resultado de la aplicación del Teorema 2.3; así como también el Teorema 2.4, que nos dará el resultado de la aplicación del Teorema 2.4. Ambos teoremas aplicarán la regla de inferencia correspondiente a la **Composición de Dependencias de Relación Funcional**, y nos permitirán obtener una nueva lista de dependencias de relación funcional en proceso de normalización.

(7) A partir del resultado de la aplicación de los Teoremas 2.3 y 2.4, se aplicará el Teorema 2.1, el cual aplicará la regla de inferencia que **halla las Dependencias con Sobreyección en el Lado Derecho**. Cada una de las dependencias halladas, deberán ser simplificadas. El resultado de la aplicación del Teorema 2.1, nos dará una nueva lista de dependencias de relación funcional ya normalizadas.

(8) A partir del resultado de la aplicación del Teorema 2.1, obtendremos la lista de dependecias de relación funcional que conformarán el **Diseño de Base de Datos para las Tablas Normalizadas**.

Dependencias No Normalizadas: En esta parte el procedimiento se bifurca hacia la obtención de dependencias no normalizadas:

(9) A partir del resultado de la aplicación del Teorema 1.3, se aplicará el Teorema 2.3, que nos dará el resultado de la aplicación del Teorema 2.3, y el cual aplicará la regla de inferencia correspondiente a la **Composición de Dependencias de Relación Funcional**. Dicho resultado nos permitirá

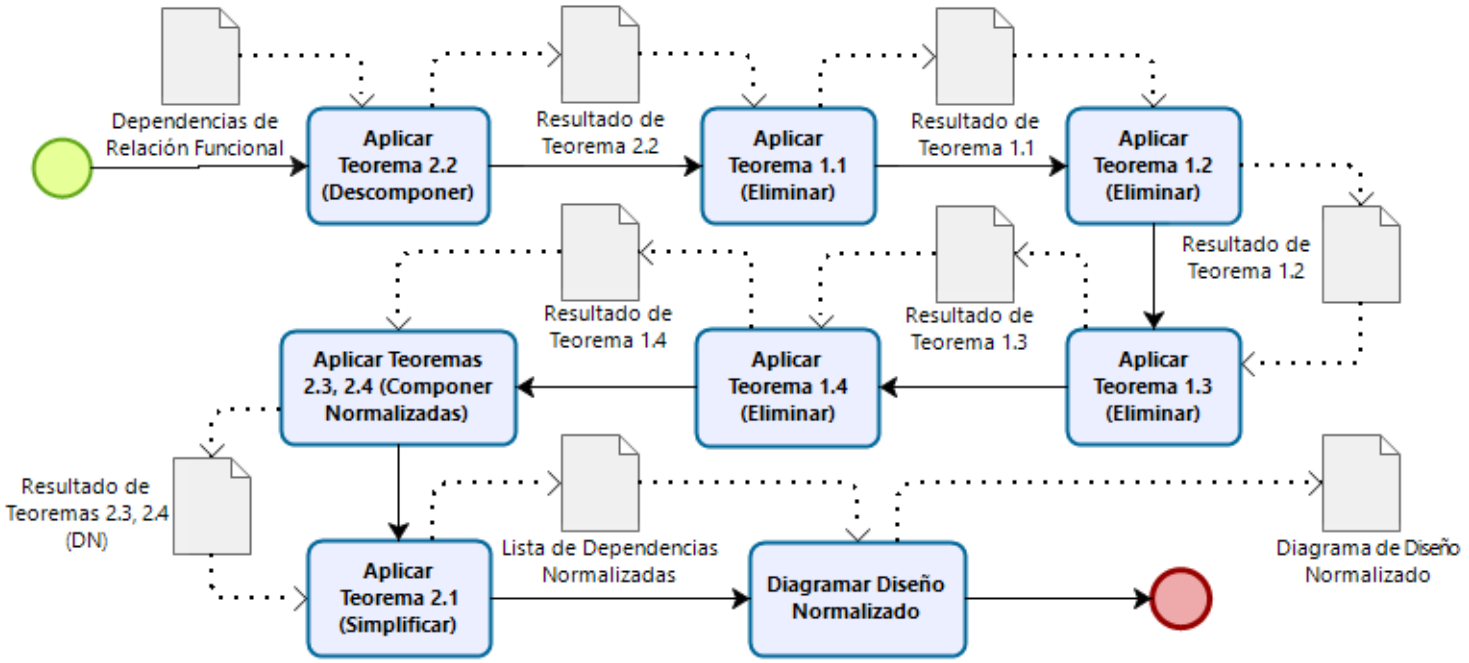
obtener una nueva lista de dependencias de relación funcional no normalizadas.

(10) A partir del resultado de la aplicación del Teorema 2.3, obtendremos la lista de dependecias de relación funcional que conformarán el **Diseño de Base de Datos para las Tablas No Normalizadas**.

2) Diseño para Tablas Normalizadas

El proceso para la obtención de las dependencias de relación funcional para el diseño de tablas Normalizadas es el siguiente:

Gráfico 14 : Diagrama de Proceso para Obtener Dependencias de Relación Funcional Normalizadas.



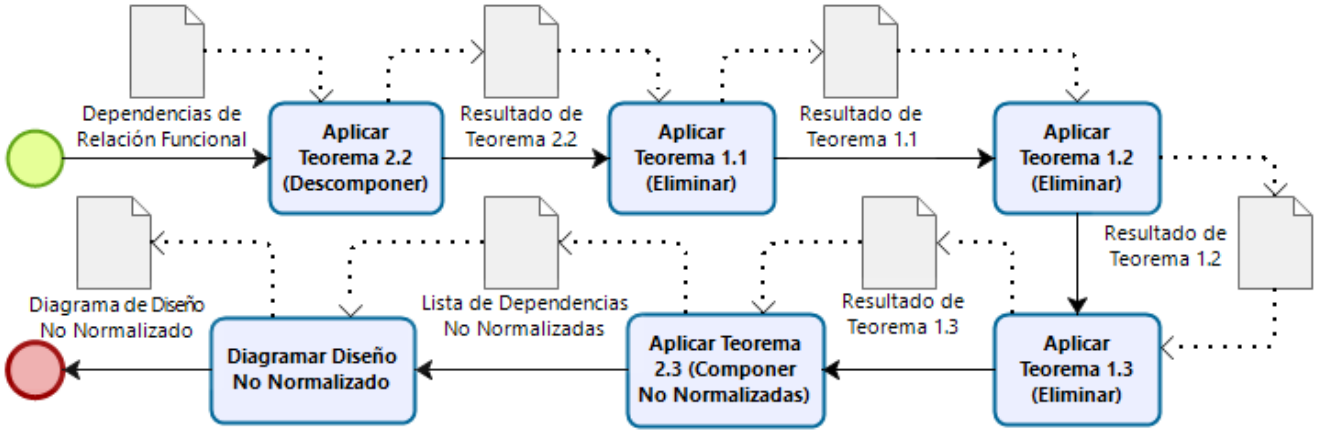
Fuente: Elaboración propia.

Para visualizar la lista de Dependencias de Relación Funcional que es parte del Diseño de Tablas Normalizadas, remitirse al **Anexo 7**.

3) Lista de Dependencias de Relación No Funcional para Tablas No Normalizadas

El proceso para la obtención de las dependencias de relación funcional para el diseño de tablas No Normalizadas es el siguiente:

Gráfico 15 : Diagrama de Proceso para Obtener Dependencias de Relación Funcional No Normalizadas.

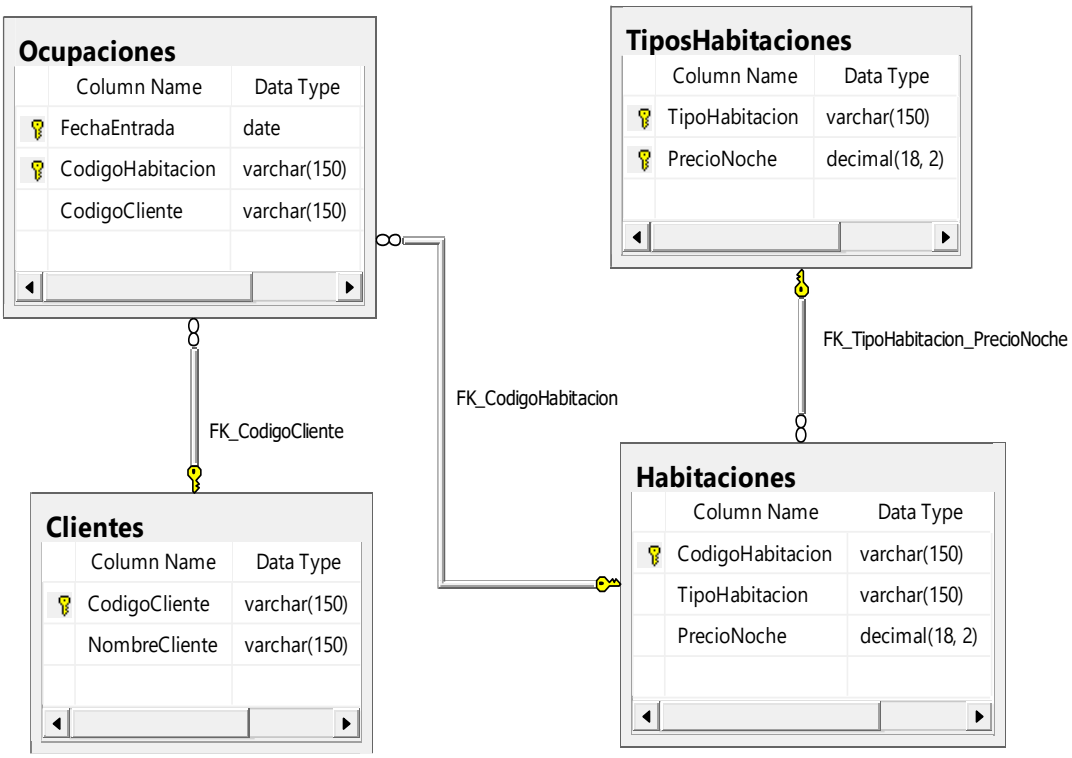


Fuente: Elaboración propia.

Para visualizar la lista de Dependencias de Relación Funcional que es parte del Diseño de Tablas No Normalizadas, remitirse al **Anexo 8**.

c. Diagrama de Diseño de Base de Datos para Tablas Normalizadas

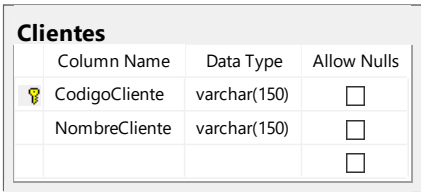
Gráfico 16 : Diagrama de Diseño de Base de Datos para Tablas Normalizadas.



Fuente: Elaboración propia.

1) Clientes

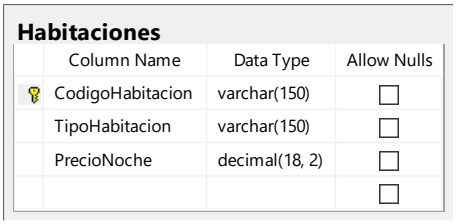
Gráfico 17 : Diagrama de Tabla Clientes.



Fuente: Elaboración propia.

2) Habitaciones

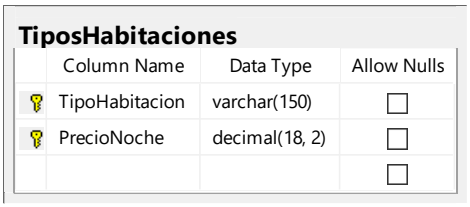
Gráfico 18 : Diagrama de Tabla Habitaciones.



Fuente: Elaboración propia.

3) TiposHabitaciones

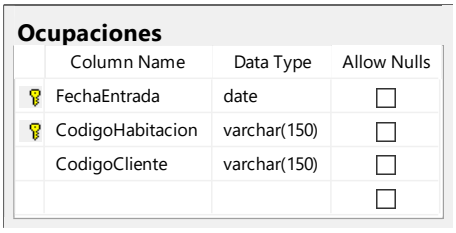
Gráfico 19 : Diagrama de Tabla TiposHabitaciones.



Fuente: Elaboración propia.

4) Ocupaciones

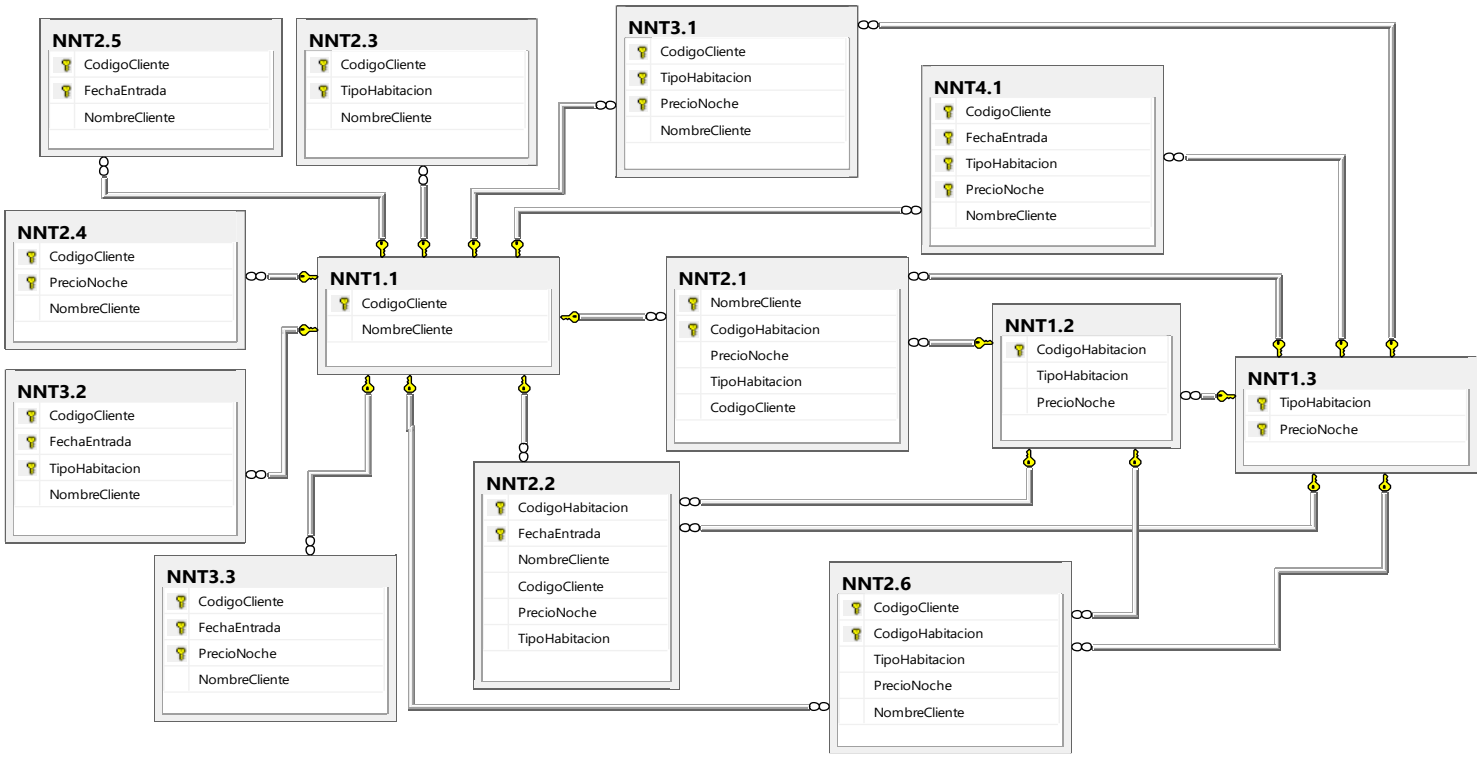
Gráfico 20 : Diagrama de Tabla Ocupaciones.



Fuente: Elaboración propia.

d. Diagrama de Diseño de Base de Datos para Tablas No Normalizadas

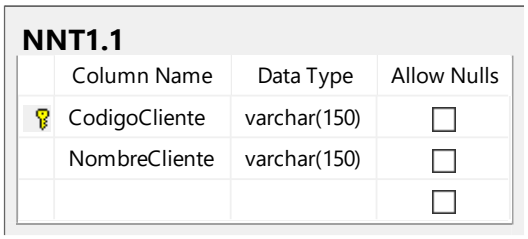
Gráfico 21 : Diagrama de Diseño de Base de Datos para Tablas No Normalizadas.



Fuente: Elaboración propia.

1) **NNT1.1**

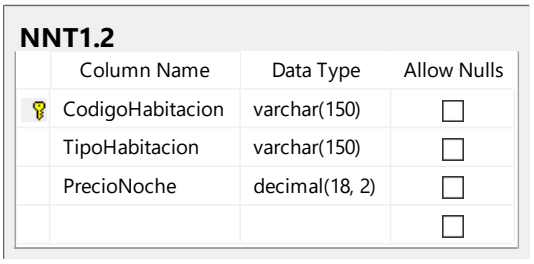
Gráfico 22 : Diagrama de Tabla NNT1.1.



Fuente: Elaboración propia.

2) **NNT1.2**

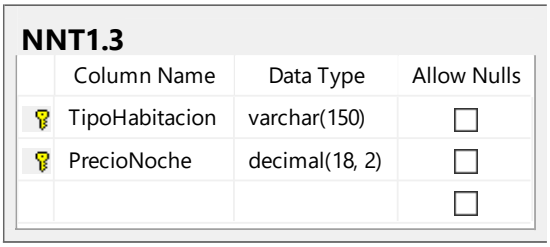
Gráfico 23 : Diagrama de Tabla NNT1.2.



Fuente: Elaboración propia.

3) **NNT1.3**

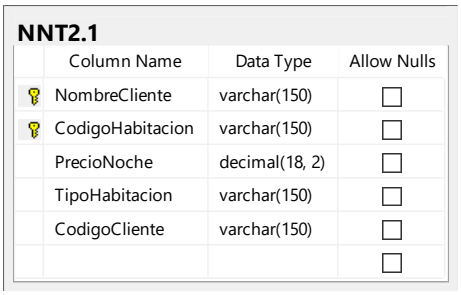
Gráfico 24 : Diagrama de Tabla NNT1.3.



Fuente: Elaboración propia.

4) **NNT2.1**

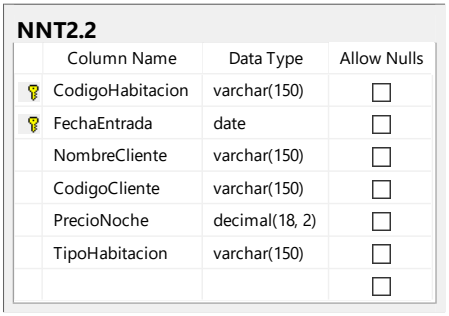
Gráfico 25 : Diagrama de Tabla NNT2.1.



Fuente: Elaboración propia.

5) **NNT2.2**

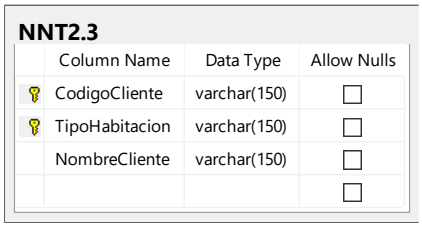
Gráfico 26 : Diagrama de Tabla NNT2.2.



Fuente: Elaboración propia.

6) **NNT2.3**

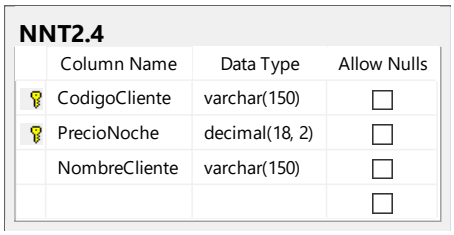
Gráfico 27 : Diagrama de Tabla NNT2.3.



Fuente: Elaboración propia.

7) **NNT2.4**

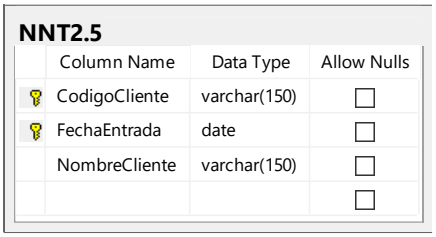
Gráfico 28 : Diagrama de Tabla NNT2.4.



Fuente: Elaboración propia.

8) **NNT2.5**

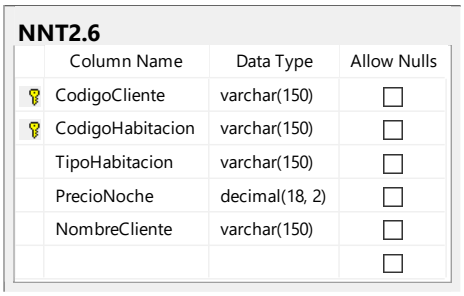
Gráfico 29 : Diagrama de Tabla NNT2.5.



Fuente: Elaboración propia.

9) **NNT2.6**

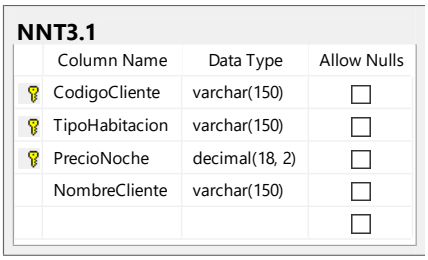
Gráfico 30 : Diagrama de Tabla NNT2.6.



Fuente: Elaboración propia.

10) NNT3.1

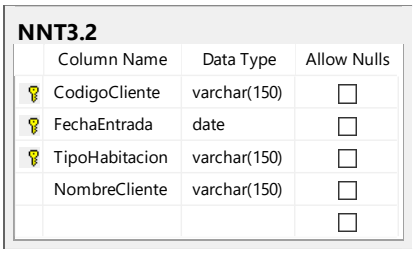
Gráfico 31 : Diagrama de Tabla NNT3.1.



Fuente: Elaboración propia.

11) NNT3.2

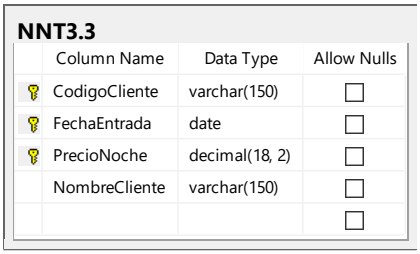
Gráfico 32 : Diagrama de Tabla NNT3.2.



Fuente: Elaboración propia.

12) NNT3.3

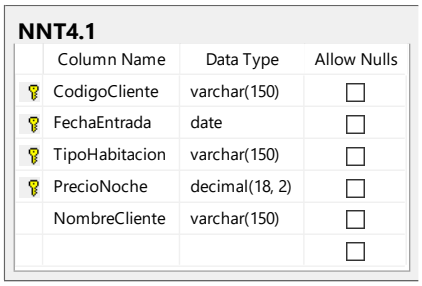
Gráfico 33 : Diagrama de Tabla NNT3.3.



Fuente: Elaboración propia.

13) NNT4.1

Gráfico 34 : Diagrama de Tabla NNT4.1.



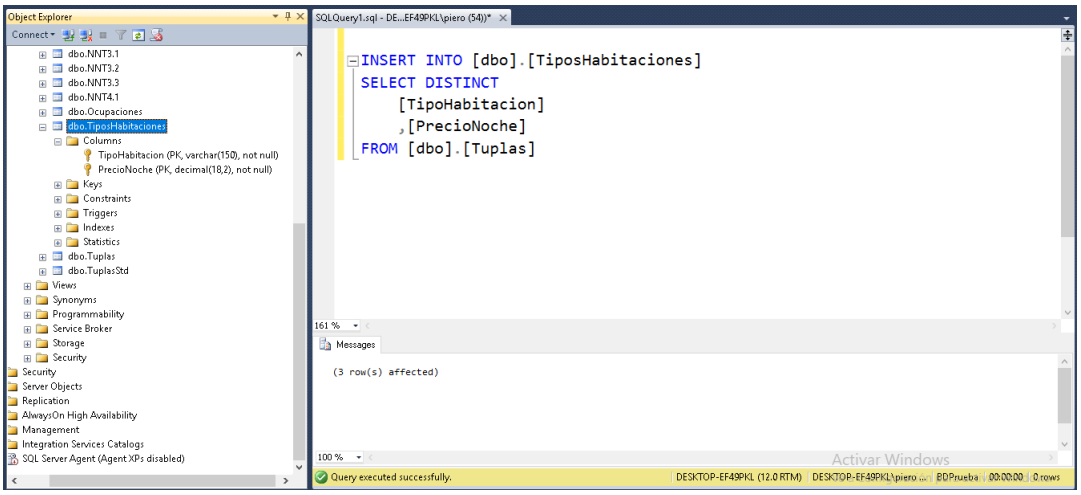
Fuente: Elaboración propia.

3. [OE3] Evaluar la Integridad de Datos del Diseño de Base de Datos creado

A. Inserción de Tuplas en Tablas Normalizadas

a. TiposHabitaciones

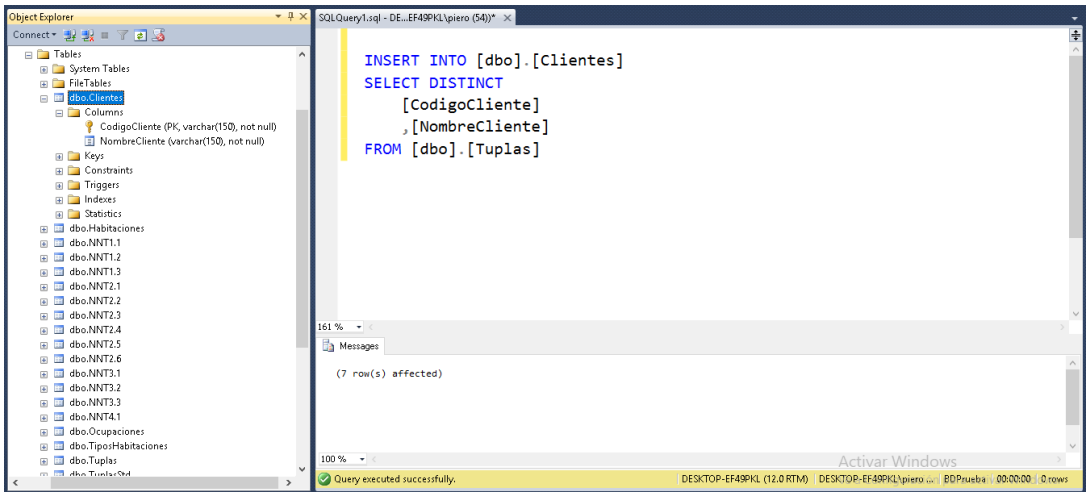
Gráfico 35 : Inserción de Tuplas en Tabla TiposHabitaciones.



Fuente: Elaboración propia.

b. Clientes

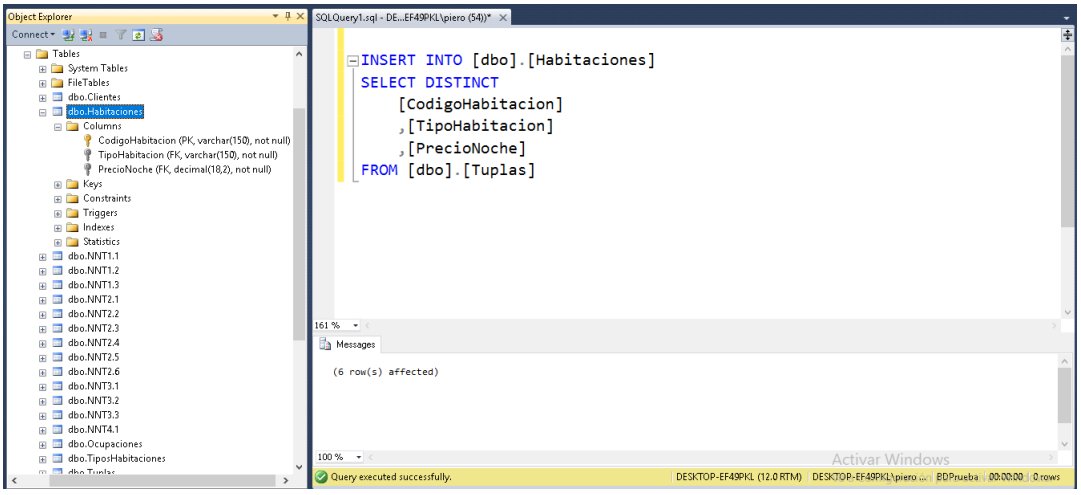
Gráfico 36 : Inserción de Tuplas en Tabla Clientes.



Fuente: Elaboración propia.

c. Habitaciones

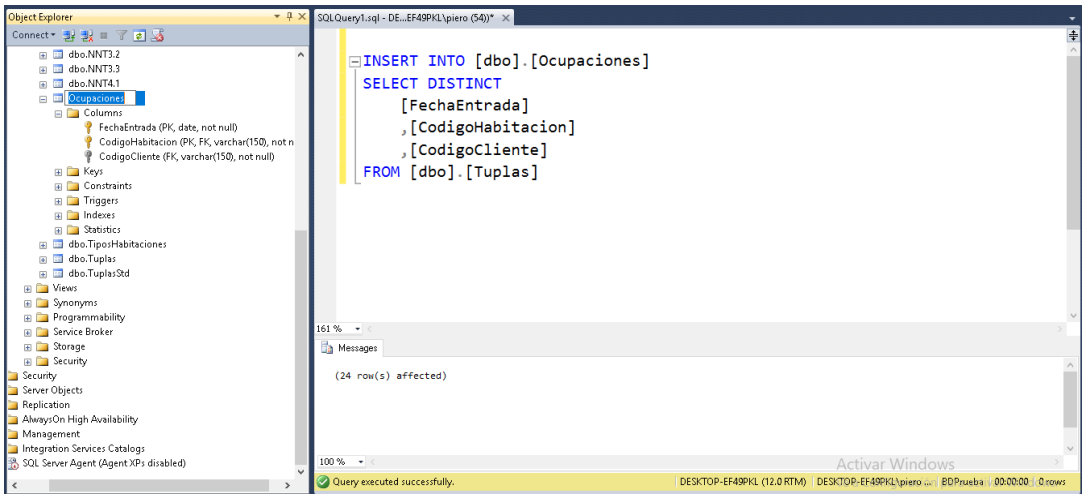
Gráfico 37 : Inserción de Tuplas en Tabla Habitaciones.



Fuente: Elaboración propia.

d. Ocupaciones

Gráfico 38 : Inserción de Tuplas en Tabla Ocupaciones.

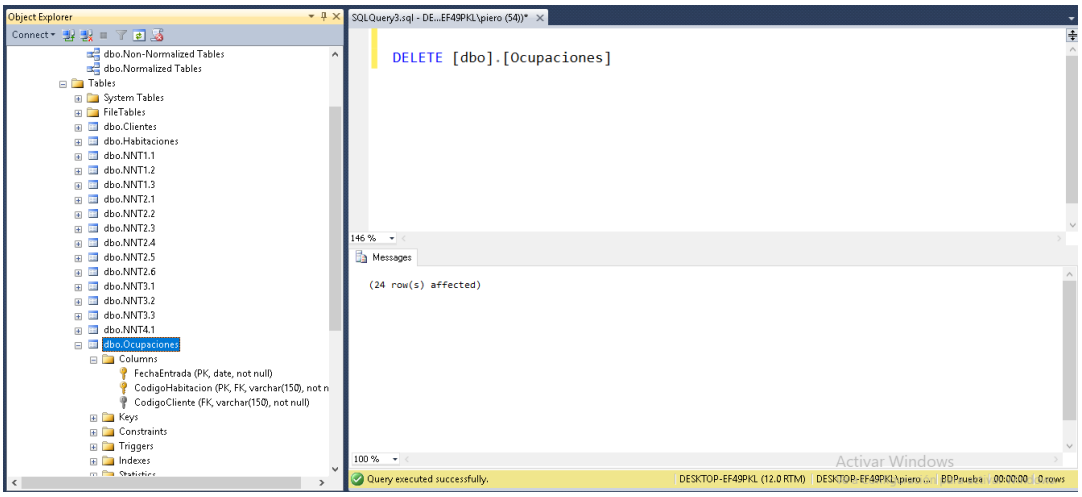


Fuente: Elaboración propia.

B. Eliminación de Tuplas en Tablas Normalizadas

a. Ocupaciones

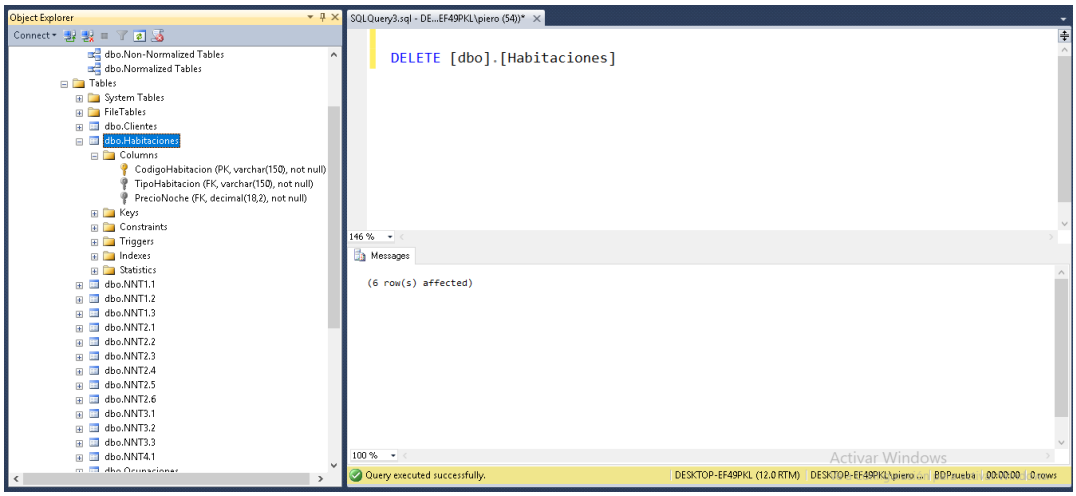
Gráfico 39 : Eliminación de Tuplas en Tabla Ocupaciones.



Fuente: Elaboración propia.

b. Habitaciones

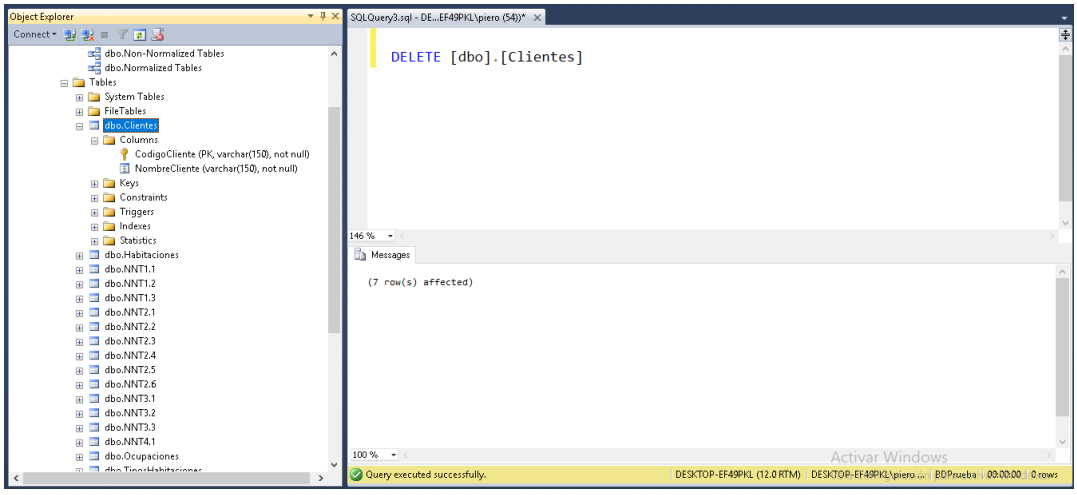
Gráfico 40 : Eliminación de Tuplas en Tabla Habitaciones.



Fuente: Elaboración propia.

c. Clientes

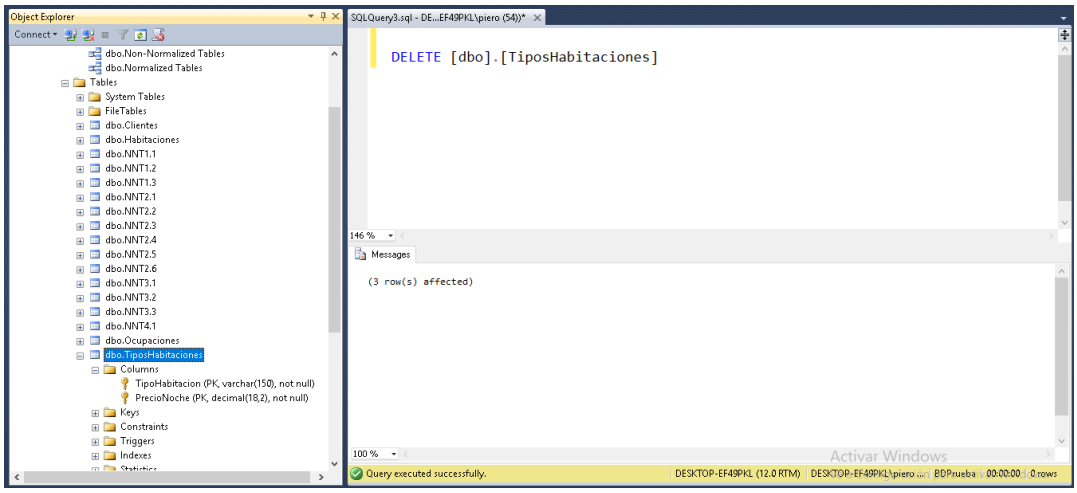
Gráfico 41 : Eliminación de Tuplas en Tabla Clientes.



Fuente: Elaboración propia.

d. TiposHabitaciones

Gráfico 42 : Eliminación de Tuplas en Tabla TipoHabitaciones.

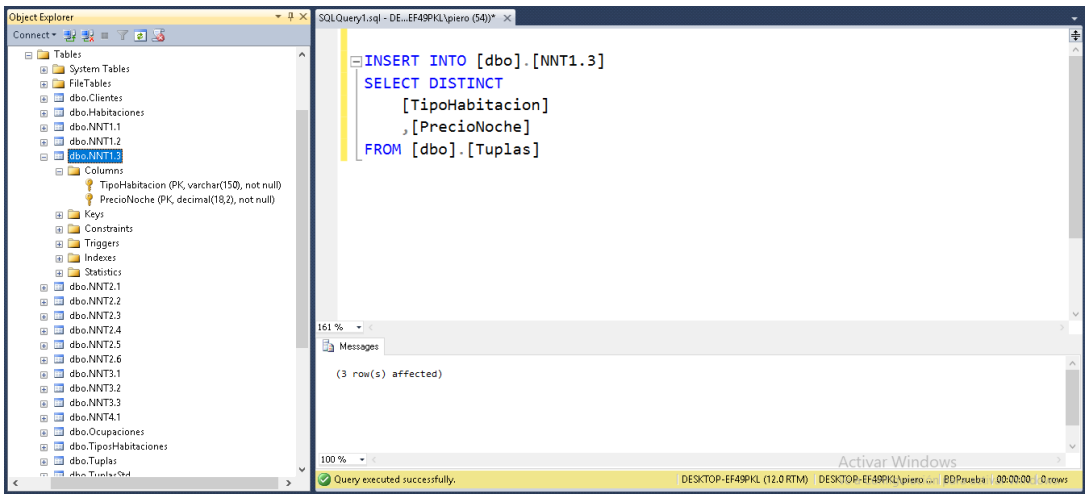


Fuente: Elaboración propia.

C. Inserción de Tuplas en Tablas No Normalizadas

a. NNT1.3

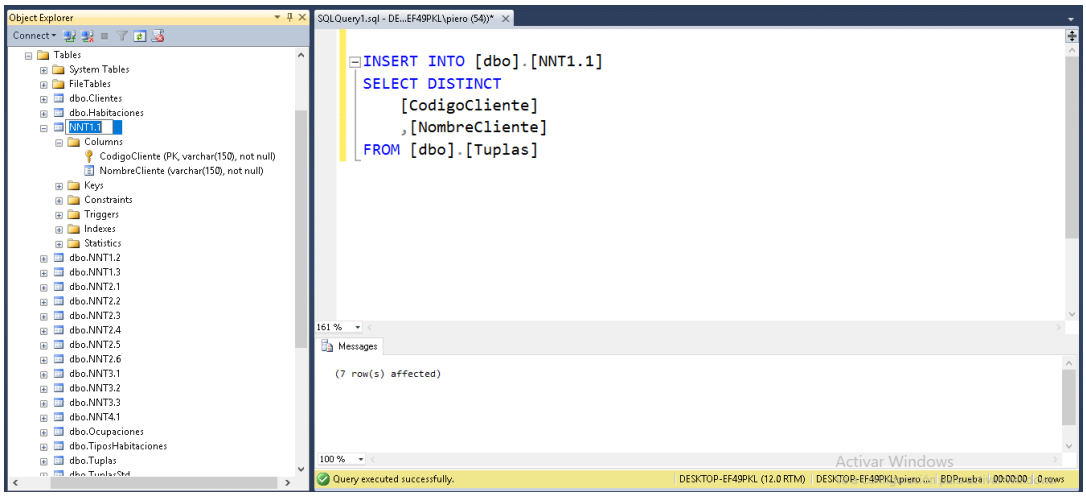
Gráfico 43 : Inserción de Tuplas en Tabla NNT1.3.



Fuente: Elaboración propia.

b. NNT1.1

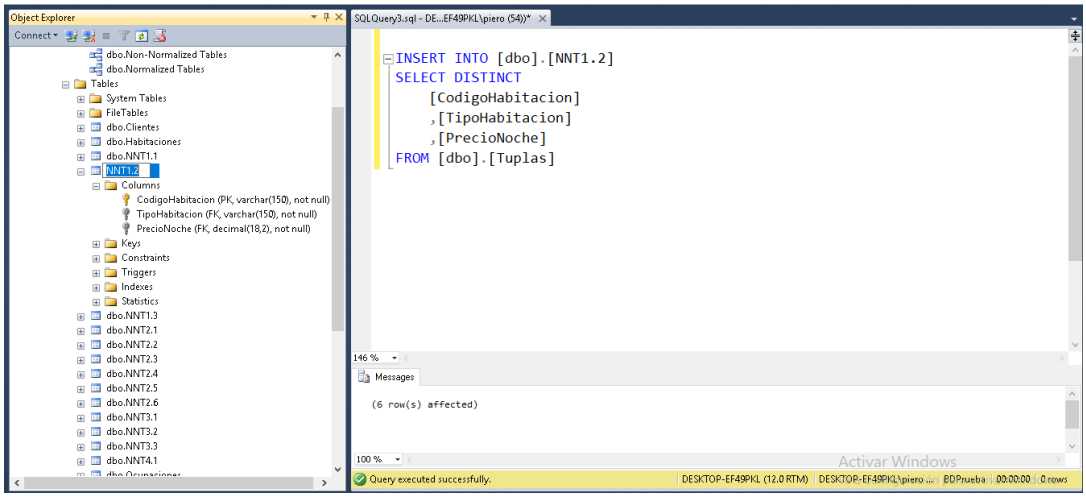
Gráfico 44 : Inserción de Tuplas en Tabla NNT1.1.



Fuente: Elaboración propia.

c. NNT1.2

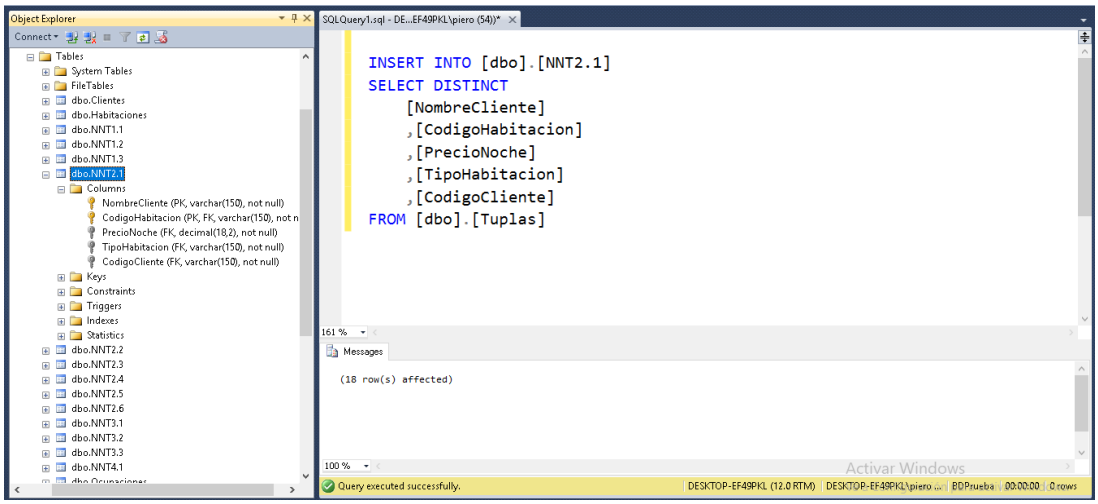
Gráfico 45 : Inserción de Tuplas en Tabla NNT1.2.



Fuente: Elaboración propia.

d. NNT2.1

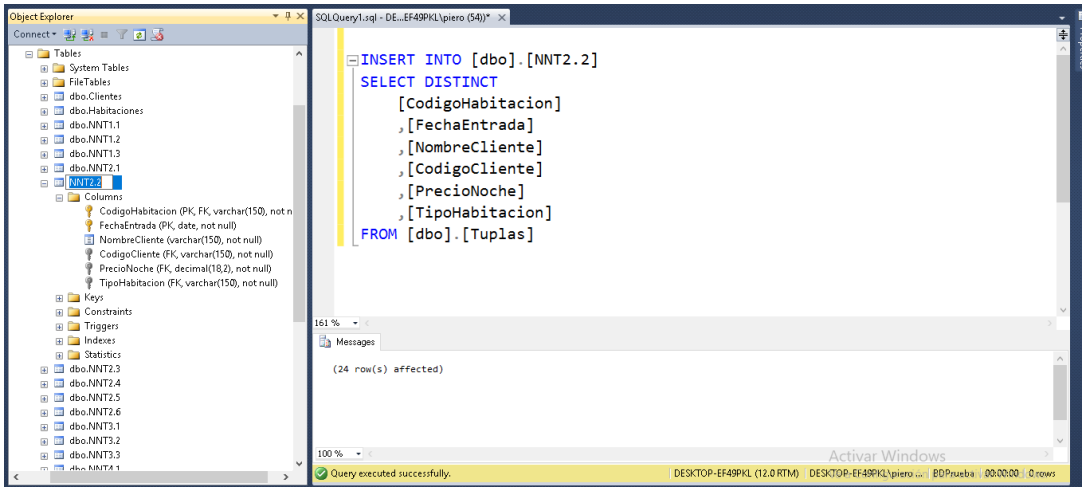
Gráfico 46 : Inserción de Tuplas en Tabla NNT2.1.



Fuente: Elaboración propia.

e. NNT2.2

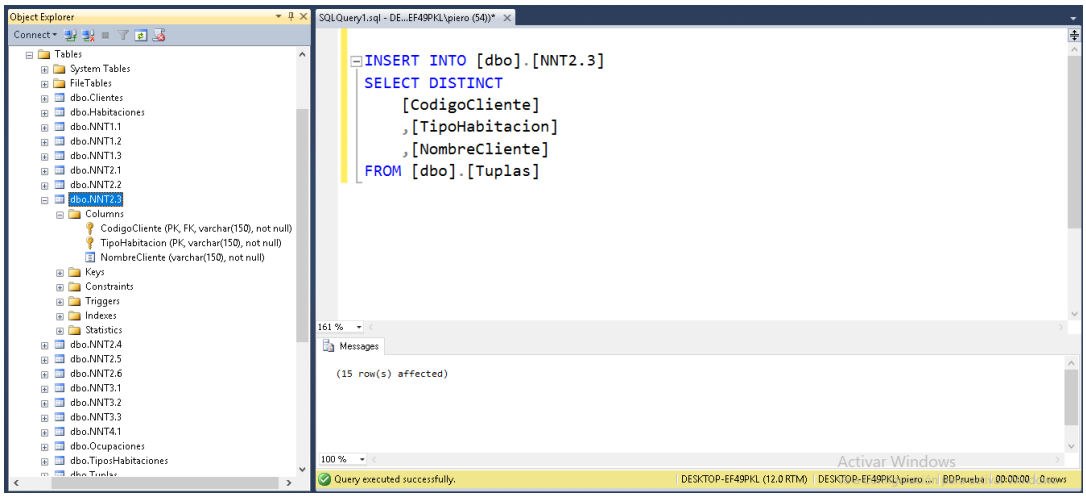
Gráfico 47 : Inserción de Tuplas en Tabla NNT2.2.



Fuente: Elaboración propia.

f. NNT2.3

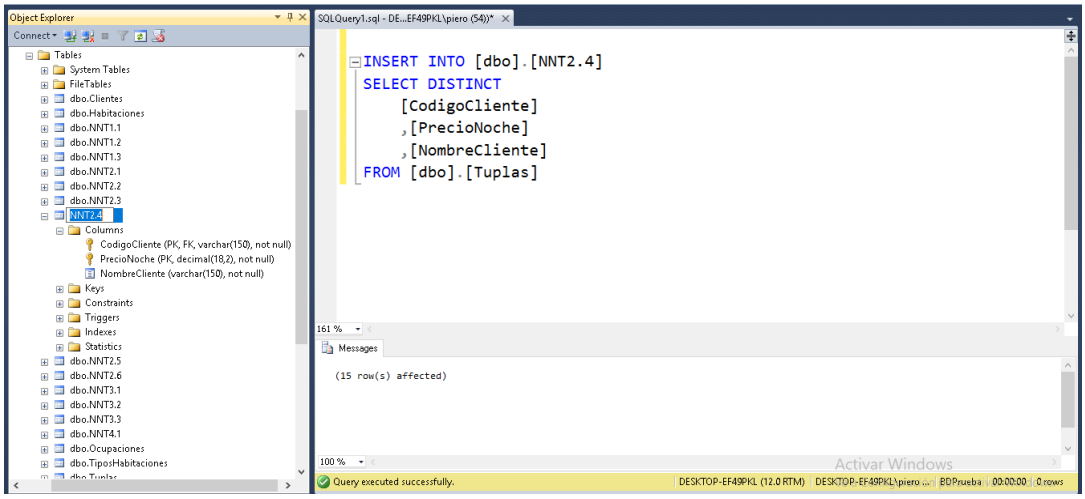
Gráfico 48 : Inserción de Tuplas en Tabla NNT2.3.



Fuente: Elaboración propia.

g. NNT2.4

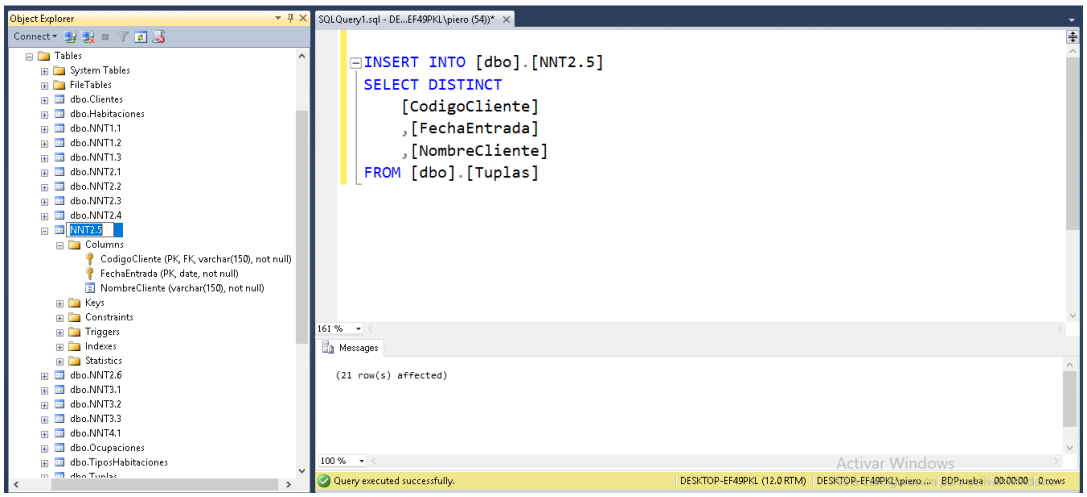
Gráfico 49 : Inserción de Tuplas en Tabla NNT2.4.



Fuente: Elaboración propia.

h. NNT2.5

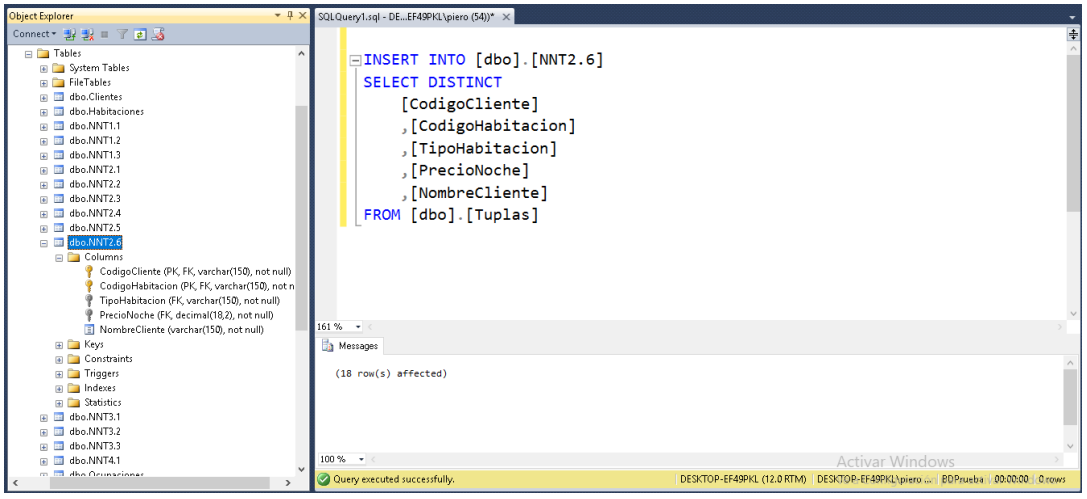
Gráfico 50 : Inserción de Tuplas en Tabla NNT2.5.



Fuente: Elaboración propia.

i. NNT2.6

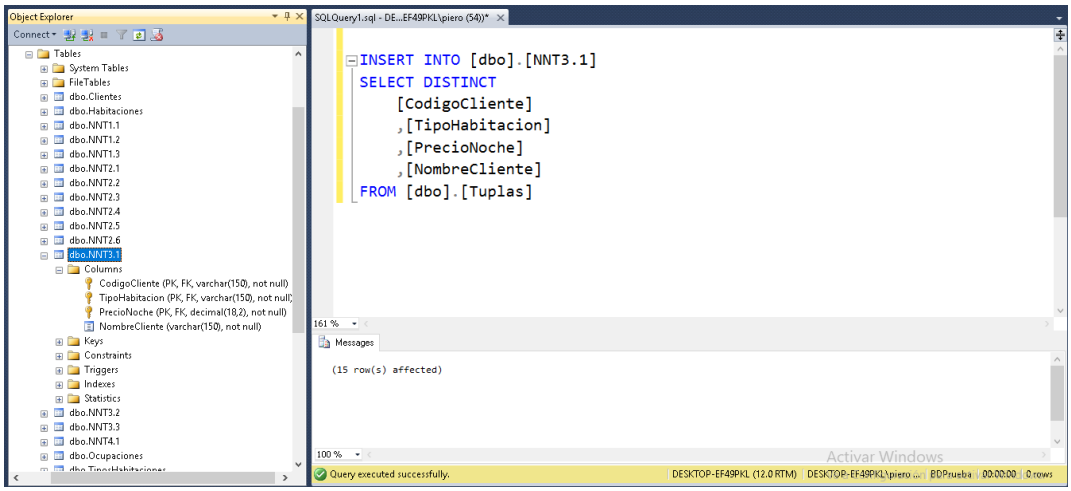
Gráfico 51 : Inserción de Tuplas en Tabla NNT2.6.



Fuente: Elaboración propia.

j. NNT3.1

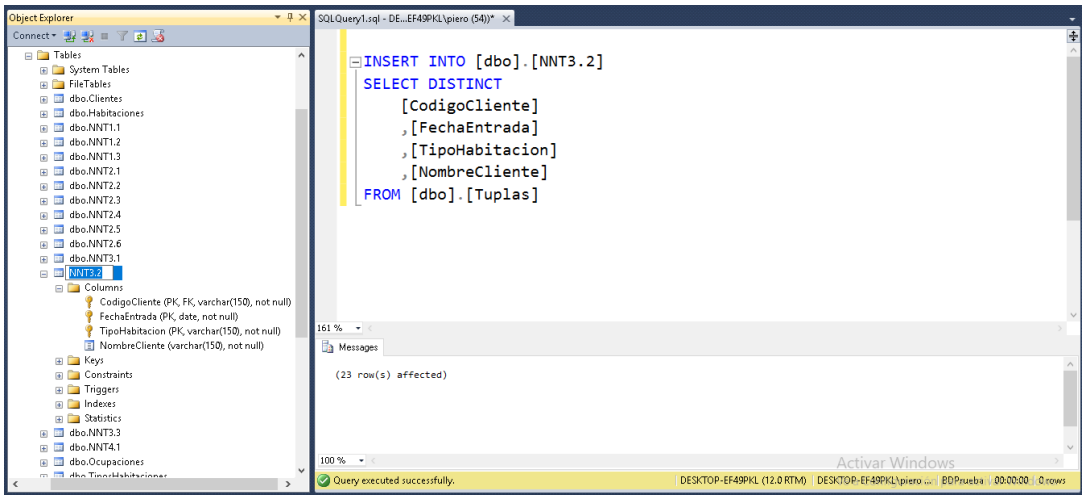
Gráfico 52 : Inserción de Tuplas en Tabla NNT3.1.



Fuente: Elaboración propia.

k. NNT3.2

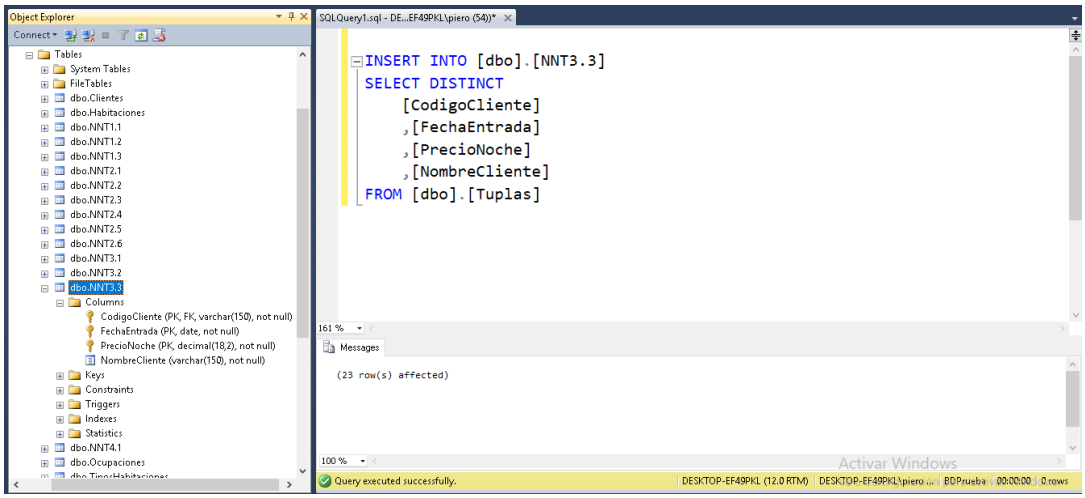
Gráfico 53 : Inserción de Tuplas en Tabla NNT3.2.



Fuente: Elaboración propia.

l. NNT3.3

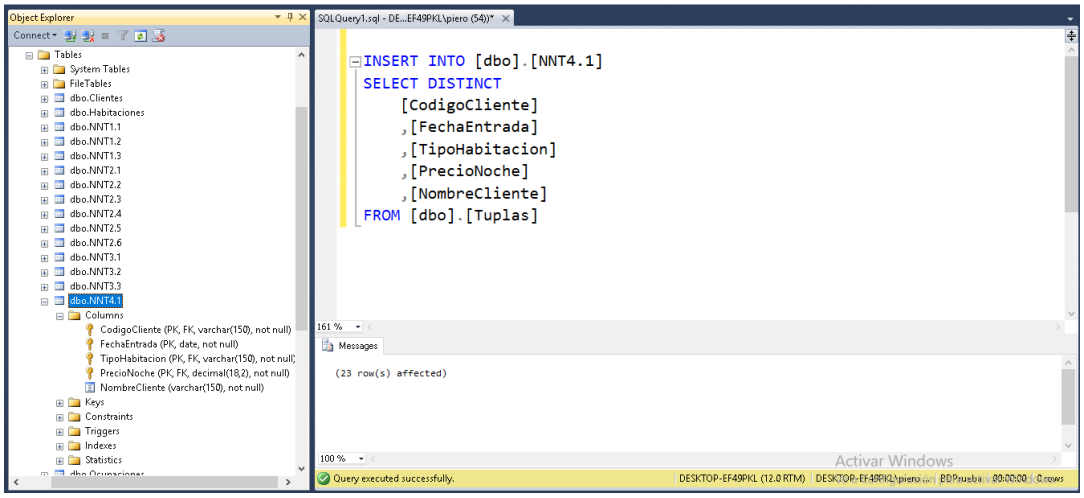
Gráfico 54 : Inserción de Tuplas en Tabla NNT3.3.



Fuente: Elaboración propia.

m. NNT4.1

Gráfico 55 : Inserción de Tuplas en Tabla NNT4.1.

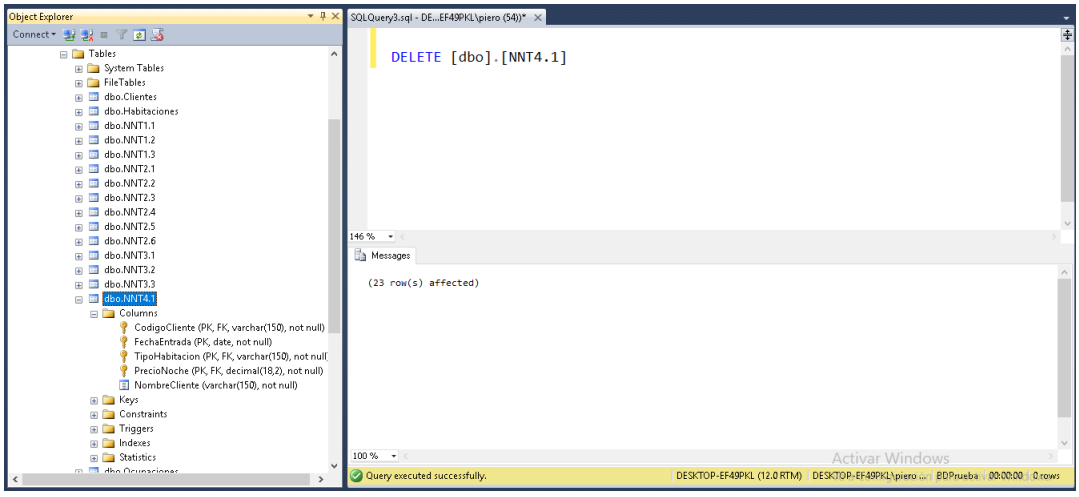


Fuente: Elaboración propia.

D. Eliminación de Tuplas en Tablas No Normalizadas

a. NNT4.1

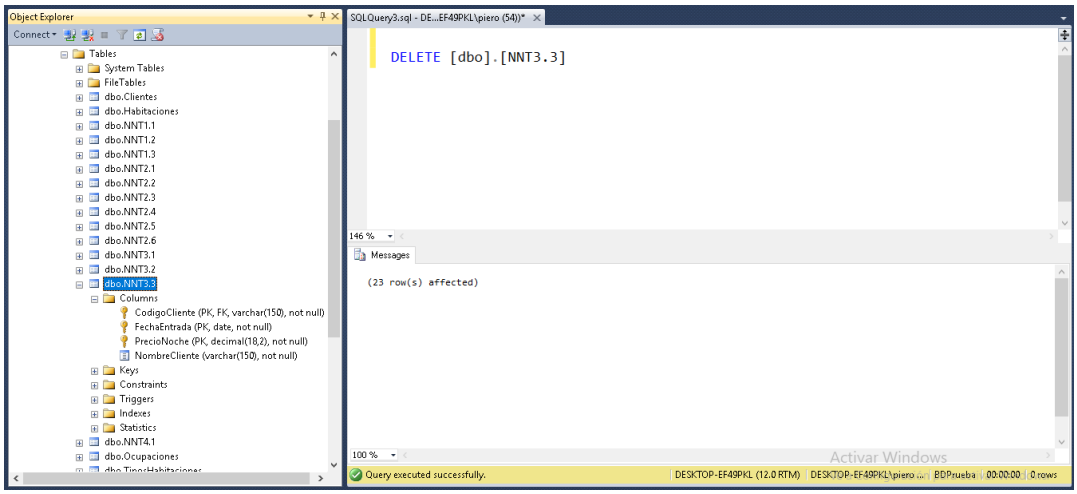
Gráfico 56 : Eliminación de Tuplas en Tabla NNT4.1.



Fuente: Elaboración propia.

b. NNT3.3

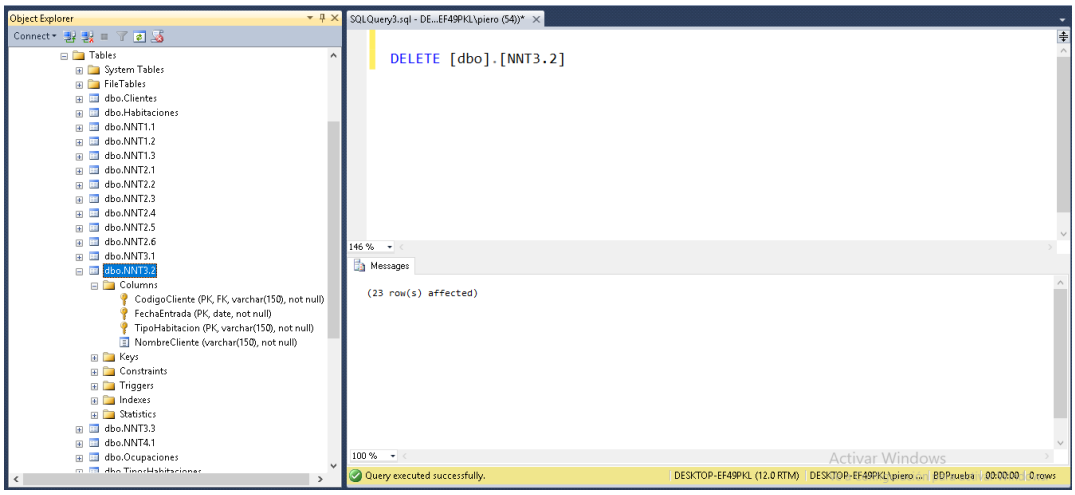
Gráfico 57 : Eliminación de Tuplas en Tabla NNT3.3.



Fuente: Elaboración propia.

c. NNT3.2

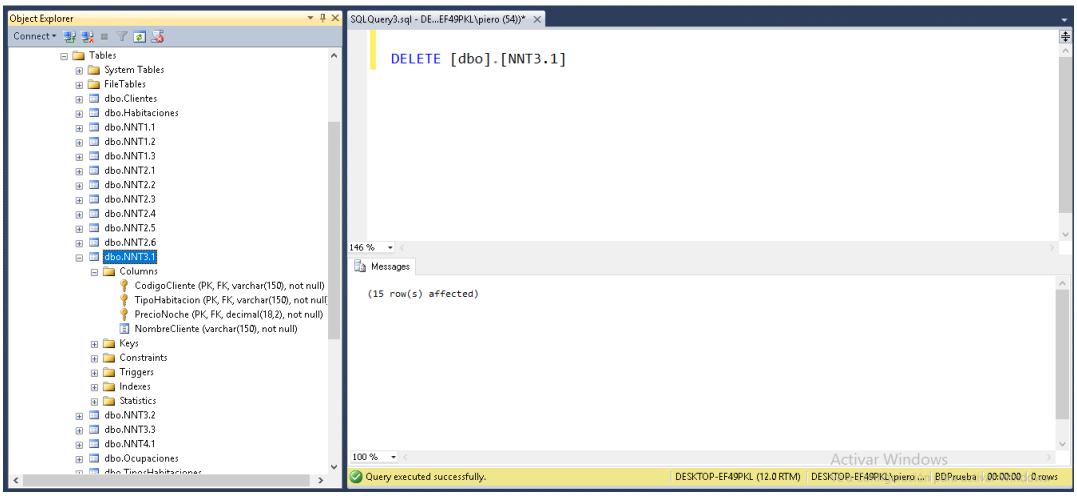
Gráfico 58 . Eliminación de Tuplas en Tabla NNT3.2.



Fuente: Elaboración propia.

d. NNT3.1

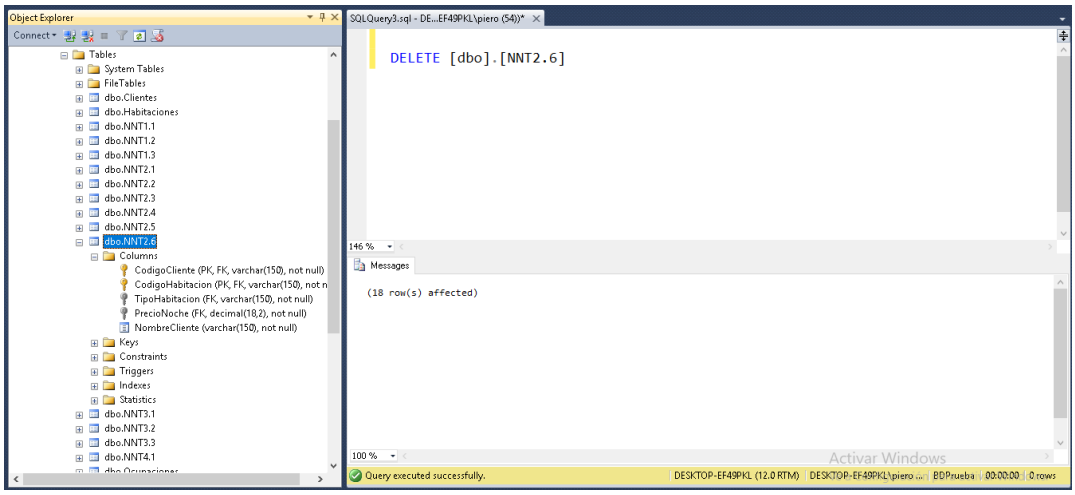
Gráfico 59 : Eliminación de Tuplas en Tabla NNT3.1.



Fuente: Elaboración propia.

e. NNT2.6

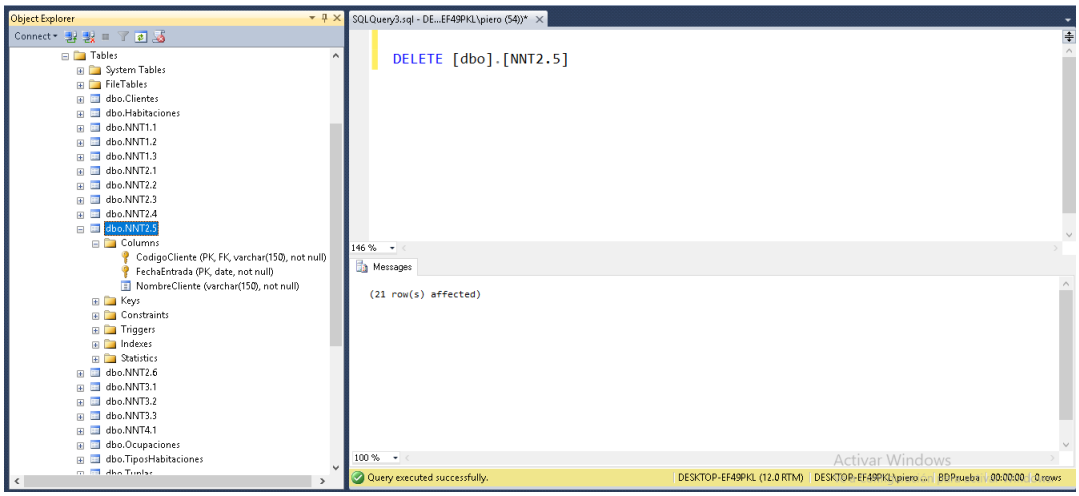
Gráfico 60 : Eliminación de Tuplas en Tabla NNT2.6.



Fuente: Elaboración propia.

f. NNT2.5

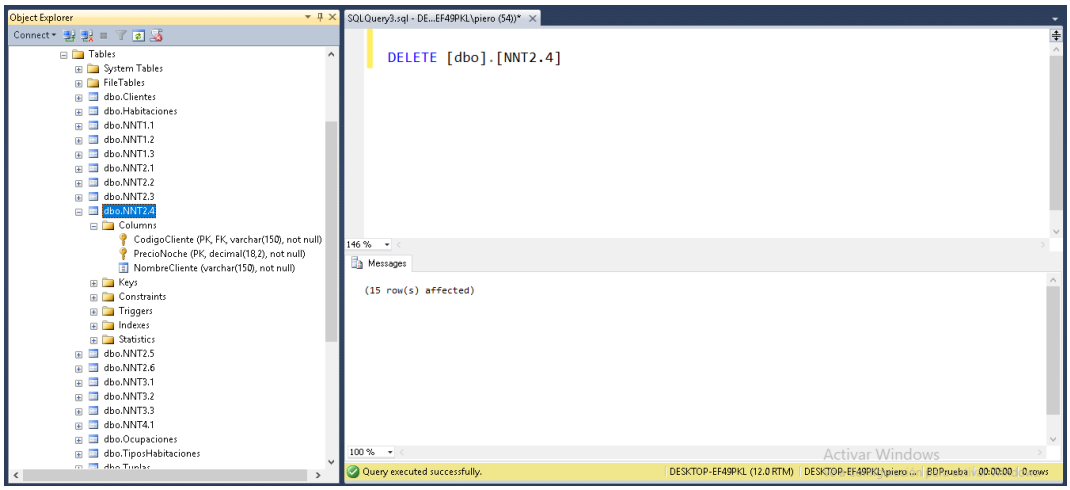
Gráfico 61 : Eliminación de Tuplas en Tabla NNT2.5.



Fuente: Elaboración propia.

g. NNT2.4

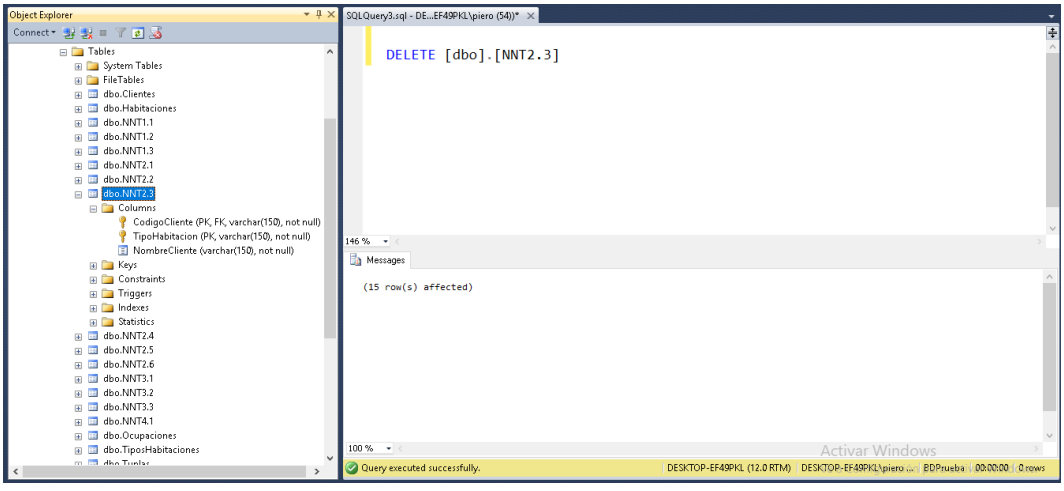
Gráfico 62 : Eliminación de Tuplas en Tabla NNT2.4.



Fuente: Elaboración propia.

h. NNT2.3

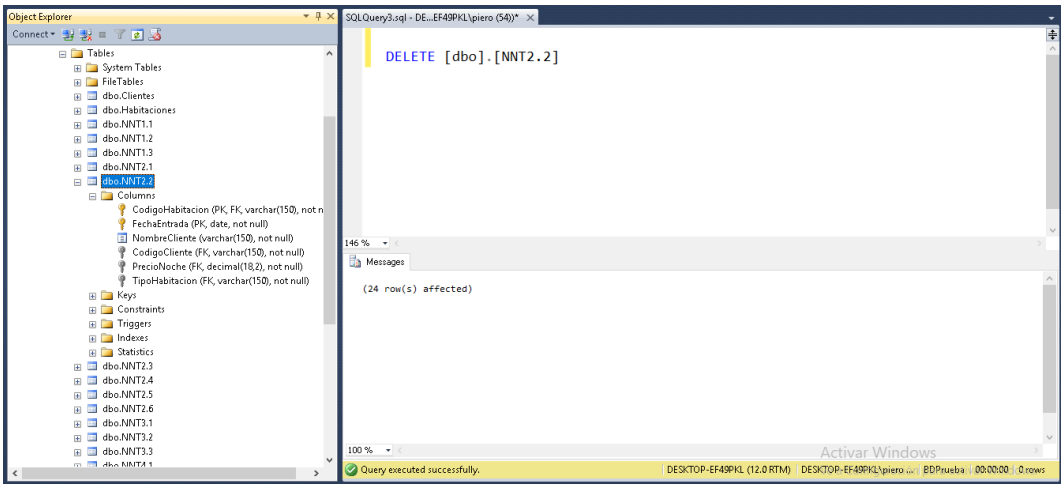
Gráfico 63 : Eliminación de Tuplas en Tabla NNT2.3.



Fuente: Elaboración propia.

i. NNT2.2

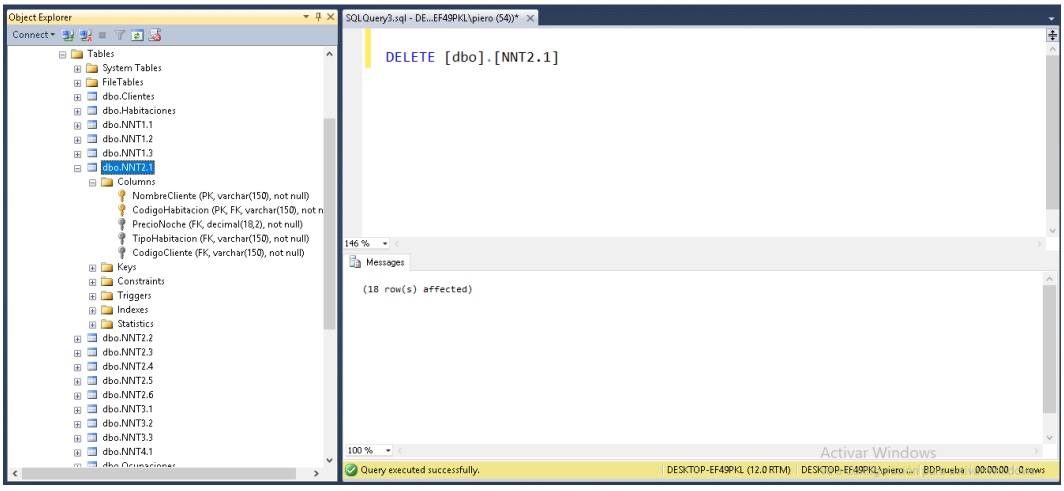
Gráfico 64 : Eliminación de Tuplas en Tabla NNT2.2.



Fuente: Elaboración propia.

j. NNT2.1

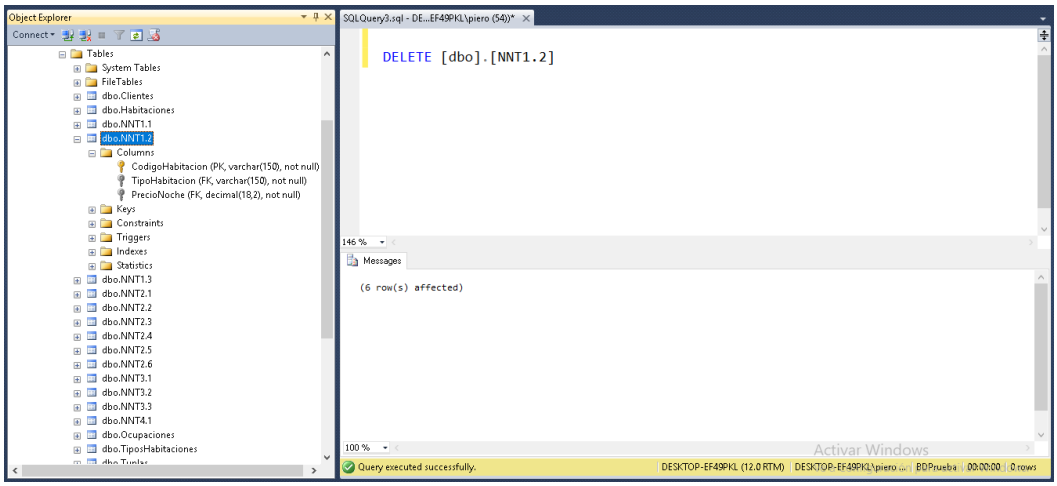
Gráfico 65 : Eliminación de Tuplas en Tabla NNT2.1.



Fuente: Elaboración propia.

k. NNT1.2

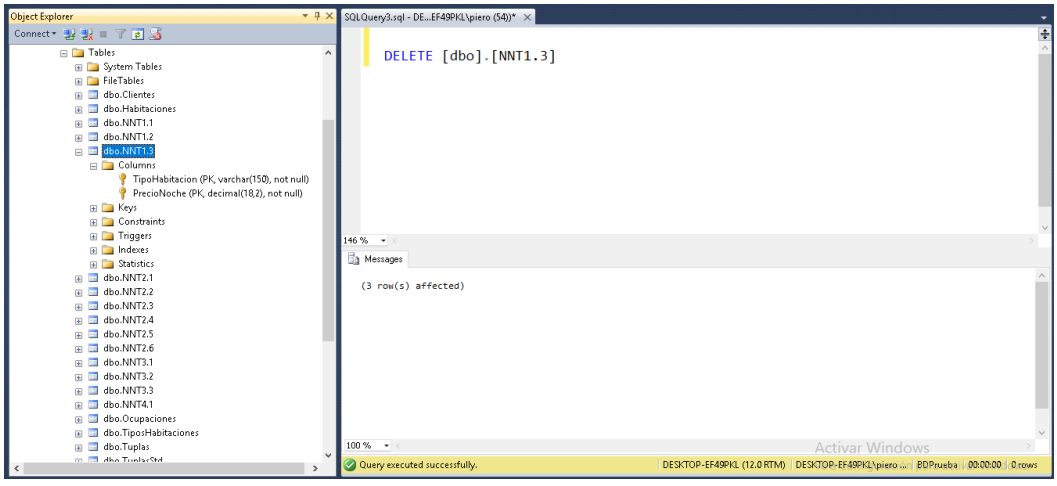
Gráfico 66 : Eliminación de Tuplas en Tabla NNT1.2.



Fuente: Elaboración propia.

l. NNT1.3

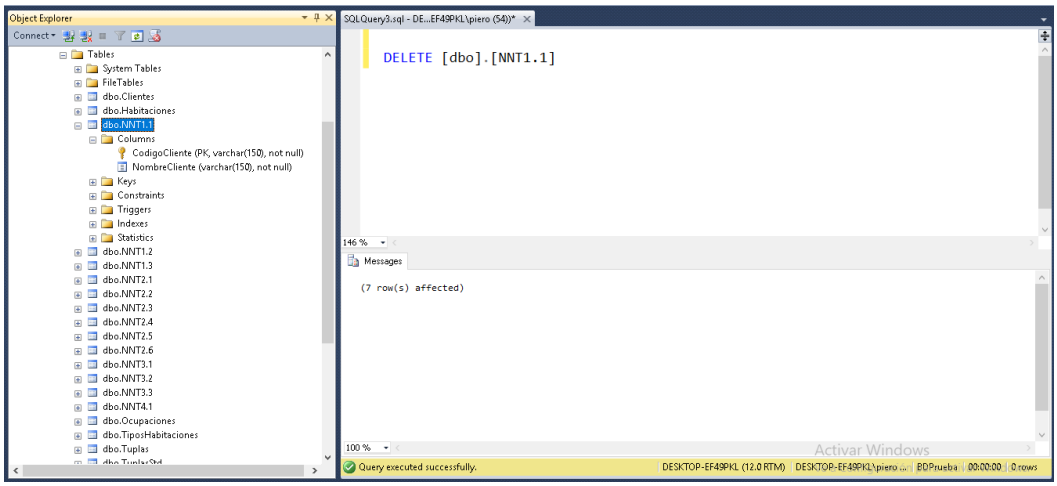
Gráfico 67 : Eliminación de Tuplas en Tabla NNT1.3.



Fuente: Elaboración propia.

m. NNT1.1

Gráfico 68 : Eliminación de Tuplas en Tabla NNT1.1.



Fuente: Elaboración propia.

E. Verificación de Restricciones de Integridad de Datos

a. Verificación de Integridad de Datos en Diseño de Tablas Normalizadas

Tabla 15 : Verificación de Integridad de Datos en el Diseño de Tablas Normalizadas.

Tablas	Restricción	Factor de Cumplimiento	¿Cumple?
Clientes(CodigoCliente[PK],NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
	Integridad Referencial	No Aplica	Sí
	Integridad de Dominio	Reglas de Negocio 5,9	Sí
Habitaciones(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)	Integridad de Entidad	CodigoHabitacion[PK]	Sí
	Integridad Referencial	No Aplica	Sí
	Integridad de Dominio	Regla de Negocio 10	Sí
TiposHabitaciones(TipoHabitacion[PK],PrecioNoche[PK])	Integridad de Entidad	TipoHabitacion[PK]	Sí
		PrecioNoche[PK]	Sí
	Integridad Referencial	No Aplica	Sí
	Integridad de Dominio	Regla de Negocio 10	Sí
Ocupaciones(CodigoHabitacion[PK],FechaEntrada[PK],CodigoCliente)	Integridad de Entidad	CodigoHabitacion[PK]	Sí
		FechaEntrada[PK]	Sí
	Integridad Referencial	CodigoCliente[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,10,11	Sí

Fuente: Elaboración propia.

b. Verificación de Integridad de Datos en Diseño de Tablas No Normalizadas

Tabla 16 : Verificación de Integridad de Datos en el Diseño de Tablas No Normalizadas.

Tablas	Restricción	Factor de Cumplimiento	¿Cumple?
NNT1.1(CodigoCliente[PK],NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
	Integridad Referencial	No Aplica	Sí
	Integridad de Dominio	Reglas de Negocio 5,9	Sí
NNT1.2(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)	Integridad de Entidad	CodigoHabitacion[PK]	Sí
	Integridad Referencial	No Aplica	Sí
	Integridad de Dominio	Regla de Negocio 10	Sí
NNT1.3(TipoHabitacion[PK],PrecioNoche[PK])	Integridad de Entidad	TipoHabitacion[PK]	Sí
		PrecioNoche[PK]	Sí
	Integridad Referencial	No Aplica	Sí
	Integridad de Dominio	Regla de Negocio 10	Sí
NNT2.1(NombreCliente[PK],CodigoHabitacion[PK],TipoHabitacion,PrecioNoche,CodigoCliente)	Integridad de Entidad	NombreCliente[PK]	Sí
		CodigoHabitacion[PK]	Sí
	Integridad Referencial	CodigoHabitacion[FK]	Sí
		TipoHabitacion[FK]	Sí
		PrecioNoche[FK]	Sí
		CodigoCliente[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,10	Sí
NNT2.2(CodigoHabitacion[PK],FechaEntrada[PK],TipoHabitacion,PrecioNoche,NombreCliente,CodigoCliente)	Integridad de Entidad	CodigoHabitacion[PK]	Sí
		FechaEntrada[PK]	Sí
	Integridad Referencial	CodigoHabitacion[FK]	Sí
		TipoHabitacion[FK]	Sí
		PrecioNoche[FK]	Sí
		CodigoCliente[FK]	Sí

	Integridad de Dominio	Reglas de Negocio 5,9,10,11	Sí
NNT2.3(CodigoCliente[PK],TipoHabitacion[PK],NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
		TipoHabitacion[PK]	Sí
	Integridad Referencial	CodigoCliente[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,10	Sí
NNT2.4(CodigoCliente[PK],PrecioNoche[PK],NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
		PrecioNoche[PK]	Sí
	Integridad Referencial	CodigoCliente[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,10	Sí
NNT2.5(CodigoCliente[PK],FechaEntrada[PK],NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
		FechaEntrada[PK]	Sí
	Integridad Referencial	CodigoCliente[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,11	Sí
NNT2.6(CodigoCliente[PK],CodigoHabitacion[PK],TipoHabitacion,PrecioNoche,NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
		CodigoHabitacion[PK]	Sí
	Integridad Referencial	CodigoHabitacion[FK]	Sí
		TipoHabitacion[FK]	Sí
		PrecioNoche[FK]	Sí
		CodigoCliente[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,10	Sí
NNT3.1(CodigoCliente[PK],TipoHabitacion[PK],PrecioNoche[PK],NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
		TipoHabitacion[PK]	Sí
		PrecioNoche[PK]	Sí
	Integridad Referencial	CodigoCliente[FK]	Sí
		TipoHabitacion[FK]	Sí
		PrecioNoche[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,10	Sí
NNT3.2(CodigoCliente[PK],FechaEntrada[PK],TipoHabitacion[PK],NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
		FechaEntrada[PK]	Sí
		TipoHabitacion[PK]	Sí

	Integridad Referencial	CodigoCliente[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,10,11	Sí
NNT3.3(CodigoCliente[PK],FechaEntrada[PK],PrecioNoche[PK],NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
		FechaEntrada[PK]	Sí
		PrecioNoche[PK]	Sí
	Integridad Referencial	CodigoCliente[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,10,11	Sí
NNT4.1(CodigoCliente[PK],FechaEntrada[PK],TipoHabitacion[PK],PrecioNoche[PK],NombreCliente)	Integridad de Entidad	CodigoCliente[PK]	Sí
		FechaEntrada[PK]	Sí
		TipoHabitacion[PK]	Sí
		PrecioNoche[PK]	Sí
	Integridad Referencial	CodigoCliente[FK]	Sí
		TipoHabitacion[FK]	Sí
		PrecioNoche[FK]	Sí
	Integridad de Dominio	Reglas de Negocio 5,9,10,11	Sí

Fuente: Elaboración propia.

F. Verificación de Cumplimiento de Normalización en Diseño de Tablas Normalizadas

Tabla 17 : Verificación de Cumplimiento de Normalización en el Diseño de Tablas Normalizadas.

Modelo Relacional	Modelo de Relación Funcional	Tablas Normalizadas	¿Cumple?
Primer Forma Normal (1FN)	Tuplas de Estructura Lógica de Datos	Clientes(CodigoCliente[PK],NombreCliente)	Sí
		Habitaciones(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)	Sí
		TiposHabitaciones(TipoHabitacion[PK],PrecioNoche[PK])	Sí

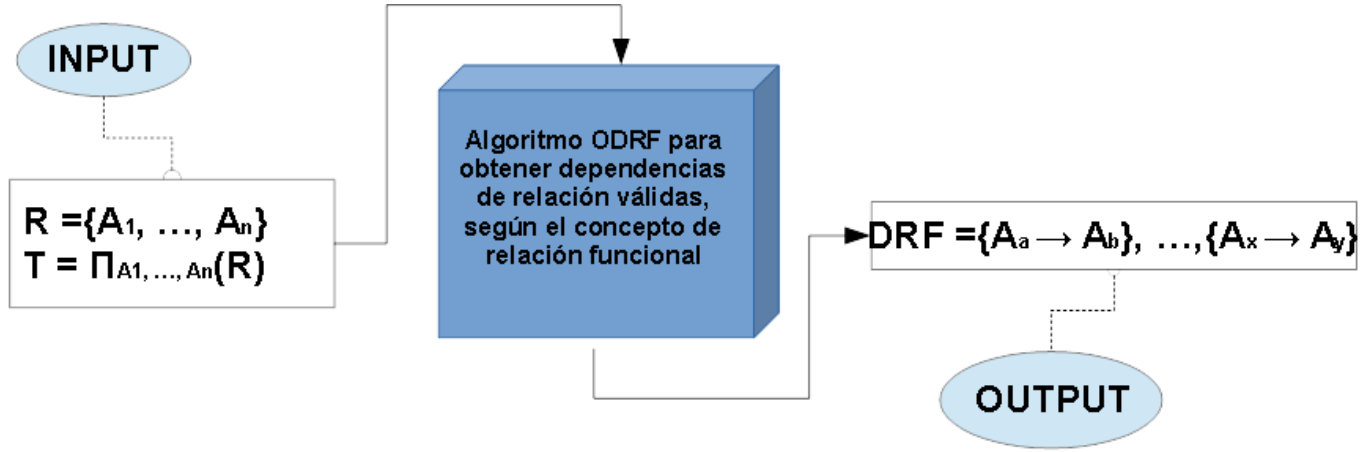
		Ocupaciones(CodigoHabitacion[PK],FechaEntrada[PK],CodigoCliente)	Sí
Segunda Forma Normal (2FN)	Teorema 1.1, Teorema 1.2, Teorema 1.3, Teorema 1.4	Clientes(CodigoCliente[PK],NombreCliente)	Sí
		Habitaciones(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)	Sí
		TiposHabitaciones(TipoHabitacion[PK],PrecioNoche[PK])	Sí
		Ocupaciones(CodigoHabitacion[PK],FechaEntrada[PK],CodigoCliente)	Sí
Tercer Forma Normal (3FN)	Teorema 2.1	Clientes(CodigoCliente[PK],NombreCliente)	Sí
		Habitaciones(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)	Sí
		TiposHabitaciones(TipoHabitacion[PK],PrecioNoche[PK])	Sí
		Ocupaciones(CodigoHabitacion[PK],FechaEntrada[PK],CodigoCliente)	Sí
Forma Normal Boyce-Codd (BCFN)	Aplicación de Restricciones Lógicas Funcionales	Clientes(CodigoCliente[PK],NombreCliente)	Sí
		Habitaciones(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)	Sí
		TiposHabitaciones(TipoHabitacion[PK],PrecioNoche[PK])	Sí
		Ocupaciones(CodigoHabitacion[PK],FechaEntrada[PK],CodigoCliente)	Sí
Cuarta Forma Normal (4FN)	Aplicación de Restricciones Lógicas Funcionales	Clientes(CodigoCliente[PK],NombreCliente)	Sí
		Habitaciones(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)	Sí
		TiposHabitaciones(TipoHabitacion[PK],PrecioNoche[PK])	Sí
		Ocupaciones(CodigoHabitacion[PK],FechaEntrada[PK],CodigoCliente)	Sí
Quinta Forma Normal (5FN)	Aplicación de Restricciones Lógicas Funcionales	Clientes(CodigoCliente[PK],NombreCliente)	Sí
		Habitaciones(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)	Sí
		TiposHabitaciones(TipoHabitacion[PK],PrecioNoche[PK])	Sí
		Ocupaciones(CodigoHabitacion[PK],FechaEntrada[PK],CodigoCliente)	Sí

Fuente: Elaboración propia.

4. [OE4] Proceso Algorítmico para el Modelo de Datos de Relación Funcional

A. Proceso Algorítmico ODRF

Gráfico 69 . Gráfico del Proceso algorítmico ODRF.



R: Esquema de relación que está conformado por sus n atributos.

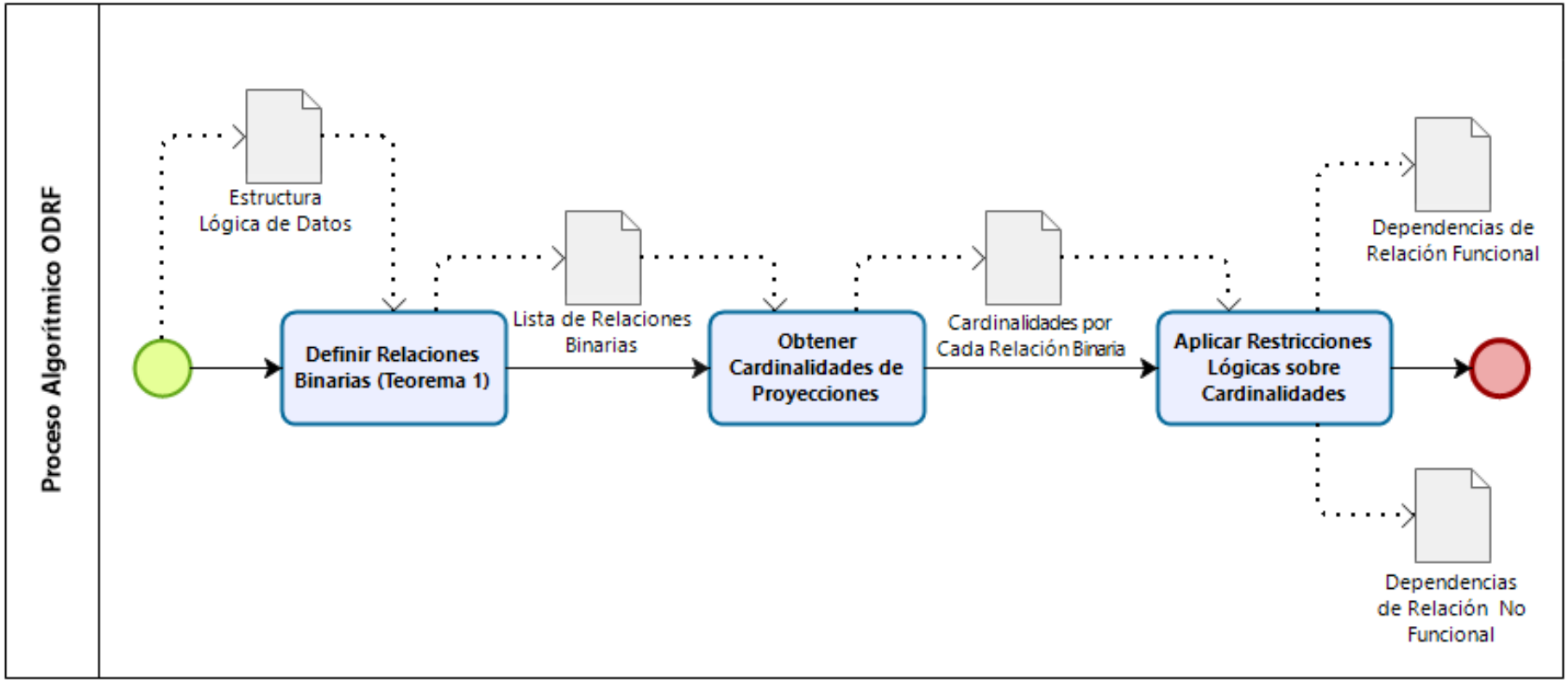
T: Proyección de tuplas de R.

DRF (Dependencias de Relación Funcional): Conjunto de dependencias funcionalmente relacionales generadas por el algoritmo ODRF.

ODRF: Algoritmo que permite la Obtención de Dependencias de Relación Funcional.

Fuente: Elaboración propia.

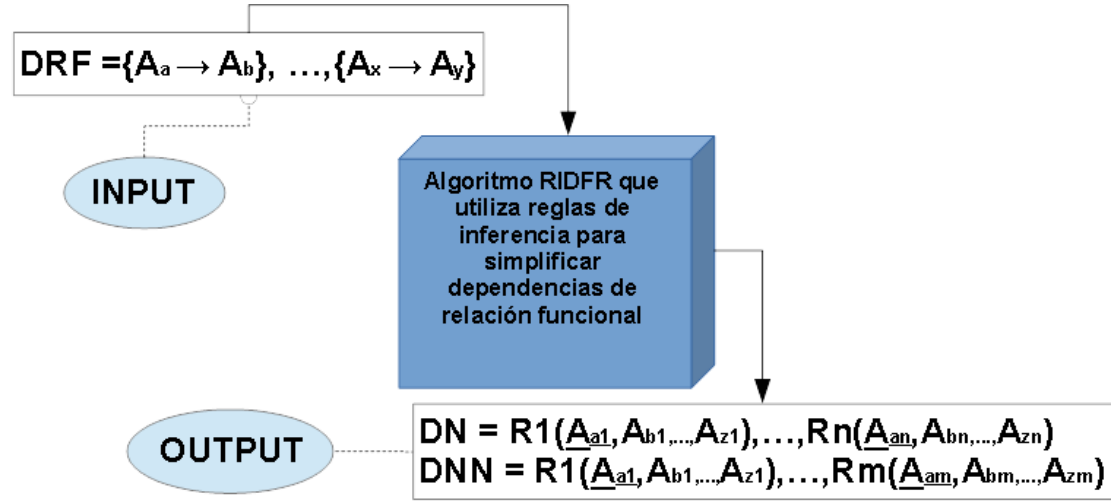
Gráfico 70 : Diagrama del Proceso Algorítmico ODRF.



Fuente: Elaboración propia.

B. Proceso Algorítmico RIDRF

Gráfico 71 : Gráfico del Proceso algorítmico RIDRF.



DRF (Dependencias de Relación Funcional): Conjunto de dependencias funcionalmente relacionales generadas por el algoritmo ODRF.

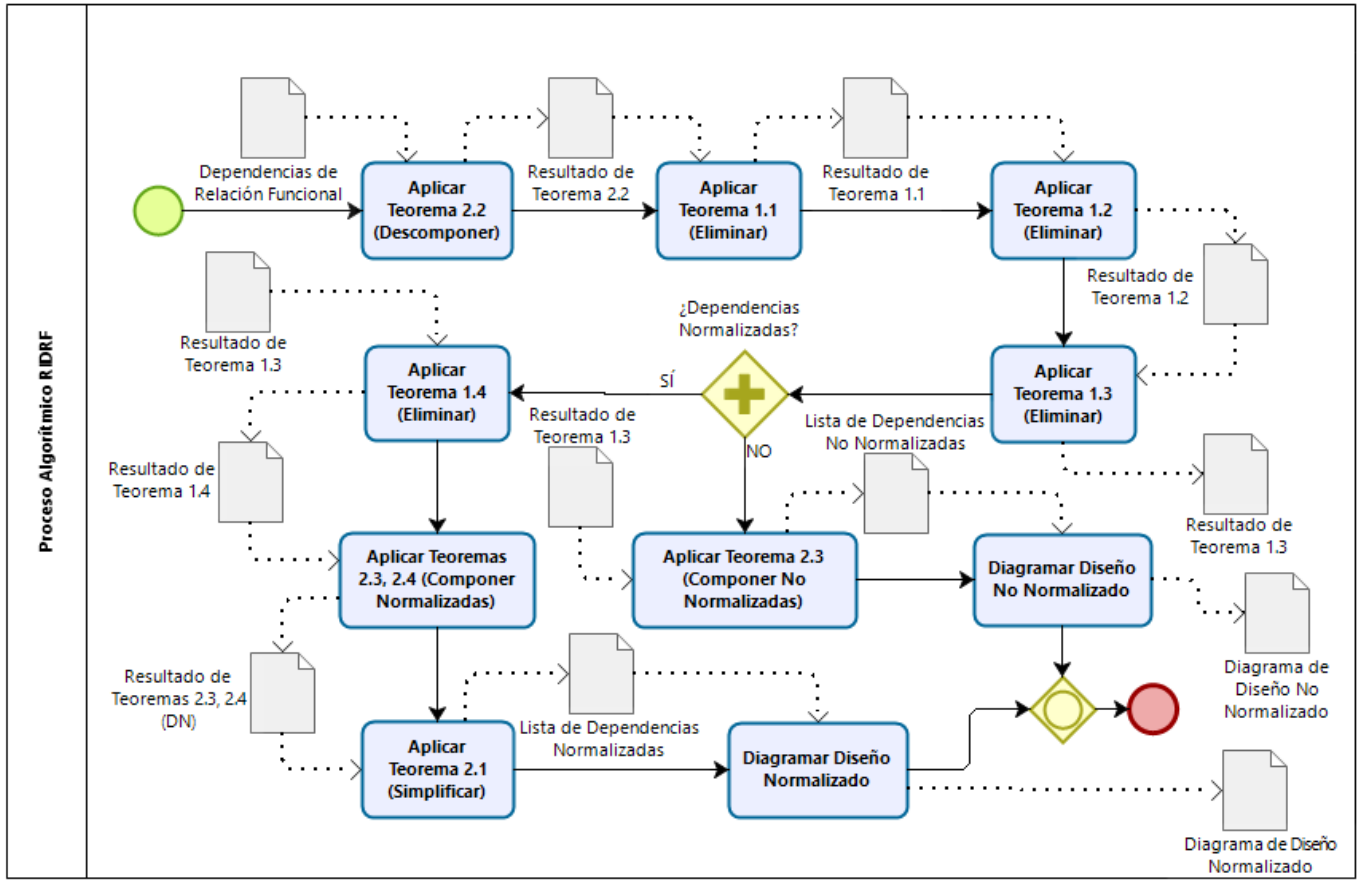
DN (Dependencias Normalizadas): Conjunto de dependencias de relación funcional normalizadas que han sido generadas por el algoritmo RIDRF.

DNN (Dependencias No Normalizadas): Conjunto de dependencias de relación funcional no normalizadas que han sido generadas por el algoritmo RIDRF.

RIDRF: Algoritmo que aplica las Reglas de Inferencia para Dependencias de Relación Funcional.

Fuente: Elaboración propia.

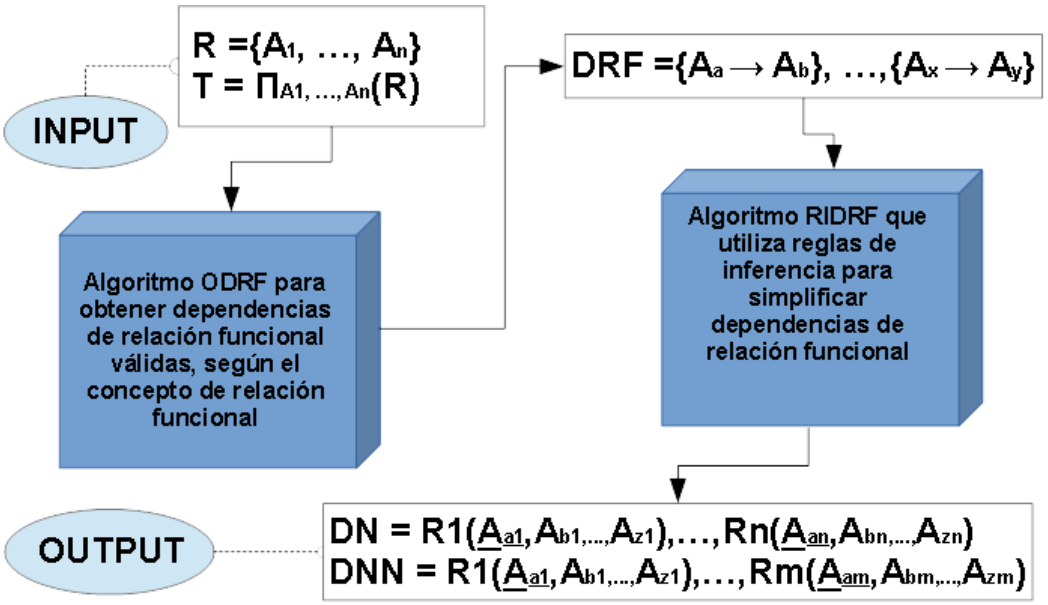
Gráfico 72 : Diagrama del Proceso Algorítmico RIDRF.



Fuente: Elaboración propia.

C. Proceso Algorítmico ODRF + RIDRF

Gráfico 73 : Gráfico del Proceso algorítmico ODRF + RIDRF.



R: Esquema de relación que está conformado por sus n atributos.

T: Proyección de tuplas de R.

DRF (Dependencias de Relación Funcional): Conjunto de dependencias funcionalmente relacionales generadas por el algoritmo ODRF.

DN (Dependencias Normalizadas): Conjunto de dependencias de relación funcional normalizadas que han sido generadas por el algoritmo RIDRF.

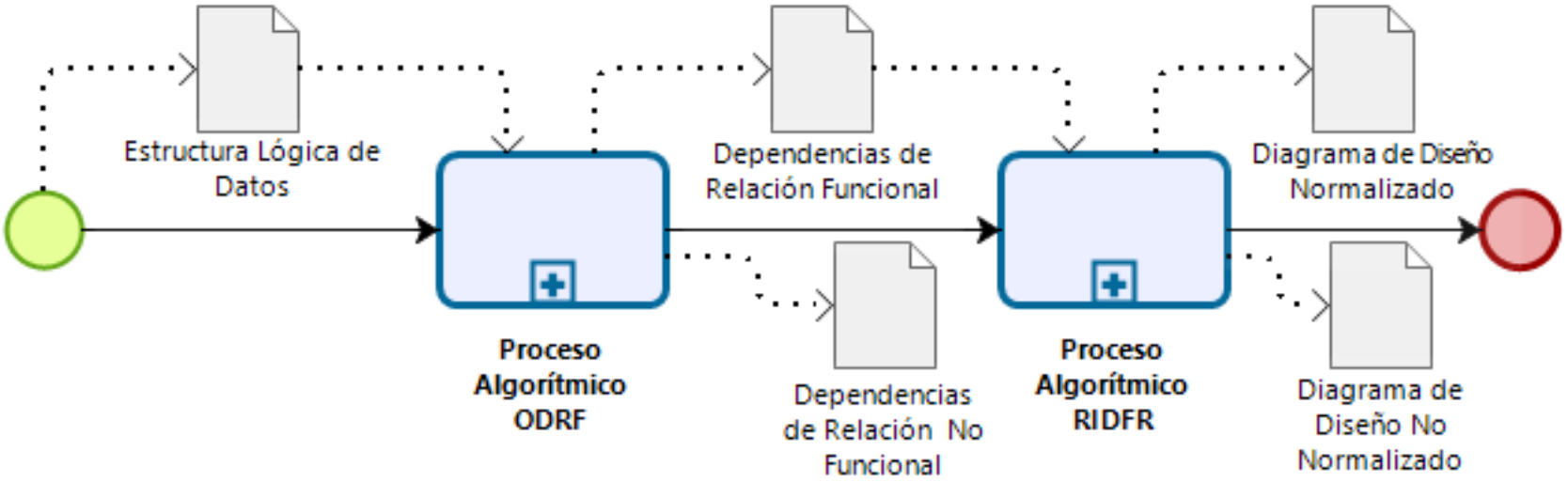
DNN (Dependencias No Normalizadas): Conjunto de dependencias de relación funcional no normalizadas que han sido generadas por el algoritmo RIDRF.

ODRF: Algoritmo que permite la Obtención de Dependencias de Relación Funcional.

RIDRF: Algoritmo que aplica las Reglas de Inferencia para Dependencias de Relación Funcional.

Fuente: Elaboración propia.

Gráfico 74 : Diagrama del Proceso Algorítmico ODRF + RIDRF.



Fuente: Elaboración propia.

D. Algoritmo ODRF + RIDRF

a. Algoritmo ODRF

Cuadro 1 : Algoritmo ODRF.

```
Algorithm: ODRF
To discover all non-functional and functional relation dependencies in a Logical Data Structure
(Dataset).
INPUT: Logical Data Structure (Dataset).
OUTPUT: All non-functional and functional relation dependencies.
BEGIN
    Set RELATION = {A1,...,An}
    Set n = |RELATION|
    Set TUPLES = ΠA1,...,An(RELATION)
    Set LIST_FUNCTIONAL_RELATION_DEPENDENCIES = Ø
    Set LIST_NONFUNCTIONAL_RELATION_DEPENDENCIES = Ø
    Set BINARY_RELATIONS = Ø
    Set COMBINATION = Ø
    Set COMBINATIONS_WITHOUT_REPETITION = Ø
    Set CARDINALITY_PROJECTIONS = Ø
    Set INDEX = 1
    Set START = 1
    if RELATION ≠ Ø and TUPLES ≠ Ø then
        Set r = START
        for each Ar ∈ RELATION do
            if r <= n – 1then
                Combinations(RELATION, START, n, r, INDEX)
        DefineBinaryRelations(RELATION, COMBINATIONS_WITHOUT_REPETITION)
        ObtainCardinalityOfProjections(RELATION,BINARY_RELATIONS)
        ApplyLogicConstraintsOnCardinalities(CARDINALITY_PROJECTIONS)
    List(LIST_FUNCTIONAL_RELATION_DEPENDENCIES)
    List(LIST_NONFUNCTIONAL_RELATION_DEPENDENCIES)
    AlgorithmRIDRF(LIST_FUNCTIONAL_RELATION_DEPENDENCIES)
END
```

Fuente: Elaboración propia.

Cuadro 2 : Procedimiento 'Combinations'.

```
Procedure: Combinations(RELATION, START, n, r, INDEX)
BEGIN
    if INDEX > r then
        Set INDEX = INDEX - 1
        Set j = INDEX
        add COMBINATION to COMBINATIONS_WITHOUT_REPETITION
        delete Aj from COMBINATION
    else
    Set i = START
    if i ≤ n and n-i+1 ≥ r-INDEX then
```

```
            for each Ai ∈ RELATION do
                add Ai to COMBINATION
                Combinations(RELATION, i+1, n, r, INDEX+1)
        Set COMBINATION = Ø
        Set INDEX = 1
END
```

Fuente: Elaboración propia.

Cuadro 3 : Procedimiento 'DefineBinaryRelations'.

```
Procedure: DefineBinaryRelations(RELATION, COMBINATIONS_WITHOUT_REPETITION)
BEGIN
    for each Cj ∈ COMBINATIONS_WITHOUT_REPETITION do
        for each Ai ∈ RELATION do
            add Cj,Ai to BINARY_RELATIONS
END
```

Fuente: Elaboración propia.

Cuadro 4 : Procedimiento 'ObtainCardinalityOfProjections'.

```
Procedure: ObtainCardinalityOfProjections(TUPLES,BINARY_RELATIONS)
BEGIN
    for each Cj,Ai ∈ BINARY_RELATIONS do
        Set cA = | Π Cj (TUPLES) |
        Set cB = | Π Ai (TUPLES) |
        Set cAB = | Π Cj,Ai (TUPLES) |
        add Cj,Ai,cA,cB,cAB to CARDINALITY_PROJECTIONS
END
```

Fuente: Elaboración propia.

Cuadro 5 : Procedimiento 'ApplyLogicConstraintsOnCardinalities'.

```
Procedure: ApplyLogicConstraintsOnCardinalities(CARDINALITY_PROJECTIONS)
BEGIN
    for each Cj,Ai,cA,cB,cAB ∈ CARDINALITY_PROJECTIONS do
        Set RT = Ø
        Set FT = Ø
        Set D = Ø
        if cAB = cA and cAB > cB then
            Set RT = SINGLE-VALUED FUNCTION
            Set FT = SURJECTIVE FUNCTION
            Set D = cA → cB
        else if cAB = cB and cAB > cA then
            Set RT = SINGLE-VALUED FUNCTION
```

```
                Set FT = SURJECTIVE FUNCTION
                Set D = cB → cA
            else if cAB = cA and cAB > cB then
                Set RT = SINGLE-VALUED FUNCTION
                Set FT = BIJECTIVE FUNCTION
                Set D = cA ↠ cB
            else if cAB > cA and cAB > cB then
                Set RT = MULTIVALUED FUNCTION
                Set FT = NO FUNCTION
                Set D = cA ⊸ cB
        add Cj,Ai,cA,cB,cAB,D,RT,FT to LIST_NONFUNCTIONAL_RELATION_DEPENDENCIES
END
```

Fuente: Elaboración propia.

Cuadro 6 : Procedimiento 'List'.

```
Procedure: List(LIST)
BEGIN
    for each Li ∈ LIST do
        print Li
END
```

Fuente: Elaboración propia.

b. Algoritmo RIDRF

Cuadro 7 : Algoritmo RIDRF.

```
Algorithm RIDRF
To obtain non-normalized and normalized functional relation dependencies from all functional
relation dependencies.
INPUT: All functional relation dependencies.
OUTPUT: Non-normalized and normalized functional relation dependencies.
BEGIN
    Set LIST_NORMALIZED_DEPENDENCIES = Ø
    Set LIST_NONNORMALIZED_DEPENDENCIES = Ø
    Set RESULT_THEOREM1_1 = Ø
    Set RESULT_THEOREM1_2 = Ø
    Set RESULT_THEOREM1_3 = Ø
    Set RESULT_THEOREM1_4 = Ø
    Set RESULT_THEOREM2_1 = Ø
    Set RESULT_THEOREM2_2 = Ø
    Set RESULT_THEOREM2_3_NND = Ø
    Set RESULT_THEOREM2_3_ND = Ø
    Set RESULT_THEOREM_TYPE = Ø
    Set TYPE = Ø
    Set NEW_DEPENDENCY = Ø
    Set COMBINATION = Ø
    Set COMBINATIONS = Ø
```

```
    if LIST_FUNCTIONAL_RELATION_DEPENDENCIES ≠ Ø then
        Theorem2.2(LIST_FUNCTIONAL_RELATION_DEPENDENCIES)
        Theorem1.1(RESULT_THEOREM2_2)
        Theorem1.2(RESULT_THEOREM1_1)
        Theorem1.3(RESULT_THEOREM1_2)
        Theorem2.3(RESULT_THEOREM1_3,NND)
        ObtainNonNormalizedDependencies(RESULT_THEOREM2_3_NND)
        Theorem1.4(RESULT_THEOREM1_3)
        Theorem2.3(RESULT_THEOREM1_4,ND)
        Theorem2.4(RESULT_THEOREM2_3_ND)
        Theorem2.1(RESULT_THEOREM2_4)
        ObtainNormalizedDependencies(RESULT_THEOREM2_1)
    List(LIST_NORMALIZED_DEPENDENCIES)
    List(LIST_NONNORMALIZED_DEPENDENCIES)
END
```

Fuente: Elaboración propia.

Cuadro 8 : Procedimiento 'Theorem2.2'.

```
Procedure: Theorem2.2 (LIST_FUNCTIONAL_RELATION_DEPENDENCIES)
BEGIN
    for each Cj,Ai,cA,cB,cAB,D,RT,FT ∈ LIST_FUNCTIONAL_RELATION_DEPENDENCIES do
        Set c = | Ai |
        if c > 1 then
            for each Ax ⊂ Ai do
                ObtainNewDependence(TUPLES,Cj, Ax)
                add NEW_DEPENDENCY to RESULT_THEOREM2_2
                NEW_DEPENDENCY = Ø
        else
            add Cj,Ai,cA,cB,cAB,D,RT,FT to RESULT_THEOREM2_2
END
```

Fuente: Elaboración propia.

Cuadro 9 : Procedimiento 'Theorem1.1'.

```
Procedure: Theorem1.1(RESULT_THEOREM2_2)
BEGIN
    Set RESULT_THEOREM1_1 = RESULT_THEOREM2_2
    for each Cj,Ai,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM2_2 do
        if Ai ⊆ Cj then
            delete Cj,Ai,cA,cB,cAB,D,RT,FT from RESULT_THEOREM1_1
END
```

Fuente: Elaboración propia.

Cuadro 10 : Procedimiento 'Theorem1.2'.

```
Procedure: Theorem1.2(RESULT_THEOREM1_1)
BEGIN
    Set RESULT_THEOREM1_2 = RESULT_THEOREM1_1
    for each Cj,Ai,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM1_2 do
        for each x ⊂ Cj do
            NEW_DEPENDENCY = Ø
            ObtainNewDependence(TUPLES,x,Ai)
            if ∃ x ↠ Ai in D ∈ NEW_DEPENDENCY then
                delete Cj,Ai,cA,cB,cAB,D,RT,FT from RESULT_THEOREM1_2
                end for each x ⊂ Cj
END
```

Fuente: Elaboración propia.

Cuadro 11 : Procedimiento 'Theorem1.3'.

```
Procedure: Theorem1.3(RESULT_THEOREM1_2)
BEGIN
    Set RESULT_THEOREM1_3 = RESULT_THEOREM1_2
    for each Cj,Ai,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM1_3 do
        Set n = | Cj |
        COMBINATIONS = Ø
        for each Ar ∈ Cj do
            if r > 1 then
                ObtainCombinations(Cj, 1, n, r, INDEX)
        for each Ck ∈ COMBINATIONS do
            for each A,B ⊂ Ck do
                ObtainNewDependence(TUPLES,A,B)
                Set A_B = NEW_DEPENDENCY
                if ∃ A → B in D ∈ A_B or ∃ B → A in D ∈ A_B then
                    delete Cj,Ai,cA,cB,cAB,D,RT,FT from RESULT_THEOREM1_3
                    end for each Ck ∈ COMBINATIONS
END
```

Fuente: Elaboración propia.

Cuadro 12 : Procedimiento 'Theorem1.4'.

```
Procedure: Theorem1.4(RESULT_THEOREM1_3)
BEGIN
    Set RESULT_THEOREM1_4 = RESULT_THEOREM1_3
    for each Cj,Ai,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM1_4 do
        Set n = | Cj |
        COMBINATIONS = Ø
        for each Ar ∈ Cj do
            if r > 1 then
                ObtainCombinations(Cj, 1, n, r, INDEX)
```

```
            for each Ck ∈ COMBINATIONS do
                for each A,B ⊂ Ck do
                    ObtainNewDependence(TUPLES,A,B)
                    Set A_B = NEW_DEPENDENCY
                    ObtainNewDependence(TUPLES,A,Ai)
                    Set A_Ai = NEW_DEPENDENCY
                    ObtainNewDependence(TUPLES,B,Ai)
                    Set B_Ai = NEW_DEPENDENCY
                    if ∃ A ⊸ B in D ∈ A_B then
                        if ∃ A → Ai in D ∈ A_Ai or ∃ B → Ai in D ∈ B_Ai then
                            delete Cj,Ai,cA,cB,cAB,D,RT,FT from RESULT_THEOREM1_4
                            end for each Ck ∈ COMBINATIONS
END
```

Fuente: Elaboración propia.

Cuadro 13 : Procedimiento 'Theorem2.3'.

```
Procedure: Theorem2.3(RESULT_THEOREM_TYPE, TYPE)
BEGIN
    for each Cj,Ai,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM_TYPE do
        if TYPE = ND
            if RESULT_THEOREM2_3_ND ≠ Ø then
                for each Cn,Am,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM2_3_ND do
                    if Cj = Cn then
                        Set B = Ai ∪ Am
                        ObtainNewDependence(TUPLES,Cj,B)
                        Set NEW_DEPENDENCY_Cj_B = NEW_DEPENDENCY
                        delete Cn,Am,cA,cB,cAB,D,RT,FT from RESULT_THEOREM2_3_ND
                        add NEW_DEPENDENCY_Cj_B to RESULT_THEOREM2_3_ND
                    else
                        add Cj,Ai,cA,cB,cAB,D,RT,FT to RESULT_THEOREM2_3_ND
            else
                add Cj,Ai,cA,cB,cAB,D,RT,FT to RESULT_THEOREM2_3_ND
        else
            if RESULT_THEOREM2_3_NND ≠ Ø then
                for each Cn,Am,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM2_3_NND do
                    if Cj = Cn then
                        Set B = Ai ∪ Am
                        ObtainNewDependence(TUPLES,Cj,B)
                        Set NEW_DEPENDENCY_Cj_B = NEW_DEPENDENCY
                        delete Cn,Am,cA,cB,cAB,D,RT,FT from RESULT_THEOREM2_3_NND
                        add NEW_DEPENDENCY_Cj_B to RESULT_THEOREM2_3_NND
                    else
                        add Cj,Ai,cA,cB,cAB,D,RT,FT to RESULT_THEOREM2_3_NND
            else
                add Cj,Ai,cA,cB,cAB,D,RT,FT to RESULT_THEOREM2_3_NND
END
```

Fuente: Elaboración propia.

Cuadro 14: Procedimiento 'Theorem2.4'.

```
Procedure: Theorem2.4(RESULT_THEOREM2_3)
BEGIN
    for each Cj,Ai,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM2_3 do
        if RESULT_THEOREM2_4 ≠ Ø then
            for each Cn,Am,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM2_4 do
                if Ai = Am then
                    ObtainNewDependence(TUPLES,Cj,Cn)
                    Set NEW_DEPENDENCY_Cj_Cn = NEW_DEPENDENCY
                    if ∃ Cj ↠ Cn ∨ Cj → Cn ∨ Cn → Cj in D ∈ NEW_DEPENDENCY_Cj_Cn then
                        Set B = Cj ∪ Cn
                        ObtainNewDependence(TUPLES,B,Ai)
                        Set NEW_DEPENDENCY_B_Ai = NEW_DEPENDENCY
                        delete Cn,Am,cA,cB,cAB,D,RT,FT from RESULT_THEOREM2_4
                        add NEW_DEPENDENCY_B_Ai to RESULT_THEOREM2_4
                    else
                        add Cj,Ai,cA,cB,cAB,D,RT,FT to RESULT_THEOREM2_4
            else
                add Cj,Ai,cA,cB,cAB,D,RT,FT to RESULT_THEOREM2_4
END
```

Fuente: Elaboración propia.

Cuadro 15 : Procedimiento 'Theorem2.1'.

```
Procedure: Theorem2.1(RESULT_THEOREM2_4)
BEGIN
    Set RESULT_THEOREM2_1 = RESULT_THEOREM2_4
    for each Cj,Ai,cA,cB,cAB,D,RT,FT ∈ RESULT_THEOREM2_1 do
        Set n = | Ai |
        COMBINATIONS = Ø
        for each Ar ∈ Ai do
            if r > 1 then
                ObtainCombinations(Ai, 1, n, r, INDEX)
        for each Ck ∈ COMBINATIONS do
            for each A,B ⊂ Ck do
                ObtainNewDependence(TUPLES,A,B)
                Set A_B = NEW_DEPENDENCY
                if ∃ A → B in D ∈ A_B and ∃ B in Ai then
                    delete B from Ai
                    replace Ai on Cj,Ai,cA,cB,cAB,D,RT,FT from RESULT_THEOREM2_1
END
```

Fuente: Elaboración propia.

Cuadro 16 : Procedimiento 'ObtainCombinations'.

```
Procedure: ObtainCombinations(RELATION, START, n, r, INDEX)
```

```
BEGIN
    if INDEX > r then
        Set INDEX = INDEX - 1
        Set j = INDEX
        add COMBINATION to COMBINATIONS
        delete Aj from COMBINATION
    else
    Set i = START
    if i ≤ n and n-i+1 ≥ r-INDEX then
        for each Ai ∈ RELATION do
            add Ai to COMBINATION
            ObtainCombinations(RELATION, i+1, n, r, INDEX+1)
   Set COMBINATION = Ø
   Set INDEX = 1
END
```

Fuente: Elaboración propia.

Cuadro 17 : Procedimiento 'ObtainNewDependence'.

```
Procedure: ObtainNewDependence(TUPLES,A,B)
BEGIN
    Set RT = Ø
    Set FT = Ø
    Set D = Ø
    Set cA = | Π A (TUPLES) |
    Set cB = | Π B (TUPLES) |
    Set cAB = | Π AB (TUPLES) |
    Set NEW_DEPENDENCY = Ø
    if cAB = cA and cAB > cB then
        Set RT = SINGLE-VALUED FUNCTION
        Set FT = SURJECTIVE FUNCTION
        Set D = cA → cB
    else if cAB = cB and cAB > cA then
        Set RT = SINGLE-VALUED FUNCTION
        Set FT = SURJECTIVE FUNCTION
        Set D = cB → cA
    if cAB = cA and cAB > cB then
        Set RT = SINGLE-VALUED FUNCTION
        Set FT = BIJECTIVE FUNCTION
        Set D = cA ↠ cB
    else if cAB > cA and cAB > cB then
        Set RT = MULTIVALUED FUNCTION
        Set FT = NO FUNCTION
        Set D = cA ⊸ cB
    add A,B,cA,cB,cAB,D,RT,FT to NEW_DEPENDENCY
END
```

Fuente: Elaboración propia.

VI. Discusiones

(1) Christopher J. Date (Date C. J., 2012, págs. 290-291) nos indica que para un científico, una teoría es un conjunto de ideas o principios que explican algún conjunto de fenómenos observables de manera coherente, ya que se ajusta a los hechos; y el Modelo Relacional es sin duda una teoría.

El propósito de la Teoría Relacional, en particular, es permitirnos construir sistemas que sean completamente prácticos, sin desviar la misma teoría subyacente, ya que nunca se han cambiado los axiomas matemáticos para el Modelo Relacional, aunque sí se han realizado, en el transcurso de los años, una serie de cambios de forma al modelo en sí. Por ejemplo, se han agregado comparaciones relacionales, pero los axiomas (que son básicamente los correspondientes a la teoría clásica de conjuntos y a la lógica de predicados clásica) no han cambiado desde el primer paper que Edgar F. Codd escribió al respecto (Codd, 1970).

Además, todos los cambios que se han producido han sido, en opinión de Date, naturalmente evolutivos, pero no reformistas. Por lo tanto, Date realmente afirma que solo hay un único Modelo Relacional, que es infranqueable, a pesar de que ha evolucionado con el tiempo y probablemente continuará haciéndolo, ya que el Modelo Relacional puede verse como una pequeña rama de las matemáticas; como tal, crece con el tiempo en la medida en que se prueban nuevos teoremas y se descubren nuevos resultados. Lo que es más, Date dice que —como las matemáticas en general— esos nuevos teoremas y resultados pueden ser probados y descubiertos por cualquiera que sea competente para hacerlo, siendo que el modelo relacional comenzó como una creación de un hombre, pero ahora pertenece al mundo.

De manera que, tal como lo afirma C. J. Date, el Modelo Relacional tiene su propia circunspección matemática muy fuerte, aunque en constante evolución, siendo que es muy perfectible todavía.

En el caso del Modelo de Relación Funcional, que se propone en la presente investigación, los axiomas matemáticos que se pretenden definir recogen los fundamentos estructurales del Modelo Relacional, aunque la variación está en el enfoque matemático que está orientado a la Teoría de la Relación Funcional.

Una Relación Funcional (Función Matemática), a diferencia de una Relación (Relación Matemática), permite saber exactamente cuál es el valor de la parte dependiente sabiendo cuál es valor de la parte independiente. Con esta misma lógica, podemos aseverar que la contribución más resaltante del Modelo de Relación Funcional propuesto está en que,

dada una Estructura Lógica de Datos, podemos afirmar exactamente qué conjunto de atributos implica a otro, así como también qué conjunto de atributos depende de otro; y de esta manera, desarrollar una Teoría de Dependencias de Relación Funcional, las cuales nos permitirán diseñar bases de datos que no solamente cumplan con las Restricciones de Integridad de Datos, sino que también sean un fiel reflejo de la información empresarial que maneja, modifica y actualiza una organización.

(2) El ***Modelo de Datos de Relación Funcional (MDRF)*** se presenta como una variación extendida del ***Modelo de Datos Relacional (MDR)***, debido a que los fundamentos de éste son utilizados por aquél. Sin embargo, la prerrogativa que se destaca en el Modelo de Datos de Relación Funcional es que éste permite que todo diseño de base de datos se haga medible de forma lógica y que posteriormente se haga verificable con respecto a las reglas de negocio, las cuales obedecen a la lógica del negocio.

Se pudo verificar, por lo tanto, que se puede aplicar el Modelo de Datos de Relación Funcional sobre la información empresarial circunscrita en Estructuras Lógicas de Datos.

Por el contrario, el Modelo de Datos Relacional, basándose en su definición, no tiene cómo extraer restricciones lógicas desde Estructuras Lógicas de Datos, más allá del concepto limitado de las Dependencias Funcionales.

La Teoría de Dependencias aplicada al Modelo de Datos Relacional indica que para que exista una dependencia funcional, entre dos conjuntos de atributos de una Relación, se necesita que existan tuplas coincidentes; al haber coincidencia de tuplas entre dos conjuntos de atributos, entonces se establecerá que un conjunto de atributos depende funcionalmente del otro conjunto de atributos (Date C. J., 2012, págs. 174-178).

Desde la perspectiva del Modelo de Datos de Relación Funcional, la definición del concepto de la dependencia funcional aplicada al Modelo de Datos Relacional no solamente es incompleta de base sino también inexacta, ya que no basta con analizar parcialmente el conjunto de tuplas de una Relación para decidir si existe dependencia; en consecuencia, es necesario evaluar, desde la perspectiva de los conjuntos de atributos, la totalidad de las tuplas de una Relación para decidir, en base a una instancia actualizada de tuplas que proporciona la información empresarial, los tipos de dependencias existentes entre los conjuntos de atributos de la Relación.

De manera que en medio de la labor de recopilación se evidenció que existen investigaciones, dentro de la perspectiva del Modelo de Datos Relacional, que definen algoritmos para obtener dependencias funcionales sobre relaciones de datos que se enmarcan en Estructuras Lógicas de Datos; sin embargo, estas investigaciones son,

aunque muy académicas, insuficientes para lograr automatizar óptimamente la obtención de dependencias funcionales y que a su vez éstas permitan un correcto diseño de base de datos circunscrito a la lógica del negocio.

Y esto sucede debido a que, como ya se indicó, las dependencias funcionales aplicadas al Modelo de Datos Relacional son también insuficientes e inexactas.

(3) Para hacer una definición de la forma más técnica posible, diremos que una dependencia funcional A → B se cumple en la Relación R si, y solo si, cada vez que dos tuplas tienen el mismo valor para A, éstas también tienen el mismo valor para B.

Dentro de las investigaciones que han tratado de definir un algoritmo que automatice la obtención de dependencias funcionales sobre Relaciones (Estructuras Lógicas de Datos) están las siguientes:

- The Inference Problem for Template Dependencies (Gurevich & Lewis, 1982).
- TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies (Huhtala, Kärkkäinen, Porkka, & Toivonen, 1999).
- FUN: An Efficient Algorithm for Mining Functional and Embedded Dependencies (Novelli & Cicchetti, 2001).
- FD_Mine: Discovering Functional Dependencies in a Database Using Equivalences (Yao, J.Hamilton, & J.Butz, 2002).
- DFD: Efficient Functional Dependency Discovery (Abedjan, Schulze, & Naumann, 2014).

Sin embargo, en el Modelo de Relación Funcional, las Dependencias de Relación Funcional se generan automáticamente al momento de aplicar, sobre una Estructura Lógica de Datos, el concepto matemático de Relación Funcional mediante Restricciones Lógicas Funcionales.

La aplicación de estas restricciones permite obtener el conjunto de dependencias estrictamente válidas, sin necesidad de aplicar algoritmos adicionales.

(4) Como se pudo verificar, la normalización tanto del Modelo Relacional como del Modelo de Relación Funcional, se dio dentro del mismo contexto.

VII. Conclusiones y Recomendaciones

1. Conclusiones

Después de recoger los datos de los resultados obtenidos, se verificó que sí se puede determinar eficazmente la aplicación del Modelo de Datos de Relación Funcional sobre un Modelo de Negocio (en este caso, la empresa de servicios de hospedaje de la ciudad de Trujillo), para que de esta manera se pueda lograr, a partir de Estructuras Lógicas de Datos correspondientes a la información de procesos del año 2014, la automatización de un Diseño de Base de Datos, y cuya eficacia en el caso tratado siempre debe estar sujeta al cumplimiento de Integridad de Datos.

La verificación del cumplimiento de la hipótesis de la investigación se dio en cuatro etapas, las cuales siguen como se explica a continuación.

(1) Los conceptos preliminares que se presentaron ayudaron a establecer y a proponer satisfactoriamente el Modelo de Datos de Relación Funcional. Para lograr la proposición de este modelo se presentó lo siguiente:

- ✓ La definición de una Ontología del Modelo de Negocio a partir de la Lógica de Negocio de una organización.
- ✓ La definición de una Estructura Lógica de Datos a partir de una Ontología del Modeo de Negocio.
- ✓ La definición y proposición de los fundamentos del Modelo de Datos de Relación Funcional.
- ✓ La definición del proceso que permitirá obtener Relaciones Binarias a partir de una Estructura Lógica de Datos que especifica tangiblemente al Modelo de Negocio de un subproceso de la empresa de servicios de hospedaje.
- ✓ La definición de las Restricciones Lógicas que se aplicarán sobre las Relaciones Binarias obtenidas para obtener Dependencias de Relación Funcional.
- ✓ La definición y proposición de los fundamentos de las Dependencias de Relación Funcional.
- ✓ La definición y proposición de las Reglas de Inferencias que se aplican sobre las Dependencias de Relación Funcional, de manera que se puedan obtener el Diseño de Tablas Normalizadas y el Diseño de Tablas No Normalizadas.

- ✓ La obtención de un Diseño de Base de Datos.

(2) Se pudo aplicar correctamente el Modelo de Datos de Relación Funcional sobre las Estructuras Lógicas de Datos de la empresa de servicios de hospedaje, aplicando las Restricciones Lógicas para la Obtención de las Dependencias de Relación Funcional. Para tal cometido, se tuvo que desarrollar lo siguiente:

- ✓ La abstracción del Modelo de Negocio de la empresa de servicios de hospedaje.
- ✓ La definición del esquema de conceptos de la Ontología del Modelo de Negocio de la empresa de servicios de hospedaje.
- ✓ La proposición de un nuevo Modelo de Datos Conceptual, de propia autoría, el cual se está circunscrito en el esquema de conceptos de la Ontología del Modelo de Negocio de la empresa de servicios de hospedaje. Este Modelo de Datos Conceptual se desarrolló y se aplicó, dando como resultado un Diagrama de Datos Conceptual.
- ✓ La definición de una Estructura Lógica de Datos, que obedece estrictamente al Diagrama de Datos Conceptual y la información de los procesos principales, los cuales especificarán explícitamente la Lógica de Negocio de la empresa de servicios de hospedaje. La información de los procesos, sostenida en la Estructura Lógica de Datos, fue correctamente proporcionada por la empresa de servicios de hospedaje.
- ✓ La correcta obtención de Dependencias de Relación Funcional (proceso algorítmico ODRF) en base a la aplicación de Restricciones Lógicas sobre una subestructura definida arbitrariamente (por temas académicos), la cual es obtenida a partir de la Estructura Lógica de Datos definida.
- ✓ La aplicación de un procedimiento (proceso algorítmico RIDRF) que permite la construcción de un Diseño de Base de Datos, permitiéndonos obtener Tablas Normalizadas y Tablas No Normalizadas.

(3) Se validó que el Diseño de Base de Datos construido cumple con la Integridad de Datos, las que se corresponden con las Reglas de Negocio, concluyendo lo siguiente:

- ✓ Se verificó que el Diseño de Base de Datos cumple con las Restricciones de Integridad de Datos.

- ✓ Se verificó que el Diseño de Tablas Normalizadas propias de una Base de Datos de Relación Funcional cumple hasta con la Quinta Forma Normal de un Diseño de Tablas Normalizadas propias de una Base de Datos Relacional.

(4) Se definió correctamente un algoritmo que cubre completamente la automatización de los Procesos Algorítmicos ODRF y RIDRF, pudiendo por lo tanto comprobar que un Diseño de Base de Datos puede ser automatizado a partir de Estructuras Lógicas de Datos que se corresponden con la Ontología de un Modelo de Negocio.

2. Recomendaciones

(1) Para poder aplicar los fundamentos del Modelo de Relación Funcional, se necesita tener en cuenta que el Modelo de Negocio, la Lógica de Negocio, los Conceptos del Negocio y la Información de los Procesos del Negocio deberán estar correctamente entendidos y definidos. De lo contrario, será difícil especificar una Estructura Lógica de Datos que permita aplicar las Dependencias de Relación Funcional.

(2) Siendo que en los casos en donde las organizaciones ya cuenten con un diseño de base de datos relacional que defina la estructura lógica de datos de un sistema de información, el algoritmo presentado en la actual investigación serviría para validar si dicho diseño cumple exactamente tanto con el conocimiento de las reglas del negocio (ya que son éstas las que la organización maneja, entiende y modifica, y que son almacenadas semánticamente como información histórica) como con las restricciones de integridad de datos (debido a que la información de una organización va constantemente creciendo y modificándose en el tiempo).

(3) Habiendo ya definido un algoritmo que automatiza la aplicación de los fundamentos de las Dependencias de Relación Funcional, como parte del Modelo de Datos de Relación Funcional, sobre la información organizacional especificada en Estructuras Lógicas de Datos, se espera que en el futuro se hicieren investigaciones donde se pueda aplicar el algoritmo propuesto en la presente investigación a través de un software comercial que sea capaz de procesar grandes Estructuras Lógicas de Datos de diferentes organizaciones.

VIII. Referencias Bibliográficas

Abedjan, Z., Schulze, P., & Naumann, F. (2014). *DFD: Efficient Functional Dependency Discovery.*

Baader, F. (2003). *The Description Logic Handbook: Theory, implementation, and applications.* Cambridge.

Beizer, D. (22 de July de 2009). *Kundra to agencies: Get ready for data deluge, Emerging technologies will create huge amounts of unstructured data*. Obtenido de washingtontechnology.com: https://washingtontechnology.com/articles/2009/07/22/kundra-data-explosion.aspx

Beynon-Davies, P. (2004). *Database Systems, Third Edition.* Houndmills, Basingstoke, Hampshire RG21 6XS and 175 Fifth Avenue, New York, N.Y. 10010: PALGRAVE MACMILLAN.

Carlos Coronel, S. M. (2011). *Database Systems: Design, Implementation, and Management* (Ninth Edition ed.). © 2011 Cengage Learning.

cl500. (s.f.). *cl500.net.* Obtenido de Basic Database Tutorial: http://www.cl500.net/logicalds.html

Codd, E. F. (June de 1970). A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM, Volume 13*(Number 6), 377-387.

Codd, E. F. (1990). *The Relational Model for Database Management, Version 2.*

Computing Services Information Security Office. (2011). *https://www.cmu.edu*. Obtenido de Information Security Essentials, Carnegie Mellon University: https://www.cmu.edu/iso/aware/presentation/tepperphd.pdf

Date, C. J. (2012). *SQL and Relational Theory: How to Write Accurate SQL Code* (Second Edition ed.). United States of America: Published by O'Reilly Media, Inc.

Date, C. J. (2013). *Relational Theory for Computer Professionals, What Relational Databases Are Really All About.*

Eriksson, H.-E., & Penker, M. (2000). *Business Modeling with UML: Business Patterns at Work; An introduction to the Unified Model Language, and lessons and examples of practical business applications for software developers* (ISBN: 0471295515 ed.). OMG. Obtenido de https://pdfs.semanticscholar.org/88cc/010afdd95ab67d732357e930dfb7a09a2b17.pdf

Faculty of Arts and the Taylor Institute of Teaching. (2017). *Sets, Logic, Computation An Open Logic Text.* Richard Zach, Creative Commons Attribution 4.0 International License.

Gacitua-Decar, V., & Pahl, C. (2010). *Ontology-based Patterns for the Integration of Business Processes and Enterprise Application Architectures.* Dublin City University, School of Computing. Dublin: This chapter appears in "Semantic Enterprise Application Integration for Business Processes" edited by G. Mentzas and A. Friesen, Copyright 2010, IGI Global, www.igi-global.com. Posted by permission of the publisher. Obtenido de https://core.ac.uk/download/pdf/11310862.pdf

Gurevich, Y., & Lewis, H. R. (1982). *The Inference Problem for Template Dependencies.*

Haan, L. d., & Koppelaars, T. (2007). *Applied Mathematics for Database Professionals* (Vols. ISBN-13: 978-1-59059-745-3;). (J. Gennick, Ed.) Springer-Verlag New York, Inc., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax 201-348-4505, e-mail orders-ny@springer-sbm.com, or visit http://www.springeronline.com., New York, United States of America: Apress.

Huhtala, Y., Kärkkäinen, J., Porkka, P., & Toivonen, H. (1999). *TANE: An Efficient Algorithm for Discovering Functional and Approximate Dependencies.*

Infosec Resources. (20 de April de 2018). *Guiding Principles in Information Security*. Obtenido de https://resources.infosecinstitute.com: https://resources.infosecinstitute.com/guiding-principles-in-information-security/#gref

Lewis, D. (2016). *Database systems, Undergraduate study in Computing and related programmes, Volume 1.* London: University of London International Programme.

Monroe, R. (27 de March de 2008). *Data Warehousing II: Extract, Transform, and Load (ETL), BI Tools and Techniques, Carnegie Mellon University*. Obtenido de https://slideplayer.com/slide/7901492/

Neeraj Sharma, L. P.-C. (2010). *Database Fundamentals, A book for the community by the community* (First Edition (November 2010) ed.). IBM Canada, 8200 Warden Avenue, Markham, ON, L6G 1C7, Canada, Canada: © Copyright IBM Corporation 2010. All rights reserved.

Novelli, N., & Cicchetti, R. (2001). *FUN: An Efficient Algorithm for Mining Functional and Embedded Dependencies.*

Ogden, C. K., & Richards, I. A. (1946). *Meaning of Meaning, A study of the Influence of Language upon Thought and the Science of Symbolism.* Cambridge: A Harvest Book, Harcourt, Brace & World, Inc.NEW YORK.

Osterwalder, A. (2004). *THE BUSINESS MODEL ONTOLOGY: A PROPOSITION IN A DESIGN SCIENCE APPROACH.* UNIVERSITE DE LAUSANNE, ECOLE DES HAUTES ETUDES COMMERCIALES. Obtenido de http://www.hec.unil.ch/aosterwa/PhD/Osterwalder_PhD_BM_Ontology.pdf

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). *Database system concepts* (Sixth Edition ed.). (I. The McGraw-Hill Companies, Ed.) New York: The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY 10020.

Teorey, T. J. (2011). *Database Modeling and Design, Logical Design* (Fifth Edition ed.).

Ullman, J. D., & Widom, J. (2008). *A First Course in Database Systems* (Third Edition ed.).

Yao, H., J.Hamilton, H., & J.Butz, C. (2002). *FD_Mine: Discovering Functional Dependencies in a Database Using Equivalences.*

IX. Anexos

1. Anexo 1: Relaciones Binarias (n-aria)

A. Relaciones Binarias de nivel 1-aria

Tabla 18 : Relaciones Binarias de nivel 1-aria

BinaryRelation
1.1.- CodigoCliente, CodigoCliente
1.2.- CodigoCliente, NombreCliente
1.3.- CodigoCliente, CodigoHabitacion
1.4.- CodigoCliente, FechaEntrada
1.5.- CodigoCliente, TipoHabitacion
1.6.- CodigoCliente, PrecioNoche
2.1.- NombreCliente, CodigoCliente
2.2.- NombreCliente, NombreCliente
2.3.- NombreCliente, CodigoHabitacion
2.4.- NombreCliente, FechaEntrada
2.5.- NombreCliente, TipoHabitacion
2.6.- NombreCliente, PrecioNoche
3.1.- CodigoHabitacion, CodigoCliente
3.2.- CodigoHabitacion, NombreCliente
3.3.- CodigoHabitacion, CodigoHabitacion
3.4.- CodigoHabitacion, FechaEntrada
3.5.- CodigoHabitacion, TipoHabitacion
3.6.- CodigoHabitacion, PrecioNoche
4.1.- FechaEntrada, CodigoCliente
4.2.- FechaEntrada, NombreCliente
4.3.- FechaEntrada, CodigoHabitacion
4.4.- FechaEntrada, FechaEntrada
4.5.- FechaEntrada, TipoHabitacion
4.6.- FechaEntrada, PrecioNoche
5.1.- TipoHabitacion, CodigoCliente
5.2.- TipoHabitacion, NombreCliente
5.3.- TipoHabitacion, CodigoHabitacion
5.4.- TipoHabitacion, FechaEntrada
5.5.- TipoHabitacion, TipoHabitacion
5.6.- TipoHabitacion, PrecioNoche
6.1.- PrecioNoche, CodigoCliente
6.2.- PrecioNoche, NombreCliente
6.3.- PrecioNoche, CodigoHabitacion
6.4.- PrecioNoche, FechaEntrada
6.5.- PrecioNoche, TipoHabitacion
6.6.- PrecioNoche, PrecioNoche

Fuente: Elaboración propia.

B. Relaciones Binarias de nivel 2-aria

Tabla 19 : Relaciones Binarias de nivel 2-aria

BinaryRelation
1.1.- {CodigoCliente,NombreCliente},CodigoCliente
1.2.- {CodigoCliente,NombreCliente}, NombreCliente
1.3.- {CodigoCliente,NombreCliente}, CodigoHabitacion
1.4.- {CodigoCliente,NombreCliente}, FechaEntrada
1.5.- {CodigoCliente,NombreCliente}, TipoHabitacion
1.6.- {CodigoCliente,NombreCliente}, PrecioNoche
2.1.- {CodigoCliente,CodigoHabitacion},CodigoCliente
2.2.- {CodigoCliente,CodigoHabitacion}, NombreCliente
2.3.- {CodigoCliente,CodigoHabitacion}, CodigoHabitacion
2.4.- {CodigoCliente,CodigoHabitacion}, FechaEntrada
2.5.- {CodigoCliente,CodigoHabitacion}, TipoHabitacion
2.6.- {CodigoCliente,CodigoHabitacion}, PrecioNoche
3.1.- {CodigoCliente,FechaEntrada}, CodigoCliente
3.2.- {CodigoCliente,FechaEntrada}, NombreCliente
3.3.- {CodigoCliente,FechaEntrada}, CodigoHabitacion
3.4.- {CodigoCliente,FechaEntrada}, FechaEntrada
3.5.- {CodigoCliente,FechaEntrada}, TipoHabitacion
3.6.- {CodigoCliente,FechaEntrada}, PrecioNoche
4.1.- {CodigoCliente,TipoHabitacion},CodigoCliente
4.2.- {CodigoCliente,TipoHabitacion}, NombreCliente
4.3.- {CodigoCliente,TipoHabitacion}, CodigoHabitacion
4.4.- {CodigoCliente,TipoHabitacion}, FechaEntrada
4.5.- {CodigoCliente,TipoHabitacion}, TipoHabitacion
4.6.- {CodigoCliente,TipoHabitacion}, PrecioNoche
5.1.- {CodigoCliente,PrecioNoche}, CodigoCliente
5.2.- {CodigoCliente,PrecioNoche}, NombreCliente
5.3.- {CodigoCliente,PrecioNoche}, CodigoHabitacion
5.4.- {CodigoCliente,PrecioNoche}, FechaEntrada
5.5.- {CodigoCliente,PrecioNoche}, TipoHabitacion
5.6.- {CodigoCliente,PrecioNoche}, PrecioNoche
6.1.- {NombreCliente,CodigoHabitacion}, CodigoCliente
6.2.- {NombreCliente,CodigoHabitacion}, NombreCliente
6.3.- {NombreCliente,CodigoHabitacion}, CodigoHabitacion
6.4.- {NombreCliente,CodigoHabitacion}, FechaEntrada
6.5.- {NombreCliente,CodigoHabitacion}, TipoHabitacion
6.6.- {NombreCliente,CodigoHabitacion}, PrecioNoche
7.1.- {NombreCliente,FechaEntrada}, CodigoCliente
7.2.- {NombreCliente,FechaEntrada}, NombreCliente
7.3.- {NombreCliente,FechaEntrada}, CodigoHabitacion
7.4.- {NombreCliente,FechaEntrada}, FechaEntrada
7.5.- {NombreCliente,FechaEntrada}, TipoHabitacion
7.6.- {NombreCliente,FechaEntrada}, PrecioNoche
8.1.- {NombreCliente,TipoHabitacion}, CodigoCliente
8.2.- {NombreCliente,TipoHabitacion}, NombreCliente
8.3.- {NombreCliente,TipoHabitacion}, CodigoHabitacion
8.4.- {NombreCliente,TipoHabitacion}, FechaEntrada

8.5.- {NombreCliente,TipoHabitacion}, TipoHabitacion
8.6.- {NombreCliente,TipoHabitacion}, PrecioNoche
9.1.- {NombreCliente,PrecioNoche}, CodigoCliente
9.2.- {NombreCliente,PrecioNoche}, NombreCliente
9.3.- {NombreCliente,PrecioNoche}, CodigoHabitacion
9.4.- {NombreCliente,PrecioNoche}, FechaEntrada
9.5.- {NombreCliente,PrecioNoche}, TipoHabitacion
9.6.- {NombreCliente,PrecioNoche}, PrecioNoche
10.1.- {CodigoHabitacion,FechaEntrada}, CodigoCliente
10.2.- {CodigoHabitacion,FechaEntrada}, NombreCliente
10.3.- {CodigoHabitacion,FechaEntrada}, CodigoHabitacion
10.4.- {CodigoHabitacion,FechaEntrada}, FechaEntrada
10.5.- {CodigoHabitacion,FechaEntrada}, TipoHabitacion
10.6.- {CodigoHabitacion,FechaEntrada}, PrecioNoche
11.1.- {CodigoHabitacion,TipoHabitacion}, CodigoCliente
11.2.- {CodigoHabitacion,TipoHabitacion}, NombreCliente
11.3.- {CodigoHabitacion,TipoHabitacion}, CodigoHabitacion
11.4.- {CodigoHabitacion,TipoHabitacion}, FechaEntrada
11.5.- {CodigoHabitacion,TipoHabitacion}, TipoHabitacion
11.6.- {CodigoHabitacion,TipoHabitacion}, PrecioNoche
12.1.- {CodigoHabitacion,PrecioNoche}, CodigoCliente
12.2.- {CodigoHabitacion,PrecioNoche}, NombreCliente
12.3.- {CodigoHabitacion,PrecioNoche}, CodigoHabitacion
12.4.- {CodigoHabitacion,PrecioNoche}, FechaEntrada
12.5.- {CodigoHabitacion,PrecioNoche}, TipoHabitacion
12.6.- {CodigoHabitacion,PrecioNoche}, PrecioNoche
13.1.- {FechaEntrada,TipoHabitacion}, CodigoCliente
13.2.- {FechaEntrada,TipoHabitacion}, NombreCliente
13.3.- {FechaEntrada,TipoHabitacion}, CodigoHabitacion
13.4.- {FechaEntrada,TipoHabitacion}, FechaEntrada
13.5.- {FechaEntrada,TipoHabitacion}, TipoHabitacion
13.6.- {FechaEntrada,TipoHabitacion}, PrecioNoche
14.1.- {FechaEntrada,PrecioNoche}, CodigoCliente
14.2.- {FechaEntrada,PrecioNoche}, NombreCliente
14.3.- {FechaEntrada,PrecioNoche}, CodigoHabitacion
14.4.- {FechaEntrada,PrecioNoche}, FechaEntrada
14.5.- {FechaEntrada,PrecioNoche}, TipoHabitacion
14.6.- {FechaEntrada,PrecioNoche}, PrecioNoche
15.1.- {TipoHabitacion,PrecioNoche}, CodigoCliente
15.2.- {TipoHabitacion,PrecioNoche}, NombreCliente
15.3.- {TipoHabitacion,PrecioNoche}, CodigoHabitacion
15.4.- {TipoHabitacion,PrecioNoche}, FechaEntrada
15.5.- {TipoHabitacion,PrecioNoche}, TipoHabitacion
15.6.- {TipoHabitacion,PrecioNoche}, PrecioNoche

Fuente: Elaboración propia.

C. Relaciones Binarias de nivel 3-aria

Tabla 20 : Relaciones Binarias de nivel 3-aria

BinaryRelation

1.1.- {CodigoCliente,NombreCliente,CodigoHabitacion}, CodigoCliente
1.2.- {CodigoCliente,NombreCliente,CodigoHabitacion}, NombreCliente
1.3.- {CodigoCliente,NombreCliente,CodigoHabitacion}, CodigoHabitacion
1.4.- {CodigoCliente,NombreCliente,CodigoHabitacion}, FechaEntrada
1.5.- {CodigoCliente,NombreCliente,CodigoHabitacion}, TipoHabitacion
1.6.- {CodigoCliente,NombreCliente,CodigoHabitacion}, PrecioNoche
2.1.- {CodigoCliente,NombreCliente,FechaEntrada}, CodigoCliente
2.2.- {CodigoCliente,NombreCliente,FechaEntrada}, NombreCliente
2.3.- {CodigoCliente,NombreCliente,FechaEntrada}, CodigoHabitacion
2.4.- {CodigoCliente,NombreCliente,FechaEntrada}, FechaEntrada
2.5.- {CodigoCliente,NombreCliente,FechaEntrada}, TipoHabitacion
2.6.- {CodigoCliente,NombreCliente,FechaEntrada}, PrecioNoche
3.1.- {CodigoCliente,NombreCliente,TipoHabitacion}, CodigoCliente
3.2.- {CodigoCliente,NombreCliente,TipoHabitacion}, NombreCliente
3.3.- {CodigoCliente,NombreCliente,TipoHabitacion}, CodigoHabitacion
3.4.- {CodigoCliente,NombreCliente,TipoHabitacion}, FechaEntrada
3.5.- {CodigoCliente,NombreCliente,TipoHabitacion}, TipoHabitacion
3.6.- {CodigoCliente,NombreCliente,TipoHabitacion}, PrecioNoche
4.1.- {CodigoCliente,NombreCliente,PrecioNoche}, CodigoCliente
4.2.- {CodigoCliente,NombreCliente,PrecioNoche}, NombreCliente
4.3.- {CodigoCliente,NombreCliente,PrecioNoche}, CodigoHabitacion
4.4.- {CodigoCliente,NombreCliente,PrecioNoche}, FechaEntrada
4.5.- {CodigoCliente,NombreCliente,PrecioNoche}, TipoHabitacion
4.6.- {CodigoCliente,NombreCliente,PrecioNoche}, PrecioNoche
5.1.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, CodigoCliente
5.2.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, NombreCliente
5.3.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, CodigoHabitacion
5.4.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, FechaEntrada
5.5.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, TipoHabitacion
5.6.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, PrecioNoche
6.1.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, CodigoCliente
6.2.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, NombreCliente
6.3.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, CodigoHabitacion
6.4.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, FechaEntrada
6.5.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, TipoHabitacion
6.6.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, PrecioNoche
7.1.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, CodigoCliente
7.2.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, NombreCliente
7.3.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, CodigoHabitacion
7.4.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, FechaEntrada
7.5.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, TipoHabitacion
7.6.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, PrecioNoche
8.1.- {CodigoCliente,FechaEntrada,TipoHabitacion}, CodigoCliente
8.2.- {CodigoCliente,FechaEntrada,TipoHabitacion}, NombreCliente
8.3.- {CodigoCliente,FechaEntrada,TipoHabitacion}, CodigoHabitacion
8.4.- {CodigoCliente,FechaEntrada,TipoHabitacion}, FechaEntrada
8.5.- {CodigoCliente,FechaEntrada,TipoHabitacion}, TipoHabitacion
8.6.- {CodigoCliente,FechaEntrada,TipoHabitacion}, PrecioNoche
9.1.- {CodigoCliente,FechaEntrada,PrecioNoche}, CodigoCliente
9.2.- {CodigoCliente,FechaEntrada,PrecioNoche}, NombreCliente
9.3.- {CodigoCliente,FechaEntrada,PrecioNoche}, CodigoHabitacion
9.4.- {CodigoCliente,FechaEntrada,PrecioNoche}, FechaEntrada
9.5.- {CodigoCliente,FechaEntrada,PrecioNoche}, TipoHabitacion
9.6.- {CodigoCliente,FechaEntrada,PrecioNoche}, PrecioNoche
10.1.- {CodigoCliente,TipoHabitacion,PrecioNoche}, CodigoCliente
10.2.- {CodigoCliente,TipoHabitacion,PrecioNoche}, NombreCliente

10.3.- {CodigoCliente,TipoHabitacion,PrecioNoche}, CodigoHabitacion
10.4.- {CodigoCliente,TipoHabitacion,PrecioNoche}, FechaEntrada
10.5.- {CodigoCliente,TipoHabitacion,PrecioNoche}, TipoHabitacion
10.6.- {CodigoCliente,TipoHabitacion,PrecioNoche}, PrecioNoche
11.1.- {NombreCliente,CodigoHabitacion,FechaEntrada}, CodigoCliente
11.2.- {NombreCliente,CodigoHabitacion,FechaEntrada}, NombreCliente
11.3.- {NombreCliente,CodigoHabitacion,FechaEntrada}, CodigoHabitacion
11.4.- {NombreCliente,CodigoHabitacion,FechaEntrada}, FechaEntrada
11.5.- {NombreCliente,CodigoHabitacion,FechaEntrada}, TipoHabitacion
11.6.- {NombreCliente,CodigoHabitacion,FechaEntrada}, PrecioNoche
12.1.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, CodigoCliente
12.2.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, NombreCliente
12.3.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, CodigoHabitacion
12.4.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, FechaEntrada
12.5.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, TipoHabitacion
12.6.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, PrecioNoche
13.1.- {NombreCliente,CodigoHabitacion,PrecioNoche}, CodigoCliente
13.2.- {NombreCliente,CodigoHabitacion,PrecioNoche}, NombreCliente
13.3.- {NombreCliente,CodigoHabitacion,PrecioNoche}, CodigoHabitacion
13.4.- {NombreCliente,CodigoHabitacion,PrecioNoche}, FechaEntrada
13.5.- {NombreCliente,CodigoHabitacion,PrecioNoche}, TipoHabitacion
13.6.- {NombreCliente,CodigoHabitacion,PrecioNoche}, PrecioNoche
14.1.- {NombreCliente,FechaEntrada,TipoHabitacion}, CodigoCliente
14.2.- {NombreCliente,FechaEntrada,TipoHabitacion}, NombreCliente
14.3.- {NombreCliente,FechaEntrada,TipoHabitacion}, CodigoHabitacion
14.4.- {NombreCliente,FechaEntrada,TipoHabitacion}, FechaEntrada
14.5.- {NombreCliente,FechaEntrada,TipoHabitacion}, TipoHabitacion
14.6.- {NombreCliente,FechaEntrada,TipoHabitacion}, PrecioNoche
15.1.- {NombreCliente,FechaEntrada,PrecioNoche}, CodigoCliente
15.2.- {NombreCliente,FechaEntrada,PrecioNoche}, NombreCliente
15.3.- {NombreCliente,FechaEntrada,PrecioNoche}, CodigoHabitacion
15.4.- {NombreCliente,FechaEntrada,PrecioNoche}, FechaEntrada
15.5.- {NombreCliente,FechaEntrada,PrecioNoche}, TipoHabitacion
15.6.- {NombreCliente,FechaEntrada,PrecioNoche}, PrecioNoche
16.1.- {NombreCliente,TipoHabitacion,PrecioNoche}, CodigoCliente
16.2.- {NombreCliente,TipoHabitacion,PrecioNoche}, NombreCliente
16.3.- {NombreCliente,TipoHabitacion,PrecioNoche}, CodigoHabitacion
16.4.- {NombreCliente,TipoHabitacion,PrecioNoche}, FechaEntrada
16.5.- {NombreCliente,TipoHabitacion,PrecioNoche}, TipoHabitacion
16.6.- {NombreCliente,TipoHabitacion,PrecioNoche}, PrecioNoche
17.1.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoCliente
17.2.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, NombreCliente
17.3.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoHabitacion
17.4.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, FechaEntrada
17.5.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, TipoHabitacion
17.6.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, PrecioNoche
18.1.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoCliente
18.2.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, NombreCliente
18.3.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoHabitacion
18.4.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, FechaEntrada
18.5.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, TipoHabitacion
18.6.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, PrecioNoche
19.1.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoCliente
19.2.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, NombreCliente
19.3.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoHabitacion
19.4.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, FechaEntrada

19.5.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, TipoHabitacion
19.6.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, PrecioNoche
20.1.- {FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente
20.2.- {FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente
20.3.- {FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion
20.4.- {FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada
20.5.- {FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion
20.6.- {FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche

Fuente: Elaboración propia.

D. Relaciones Binarias de nivel 4-aria

Tabla 21 : Relaciones Binarias de nivel 4-aria

BinaryRelation
1.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, CodigoCliente
1.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, NombreCliente
1.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, CodigoHabitacion
1.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, FechaEntrada
1.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, TipoHabitacion
1.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, PrecioNoche
2.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, CodigoCliente
2.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, NombreCliente
2.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, CodigoHabitacion
2.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, FechaEntrada
2.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, TipoHabitacion
2.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, PrecioNoche
3.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, CodigoCliente
3.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, NombreCliente
3.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, CodigoHabitacion
3.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, FechaEntrada
3.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, TipoHabitacion
3.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, PrecioNoche
4.1.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, CodigoCliente
4.2.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, NombreCliente
4.3.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, CodigoHabitacion
4.4.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, FechaEntrada
4.5.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, TipoHabitacion
4.6.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, PrecioNoche
5.1.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, CodigoCliente
5.2.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, NombreCliente
5.3.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, CodigoHabitacion
5.4.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, FechaEntrada
5.5.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, TipoHabitacion
5.6.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, PrecioNoche
6.1.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, CodigoCliente
6.2.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, NombreCliente
6.3.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, CodigoHabitacion
6.4.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, FechaEntrada
6.5.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, TipoHabitacion
6.6.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, PrecioNoche

7.1.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoCliente
7.2.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, NombreCliente
7.3.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoHabitacion
7.4.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, FechaEntrada
7.5.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, TipoHabitacion
7.6.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, PrecioNoche
8.1.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoCliente
8.2.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, NombreCliente
8.3.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoHabitacion
8.4.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, FechaEntrada
8.5.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, TipoHabitacion
8.6.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, PrecioNoche
9.1.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoCliente
9.2.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, NombreCliente
9.3.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoHabitacion
9.4.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, FechaEntrada
9.5.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, TipoHabitacion
9.6.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, PrecioNoche
10.1.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente
10.2.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente
10.3.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion
10.4.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada
10.5.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion
10.6.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche
11.1.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoCliente
11.2.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, NombreCliente
11.3.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoHabitacion
11.4.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, FechaEntrada
11.5.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, TipoHabitacion
11.6.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, PrecioNoche
12.1.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoCliente
12.2.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, NombreCliente
12.3.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoHabitacion
12.4.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, FechaEntrada
12.5.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, TipoHabitacion
12.6.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, PrecioNoche
13.1.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoCliente
13.2.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, NombreCliente
13.3.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoHabitacion
13.4.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, FechaEntrada
13.5.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, TipoHabitacion
13.6.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, PrecioNoche
14.1.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente
14.2.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente
14.3.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion
14.4.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada
14.5.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion
14.6.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche
15.1.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente
15.2.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente
15.3.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion
15.4.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada
15.5.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion
15.6.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche

Fuente: Elaboración propia.

E. Relaciones Binarias de nivel 5-aria

Tabla 22 : Relaciones Binarias de nivel 5-aria

BinaryRelation
1.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},CodigoCliente
1.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},NombreCliente
1.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},CodigoHabitacion
1.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},FechaEntrada
1.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},TipoHabitacion
1.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},PrecioNoche
2.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoCliente
2.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, NombreCliente
2.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoHabitacion
2.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, FechaEntrada
2.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, TipoHabitacion
2.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, PrecioNoche
3.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoCliente
3.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, NombreCliente
3.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoHabitacion
3.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, FechaEntrada
3.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, TipoHabitacion
3.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, PrecioNoche
4.1.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente
4.2.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente
4.3.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion
4.4.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada
4.5.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion
4.6.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche
5.1.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente
5.2.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente
5.3.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion
5.4.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada
5.5.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion
5.6.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche
6.1.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente
6.2.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente
6.3.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion
6.4.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada
6.5.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion
6.6.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche

Fuente: Elaboración propia.

2. Anexo 2: Cardinalidades de las Proyecciones de las Tuplas

A. Cardinalidad de las Proyecciones de las Tuplas sobre las Relaciones Binarias de nivel 1-aria

Tabla 23 : Cardinalidad de las Proyecciones de nivel 1-aria

BinaryRelation	|ΠAB(R)|	|ΠA(R)|	|ΠB(R)|
1.1.- CodigoCliente, CodigoCliente	7	7	7
1.2.- CodigoCliente, NombreCliente	7	7	6
1.3.- CodigoCliente, CodigoHabitacion	18	7	6
1.4.- CodigoCliente, FechaEntrada	20	7	10
1.5.- CodigoCliente, TipoHabitacion	15	7	3
1.6.- CodigoCliente, PrecioNoche	15	7	3
2.1.- NombreCliente, CodigoCliente	7	6	7
2.2.- NombreCliente, NombreCliente	6	6	6
2.3.- NombreCliente, CodigoHabitacion	18	6	6
2.4.- NombreCliente, FechaEntrada	19	6	10
2.5.- NombreCliente, TipoHabitacion	13	6	3
2.6.- NombreCliente, PrecioNoche	13	6	3
3.1.- CodigoHabitacion, CodigoCliente	18	6	7
3.2.- CodigoHabitacion, NombreCliente	18	6	6
3.3.- CodigoHabitacion, CodigoHabitacion	6	6	6
3.4.- CodigoHabitacion, FechaEntrada	24	6	10
3.5.- CodigoHabitacion, TipoHabitacion	6	6	3
3.6.- CodigoHabitacion, PrecioNoche	6	6	3
4.1.- FechaEntrada, CodigoCliente	20	10	7
4.2.- FechaEntrada, NombreCliente	19	10	6
4.3.- FechaEntrada, CodigoHabitacion	24	10	6
4.4.- FechaEntrada, FechaEntrada	10	10	10
4.5.- FechaEntrada, TipoHabitacion	17	10	3
4.6.- FechaEntrada, PrecioNoche	17	10	3
5.1.- TipoHabitacion, CodigoCliente	15	3	7
5.2.- TipoHabitacion, NombreCliente	13	3	6
5.3.- TipoHabitacion, CodigoHabitacion	6	3	6
5.4.- TipoHabitacion, FechaEntrada	17	3	10
5.5.- TipoHabitacion, TipoHabitacion	3	3	3
5.6.- TipoHabitacion, PrecioNoche	3	3	3
6.1.- PrecioNoche, CodigoCliente	15	3	7
6.2.- PrecioNoche, NombreCliente	13	3	6
6.3.- PrecioNoche, CodigoHabitacion	6	3	6
6.4.- PrecioNoche, FechaEntrada	17	3	10
6.5.- PrecioNoche, TipoHabitacion	3	3	3
6.6.- PrecioNoche, PrecioNoche	3	3	3

Fuente: Elaboración propia.

B. Cardinalidad de las Proyecciones de las Tuplas sobre las Relaciones Binarias de nivel 2-aria

Tabla 24 : Cardinalidad de las Proyecciones de nivel 2-aria

BinaryRelation	|ΠAB(R)|	|ΠA(R)|	|ΠB(R)|
1.1.- {CodigoCliente,NombreCliente},CodigoCliente	7	7	7
1.2.- {CodigoCliente,NombreCliente}, NombreCliente	7	7	6
1.3.- {CodigoCliente,NombreCliente}, CodigoHabitacion	18	7	6
1.4.- {CodigoCliente,NombreCliente}, FechaEntrada	20	7	10
1.5.- {CodigoCliente,NombreCliente}, TipoHabitacion	15	7	3
1.6.- {CodigoCliente,NombreCliente}, PrecioNoche	15	7	3
2.1.- {CodigoCliente,CodigoHabitacion},CodigoCliente	18	18	7
2.2.- {CodigoCliente,CodigoHabitacion}, NombreCliente	18	18	6
2.3.- {CodigoCliente,CodigoHabitacion}, CodigoHabitacion	18	18	6
2.4.- {CodigoCliente,CodigoHabitacion}, FechaEntrada	24	18	10
2.5.- {CodigoCliente,CodigoHabitacion}, TipoHabitacion	18	18	3
2.6.- {CodigoCliente,CodigoHabitacion}, PrecioNoche	18	18	3
3.1.- {CodigoCliente,FechaEntrada}, CodigoCliente	20	20	7
3.2.- {CodigoCliente,FechaEntrada}, NombreCliente	20	20	6
3.3.- {CodigoCliente,FechaEntrada}, CodigoHabitacion	24	20	6
3.4.- {CodigoCliente,FechaEntrada}, FechaEntrada	20	20	10
3.5.- {CodigoCliente,FechaEntrada}, TipoHabitacion	23	20	3
3.6.- {CodigoCliente,FechaEntrada}, PrecioNoche	23	20	3
4.1.- {CodigoCliente,TipoHabitacion},CodigoCliente	15	15	7
4.2.- {CodigoCliente,TipoHabitacion}, NombreCliente	15	15	6
4.3.- {CodigoCliente,TipoHabitacion}, CodigoHabitacion	18	15	6
4.4.- {CodigoCliente,TipoHabitacion}, FechaEntrada	23	15	10
4.5.- {CodigoCliente,TipoHabitacion}, TipoHabitacion	15	15	3
4.6.- {CodigoCliente,TipoHabitacion}, PrecioNoche	15	15	3
5.1.- {CodigoCliente,PrecioNoche}, CodigoCliente	15	15	7
5.2.- {CodigoCliente,PrecioNoche}, NombreCliente	15	15	6
5.3.- {CodigoCliente,PrecioNoche}, CodigoHabitacion	18	15	6
5.4.- {CodigoCliente,PrecioNoche}, FechaEntrada	23	15	10
5.5.- {CodigoCliente,PrecioNoche}, TipoHabitacion	15	15	3
5.6.- {CodigoCliente,PrecioNoche}, PrecioNoche	15	15	3
6.1.- {NombreCliente,CodigoHabitacion}, CodigoCliente	18	18	7
6.2.- {NombreCliente,CodigoHabitacion}, NombreCliente	18	18	6
6.3.- {NombreCliente,CodigoHabitacion}, CodigoHabitacion	18	18	6
6.4.- {NombreCliente,CodigoHabitacion}, FechaEntrada	24	18	10
6.5.- {NombreCliente,CodigoHabitacion}, TipoHabitacion	18	18	3
6.6.- {NombreCliente,CodigoHabitacion}, PrecioNoche	18	18	3
7.1.- {NombreCliente,FechaEntrada}, CodigoCliente	20	19	7
7.2.- {NombreCliente,FechaEntrada}, NombreCliente	19	19	6
7.3.- {NombreCliente,FechaEntrada}, CodigoHabitacion	24	19	6
7.4.- {NombreCliente,FechaEntrada}, FechaEntrada	19	19	10
7.5.- {NombreCliente,FechaEntrada}, TipoHabitacion	22	19	3
7.6.- {NombreCliente,FechaEntrada}, PrecioNoche	22	19	3
8.1.- {NombreCliente,TipoHabitacion}, CodigoCliente	15	13	7
8.2.- {NombreCliente,TipoHabitacion}, NombreCliente	13	13	6
8.3.- {NombreCliente,TipoHabitacion}, CodigoHabitacion	18	13	6
8.4.- {NombreCliente,TipoHabitacion}, FechaEntrada	22	13	10
8.5.- {NombreCliente,TipoHabitacion}, TipoHabitacion	13	13	3
8.6.- {NombreCliente,TipoHabitacion}, PrecioNoche	13	13	3
9.1.- {NombreCliente,PrecioNoche}, CodigoCliente	15	13	7
9.2.- {NombreCliente,PrecioNoche}, NombreCliente	13	13	6
9.3.- {NombreCliente,PrecioNoche}, CodigoHabitacion	18	13	6
9.4.- {NombreCliente,PrecioNoche}, FechaEntrada	22	13	10
9.5.- {NombreCliente,PrecioNoche}, TipoHabitacion	13	13	3
9.6.- {NombreCliente,PrecioNoche}, PrecioNoche	13	13	3
10.1.- {CodigoHabitacion,FechaEntrada}, CodigoCliente	24	24	7

10.2.- {CodigoHabitacion,FechaEntrada}, NombreCliente	24	24	6
10.3.- {CodigoHabitacion,FechaEntrada}, CodigoHabitacion	24	24	6
10.4.- {CodigoHabitacion,FechaEntrada}, FechaEntrada	24	24	10
10.5.- {CodigoHabitacion,FechaEntrada}, TipoHabitacion	24	24	3
10.6.- {CodigoHabitacion,FechaEntrada}, PrecioNoche	24	24	3
11.1.- {CodigoHabitacion,TipoHabitacion}, CodigoCliente	18	6	7
11.2.- {CodigoHabitacion,TipoHabitacion}, NombreCliente	18	6	6
11.3.- {CodigoHabitacion,TipoHabitacion}, CodigoHabitacion	6	6	6
11.4.- {CodigoHabitacion,TipoHabitacion}, FechaEntrada	24	6	10
11.5.- {CodigoHabitacion,TipoHabitacion}, TipoHabitacion	6	6	3
11.6.- {CodigoHabitacion,TipoHabitacion}, PrecioNoche	6	6	3
12.1.- {CodigoHabitacion,PrecioNoche}, CodigoCliente	18	6	7
12.2.- {CodigoHabitacion,PrecioNoche}, NombreCliente	18	6	6
12.3.- {CodigoHabitacion,PrecioNoche}, CodigoHabitacion	6	6	6
12.4.- {CodigoHabitacion,PrecioNoche}, FechaEntrada	24	6	10
12.5.- {CodigoHabitacion,PrecioNoche}, TipoHabitacion	6	6	3
12.6.- {CodigoHabitacion,PrecioNoche}, PrecioNoche	6	6	3
13.1.- {FechaEntrada,TipoHabitacion}, CodigoCliente	23	17	7
13.2.- {FechaEntrada,TipoHabitacion}, NombreCliente	22	17	6
13.3.- {FechaEntrada,TipoHabitacion}, CodigoHabitacion	24	17	6
13.4.- {FechaEntrada,TipoHabitacion}, FechaEntrada	17	17	10
13.5.- {FechaEntrada,TipoHabitacion}, TipoHabitacion	17	17	3
13.6.- {FechaEntrada,TipoHabitacion}, PrecioNoche	17	17	3
14.1.- {FechaEntrada,PrecioNoche}, CodigoCliente	23	17	7
14.2.- {FechaEntrada,PrecioNoche}, NombreCliente	22	17	6
14.3.- {FechaEntrada,PrecioNoche}, CodigoHabitacion	24	17	6
14.4.- {FechaEntrada,PrecioNoche}, FechaEntrada	17	17	10
14.5.- {FechaEntrada,PrecioNoche}, TipoHabitacion	17	17	3
14.6.- {FechaEntrada,PrecioNoche}, PrecioNoche	17	17	3
15.1.- {TipoHabitacion,PrecioNoche}, CodigoCliente	15	3	7
15.2.- {TipoHabitacion,PrecioNoche}, NombreCliente	13	3	6
15.3.- {TipoHabitacion,PrecioNoche}, CodigoHabitacion	6	3	6
15.4.- {TipoHabitacion,PrecioNoche}, FechaEntrada	17	3	10
15.5.- {TipoHabitacion,PrecioNoche}, TipoHabitacion	3	3	3
15.6.- {TipoHabitacion,PrecioNoche}, PrecioNoche	3	3	3

Fuente: Elaboración propia.

C. Cardinalidad de las Proyecciones de las Tuplas sobre las Relaciones Binarias de nivel 3-aria

Tabla 25 : Cardinalidad de las Proyecciones de nivel 3-aria

BinaryRelation	|ΠAB(R)|	|ΠA(R)|	|ΠB(R)|
1.1.- {CodigoCliente,NombreCliente,CodigoHabitacion}, CodigoCliente	18	18	7
1.2.- {CodigoCliente,NombreCliente,CodigoHabitacion}, NombreCliente	18	18	6
1.3.- {CodigoCliente,NombreCliente,CodigoHabitacion}, CodigoHabitacion	18	18	6
1.4.- {CodigoCliente,NombreCliente,CodigoHabitacion}, FechaEntrada	24	18	10
1.5.- {CodigoCliente,NombreCliente,CodigoHabitacion}, TipoHabitacion	18	18	3
1.6.- {CodigoCliente,NombreCliente,CodigoHabitacion}, PrecioNoche	18	18	3
2.1.- {CodigoCliente,NombreCliente,FechaEntrada}, CodigoCliente	20	20	7

2.2.- {CodigoCliente,NombreCliente,FechaEntrada}, NombreCliente	20	20	6
2.3.- {CodigoCliente,NombreCliente,FechaEntrada}, CodigoHabitacion	24	20	6
2.4.- {CodigoCliente,NombreCliente,FechaEntrada}, FechaEntrada	20	20	10
2.5.- {CodigoCliente,NombreCliente,FechaEntrada}, TipoHabitacion	23	20	3
2.6.- {CodigoCliente,NombreCliente,FechaEntrada}, PrecioNoche	23	20	3
3.1.- {CodigoCliente,NombreCliente,TipoHabitacion}, CodigoCliente	15	15	7
3.2.- {CodigoCliente,NombreCliente,TipoHabitacion}, NombreCliente	15	15	6
3.3.- {CodigoCliente,NombreCliente,TipoHabitacion}, CodigoHabitacion	18	15	6
3.4.- {CodigoCliente,NombreCliente,TipoHabitacion}, FechaEntrada	23	15	10
3.5.- {CodigoCliente,NombreCliente,TipoHabitacion}, TipoHabitacion	15	15	3
3.6.- {CodigoCliente,NombreCliente,TipoHabitacion}, PrecioNoche	15	15	3
4.1.- {CodigoCliente,NombreCliente,PrecioNoche}, CodigoCliente	15	15	7
4.2.- {CodigoCliente,NombreCliente,PrecioNoche}, NombreCliente	15	15	6
4.3.- {CodigoCliente,NombreCliente,PrecioNoche}, CodigoHabitacion	18	15	6
4.4.- {CodigoCliente,NombreCliente,PrecioNoche}, FechaEntrada	23	15	10
4.5.- {CodigoCliente,NombreCliente,PrecioNoche}, TipoHabitacion	15	15	3
4.6.- {CodigoCliente,NombreCliente,PrecioNoche}, PrecioNoche	15	15	3
5.1.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, CodigoCliente	24	24	7
5.2.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, NombreCliente	24	24	6
5.3.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, CodigoHabitacion	24	24	6
5.4.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, FechaEntrada	24	24	10
5.5.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, TipoHabitacion	24	24	3
5.6.- {CodigoCliente,CodigoHabitacion,FechaEntrada}, PrecioNoche	24	24	3
6.1.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, CodigoCliente	18	18	7
6.2.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, NombreCliente	18	18	6
6.3.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, CodigoHabitacion	18	18	6
6.4.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, FechaEntrada	24	18	10
6.5.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, TipoHabitacion	18	18	3
6.6.- {CodigoCliente,CodigoHabitacion,TipoHabitacion}, PrecioNoche	18	18	3
7.1.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, CodigoCliente	18	18	7
7.2.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, NombreCliente	18	18	6
7.3.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, CodigoHabitacion	18	18	6
7.4.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, FechaEntrada	24	18	10
7.5.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, TipoHabitacion	18	18	3
7.6.- {CodigoCliente,CodigoHabitacion,PrecioNoche}, PrecioNoche	18	18	3
8.1.- {CodigoCliente,FechaEntrada,TipoHabitacion}, CodigoCliente	23	23	7
8.2.- {CodigoCliente,FechaEntrada,TipoHabitacion}, NombreCliente	23	23	6
8.3.- {CodigoCliente,FechaEntrada,TipoHabitacion}, CodigoHabitacion	24	23	6
8.4.- {CodigoCliente,FechaEntrada,TipoHabitacion}, FechaEntrada	23	23	10
8.5.- {CodigoCliente,FechaEntrada,TipoHabitacion}, TipoHabitacion	23	23	3
8.6.- {CodigoCliente,FechaEntrada,TipoHabitacion}, PrecioNoche	23	23	3
9.1.- {CodigoCliente,FechaEntrada,PrecioNoche}, CodigoCliente	23	23	7
9.2.- {CodigoCliente,FechaEntrada,PrecioNoche}, NombreCliente	23	23	6
9.3.- {CodigoCliente,FechaEntrada,PrecioNoche}, CodigoHabitacion	24	23	6
9.4.- {CodigoCliente,FechaEntrada,PrecioNoche}, FechaEntrada	23	23	10
9.5.- {CodigoCliente,FechaEntrada,PrecioNoche}, TipoHabitacion	23	23	3
9.6.- {CodigoCliente,FechaEntrada,PrecioNoche}, PrecioNoche	23	23	3
10.1.- {CodigoCliente,TipoHabitacion,PrecioNoche}, CodigoCliente	15	15	7
10.2.- {CodigoCliente,TipoHabitacion,PrecioNoche}, NombreCliente	15	15	6
10.3.- {CodigoCliente,TipoHabitacion,PrecioNoche}, CodigoHabitacion	18	15	6
10.4.- {CodigoCliente,TipoHabitacion,PrecioNoche}, FechaEntrada	23	15	10
10.5.- {CodigoCliente,TipoHabitacion,PrecioNoche}, TipoHabitacion	15	15	3
10.6.- {CodigoCliente,TipoHabitacion,PrecioNoche}, PrecioNoche	15	15	3
11.1.- {NombreCliente,CodigoHabitacion,FechaEntrada}, CodigoCliente	24	24	7
11.2.- {NombreCliente,CodigoHabitacion,FechaEntrada}, NombreCliente	24	24	6
11.3.- {NombreCliente,CodigoHabitacion,FechaEntrada}, CodigoHabitacion	24	24	6

11.4.- {NombreCliente,CodigoHabitacion,FechaEntrada}, FechaEntrada	24	24	10
11.5.- {NombreCliente,CodigoHabitacion,FechaEntrada}, TipoHabitacion	24	24	3
11.6.- {NombreCliente,CodigoHabitacion,FechaEntrada}, PrecioNoche	24	24	3
12.1.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, CodigoCliente	18	18	7
12.2.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, NombreCliente	18	18	6
12.3.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, CodigoHabitacion	18	18	6
12.4.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, FechaEntrada	24	18	10
12.5.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, TipoHabitacion	18	18	3
12.6.- {NombreCliente,CodigoHabitacion,TipoHabitacion}, PrecioNoche	18	18	3
13.1.- {NombreCliente,CodigoHabitacion,PrecioNoche}, CodigoCliente	18	18	7
13.2.- {NombreCliente,CodigoHabitacion,PrecioNoche}, NombreCliente	18	18	6
13.3.- {NombreCliente,CodigoHabitacion,PrecioNoche}, CodigoHabitacion	18	18	6
13.4.- {NombreCliente,CodigoHabitacion,PrecioNoche}, FechaEntrada	24	18	10
13.5.- {NombreCliente,CodigoHabitacion,PrecioNoche}, TipoHabitacion	18	18	3
13.6.- {NombreCliente,CodigoHabitacion,PrecioNoche}, PrecioNoche	18	18	3
14.1.- {NombreCliente,FechaEntrada,TipoHabitacion}, CodigoCliente	23	22	7
14.2.- {NombreCliente,FechaEntrada,TipoHabitacion}, NombreCliente	22	22	6
14.3.- {NombreCliente,FechaEntrada,TipoHabitacion}, CodigoHabitacion	24	22	6
14.4.- {NombreCliente,FechaEntrada,TipoHabitacion}, FechaEntrada	22	22	10
14.5.- {NombreCliente,FechaEntrada,TipoHabitacion}, TipoHabitacion	22	22	3
14.6.- {NombreCliente,FechaEntrada,TipoHabitacion}, PrecioNoche	22	22	3
15.1.- {NombreCliente,FechaEntrada,PrecioNoche}, CodigoCliente	23	22	7
15.2.- {NombreCliente,FechaEntrada,PrecioNoche}, NombreCliente	22	22	6
15.3.- {NombreCliente,FechaEntrada,PrecioNoche}, CodigoHabitacion	24	22	6
15.4.- {NombreCliente,FechaEntrada,PrecioNoche}, FechaEntrada	22	22	10
15.5.- {NombreCliente,FechaEntrada,PrecioNoche}, TipoHabitacion	22	22	3
15.6.- {NombreCliente,FechaEntrada,PrecioNoche}, PrecioNoche	22	22	3
16.1.- {NombreCliente,TipoHabitacion,PrecioNoche}, CodigoCliente	15	13	7
16.2.- {NombreCliente,TipoHabitacion,PrecioNoche}, NombreCliente	13	13	6
16.3.- {NombreCliente,TipoHabitacion,PrecioNoche}, CodigoHabitacion	18	13	6
16.4.- {NombreCliente,TipoHabitacion,PrecioNoche}, FechaEntrada	22	13	10
16.5.- {NombreCliente,TipoHabitacion,PrecioNoche}, TipoHabitacion	13	13	3
16.6.- {NombreCliente,TipoHabitacion,PrecioNoche}, PrecioNoche	13	13	3
17.1.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoCliente	24	24	7
17.2.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, NombreCliente	24	24	6
17.3.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoHabitacion	24	24	6
17.4.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, FechaEntrada	24	24	10
17.5.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, TipoHabitacion	24	24	3
17.6.- {CodigoHabitacion,FechaEntrada,TipoHabitacion}, PrecioNoche	24	24	3
18.1.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoCliente	24	24	7
18.2.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, NombreCliente	24	24	6
18.3.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoHabitacion	24	24	6
18.4.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, FechaEntrada	24	24	10
18.5.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, TipoHabitacion	24	24	3
18.6.- {CodigoHabitacion,FechaEntrada,PrecioNoche}, PrecioNoche	24	24	3
19.1.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoCliente	18	6	7
19.2.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, NombreCliente	18	6	6
19.3.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoHabitacion	6	6	6
19.4.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, FechaEntrada	24	6	10
19.5.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, TipoHabitacion	6	6	3
19.6.- {CodigoHabitacion,TipoHabitacion,PrecioNoche}, PrecioNoche	6	6	3
20.1.- {FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente	23	17	7
20.2.- {FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente	22	17	6
20.3.- {FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion	24	17	6
20.4.- {FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada	17	17	10
20.5.- {FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion	17	17	3

20.6.- {FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche	17	17	3

Fuente: Elaboración propia.

D. Cardinalidad de las Proyecciones de las Tuplas sobre las Relaciones Binarias de nivel 4-aria

Tabla 26 : Cardinalidad de las Proyecciones de nivel 4-aria

BinaryRelation	|ΠAB(R)|	|ΠA(R)|	|ΠB(R)|
1.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, CodigoCliente	24	24	7
1.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, NombreCliente	24	24	6
1.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, CodigoHabitacion	24	24	6
1.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, FechaEntrada	24	24	10
1.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, TipoHabitacion	24	24	3
1.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada}, PrecioNoche	24	24	3
2.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, CodigoCliente	18	18	7
2.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, NombreCliente	18	18	6
2.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, CodigoHabitacion	18	18	6
2.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, FechaEntrada	24	18	10
2.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, TipoHabitacion	18	18	3
2.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion}, PrecioNoche	18	18	3
3.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, CodigoCliente	18	18	7
3.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, NombreCliente	18	18	6
3.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, CodigoHabitacion	18	18	6
3.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, FechaEntrada	24	18	10
3.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, TipoHabitacion	18	18	3
3.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche}, PrecioNoche	18	18	3
4.1.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, CodigoCliente	23	23	7
4.2.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, NombreCliente	23	23	6
4.3.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, CodigoHabitacion	24	23	6
4.4.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, FechaEntrada	23	23	10
4.5.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, TipoHabitacion	23	23	3
4.6.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion}, PrecioNoche	23	23	3
5.1.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, CodigoCliente	23	23	7
5.2.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, NombreCliente	23	23	6
5.3.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, CodigoHabitacion	24	23	6
5.4.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, FechaEntrada	23	23	10
5.5.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, TipoHabitacion	23	23	3
5.6.- {CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche}, PrecioNoche	23	23	3
6.1.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, CodigoCliente	15	15	7
6.2.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, NombreCliente	15	15	6
6.3.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, CodigoHabitacion	18	15	6
6.4.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, FechaEntrada	23	15	10
6.5.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, TipoHabitacion	15	15	3
6.6.- {CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche}, PrecioNoche	15	15	3
7.1.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoCliente	24	24	7
7.2.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, NombreCliente	24	24	6
7.3.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoHabitacion	24	24	6

7.4.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, FechaEntrada	24	24	10
7.5.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, TipoHabitacion	24	24	3
7.6.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, PrecioNoche	24	24	3
8.1.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoCliente	24	24	7
8.2.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, NombreCliente	24	24	6
8.3.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoHabitacion	24	24	6
8.4.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, FechaEntrada	24	24	10
8.5.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, TipoHabitacion	24	24	3
8.6.- {CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, PrecioNoche	24	24	3
9.1.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoCliente	18	18	7
9.2.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, NombreCliente	18	18	6
9.3.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoHabitacion	18	18	6
9.4.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, FechaEntrada	24	18	10
9.5.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, TipoHabitacion	18	18	3
9.6.- {CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, PrecioNoche	18	18	3
10.1.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente	23	23	7
10.2.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente	23	23	6
10.3.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion	24	23	6
10.4.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada	23	23	10
10.5.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion	23	23	3
10.6.- {CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche	23	23	3
11.1.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoCliente	24	24	7
11.2.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, NombreCliente	24	24	6
11.3.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, CodigoHabitacion	24	24	6
11.4.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, FechaEntrada	24	24	10
11.5.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, TipoHabitacion	24	24	3
11.6.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion}, PrecioNoche	24	24	3
12.1.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoCliente	24	24	7
12.2.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, NombreCliente	24	24	6
12.3.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoHabitacion	24	24	6
12.4.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, FechaEntrada	24	24	10
12.5.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, TipoHabitacion	24	24	3
12.6.- {NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, PrecioNoche	24	24	3
13.1.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoCliente	18	18	7
13.2.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, NombreCliente	18	18	6
13.3.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoHabitacion	18	18	6
13.4.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, FechaEntrada	24	18	10
13.5.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, TipoHabitacion	18	18	3
13.6.- {NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, PrecioNoche	18	18	3
14.1.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente	23	22	7
14.2.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente	22	22	6
14.3.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion	24	22	6
14.4.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada	22	22	10
14.5.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion	22	22	3
14.6.- {NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche	22	22	3
15.1.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente	24	24	7
15.2.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente	24	24	6
15.3.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion	24	24	6
15.4.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada	24	24	10
15.5.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion	24	24	3
15.6.- {CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche	24	24	3

Fuente: Elaboración propia.

E. Cardinalidad de las Proyecciones de las Tuplas sobre las Relaciones Binarias de nivel 5-aria

Tabla 27 : Cardinalidad de las Proyecciones de nivel 5-aria

BinaryRelation	|ΠAB(R)|	|ΠA(R)|	|ΠB(R)|
1.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},CodigoCliente	24	24	7
1.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},NombreCliente	24	24	6
1.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},CodigoHabitacion	24	24	6
1.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},FechaEntrada	24	24	10
1.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},TipoHabitacion	24	24	3
1.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion},PrecioNoche	24	24	3
2.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoCliente	24	24	7
2.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, NombreCliente	24	24	6
2.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, CodigoHabitacion	24	24	6
2.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, FechaEntrada	24	24	10
2.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, TipoHabitacion	24	24	3
2.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche}, PrecioNoche	24	24	3
3.1.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoCliente	18	18	7
3.2.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, NombreCliente	18	18	6
3.3.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, CodigoHabitacion	18	18	6
3.4.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, FechaEntrada	24	18	10
3.5.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, TipoHabitacion	18	18	3
3.6.- {CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche}, PrecioNoche	18	18	3
4.1.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente	23	23	7
4.2.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente	23	23	6
4.3.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion	24	23	6
4.4.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada	23	23	10
4.5.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion	23	23	3
4.6.- {CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche	23	23	3
5.1.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente	24	24	7
5.2.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente	24	24	6
5.3.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion	24	24	6
5.4.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada	24	24	10
5.5.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion	24	24	3

5.6.- {CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche	24	24	3
6.1.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoCliente	24	24	7
6.2.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, NombreCliente	24	24	6
6.3.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, CodigoHabitacion	24	24	6
6.4.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, FechaEntrada	24	24	10
6.5.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, TipoHabitacion	24	24	3
6.6.- {NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche}, PrecioNoche	24	24	3

Fuente: Elaboración propia.

3. Anexo 3: Restricciones Lógicas sobre las Cardinalidades de Proyecciones

A. Restricciones Lógicas sobre las Cardinalidades de las Proyecciones de cada Relación Binaria de nivel 1-aria

Tabla 28 : Restricciones Lógicas de nivel 1-aria

Dependencies (Logic Constraint)	RelationTypes	FunctionTypes
CodigoCliente -->> CodigoCliente	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
CodigoCliente ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
CodigoCliente ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
CodigoCliente ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
CodigoCliente ---o TipoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
CodigoCliente ---o PrecioNoche	MULTIVALUED FUNCTION	NO FUNCTION
CodigoCliente ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
NombreCliente -->> NombreCliente	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
NombreCliente ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
NombreCliente ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
NombreCliente ---o TipoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
NombreCliente ---o PrecioNoche	MULTIVALUED FUNCTION	NO FUNCTION
CodigoHabitacion ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
CodigoHabitacion ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
CodigoHabitacion -->> CodigoHabitacion	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
CodigoHabitacion ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
CodigoHabitacion ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
CodigoHabitacion ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
FechaEntrada ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
FechaEntrada ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
FechaEntrada ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
FechaEntrada -->> FechaEntrada	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
FechaEntrada ---o TipoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
FechaEntrada ---o PrecioNoche	MULTIVALUED FUNCTION	NO FUNCTION
TipoHabitacion ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
TipoHabitacion ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
CodigoHabitacion ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
TipoHabitacion ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
TipoHabitacion -->> TipoHabitacion	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
TipoHabitacion -->> PrecioNoche	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
PrecioNoche ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
PrecioNoche ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
CodigoHabitacion ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
PrecioNoche ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
PrecioNoche -->> TipoHabitacion	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
PrecioNoche -->> PrecioNoche	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION

Fuente: Elaboración propia.

B. Restricciones Lógicas sobre las Cardinalidades de las Proyecciones de cada Relación Binaria de nivel 2-aria

Tabla 29 : Restricciones Lógicas de nivel 2-aria

Dependencies (Logic Constraint)	RelationTypes	FunctionTypes
{CodigoCliente,NombreCliente} -->> CodigoCliente	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
{CodigoCliente,NombreCliente} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente} ---o TipoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente} ---o PrecioNoche	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,CodigoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,FechaEntrada} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada} ---o TipoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,FechaEntrada} ---o PrecioNoche	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,TipoHabitacion} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,TipoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,CodigoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,FechaEntrada} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,FechaEntrada} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada} ---o TipoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,FechaEntrada} ---o PrecioNoche	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,TipoHabitacion} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,TipoHabitacion} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,TipoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

{NombreCliente,PrecioNoche} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,TipoHabitacion} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoHabitacion,TipoHabitacion} ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoHabitacion,TipoHabitacion} -->> CodigoHabitacion	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
{CodigoHabitacion,TipoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,PrecioNoche} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoHabitacion,PrecioNoche} ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoHabitacion,PrecioNoche} -->> CodigoHabitacion	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
{CodigoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{FechaEntrada,TipoHabitacion} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{FechaEntrada,TipoHabitacion} ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
{FechaEntrada,TipoHabitacion} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{FechaEntrada,TipoHabitacion} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{FechaEntrada,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{FechaEntrada,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{FechaEntrada,PrecioNoche} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{FechaEntrada,PrecioNoche} ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
{FechaEntrada,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{FechaEntrada,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{FechaEntrada,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{FechaEntrada,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{TipoHabitacion,PrecioNoche} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{TipoHabitacion,PrecioNoche} ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
CodigoHabitacion ---> {TipoHabitacion,PrecioNoche}	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{TipoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{TipoHabitacion,PrecioNoche} -->> TipoHabitacion	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
{TipoHabitacion,PrecioNoche} -->> PrecioNoche	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION

Fuente: Elaboración propia.

C. Restricciones Lógicas sobre las Cardinalidades de las Proyecciones de cada Relación Binaria de nivel 3-aria

Tabla 30 : Restricciones Lógicas de nivel 3-aria

Dependencies (Logic Constraint)	RelationTypes	FunctionTypes
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada} ---o TipoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada} ---o PrecioNoche	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,TipoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

{NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,PrecioNoche} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,TipoHabitacion,PrecioNoche} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,TipoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoHabitacion,TipoHabitacion,PrecioNoche} -->> CodigoHabitacion	SINGLE-VALUED FUNCTION	BIJECTIVE FUNCTION
{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{FechaEntrada,TipoHabitacion,PrecioNoche} ---o NombreCliente	MULTIVALUED FUNCTION	NO FUNCTION
{FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

Fuente: Elaboración propia.

D. Restricciones Lógicas sobre las Cardinalidades de las Proyecciones de cada Relación Binaria de nivel 4-aria

Tabla 31 : Restricciones Lógicas de nivel 4-aria

Dependencies (Logic Constraint)	RelationTypes	FunctionTypes
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoCliente	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

Fuente: Elaboración propia.

E. Restricciones Lógicas sobre las Cardinalidades de las Proyecciones de cada Relación Binaria de nivel 5-aria

Tabla 32 : Restricciones Lógicas de nivel 5-aria

Dependencies (Logic Constraint)	RelationTypes	FunctionTypes
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o FechaEntrada	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion	MULTIVALUED FUNCTION	NO FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	SINGLE-VALUED FUNCTION	SURJECTIVE FUNCTION

Fuente: Elaboración propia.

4. Anexo 4: Dependencias de Relación Funcional (Válidas)

Tabla 33 : Dependencias de Relación Funcional.

Nº	Functional Relationship Dependencies
1	CodigoCliente -->> CodigoCliente
2	CodigoCliente ---> NombreCliente
3	NombreCliente -->> NombreCliente
4	CodigoHabitacion -->> CodigoHabitacion
5	CodigoHabitacion ---> TipoHabitacion
6	CodigoHabitacion ---> PrecioNoche
7	FechaEntrada -->> FechaEntrada
8	TipoHabitacion -->> TipoHabitacion
9	TipoHabitacion -->> PrecioNoche
10	PrecioNoche -->> TipoHabitacion
11	PrecioNoche -->> PrecioNoche
12	{CodigoCliente,NombreCliente} -->> CodigoCliente
13	{CodigoCliente,NombreCliente} ---> NombreCliente
14	{CodigoCliente,CodigoHabitacion} ---> CodigoCliente
15	{CodigoCliente,CodigoHabitacion} ---> NombreCliente
16	{CodigoCliente,CodigoHabitacion} ---> CodigoHabitacion
17	{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion
18	{CodigoCliente,CodigoHabitacion} ---> PrecioNoche
19	{CodigoCliente,FechaEntrada} ---> CodigoCliente
20	{CodigoCliente,FechaEntrada} ---> NombreCliente
21	{CodigoCliente,FechaEntrada} ---> FechaEntrada
22	{CodigoCliente,TipoHabitacion} ---> CodigoCliente
23	{CodigoCliente,TipoHabitacion} ---> NombreCliente
24	{CodigoCliente,TipoHabitacion} ---> TipoHabitacion
25	{CodigoCliente,TipoHabitacion} ---> PrecioNoche
26	{CodigoCliente,PrecioNoche} ---> CodigoCliente
27	{CodigoCliente,PrecioNoche} ---> NombreCliente
28	{CodigoCliente,PrecioNoche} ---> TipoHabitacion
29	{CodigoCliente,PrecioNoche} ---> PrecioNoche
30	{NombreCliente,CodigoHabitacion} ---> CodigoCliente
31	{NombreCliente,CodigoHabitacion} ---> NombreCliente
32	{NombreCliente,CodigoHabitacion} ---> CodigoHabitacion
33	{NombreCliente,CodigoHabitacion} ---> TipoHabitacion
34	{NombreCliente,CodigoHabitacion} ---> PrecioNoche
35	{NombreCliente,FechaEntrada} ---> NombreCliente
36	{NombreCliente,FechaEntrada} ---> FechaEntrada
37	{NombreCliente,TipoHabitacion} ---> NombreCliente
38	{NombreCliente,TipoHabitacion} ---> TipoHabitacion
39	{NombreCliente,TipoHabitacion} ---> PrecioNoche
40	{NombreCliente,PrecioNoche} ---> NombreCliente
41	{NombreCliente,PrecioNoche} ---> TipoHabitacion
42	{NombreCliente,PrecioNoche} ---> PrecioNoche
43	{CodigoHabitacion,FechaEntrada} ---> CodigoCliente
44	{CodigoHabitacion,FechaEntrada} ---> NombreCliente
45	{CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion
46	{CodigoHabitacion,FechaEntrada} ---> FechaEntrada
47	{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
48	{CodigoHabitacion,FechaEntrada} ---> PrecioNoche
49	{CodigoHabitacion,TipoHabitacion} -->> CodigoHabitacion
50	{CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion

51	{CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
52	{CodigoHabitacion,PrecioNoche} -->> CodigoHabitacion
53	{CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
54	{CodigoHabitacion,PrecioNoche} ---> PrecioNoche
55	{FechaEntrada,TipoHabitacion} ---> FechaEntrada
56	{FechaEntrada,TipoHabitacion} ---> TipoHabitacion
57	{FechaEntrada,TipoHabitacion} ---> PrecioNoche
58	{FechaEntrada,PrecioNoche} ---> FechaEntrada
59	{FechaEntrada,PrecioNoche} ---> TipoHabitacion
60	{FechaEntrada,PrecioNoche} ---> PrecioNoche
61	CodigoHabitacion ---> {TipoHabitacion,PrecioNoche}
62	{TipoHabitacion,PrecioNoche} -->> TipoHabitacion
63	{TipoHabitacion,PrecioNoche} -->> PrecioNoche
64	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoCliente
65	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> NombreCliente
66	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoHabitacion
67	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> TipoHabitacion
68	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> PrecioNoche
69	{CodigoCliente,NombreCliente,FechaEntrada} ---> CodigoCliente
70	{CodigoCliente,NombreCliente,FechaEntrada} ---> NombreCliente
71	{CodigoCliente,NombreCliente,FechaEntrada} ---> FechaEntrada
72	{CodigoCliente,NombreCliente,TipoHabitacion} ---> CodigoCliente
73	{CodigoCliente,NombreCliente,TipoHabitacion} ---> NombreCliente
74	{CodigoCliente,NombreCliente,TipoHabitacion} ---> TipoHabitacion
75	{CodigoCliente,NombreCliente,TipoHabitacion} ---> PrecioNoche
76	{CodigoCliente,NombreCliente,PrecioNoche} ---> CodigoCliente
77	{CodigoCliente,NombreCliente,PrecioNoche} ---> NombreCliente
78	{CodigoCliente,NombreCliente,PrecioNoche} ---> TipoHabitacion
79	{CodigoCliente,NombreCliente,PrecioNoche} ---> PrecioNoche
80	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente
81	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente
82	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion
83	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada
84	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
85	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
86	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente
87	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente
88	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion
89	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion
90	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
91	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente
92	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente
93	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion
94	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
95	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche
96	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente
97	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente
98	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada
99	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
100	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche
101	{CodigoCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente
102	{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
103	{CodigoCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada
104	{CodigoCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion
105	{CodigoCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche
106	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente

107	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
108	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
109	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche
110	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente
111	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente
112	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion
113	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada
114	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
115	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
116	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente
117	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente
118	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion
119	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion
120	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
121	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente
122	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente
123	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion
124	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
125	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche
126	{NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente
127	{NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada
128	{NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
129	{NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche
130	{NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
131	{NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada
132	{NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion
133	{NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche
134	{NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
135	{NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
136	{NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche
137	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
138	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
139	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion
140	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada
141	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
142	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
143	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
144	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
145	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion
146	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada
147	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
148	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche
149	{CodigoHabitacion,TipoHabitacion,PrecioNoche} -->> CodigoHabitacion
150	{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
151	{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche
152	{FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
153	{FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
154	{FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
155	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente
156	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente
157	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion
158	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada
159	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
160	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
161	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente
162	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente

163	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion
164	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion
165	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
166	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente
167	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente
168	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion
169	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
170	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche
171	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente
172	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente
173	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada
174	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
175	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche
176	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente
177	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
178	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada
179	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion
180	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche
181	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente
182	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
183	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
184	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche
185	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
186	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
187	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion
188	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada
189	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
190	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
191	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
192	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
193	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion
194	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada
195	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
196	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche
197	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente
198	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente
199	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
200	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
201	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche
202	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
203	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
204	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
205	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
206	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
207	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
208	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
209	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion
210	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada
211	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
212	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
213	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
214	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
215	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion
216	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada
217	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
218	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche

219	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente
220	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente
221	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
222	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
223	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche
224	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
225	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
226	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
227	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
228	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
229	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
230	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
231	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
232	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
233	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
234	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
235	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
236	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion
237	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada
238	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
239	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
240	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
241	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
242	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion
243	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada
244	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
245	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche
246	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente
247	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente
248	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
249	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
250	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche
251	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
252	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
253	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
254	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
255	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
256	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
257	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
258	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
259	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
260	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
261	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
262	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
263	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
264	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
265	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
266	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
267	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche

Fuente: Elaboración propia.

5. Anexo 5: Dependencias de Relación No Funcional (Inválidas)

Tabla 34 : Dependencias de Relación No Funcional

Nº	Non Functional Relationship Dependencies
1	CodigoCliente ---o CodigoHabitacion
2	CodigoCliente ---o FechaEntrada
3	CodigoCliente ---o TipoHabitacion
4	CodigoCliente ---o PrecioNoche
5	NombreCliente ---o CodigoHabitacion
6	NombreCliente ---o FechaEntrada
7	NombreCliente ---o TipoHabitacion
8	NombreCliente ---o PrecioNoche
9	CodigoHabitacion ---o CodigoCliente
10	CodigoHabitacion ---o NombreCliente
11	CodigoHabitacion ---o FechaEntrada
12	FechaEntrada ---o CodigoCliente
13	FechaEntrada ---o NombreCliente
14	FechaEntrada ---o CodigoHabitacion
15	FechaEntrada ---o TipoHabitacion
16	FechaEntrada ---o PrecioNoche
17	TipoHabitacion ---o CodigoCliente
18	TipoHabitacion ---o NombreCliente
19	TipoHabitacion ---o FechaEntrada
20	PrecioNoche ---o CodigoCliente
21	PrecioNoche ---o NombreCliente
22	PrecioNoche ---o FechaEntrada
23	{CodigoCliente,NombreCliente} ---o CodigoHabitacion
24	{CodigoCliente,NombreCliente} ---o FechaEntrada
25	{CodigoCliente,NombreCliente} ---o TipoHabitacion
26	{CodigoCliente,NombreCliente} ---o PrecioNoche
27	{CodigoCliente,CodigoHabitacion} ---o FechaEntrada
28	{CodigoCliente,FechaEntrada} ---o CodigoHabitacion
29	{CodigoCliente,FechaEntrada} ---o TipoHabitacion
30	{CodigoCliente,FechaEntrada} ---o PrecioNoche
31	{CodigoCliente,TipoHabitacion} ---o CodigoHabitacion
32	{CodigoCliente,TipoHabitacion} ---o FechaEntrada
33	{CodigoCliente,PrecioNoche} ---o CodigoHabitacion
34	{CodigoCliente,PrecioNoche} ---o FechaEntrada
35	{NombreCliente,CodigoHabitacion} ---o FechaEntrada
36	{NombreCliente,FechaEntrada} ---o CodigoCliente
37	{NombreCliente,FechaEntrada} ---o CodigoHabitacion
38	{NombreCliente,FechaEntrada} ---o TipoHabitacion
39	{NombreCliente,FechaEntrada} ---o PrecioNoche
40	{NombreCliente,TipoHabitacion} ---o CodigoCliente
41	{NombreCliente,TipoHabitacion} ---o CodigoHabitacion
42	{NombreCliente,TipoHabitacion} ---o FechaEntrada
43	{NombreCliente,PrecioNoche} ---o CodigoCliente
44	{NombreCliente,PrecioNoche} ---o CodigoHabitacion
45	{NombreCliente,PrecioNoche} ---o FechaEntrada
46	{CodigoHabitacion,TipoHabitacion} ---o CodigoCliente
47	{CodigoHabitacion,TipoHabitacion} ---o NombreCliente
48	{CodigoHabitacion,TipoHabitacion} ---o FechaEntrada
49	{CodigoHabitacion,PrecioNoche} ---o CodigoCliente
50	{CodigoHabitacion,PrecioNoche} ---o NombreCliente

51	{CodigoHabitacion,PrecioNoche} ---o FechaEntrada
52	{FechaEntrada,TipoHabitacion} ---o CodigoCliente
53	{FechaEntrada,TipoHabitacion} ---o NombreCliente
54	{FechaEntrada,TipoHabitacion} ---o CodigoHabitacion
55	{FechaEntrada,PrecioNoche} ---o CodigoCliente
56	{FechaEntrada,PrecioNoche} ---o NombreCliente
57	{FechaEntrada,PrecioNoche} ---o CodigoHabitacion
58	{TipoHabitacion,PrecioNoche} ---o CodigoCliente
59	{TipoHabitacion,PrecioNoche} ---o NombreCliente
60	{TipoHabitacion,PrecioNoche} ---o FechaEntrada
61	{CodigoCliente,NombreCliente,CodigoHabitacion} ---o FechaEntrada
62	{CodigoCliente,NombreCliente,FechaEntrada} ---o CodigoHabitacion
63	{CodigoCliente,NombreCliente,FechaEntrada} ---o TipoHabitacion
64	{CodigoCliente,NombreCliente,FechaEntrada} ---o PrecioNoche
65	{CodigoCliente,NombreCliente,TipoHabitacion} ---o CodigoHabitacion
66	{CodigoCliente,NombreCliente,TipoHabitacion} ---o FechaEntrada
67	{CodigoCliente,NombreCliente,PrecioNoche} ---o CodigoHabitacion
68	{CodigoCliente,NombreCliente,PrecioNoche} ---o FechaEntrada
69	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---o FechaEntrada
70	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---o FechaEntrada
71	{CodigoCliente,FechaEntrada,TipoHabitacion} ---o CodigoHabitacion
72	{CodigoCliente,FechaEntrada,PrecioNoche} ---o CodigoHabitacion
73	{CodigoCliente,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion
74	{CodigoCliente,TipoHabitacion,PrecioNoche} ---o FechaEntrada
75	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---o FechaEntrada
76	{NombreCliente,CodigoHabitacion,PrecioNoche} ---o FechaEntrada
77	{NombreCliente,FechaEntrada,TipoHabitacion} ---o CodigoCliente
78	{NombreCliente,FechaEntrada,TipoHabitacion} ---o CodigoHabitacion
79	{NombreCliente,FechaEntrada,PrecioNoche} ---o CodigoCliente
80	{NombreCliente,FechaEntrada,PrecioNoche} ---o CodigoHabitacion
81	{NombreCliente,TipoHabitacion,PrecioNoche} ---o CodigoCliente
82	{NombreCliente,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion
83	{NombreCliente,TipoHabitacion,PrecioNoche} ---o FechaEntrada
84	{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o CodigoCliente
85	{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o NombreCliente
86	{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o FechaEntrada
87	{FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoCliente
88	{FechaEntrada,TipoHabitacion,PrecioNoche} ---o NombreCliente
89	{FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion
90	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---o FechaEntrada
91	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---o FechaEntrada
92	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---o CodigoHabitacion
93	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---o CodigoHabitacion
94	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion
95	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---o FechaEntrada
96	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o FechaEntrada
97	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion
98	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o FechaEntrada
99	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoCliente
100	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion
101	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---o FechaEntrada
102	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---o CodigoHabitacion

Fuente: Elaboración propia.

6. Anexo 6: Reglas de Inferencia sobre las Dependencias de Relación Funcional

A. Teorema 2.2: Descomposición

Tabla 35 : Teorema 2.2: Descomposición

Theorem 2.2 (Descomposición)	Result
CodigoCliente -->> CodigoCliente	CodigoCliente -->> CodigoCliente
CodigoCliente ---> NombreCliente	CodigoCliente ---> NombreCliente
NombreCliente -->> NombreCliente	NombreCliente -->> NombreCliente
CodigoHabitacion -->> CodigoHabitacion	CodigoHabitacion -->> CodigoHabitacion
CodigoHabitacion ---> TipoHabitacion	CodigoHabitacion ---> TipoHabitacion
CodigoHabitacion ---> PrecioNoche	CodigoHabitacion ---> PrecioNoche
FechaEntrada -->> FechaEntrada	FechaEntrada -->> FechaEntrada
TipoHabitacion -->> TipoHabitacion	TipoHabitacion -->> TipoHabitacion
TipoHabitacion -->> PrecioNoche	TipoHabitacion -->> PrecioNoche
PrecioNoche -->> TipoHabitacion	PrecioNoche -->> TipoHabitacion
PrecioNoche -->> PrecioNoche	PrecioNoche -->> PrecioNoche
{CodigoCliente,NombreCliente} -->> CodigoCliente	{CodigoCliente,NombreCliente} -->> CodigoCliente
{CodigoCliente,NombreCliente} ---> NombreCliente	{CodigoCliente,NombreCliente} ---> NombreCliente
{CodigoCliente,CodigoHabitacion} ---> CodigoCliente	{CodigoCliente,CodigoHabitacion} ---> CodigoCliente
{CodigoCliente,CodigoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion} ---> CodigoHabitacion	{CodigoCliente,CodigoHabitacion} ---> CodigoHabitacion
{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion} ---> PrecioNoche
{CodigoCliente,FechaEntrada} ---> CodigoCliente	{CodigoCliente,FechaEntrada} ---> CodigoCliente
{CodigoCliente,FechaEntrada} ---> NombreCliente	{CodigoCliente,FechaEntrada} ---> NombreCliente
{CodigoCliente,FechaEntrada} ---> FechaEntrada	{CodigoCliente,FechaEntrada} ---> FechaEntrada
{CodigoCliente,TipoHabitacion} ---> CodigoCliente	{CodigoCliente,TipoHabitacion} ---> CodigoCliente
{CodigoCliente,TipoHabitacion} ---> NombreCliente	{CodigoCliente,TipoHabitacion} ---> NombreCliente
{CodigoCliente,TipoHabitacion} ---> TipoHabitacion	{CodigoCliente,TipoHabitacion} ---> TipoHabitacion
{CodigoCliente,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,PrecioNoche} ---> CodigoCliente	{CodigoCliente,PrecioNoche} ---> CodigoCliente
{CodigoCliente,PrecioNoche} ---> NombreCliente	{CodigoCliente,PrecioNoche} ---> NombreCliente
{CodigoCliente,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,PrecioNoche} ---> PrecioNoche	{CodigoCliente,PrecioNoche} ---> PrecioNoche
{NombreCliente,CodigoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion} ---> NombreCliente	{NombreCliente,CodigoHabitacion} ---> NombreCliente
{NombreCliente,CodigoHabitacion} ---> CodigoHabitacion	{NombreCliente,CodigoHabitacion} ---> CodigoHabitacion
{NombreCliente,CodigoHabitacion} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion} ---> TipoHabitacion

{NombreCliente,CodigoHabitacion} ---> PrecioNoche	{NombreCliente,CodigoHabitacion} ---> PrecioNoche
{NombreCliente,FechaEntrada} ---> NombreCliente	{NombreCliente,FechaEntrada} ---> NombreCliente
{NombreCliente,FechaEntrada} ---> FechaEntrada	{NombreCliente,FechaEntrada} ---> FechaEntrada
{NombreCliente,TipoHabitacion} ---> NombreCliente	{NombreCliente,TipoHabitacion} ---> NombreCliente
{NombreCliente,TipoHabitacion} ---> TipoHabitacion	{NombreCliente,TipoHabitacion} ---> TipoHabitacion
{NombreCliente,TipoHabitacion} ---> PrecioNoche	{NombreCliente,TipoHabitacion} ---> PrecioNoche
{NombreCliente,PrecioNoche} ---> NombreCliente	{NombreCliente,PrecioNoche} ---> NombreCliente
{NombreCliente,PrecioNoche} ---> TipoHabitacion	{NombreCliente,PrecioNoche} ---> TipoHabitacion
{NombreCliente,PrecioNoche} ---> PrecioNoche	{NombreCliente,PrecioNoche} ---> PrecioNoche
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada} ---> NombreCliente	{CodigoHabitacion,FechaEntrada} ---> NombreCliente
{CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	{CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion
{CodigoHabitacion,FechaEntrada} ---> FechaEntrada	{CodigoHabitacion,FechaEntrada} ---> FechaEntrada
{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{CodigoHabitacion,TipoHabitacion} -->> CodigoHabitacion	{CodigoHabitacion,TipoHabitacion} -->> CodigoHabitacion
{CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	{CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion
{CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	{CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
{CodigoHabitacion,PrecioNoche} -->> CodigoHabitacion	{CodigoHabitacion,PrecioNoche} -->> CodigoHabitacion
{CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoHabitacion,PrecioNoche} ---> PrecioNoche
{FechaEntrada,TipoHabitacion} ---> FechaEntrada	{FechaEntrada,TipoHabitacion} ---> FechaEntrada
{FechaEntrada,TipoHabitacion} ---> TipoHabitacion	{FechaEntrada,TipoHabitacion} ---> TipoHabitacion
{FechaEntrada,TipoHabitacion} ---> PrecioNoche	{FechaEntrada,TipoHabitacion} ---> PrecioNoche
{FechaEntrada,PrecioNoche} ---> FechaEntrada	{FechaEntrada,PrecioNoche} ---> FechaEntrada
{FechaEntrada,PrecioNoche} ---> TipoHabitacion	{FechaEntrada,PrecioNoche} ---> TipoHabitacion
{FechaEntrada,PrecioNoche} ---> PrecioNoche	{FechaEntrada,PrecioNoche} ---> PrecioNoche
CodigoHabitacion ---> {TipoHabitacion,PrecioNoche}	CodigoHabitacion ---> TipoHabitacion
	CodigoHabitacion ---> PrecioNoche
{TipoHabitacion,PrecioNoche} -->> TipoHabitacion	{TipoHabitacion,PrecioNoche} -->> TipoHabitacion
{TipoHabitacion,PrecioNoche} -->> PrecioNoche	{TipoHabitacion,PrecioNoche} -->> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoCliente	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoCliente
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> NombreCliente	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> NombreCliente
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,FechaEntrada} ---> CodigoCliente	{CodigoCliente,NombreCliente,FechaEntrada} ---> CodigoCliente
{CodigoCliente,NombreCliente,FechaEntrada} ---> NombreCliente	{CodigoCliente,NombreCliente,FechaEntrada} ---> NombreCliente
{CodigoCliente,NombreCliente,FechaEntrada} ---> FechaEntrada	{CodigoCliente,NombreCliente,FechaEntrada} ---> FechaEntrada
{CodigoCliente,NombreCliente,TipoHabitacion} ---> CodigoCliente	{CodigoCliente,NombreCliente,TipoHabitacion} ---> CodigoCliente
{CodigoCliente,NombreCliente,TipoHabitacion} ---> NombreCliente	{CodigoCliente,NombreCliente,TipoHabitacion} ---> NombreCliente
{CodigoCliente,NombreCliente,TipoHabitacion} ---> TipoHabitacion	{CodigoCliente,NombreCliente,TipoHabitacion} ---> TipoHabitacion
{CodigoCliente,NombreCliente,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,PrecioNoche} ---> CodigoCliente	{CodigoCliente,NombreCliente,PrecioNoche} ---> CodigoCliente

{CodigoCliente,NombreCliente,PrecioNoche} ---> NombreCliente	{CodigoCliente,NombreCliente,PrecioNoche} ---> NombreCliente
{CodigoCliente,NombreCliente,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,PrecioNoche} ---> PrecioNoche	{CodigoCliente,NombreCliente,PrecioNoche} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente	{CodigoCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente
{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada	{CodigoCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada
{CodigoCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche	{CodigoCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche

{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche
{NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	{NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente
{NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada	{NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada
{NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	{NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
{NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	{NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
{NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada	{NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada
{NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche	{NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche
{NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	{NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche
{CodigoHabitacion,TipoHabitacion,PrecioNoche} -->> CodigoHabitacion	{CodigoHabitacion,TipoHabitacion,PrecioNoche} -->> CodigoHabitacion
{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	{FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
{FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion

{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente

{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente

{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche

Fuente: Elaboración propia.

B. Teorema 1.1: Eliminación de Dependencias Reflexivas

Tabla 36 : Teorema 1.1: Eliminación de Dependencias Reflexivas

Theorem 1.1 (Eliminación de Dependencias Reflexivas)	Result
CodigoCliente -->> CodigoCliente	
CodigoCliente ---> NombreCliente	CodigoCliente ---> NombreCliente
NombreCliente -->> NombreCliente	
CodigoHabitacion -->> CodigoHabitacion	
CodigoHabitacion ---> TipoHabitacion	CodigoHabitacion ---> TipoHabitacion
CodigoHabitacion ---> PrecioNoche	CodigoHabitacion ---> PrecioNoche
FechaEntrada -->> FechaEntrada	
TipoHabitacion -->> TipoHabitacion	
TipoHabitacion -->> PrecioNoche	TipoHabitacion -->> PrecioNoche
PrecioNoche -->> TipoHabitacion	PrecioNoche -->> TipoHabitacion
PrecioNoche -->> PrecioNoche	
{CodigoCliente,NombreCliente} -->> CodigoCliente	
{CodigoCliente,NombreCliente} ---> NombreCliente	
{CodigoCliente,CodigoHabitacion} ---> CodigoCliente	

{CodigoCliente,CodigoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion} ---> CodigoHabitacion	
{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion} ---> PrecioNoche
{CodigoCliente,FechaEntrada} ---> CodigoCliente	
{CodigoCliente,FechaEntrada} ---> NombreCliente	{CodigoCliente,FechaEntrada} ---> NombreCliente
{CodigoCliente,FechaEntrada} ---> FechaEntrada	
{CodigoCliente,TipoHabitacion} ---> CodigoCliente	
{CodigoCliente,TipoHabitacion} ---> NombreCliente	{CodigoCliente,TipoHabitacion} ---> NombreCliente
{CodigoCliente,TipoHabitacion} ---> TipoHabitacion	
{CodigoCliente,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,PrecioNoche} ---> NombreCliente	{CodigoCliente,PrecioNoche} ---> NombreCliente
{CodigoCliente,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,PrecioNoche} ---> PrecioNoche	
{NombreCliente,CodigoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion} ---> NombreCliente	
{NombreCliente,CodigoHabitacion} ---> CodigoHabitacion	
{NombreCliente,CodigoHabitacion} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion} ---> PrecioNoche	{NombreCliente,CodigoHabitacion} ---> PrecioNoche
{NombreCliente,FechaEntrada} ---> NombreCliente	
{NombreCliente,FechaEntrada} ---> FechaEntrada	
{NombreCliente,TipoHabitacion} ---> NombreCliente	
{NombreCliente,TipoHabitacion} ---> TipoHabitacion	
{NombreCliente,TipoHabitacion} ---> PrecioNoche	{NombreCliente,TipoHabitacion} ---> PrecioNoche
{NombreCliente,PrecioNoche} ---> NombreCliente	
{NombreCliente,PrecioNoche} ---> TipoHabitacion	{NombreCliente,PrecioNoche} ---> TipoHabitacion
{NombreCliente,PrecioNoche} ---> PrecioNoche	
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada} ---> NombreCliente	{CodigoHabitacion,FechaEntrada} ---> NombreCliente
{CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	
{CodigoHabitacion,FechaEntrada} ---> FechaEntrada	
{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{CodigoHabitacion,TipoHabitacion} -->> CodigoHabitacion	
{CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	
{CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	{CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
{CodigoHabitacion,PrecioNoche} -->> CodigoHabitacion	
{CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoHabitacion,PrecioNoche} ---> PrecioNoche	
{FechaEntrada,TipoHabitacion} ---> FechaEntrada	
{FechaEntrada,TipoHabitacion} ---> TipoHabitacion	
{FechaEntrada,TipoHabitacion} ---> PrecioNoche	{FechaEntrada,TipoHabitacion} ---> PrecioNoche
{FechaEntrada,PrecioNoche} ---> FechaEntrada	

{FechaEntrada,PrecioNoche} ---> TipoHabitacion	{FechaEntrada,PrecioNoche} ---> TipoHabitacion
{FechaEntrada,PrecioNoche} ---> PrecioNoche	
{TipoHabitacion,PrecioNoche} -->> TipoHabitacion	
{TipoHabitacion,PrecioNoche} -->> PrecioNoche	
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> NombreCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> CodigoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,FechaEntrada} ---> CodigoCliente	
{CodigoCliente,NombreCliente,FechaEntrada} ---> NombreCliente	
{CodigoCliente,NombreCliente,FechaEntrada} ---> FechaEntrada	
{CodigoCliente,NombreCliente,TipoHabitacion} ---> CodigoCliente	
{CodigoCliente,NombreCliente,TipoHabitacion} ---> NombreCliente	
{CodigoCliente,NombreCliente,TipoHabitacion} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,NombreCliente,PrecioNoche} ---> NombreCliente	
{CodigoCliente,NombreCliente,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada	
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion	
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion	
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente	
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada	
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada	

{CodigoCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada	
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion	
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion	
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche	
{NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	
{NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada	
{NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	
{NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	
{NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada	
{NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche	
{NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion

{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	
{CodigoHabitacion,TipoHabitacion,PrecioNoche} -->> CodigoHabitacion	
{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	
{FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> FechaEntrada	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> CodigoCliente	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> FechaEntrada	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> FechaEntrada	
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	

{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	

{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> FechaEntrada	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> FechaEntrada	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoHabitacion	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> FechaEntrada	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> TipoHabitacion	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> PrecioNoche	

Fuente: Elaboración propia.

C. Teorema 1.2: Eliminar Dependencias con Biyección Parcial

Tabla 37 : Teorema 1.2: Eliminar Dependencias con Biyección Parcial.

Theorem 1.2 (Eliminar Dependencias con Biyección Parcial)	Result
CodigoCliente ---> NombreCliente	CodigoCliente ---> NombreCliente
CodigoHabitacion ---> TipoHabitacion	CodigoHabitacion ---> TipoHabitacion
CodigoHabitacion ---> PrecioNoche	CodigoHabitacion ---> PrecioNoche
TipoHabitacion -->> PrecioNoche	TipoHabitacion -->> PrecioNoche
PrecioNoche -->> TipoHabitacion	PrecioNoche -->> TipoHabitacion
{CodigoCliente,CodigoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion} ---> PrecioNoche
{CodigoCliente,FechaEntrada} ---> NombreCliente	{CodigoCliente,FechaEntrada} ---> NombreCliente
{CodigoCliente,TipoHabitacion} ---> NombreCliente	{CodigoCliente,TipoHabitacion} ---> NombreCliente
{CodigoCliente,TipoHabitacion} ---> PrecioNoche	
{CodigoCliente,PrecioNoche} ---> NombreCliente	{CodigoCliente,PrecioNoche} ---> NombreCliente
{CodigoCliente,PrecioNoche} ---> TipoHabitacion	
{NombreCliente,CodigoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion} ---> PrecioNoche	{NombreCliente,CodigoHabitacion} ---> PrecioNoche
{NombreCliente,TipoHabitacion} ---> PrecioNoche	
{NombreCliente,PrecioNoche} ---> TipoHabitacion	
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada} ---> NombreCliente	{CodigoHabitacion,FechaEntrada} ---> NombreCliente
{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	
{CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	
{FechaEntrada,TipoHabitacion} ---> PrecioNoche	
{FechaEntrada,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion} ---> PrecioNoche
{CodigoCliente,NombreCliente,TipoHabitacion} ---> PrecioNoche	
{CodigoCliente,NombreCliente,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche

{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	
{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	
{NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	
{NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion,TipoHabitacion} ---> PrecioNoche	
{CodigoCliente,NombreCliente,CodigoHabitacion,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,FechaEntrada,TipoHabitacion} ---> PrecioNoche	
{CodigoCliente,NombreCliente,FechaEntrada,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> PrecioNoche	

{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente

Fuente: Elaboración propia.

D. Teorema 1.3: Eliminar Dependencias con Sobreyección en el Lado Izquierdo

Tabla 38 : Teorema 1.3: Eliminar Dependencias con Sobreyección en el Lado Izquierdo.

Theorem 1.3 (Eliminar Dependencias con Sobreyección en el Lado Izquierdo)	Result
CodigoCliente ---> NombreCliente	CodigoCliente ---> NombreCliente
CodigoHabitacion ---> TipoHabitacion	CodigoHabitacion ---> TipoHabitacion
CodigoHabitacion ---> PrecioNoche	CodigoHabitacion ---> PrecioNoche
TipoHabitacion -->> PrecioNoche	TipoHabitacion -->> PrecioNoche
PrecioNoche -->> TipoHabitacion	PrecioNoche -->> TipoHabitacion
{CodigoCliente,CodigoHabitacion} ---> NombreCliente	{CodigoCliente,CodigoHabitacion} ---> NombreCliente
{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion
{CodigoCliente,CodigoHabitacion} ---> PrecioNoche	{CodigoCliente,CodigoHabitacion} ---> PrecioNoche
{CodigoCliente,FechaEntrada} ---> NombreCliente	{CodigoCliente,FechaEntrada} ---> NombreCliente
{CodigoCliente,TipoHabitacion} ---> NombreCliente	{CodigoCliente,TipoHabitacion} ---> NombreCliente
{CodigoCliente,PrecioNoche} ---> NombreCliente	{CodigoCliente,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion} ---> CodigoCliente	{NombreCliente,CodigoHabitacion} ---> CodigoCliente
{NombreCliente,CodigoHabitacion} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion} ---> TipoHabitacion
{NombreCliente,CodigoHabitacion} ---> PrecioNoche	{NombreCliente,CodigoHabitacion} ---> PrecioNoche
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada} ---> NombreCliente	{CodigoHabitacion,FechaEntrada} ---> NombreCliente
{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion
{CodigoHabitacion,FechaEntrada} ---> PrecioNoche	{CodigoHabitacion,FechaEntrada} ---> PrecioNoche
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> NombreCliente	
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion,TipoHabitacion} ---> NombreCliente	
{CodigoCliente,CodigoHabitacion,PrecioNoche} ---> NombreCliente	
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente

{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> CodigoCliente	
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	
{NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	
{NombreCliente,CodigoHabitacion,TipoHabitacion} ---> CodigoCliente	
{NombreCliente,CodigoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	
{CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	
{CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	
{CodigoCliente,NombreCliente,CodigoHabitacion,FechaEntrada} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> NombreCliente	
{CodigoCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> NombreCliente	
{CodigoCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion} ---> CodigoCliente	
{NombreCliente,CodigoHabitacion,FechaEntrada,PrecioNoche} ---> CodigoCliente	
{NombreCliente,CodigoHabitacion,TipoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	
{CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{CodigoCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{NombreCliente,CodigoHabitacion,FechaEntrada,TipoHabitacion,PrecioNoche} ---> CodigoCliente	

Fuente: Elaboración propia.

E. Teorema 1.4: Eliminar Dependencias con Sobreyección Parcial

Tabla 39 : Teorema 1.4: Eliminar Dependencias con Sobreyección Parcial (Normalizadas).

Normalized Dependencies (Dependencies from Result of Theorem 1.3)	
Theorem 1.4 (Eliminar Dependencias con Sobreyección Parcial)	Result
CodigoCliente ---> NombreCliente	CodigoCliente ---> NombreCliente
CodigoHabitacion ---> TipoHabitacion	CodigoHabitacion ---> TipoHabitacion
CodigoHabitacion ---> PrecioNoche	CodigoHabitacion ---> PrecioNoche
TipoHabitacion -->> PrecioNoche	TipoHabitacion -->> PrecioNoche

PrecioNoche -->> TipoHabitacion	PrecioNoche -->> TipoHabitacion
{CodigoCliente,CodigoHabitacion} ---> NombreCliente	
{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion	
{CodigoCliente,CodigoHabitacion} ---> PrecioNoche	
{CodigoCliente,FechaEntrada} ---> NombreCliente	
{CodigoCliente,TipoHabitacion} ---> NombreCliente	
{CodigoCliente,PrecioNoche} ---> NombreCliente	
{NombreCliente,CodigoHabitacion} ---> CodigoCliente	
{NombreCliente,CodigoHabitacion} ---> TipoHabitacion	
{NombreCliente,CodigoHabitacion} ---> PrecioNoche	
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada} ---> CodigoCliente
{CodigoHabitacion,FechaEntrada} ---> NombreCliente	{CodigoHabitacion,FechaEntrada} ---> NombreCliente
{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	
{CodigoHabitacion,FechaEntrada} ---> PrecioNoche	
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	
{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	

Fuente: Elaboración propia.

F. Teoremas 2.3: Composición

Tabla 40 : Teorema 2.3: Composición de Dependencias (Normalizadas).

Normalized Dependencies (Dependencies from Result of Theorem 1.4)	
Theorem 2.3 (Composición)	Result
CodigoCliente ---> NombreCliente	CodigoCliente ---> NombreCliente
CodigoHabitacion ---> TipoHabitacion	CodigoHabitacion ---> TipoHabitacion, PrecioNoche
CodigoHabitacion ---> PrecioNoche	
TipoHabitacion -->> PrecioNoche	TipoHabitacion -->> PrecioNoche
PrecioNoche -->> TipoHabitacion	PrecioNoche -->> TipoHabitacion
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente	{CodigoHabitacion,FechaEntrada} ---> CodigoCliente, NombreCliente
{CodigoHabitacion,FechaEntrada} ---> NombreCliente	

Fuente: Elaboración propia.

Tabla 41: Teorema 2.3: Composición de Dependencias (No Normalizadas).

Non-Normalized Dependencies (Dependencies from Result of Theorem 1.3)	
Theorem 2.3 (Composición)	**Result**
TipoHabitacion -->> PrecioNoche	TipoHabitacion -->> PrecioNoche
PrecioNoche -->> TipoHabitacion	PrecioNoche -->> TipoHabitacion
CodigoHabitacion ---> TipoHabitacion	CodigoHabitacion ---> {TipoHabitacion,PrecioNoche}
CodigoHabitacion ---> PrecioNoche	
CodigoCliente ---> NombreCliente	CodigoCliente ---> NombreCliente
{NombreCliente,CodigoHabitacion} ---> TipoHabitacion	{NombreCliente,CodigoHabitacion} ---> {TipoHabitacion,PrecioNoche,CodigoCliente}
{NombreCliente,CodigoHabitacion} ---> PrecioNoche	
{NombreCliente,CodigoHabitacion} ---> CodigoCliente	
{CodigoHabitacion,FechaEntrada} ---> TipoHabitacion	{CodigoHabitacion,FechaEntrada} ---> {TipoHabitacion,PrecioNoche,NombreCliente,CodigoCliente}
{CodigoHabitacion,FechaEntrada} ---> PrecioNoche	
{CodigoHabitacion,FechaEntrada} ---> NombreCliente	
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente	
{CodigoCliente,TipoHabitacion} ---> NombreCliente	{CodigoCliente,TipoHabitacion} ---> NombreCliente
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,PrecioNoche} ---> NombreCliente	{CodigoCliente,PrecioNoche} ---> NombreCliente
{CodigoCliente,FechaEntrada} ---> NombreCliente	{CodigoCliente,FechaEntrada} ---> NombreCliente
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente
{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente
{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente
{CodigoCliente,CodigoHabitacion} ---> TipoHabitacion	{CodigoCliente,CodigoHabitacion} ---> {TipoHabitacion,PrecioNoche,NombreCliente}
{CodigoCliente,CodigoHabitacion} ---> PrecioNoche	
{CodigoCliente,CodigoHabitacion} ---> NombreCliente	

Fuente: Elaboración propia.

G. Teoremas 2.4: Composición

Tabla 42: Teorema 2.4: Composición de Dependencias (Normalizadas).

Normalized Dependencies (Dependencies from Result of Theorem 2.3)	
Theorem 2.4 (Composición)	**Result**
CodigoCliente ---> NombreCliente	CodigoCliente ---> NombreCliente
CodigoHabitacion ---> TipoHabitacion, PrecioNoche	CodigoHabitacion ---> TipoHabitacion, PrecioNoche
TipoHabitacion -->> PrecioNoche	TipoHabitacion -->> PrecioNoche
PrecioNoche -->> TipoHabitacion	PrecioNoche -->> TipoHabitacion
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente, NombreCliente	{CodigoHabitacion,FechaEntrada} ---> CodigoCliente, NombreCliente

Fuente: Elaboración propia.

H. Teorema 2.1: Simplificar Dependencias con Sobreyección en el Lado Derecho

Tabla 43 : Teorema 2.1: Simplificar Dependencias con Sobreyección en el Lado Derecho (Normalizadas).

Normalized Dependencies (Dependencies from Result of Theorem 2.4)	
Theorem 2.1 (Simplificar Dependencias con Sobreyección en el Lado Derecho)	**Result**
CodigoCliente ---> NombreCliente	CodigoCliente ---> NombreCliente
CodigoHabitacion ---> TipoHabitacion, PrecioNoche	CodigoHabitacion ---> TipoHabitacion, PrecioNoche
TipoHabitacion -->> PrecioNoche	TipoHabitacion -->> PrecioNoche
PrecioNoche -->> TipoHabitacion	PrecioNoche -->> TipoHabitacion
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente, Nombre	{CodigoHabitacion,FechaEntrada} ---> CodigoCliente

Fuente: Elaboración propia.

I. Resultado de Tablas Normalizadas

Tabla 44 : Resultado de Tablas Normalizadas.

Normalized Dependencies (Dependencies from Result of Theorem 2.1)	
List of Normalized Dependencies	**Normalized Tables**
CodigoCliente ---> NombreCliente	NT1.1(CodigoCliente[PK],NombreCliente)
CodigoHabitacion ---> TipoHabitacion, PrecioNoche	NT1.2(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)
TipoHabitacion -->> PrecioNoche	NT1.3(TipoHabitacion[PK],PrecioNoche[PK])
PrecioNoche -->> TipoHabitacion	
{CodigoHabitacion,FechaEntrada} ---> CodigoCliente	NT2.1(CodigoHabitacion[PK],FechaEntrada[PK],CodigoCliente)

Fuente: Elaboración propia.

J. Resultado de Tablas No Normalizadas

Tabla 45 : Resultado de Tablas No Normalizadas.

Non-Normalized Dependencies (Dependencies from Result of Theorem 2.3)	
List of Non-Normalized Dependencies	**Non-normalized Tables**
TipoHabitacion -->> PrecioNoche	NNT1.3(TipoHabitacion[PK],PrecioNoche[PK])
PrecioNoche -->> TipoHabitacion	
CodigoHabitacion ---> {TipoHabitacion,PrecioNoche}	NNT1.2(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)
CodigoCliente ---> NombreCliente	NNT1.1(CodigoCliente[PK],NombreCliente)
{NombreCliente,CodigoHabitacion} ---> {TipoHabitacion,PrecioNoche,CodigoCliente}	NNT2.1(NombreCliente[PK],CodigoHabitacion[PK],TipoHabitacion,PrecioNoche,CodigoCliente)
{CodigoHabitacion,FechaEntrada} ---> {TipoHabitacion,PrecioNoche,NombreCliente,CodigoCliente}	NNT2.2(CodigoHabitacion[PK],FechaEntrada[PK],TipoHabitacion,PrecioNoche,NombreCliente,CodigoCliente)
{CodigoCliente,TipoHabitacion} ---> NombreCliente	NNT2.3(CodigoCliente[PK],TipoHabitacion[PK],NombreCliente)
{CodigoCliente,TipoHabitacion,PrecioNoche} ---> NombreCliente	NNT3.1(CodigoCliente[PK],TipoHabitacion[PK],PrecioNoche[PK],NombreCliente)
{CodigoCliente,PrecioNoche} ---> NombreCliente	NNT2.4(CodigoCliente[PK],PrecioNoche[PK],NombreCliente)
{CodigoCliente,FechaEntrada} ---> NombreCliente	NNT2.5(CodigoCliente[PK],FechaEntrada[PK],NombreCliente)
{CodigoCliente,FechaEntrada,TipoHabitacion} ---> NombreCliente	NNT3.2(CodigoCliente[PK],FechaEntrada[PK],TipoHabitacion[PK],NombreCliente)

{CodigoCliente,FechaEntrada,TipoHabitacion,PrecioNoche} ---> NombreCliente	NNT4.1(CodigoCliente[PK],FechaEntrada[PK],TipoHabitacion[PK],PrecioNoche[PK],NombreCliente)
{CodigoCliente,FechaEntrada,PrecioNoche} ---> NombreCliente	NNT3.3(CodigoCliente[PK],FechaEntrada[PK],PrecioNoche[PK],NombreCliente)
{CodigoCliente,CodigoHabitacion} ---> {TipoHabitacion,PrecioNoche,NombreCliente}	NNT2.6(CodigoCliente[PK],CodigoHabitacion[PK],TipoHabitacion,PrecioNoche,NombreCliente)

Fuente: Elaboración propia.

7. Anexo 7: Tablas Normalizadas

Tabla 46 : Tablas Normalizadas.

Normalized Tables
Clientes(CodigoCliente[PK],NombreCliente)
Habitaciones(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)
TiposHabitaciones(TipoHabitacion[PK],PrecioNoche[PK])
Ocupaciones(CodigoHabitacion[PK],FechaEntrada[PK],CodigoCliente)

Fuente: Elaboración propia.

8. Anexo 8: Tablas No Normalizadas

Tabla 47 : Tablas No Normalizadas.

Non-normalized Tables
NNT1.1(CodigoCliente[PK],NombreCliente)
NNT1.2(CodigoHabitacion[PK],TipoHabitacion,PrecioNoche)
NNT1.3(TipoHabitacion[PK],PrecioNoche[PK])
NNT2.1(NombreCliente[PK],CodigoHabitacion[PK],TipoHabitacion,PrecioNoche,CodigoCliente)

NNT2.2(CodigoHabitacion[PK],FechaEntrada[PK],TipoHabitacion,PrecioNoche,NombreCliente,CodigoCliente)
NNT2.3(CodigoCliente[PK],TipoHabitacion[PK],NombreCliente)
NNT2.4(CodigoCliente[PK],PrecioNoche[PK],NombreCliente)
NNT2.5(CodigoCliente[PK],FechaEntrada[PK],NombreCliente)
NNT2.6(CodigoCliente[PK],CodigoHabitacion[PK],TipoHabitacion,PrecioNoche,NombreCliente)
NNT3.1(CodigoCliente[PK],TipoHabitacion[PK],PrecioNoche[PK],NombreCliente)
NNT3.2(CodigoCliente[PK],FechaEntrada[PK],TipoHabitacion[PK],NombreCliente)
NNT3.3(CodigoCliente[PK],FechaEntrada[PK],PrecioNoche[PK],NombreCliente)
NNT4.1(CodigoCliente[PK],FechaEntrada[PK],TipoHabitacion[PK],PrecioNoche[PK],NombreCliente)

Fuente: Elaboración propia.