



UNIDAD 2 | RELATIONAL DATABASE MODELING AND IMPLEMENTATION

# RELATIONAL DATABASE DESIGN: PART I



Al finalizar la unidad de aprendizaje, el estudiante diseña, implementa y realiza operaciones de manipulación de datos sobre una base de datos relacional, satisfaciendo requisitos para un dominio y contexto determinados.

### **Logro de la semana:**

Al finalizar la semana de aprendizaje, el estudiante aplica el enfoque relacional para la elaboración de diseños que incluyen entidades de datos, sus características y relaciones.

---

# AGENDA

INTRODUCCION

RELATIONAL DATABASE DESIGN

ENTITIES & RELATIONSHIPS



# Database

Una colección de información coherente, siendo información una forma procesada de datos.

Las bases de datos se asocian normalmente con datos estructurados: datos que se almacenan según una clase de patrón regular, permitiendo que una computadora pueda procesarlos y filtrarlos, así como inferir respuestas a preguntas que no están establecidas como entradas originales en los datos.

# Data model

Un conjunto de reglas, supuestos y convenciones que sirven para transformar conceptos abstractos en el mundo real, en un modelo finito de objetos en la computadora por medio de un número fijo de conceptos reutilizables.

El **modelo relacional** es uno de los más conocidos y utilizados comercialmente, aunque no es el único.

---

# AGENDA

INTRODUCCION

RELATIONAL DATABASE DESIGN

ENTITIES & RELATIONSHIPS



# Relational database

Una base de datos relacional es aquella que organiza sus datos en colecciones de tablas, filas, atributos y dominios.

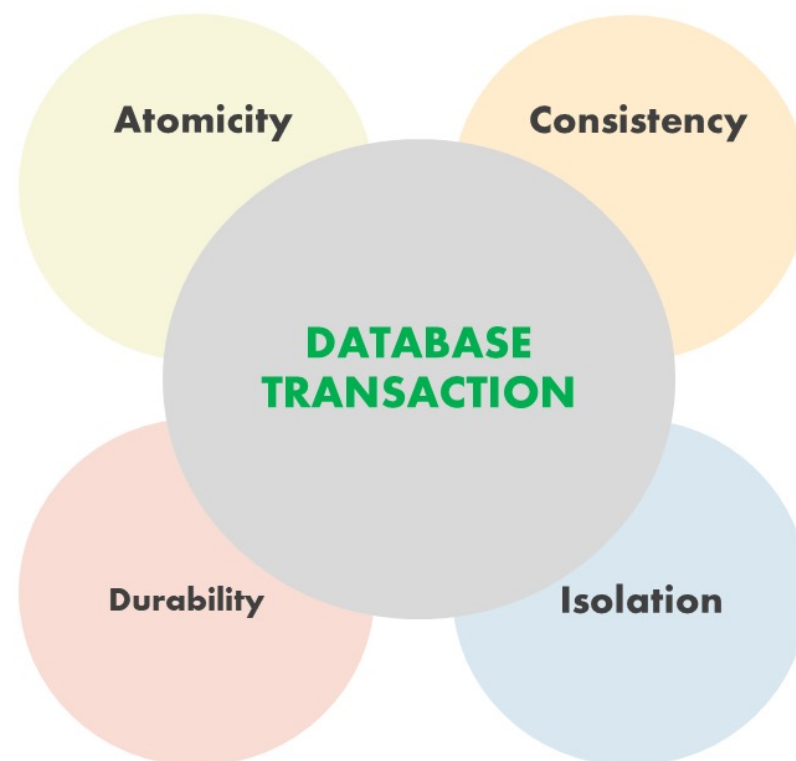
También conocida como SQL Database, en ella se utiliza la lógica de predicados para describir la información contenida en la base de datos y para realizar consultas sobre ella.



# Relation database design

El diseño de bases de datos SQL se apoya a menudo en técnicas como **normalization**, cuyo objetivo es reducir o eliminar datos duplicados en una base de datos para reducir errores en los datos almacenados. Cada tabla en una base de datos relacional contiene datos de un mismo tipo o concepto, por ejemplo, addresses.

Relational databases utilizan un conjunto de reglas conocidas como **ACID** (**A**tomicity, **C**onsistency, **I**ntegrity, **D**urability) que guían el database design.



# ACID

**Atomicidad:** O ocurre toda una transacción o no ocurre nada.

**Consistencia:** Una transacción lleva la base de datos de un estado consistente al siguiente.

**Aislamiento:** Los efectos de una transacción pueden no ser visibles para otras transacciones hasta que la transacción se haya confirmado.

**Durabilidad:** Una vez comprometida la transacción, es permanente.

# Relation database design: Steps

- Definir el propósito de la base de datos.
- Investigar y recolectar toda la información sobre los datos a incluirse y el propósito de la base de datos en el tiempo.
- Dividir los datos a incluirse en subjects o types, por ejemplo, información de user account. Cada uno de ellos se convertirá (o debería convertirse) en tablas de base de datos.
- Para cada tabla de base de datos, identificar los data point a incluir. Cada data point se convertirá en una columna en la tabla de base de datos.
- Para cada tabla de base de datos, identificar la óptima primary key para identificar de manera única a cada fila de la tabla.
- Comparar y evaluar cómo se relacionan entre sí los datos en las tablas. Adicionar campos o tablas, para aclarar las relaciones de datos en cada tabla. Por ejemplo, una base de datos con información de contactos para compañías podría necesitar incluir múltiples direcciones, números de teléfono entre otros datos para cada compañía.
- Probar el diseño de la base de datos en papel, escribiendo consultas para las tareas más recurrentes. Refinar el diseño de tablas según se requiera.
- Normalizar el diseño de base de datos para asegurarse que cada tabla represente una cosa o concepto, con referencias y relaciones hacia otras tablas según se necesite.

---

# AGENDA

INTRODUCCION

RELATIONAL DATABASE DESIGN

ENTITIES AND RELATIONSHIPS



# Entities & Their Attributes

- La mayoría de los diseños de bases de datos comienzan identificando entidades.
- Una entidad es algo sobre lo que **almacenamos datos**.
- Las entidades no son necesariamente tangibles.
- Las entidades tienen datos que las describen (sus atributos).
- Cuando representamos entidades en una base de datos, en realidad **almacenamos solo los atributos**.

Cada grupo de atributos que describe una única ocurrencia del mundo real de una entidad actúa para representar una **instancia** de una entidad. Por ejemplo, una entidad de cliente generalmente se describe mediante un número de cliente, nombre, apellido, calle, ciudad, estado, código postal y número de teléfono.



# Entity Identifiers

**Recuperar los datos** en una fecha posterior es el propósito de colocar los datos que describen una entidad en una base de datos (una instancia de la entidad).

Cada instancia de entidad tiene algunos valores de atributo que la distinguen de cualquier otra instancia de la misma entidad en la base de datos: un **identificador** de entidad.

# Single-Valued Versus Multivalued Attributes

Los atributos en nuestro modelo de datos deben tener un solo valor. Esto significa que para una instancia dada de una entidad, cada atributo puede tener solo un valor: **monovalorado**.

La existencia de más de un valor para un atributo lo convierte en un atributo **multivalorado**.

Debido a que una entidad en una base de datos relacional no puede tener atributos de varios valores, debes manejar esos atributos **creando una entidad** para contenerlos.

# Domains

Cada atributo tiene un dominio, una expresión de los valores permitidos para ese atributo.

Un DBMS impone un dominio a través de un *constraint* de dominio. Cada vez que se almacena un valor en la base de datos, el DBMS verifica que proviene del dominio especificado del atributo.

El dominio nos asegura que estamos obteniendo datos del tipo correcto. Por ejemplo, un DBMS puede evitar que un usuario almacene  $123 \times 50$  en un atributo cuyo dominio son valores de moneda.

Los dominios para los atributos teóricamente deberían ser independientes del DBMS que se utilizará. Sin embargo, en términos prácticos, tiene poco sentido asignar dominios que no puede implementar. Por lo tanto, el diseñador de la base de datos que deber dar un vistazo al DBMS para ver qué tipos de datos son compatibles.



# Entity Relationship Diagram

**Entity Relational (ER) Model** es un diagrama conceptual de alto nivel de un modelo de datos. Representa entidades del mundo real y sus relaciones entre ellas.

Un **Entity-Relationship Diagram (ERD)**, es una herramienta visual de utilidad para representar un Entity Relationship Model.

Entity-Relationship Diagram proporcionan una forma de documentar las entidades en una base de datos, junto con los atributos que las describen.

Hay varios estilos de diagramas ER. Hoy en día, existen tres métodos principales: el modelo Chen (llamado así por el creador del modelado ER, el Dr. Peter P.S. Chen), Information Engineering y Unified Modeling Language (UML).

# Entity Relationship Diagram

Components:

- Entities
- Attributes
- Relationships

# Components

**Entity.** Concepto del mundo real.

**Attribute.** Características o propiedades de un entity.

**Relationship.** Dependencia o asociación entre entities.

Ejemplos:

*Customer* y *Product* son entidades.

*Customer number* y *name* son atributos de *Customer*.

*Product name* y *price* son atributos de *Product*.

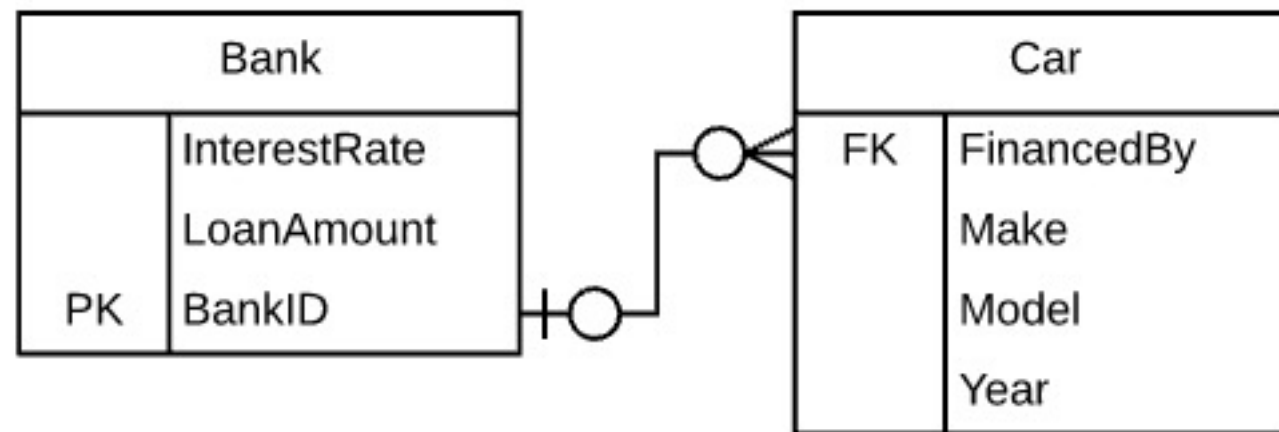
*Sale* es la relación entre *Customer* y *Product*.

# Levels

**Conceptual.** Entities y Relationships

**Logical.** Entities, Attributes y Relationships.

**Physical.** Todas las estructuras de tables, incluyendo column name, column data type, column constraints, primary key, foreign key y relationships entre tables.



# Basic Data Relationships

Antes de pasar a los tipos de relaciones en sí, hay una cosa importante a tener en cuenta: las relaciones que se almacenan en una base de datos son entre instancias de entidades.

**One-to-One Relationships:** Si tenemos instancias de dos entidades (A y B) llamadas  $A_i$  y  $B_i$ , entonces existe una relación de uno a uno si en todo momento  $A_i$  no está relacionado con ninguna instancia de la entidad B o con una instancia de la entidad B, y  $B_i$  está relacionado a ninguna instancia de la entidad A o a una instancia de la entidad A. Las verdaderas relaciones uno a uno son muy raras en los negocios.

**One-to-Many Relationships:** El tipo de relación más común es una relación de uno a muchos. Si tenemos instancias de dos entidades (A y B), entonces existe una relación de uno a muchos entre dos instancias ( $A_i$  y  $B_i$ ) si  $A_i$  está relacionada con cero, una o más instancias de la entidad B, y  $B_i$  está relacionada a cero o una instancia de la entidad A.

**Many-to-Many Relationships:** Existe una relación de muchos a muchos entre las entidades A y B si para dos instancias de esas entidades ( $A_i$  y  $B_i$ ),  $A_i$  se puede relacionar con cero, una o más instancias de la entidad B, y  $B_i$  se puede relacionar con cero, una o más instancias de la entidad A.







# Weak Entities and Mandatory Relationships

La identificación de entidades débiles y sus relaciones obligatorias asociadas puede ser muy importante para mantener la coherencia y la integridad de la base de datos. Considere el efecto, por ejemplo, de almacenar un pedido sin conocer el cliente al que pertenece. No habría forma de enviar el artículo al cliente, lo que provocaría que la empresa perdiera negocios.

# Cardinality & ordinality

**Cardinality.** Máximo número de veces que una instancia de entidad se puede relacionar con instancias de otra entidad.

**Ordinality.** Mínimo número de veces que una instancia de una entidad puede asociarse con una instancia de la entidad relacionada.

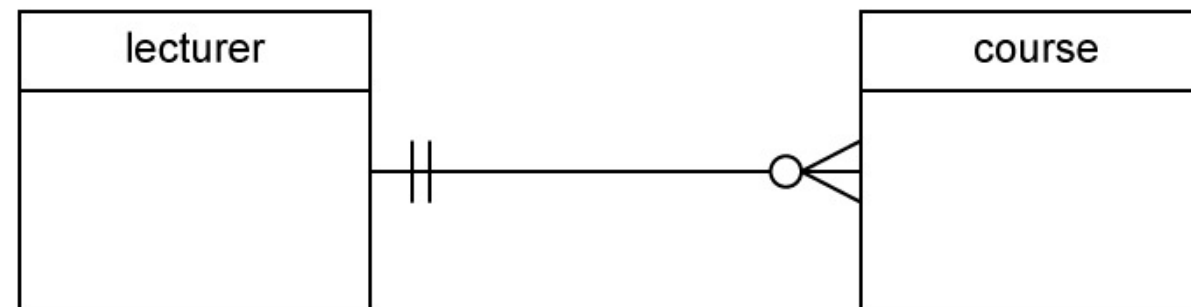
	One
	Many
	One (and only one)
	Zero or one
	One or many
	Zero or many

# Cardinality & ordinality

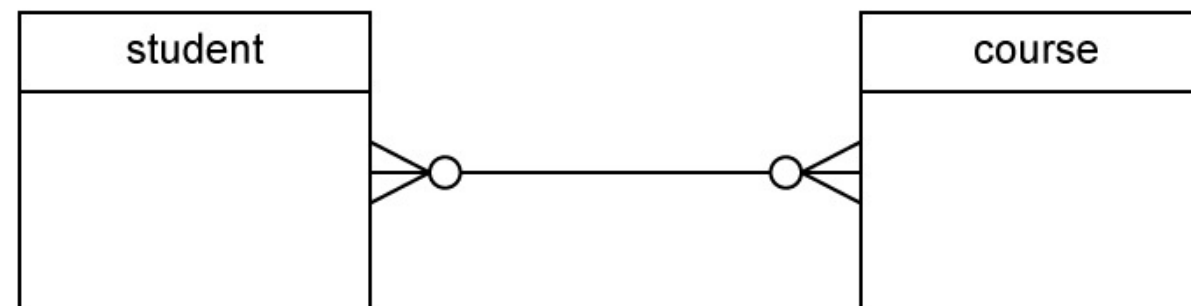
One-to-one



One-to-many

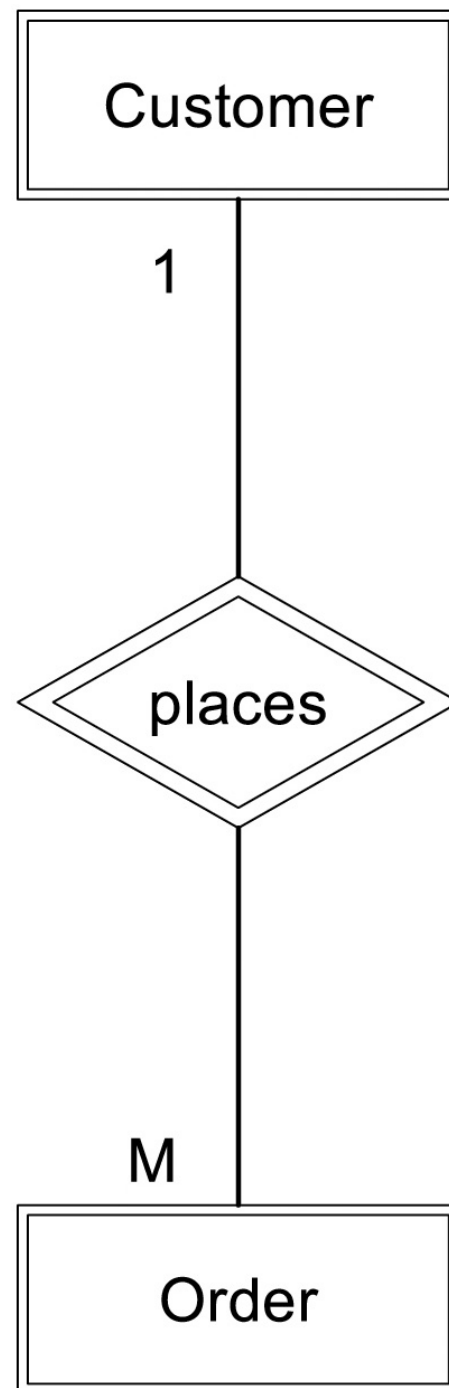


Many-to many

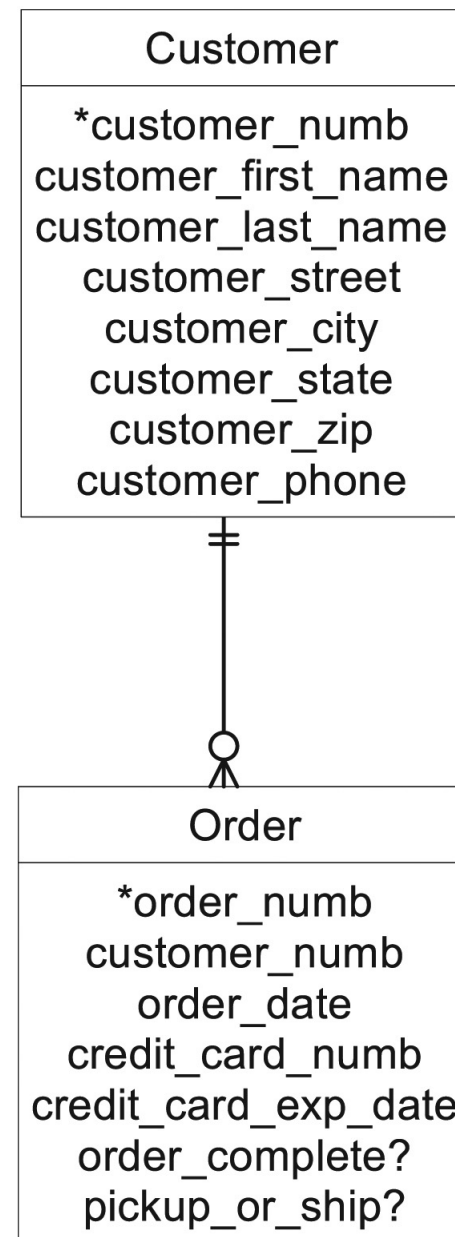




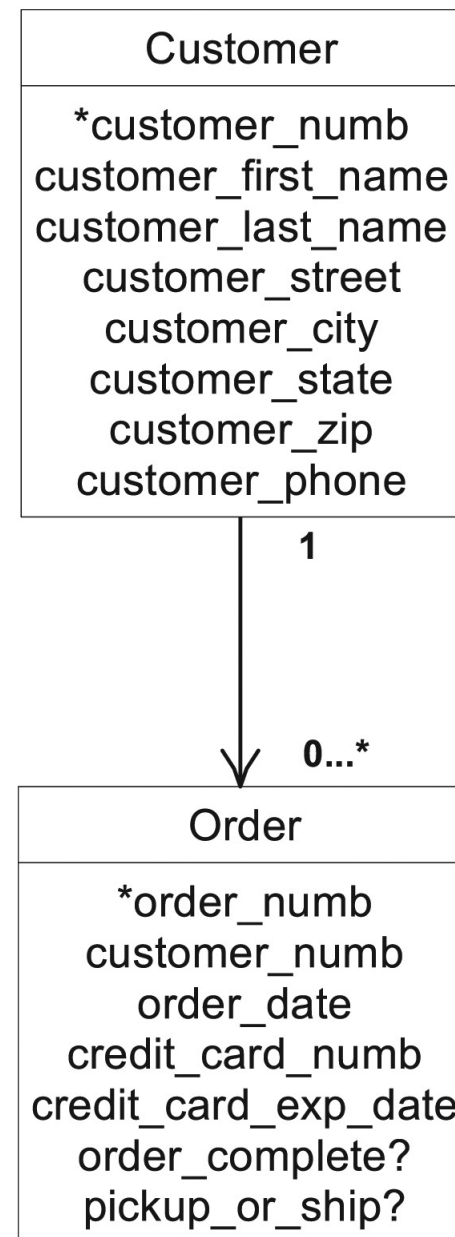
# The Chen Method



# IE Style Diagrams



# UML Style Diagrams



# ERD logical naming conventions

Elemento	Convención	Ejemplo
Entity	Noun, Upper Camel Case, Singular	Person, Customer
Attribute	Noun, Lower Camel Case, Singular	name, starsCount
Relationship	Verb, Present Tense, Third-Person, Upper Camel Case	Sales, Produces, Holds

# ERD physical naming conventions

Elemento	Convención	Ejemplo
Table	Noun, Snake Case, Plural	people, customers, order_items
Column	Noun, Snake Case Singular	name, stars_count
Primary Key Column	id	id
Foreign Key Column	<entity_name>_id	customer_id, product_id

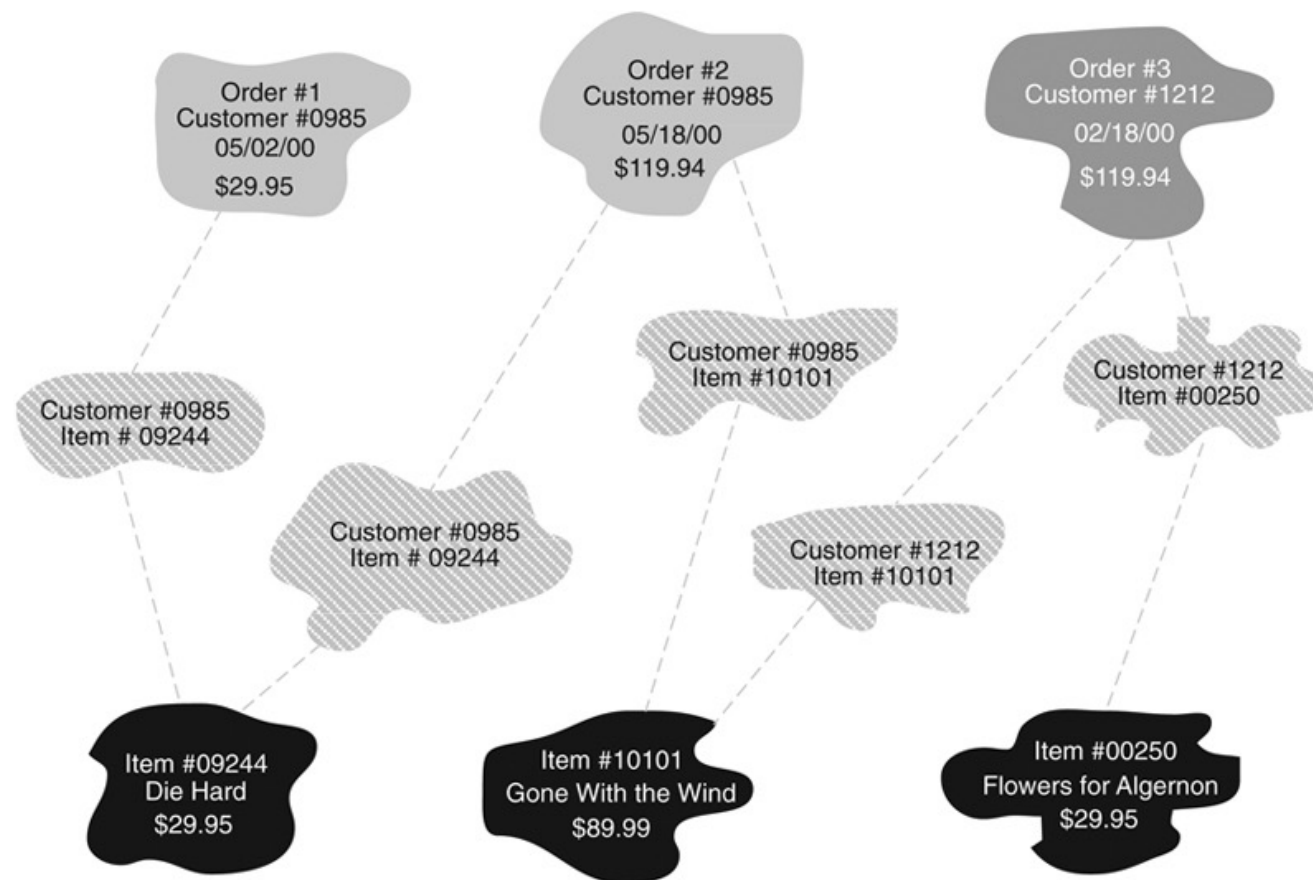
# Audit trail fields naming conventions

Elemento	Logical	Physical
Creation Date	createdAt	created_at
Creation User ID	createdBy	created_by
Last Update Date	updatedAt	updated_at
Last Update User ID	updatedBy	updated_by

# Dealing with Many-to-Many Relationships

El modelo de datos relacional no puede manejar relaciones de muchos a muchos directamente; se limita a relaciones de uno a uno y de uno a muchos.

**Composite Entities:** Las entidades que existen para representar la relación entre dos o más entidades se conocen como entidades compuestas.



# Many-to-Many relationship table naming conventions

**Recomendable.** Identificar un término que represente el concepto de la relación.

**Válido.** Término compuesto por nombres de ambas entidades que se relacionan.

From Entity	To Entity	From Table	To Table	Join Table
Warehouse	Product	warehouses	products	inventories
Student	Course	students	courses	course_enrolments
User	Plan	users	plans	subscriptions
Agent	Property	agents	properties	agent_assignments



---

# RESUMEN

## Recordemos

Relational database design

Entities & relationships

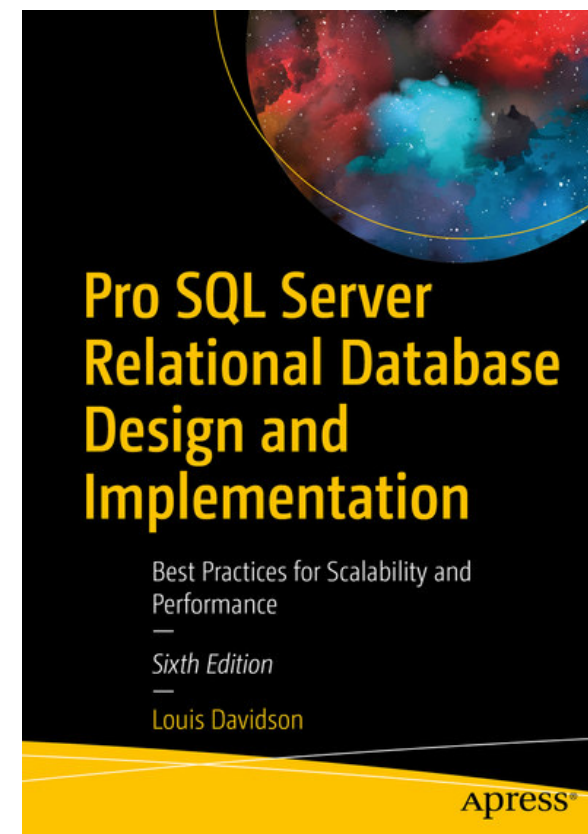
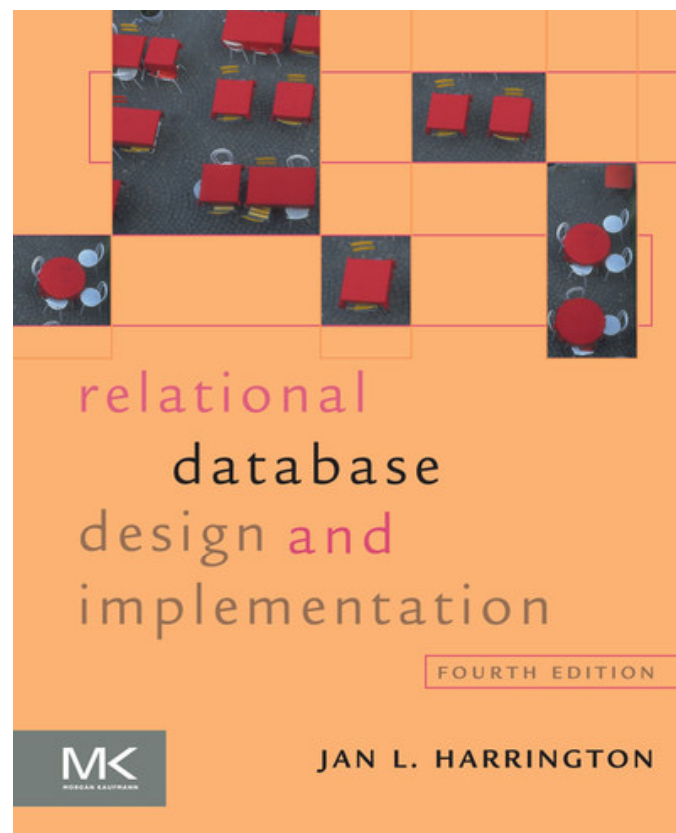


# REFERENCIAS

## Para profundizar

Relational Database Design and Implementation

Pro SQL Server Relational Database Design and Implementation



WWW

# PREGRADO

## Ingeniería de Software

Escuela de Ingeniería de Sistemas y Computación | Facultad de Ingeniería



**UPC**

Universidad Peruana  
de Ciencias Aplicadas

Prolongación Primavera 2390,  
Monterrico, Santiago de Surco  
Lima 33 - Perú  
T 511 313 3333  
<https://www.upc.edu.pe>

***exígete, innova***