

# Capítulo 1

## Introducción al Diseño de Bases de Datos Relacionales

### Objetivos

Al final de este módulo, Ud. Será capaz de:

- Definir el término *base de datos*
- Definir el término *modelo de datos*
- Identificar los factores de éxito críticos para diseñar una base de datos relacional

# ¿Qué es una Base de Datos?

Una base de datos es una colección de datos relacionados que pueden servir a múltiples propósitos y soportar varios usuarios

Una base de datos es una colección de datos relacionados. Una base de datos permite que la definición de datos y las relaciones entre los datos estén separados de las aplicaciones individuales.

Una base de datos puede tener mas de un uso y puede ser usada por varios usuarios simultáneamente. Los datos contenidos en una base de datos pueden ser vistos por los usuarios en muchas diferentes formas. El compartir los datos es el principal objetivo de un sistema de base de datos.

## **Sistemas de archivos convencionales: Desventajas**

- Redundancia de datos
- Problemas de integridad de datos
- Compartición de datos limitada
- Restricciones en la disponibilidad de datos
- Dificultades para la administración de los datos

Los sistemas de archivos convencionales tienen muchas desventajas. Por ello, los negocios están cada vez más girando hacia los sistemas de bases de datos que proveen una poderosa funcionalidad y mayor flexibilidad que los sistemas de archivos convencionales.

Muchos elementos de datos son usados repetidamente en múltiples aplicaciones. Los sistemas de archivos convencionales requieren que la misma información sea almacenada en múltiples archivos. Almacenar datos repetidamente es conocido como *redundancia de datos*.

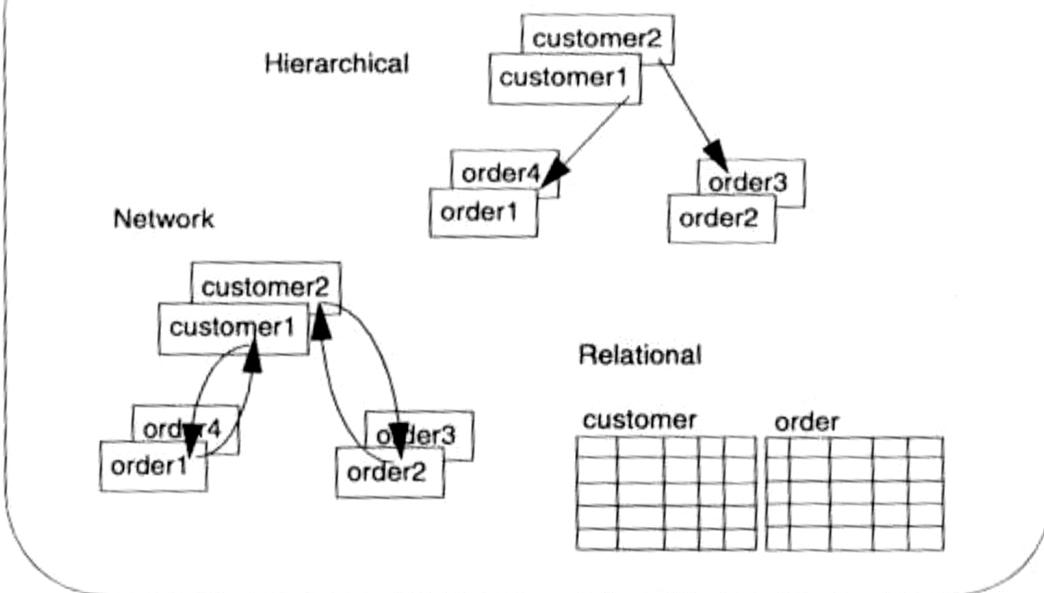
La redundancia de datos lleva hacia muchos problemas concernientes a la integridad de los datos. La inconsistencia de datos es más probable que ocurra cuando los datos están almacenados en mas de un lugar durante las operaciones de captura, actualización y eliminación.

Los sistemas de archivos convencionales proveen una capacidad limitada de compartir los datos. La mayoría de los archivos de datos convencionales están implementados como unidades separadas, haciendo difícil, si no imposible, compartir los datos entre múltiples aplicaciones.

En los negocios del día de hoy, los datos deben ser estar disponibles para su lectura dondequiera que el usuario lo requiera. Si los datos requeridos son recuperados desde varios archivos, la disponibilidad de los datos puede ser restringida.

Sistemas de archivos con redundancia de datos hacen al manejo de los datos difícil de controlar. Si un administrador de datos necesita introducir un nuevo cambio a través de la organización, pueden ocurrir algunas dificultades en su implementación.

## Types of Databases



Introduction to Relational Database Design 1860 01-96 4

Previo a las bases de datos relacionales, se han usado principalmente dos tipos de bases de datos: jerárquicas y de red.

El modelo de base de datos jerárquica está constituida por archivos con apunadores desde los archivos padre a los archivos hijo, ligando la información relacionada. Por ejemplo, si una base de datos jerárquica tuviera información acerca de los clientes y sus pedidos, el archivo **cliente** podría tener apunadores desde cada cliente a la localidad física de cada pedido del cliente en el archivo **pedidos**.

Con este modelo, las búsquedas de datos de arriba hacia abajo de la jerarquía son fáciles (por ejemplo, qué pedidos ha hecho cada cliente); sin embargo, las búsquedas de información hacia arriba en la jerarquía son más difíciles (por ejemplo, cuál cliente hizo este pedido). También es difícil de representar datos no-jerárquicos usando este modelo.

Las bases de datos tipo red son como las bases de datos jerárquicas, excepto que hay apunadores en ambas direcciones ligando la información relacionada. En el ejemplo de clientes y pedidos descrito arriba, podría haber apunadores no solamente entre cada cliente y sus pedidos, sino también habría un apuntador desde cada pedido de regreso hacia el cliente que hizo ese pedido. Mientras que este modelo resuelve algunos de los problemas asociados con el modelo jerárquico, el ejecutar consultas simples es un proceso bastante complicado. Además, debido a que la lógica de un procedimiento depende de la organización física de los datos, este modelo no es completamente independiente de la aplicación.

El modelo relacional representa los datos como un juego de relaciones en las cuales los datos están almacenados. Las ventajas ofrecidas por esta aproximación están descritas en las siguientes páginas.

Este curso se refiere al diseño de bases de datos relacionales. No cubre los modelos jerárquicos o de red.

## Ventajas de una base de datos relacional

- Reduce la redundancia de datos
- Asegura la integridad de los datos
- Refuerza la seguridad de los datos
- Puede ser usada concurrentemente por mucha gente
- Soporta la compartición de datos
- Se ajusta a las necesidades de cambios, crecimiento y nueva información

Una base de datos relacional provee de muchas ventajas sobre los sistemas de archivos convencionales y las bases de datos jerárquicos y de red. Una base de datos relacional permite una poderosa manipulación de datos a través de una variedad de sentencias del lenguaje de consulta estructurado (SQL) y los operadores relacionales.

Una base de datos relacional reduce la redundancia de los datos. Los datos pueden ser almacenados una vez y relacionados con otros a través de una variedad de construcciones. Limitar la redundancia de datos mejora la integridad de los datos y puede reducir significativamente el monto de espacio en disco requerido.

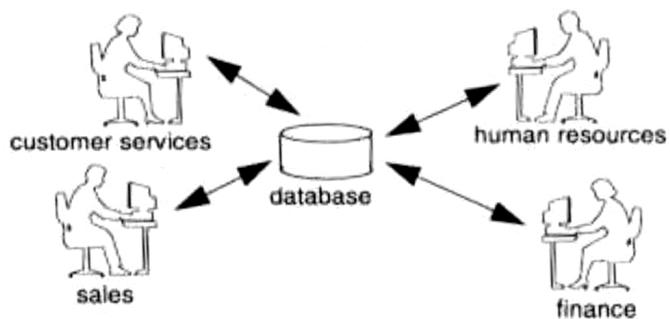
Las reglas de integridad de los datos pueden ser incorporadas en los datos contenidos en una base de datos relacional. Estas reglas, las cuales son introducidas en un modulo posterior, son conocidas como restricciones (*constraints*) *referenciales* y de *integridad de entidades*.

Una base de datos relacional soporta la concurrencia de uso por mucha gente y muchas diferentes aplicaciones. Los datos almacenados en una base de datos relacional pueden tener mas de un uso y pueden ser compartidos por muchos usuarios.

Una base de datos relacional puede ser fácilmente modificada para acomodarse al crecimiento y requerimientos de nueva información. Una base de datos relacional provee de facilidades para construir un diseño, flexibilidad para cambiar el diseño e independencia de la aplicación.

## Relational Database Management System

An *RDBMS* is a system that integrates data files into a database and makes it possible to access integrated data that crosses operational, functional, and organizational boundaries within an enterprise.

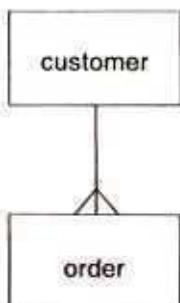


Un sistema de base de datos relacional (*relational database management system RDBMS*) puede satisfacer varios objetivos muy importantes:

- Un RDBMS puede servir efectivamente a diferentes funciones de los negocios de la empresa.
- El monto de redundancia de datos puede ser minimizado.
- Se puede obtener información consistente y precisa.
- Puede aplicarse e implementarse controles en la seguridad de los datos.
- Existen utilerías para el respaldo y recuperación.
- Los programas de aplicación pueden ser desarrollados, cambiados y mantenidos más rápidamente, mas económicoamente y con personal menos capacitado.
- La reorganización física de los datos puede ser fácilmente completada.
- Es posible la administración y control centralizados de la base de datos.

## The Data Model

A *data model* is a logical representation that defines the units of data and specifies how each unit of data is related to the others.



El *modelo de datos* es una representación lógica de los datos organizados. Sirve como una herramienta de comunicación para los usuarios finales y diseñadores de bases de datos. El modelo de datos se usa para organizar, visualizar, planear y comunicar ideas.

El dibujo mostrado arriba enseña un modelo de datos simple para la relación entre clientes y pedidos.

Hay muchas metodologías diferentes para el diseño de base de datos relacional. Este curso enseña el diseño de base de datos relacional usando los modelos de *entidad-relación*. Al terminar este curso, Ud. podría desear evaluar otras metodologías para determinar cual trabaja mejor para Ud.

## The Entity-Relationship Data Model

- Schema diagram formalized in the late 1960s by C.W. Bachman
- Emphasis on simplicity and readability
- Provides an accurate logical representation of the business enterprise
- Many CASE tools offer interactive modeling capability using the entity-relationship (ER) approach.

Cuando se va a diseñar una base de datos relacional, se crea un diagrama para ayudar a entender los requerimientos de datos. El diagrama puede también servir como una definición, o *esquema* de la base de datos.

Los primeros diagramas de esquema fueron formalizados por C.W. Bachman al final de los 60's. La teoría relacional fue formalizada por el Dr. E.F. Codd en 1970. La aproximación entidad-relación para modelar datos ha sufrido muchos cambios para soportar extensiones para modelos más complejos que desarrollados como teoría relacional evolucionaron desde el ambiente académico hacia el ambiente de los negocios de RDBMS.

Existen dos niveles de definiciones usados en la definición de modelos de datos: simple y complejo. El nivel simple es usado por muchas herramientas CASE (Computer-Aided Software Engineering) y es el nivel de definición usado en este curso. El nivel simple de definición es suficiente para modelar la mayoría de las empresas de negocios existentes. El segundo y más complejo nivel incluye conceptos para modelos semánticos o inteligencia artificial (AI) y modelos conceptuales usados en el método de análisis de información de Nijssen (NIAM).

## Benefits of an ER Model

- Focuses discussions on the importance of relationships
- Uses diagrammatic syntax that conveys a great amount of information in a readily understandable form
- Captures real-world data requirements that are meaningful to the end user and database designer
- Can be used to build the physical database schema

Hay muchos beneficios asociados al modelo ER:

- El modelo ER se convierte en una poderosa herramienta al concentrar una gran cantidad de información en una forma simple y de fácil lectura.
- El modelo ER puede ser usado como una herramienta de comunicación entre el usuario final y el diseñador de la base de datos.
- El modelo ER se transforma en una excelente fuente de documentación para los administradores de la base de datos y los desarrolladores de aplicaciones.

En un modelo ER, los objetos importantes de información y sus relaciones pueden ser organizados y fácilmente entendidos. Este modelo lógico puede ser usado como un mapa para construir la base de datos física.

## Critical Success Factors for Database Design

- Comprehensive and in-depth understanding of the business requirements
- Communication with end users throughout the design process
- Employment of a structured methodology throughout all phases of development
- Use of diagrams to represent the data model

Hay muchos factores importantes que contribuyen al exitoso diseño de un modelo de datos. Aprendiendo y aplicando las varias técnicas de modelado de datos aisladamente no garantizan un modelo exitoso. Un diseñador de base de datos debe también tener un conocimiento amplio y profundo de los requerimientos y negocios de la empresa.

Debe existir la interacción y comunicación productiva entre el usuario y el diseñador de la base de datos a través del proceso de diseño. Además, el uso de la metodología estructurada es crítico durante el ciclo de vida del desarrollo.

## Where Database Design Fits into the Application Development Process

- Feasibility
- Functional requirements
- Technical design
- Coding and testing
- Implementation
- Maintenance

Database design

Existen muchas fases y entregables asociados con una metodología de desarrollo de aplicaciones. El diseño de la base de datos ocurre durante las fases de requerimientos funcionales y diseño técnico.

## Database Design Approach

- Gain an understanding of the business.
- Identify the principal data objects (entities, attributes, and relationships).
- Diagram the data objects using the entity-relationship approach.
- Resolve the logical data model.
- Determine attribute specifications and data types.
- Verify the logical data model through normalization.
- **Use SQL to convert the logical data model to a physical database schema.**

Los pasos descritos arriba representan la aproximación al diseño de la base de datos que serán cubiertos en este curso.

# Capítulo 2

## Conceptos y terminología Entidad-Relación

### Objetivos

- Al final de este módulo, Ud. será capaz de:
- Describir los conceptos básicos de la tecnología relacional
- Definir los términos usados en el diseño de base de datos relacional
- Identificar los objetos básicos en un modelo Entidad-Relación

## Database Design Approach

- Gain an understanding of the business.
- Identify the principal data objects (entities, attributes, and relationships).
- Diagram the data objects using the entity-relationship approach.
- Resolve the logical data model.
- Determine attribute specifications and data types.
- Verify the logical data model through normalization.
- **Use SQL to convert the logical data model to a physical database schema.**

Este módulo cubre los primeros dos pasos de la aproximación de diseño de base de datos descrita arriba.

## Three Classes of Objects

- Entities
- Relationships
- Attributes

Existen tres clases de objetos representados en los modelos de datos entidad-relación: entidades, relaciones y atributos.

## Entity

An *entity* is a major data object that is of significant interest to the user. It is usually a person, place, thing, or event.

employee

Una *entidad* es un objeto de datos que tiene particular interés para el usuario. Una entidad puede representar algo real, tangible o abstracto. Las entidades usualmente son nombradas como una persona, lugar, cosa o evento. Por ejemplo, *empleado* podría ser una entidad en un sistema de recursos humanos.

Un buen momento para descubrir entidades es durante el proceso de entrevista al usuario. Pida al usuario que describa sus actividades de negocio. Registre cualquier información acerca de cosas que el usuario mencione durante las entrevistas. Busque por los principales objetos de interés (gente, lugares, cosas o eventos). La lista inicial de entidades puede cambiar durante el proceso de entrevista. Es práctica común listar las entidades en forma singular.

### **Consejo:**

Si necesita ayuda para iniciar el proceso de entrevista, pida a los usuarios nombrar gente, lugares, cosas o eventos que ellos necesiten registrar.

*entity type*

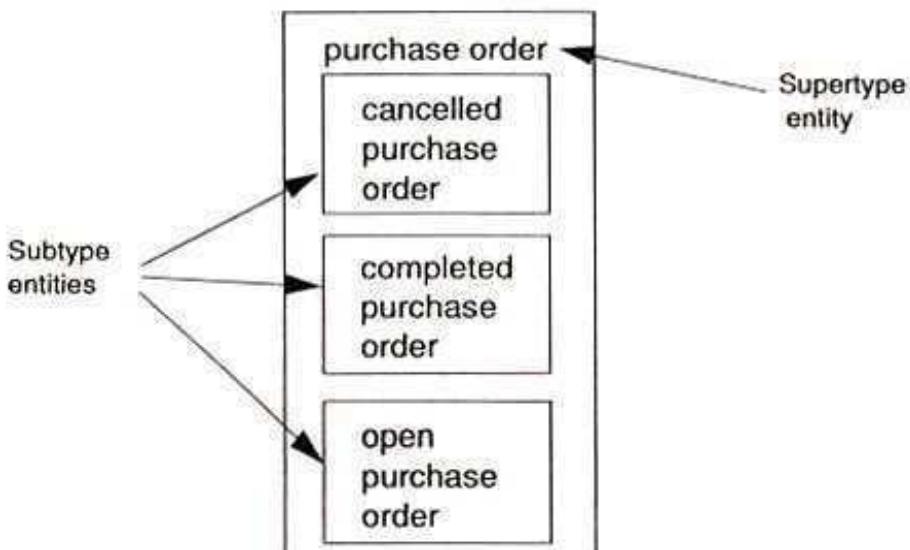
## Entity Instance

A particular occurrence or individual instance of an entity.

employee number:	537
name:	<i>Chris Parker</i>
address:	1234 Main Street
city:	New York
state:	NY
zip:	10017
phone:	212-555-9876

Una instancia individual de una entidad es referida como una *instancia* de la entidad ó una *ocurrencia* de una entidad. Por ejemplo, podría haber muchos empleados representados en una entidad **empleado**. Una instancia podría ser un empleado específico.

## Subtype and Supertype Entities



Las entidades *subtipo* son entidades que representan los mismos objetos del mundo real que otra entidad, pero con una ligera diferencia o características mas específicas. Ud. puede identificar varias entidades que son realmente diferentes formas de la misma entidad padre. La entidad padre es llamada entidad *supertipo*. Las entidades subtipo son a menudo, pero no siempre, mutuamente exclusivas. Esto significa que cada instancia de una entidad supertipo puede pertenecer a solamente una entidad subtipo en un momento dado.

En el ejemplo de arriba, **purchase order** (orden de compra) es la entidad supertipo y las entidades subtipo son **cancelled purchase order**, **complete purchase order** y **open purchase order**. Dado que cada orden de compra cae dentro de una de estas categorías en un momento dado, las entidades subtipo en este ejemplo son mutuamente exclusivas.

Es muy benefico poder diagramar subtipo en su modelo de datos. El uso de subtipos puede ayudar a verificar que su modelo tiene *sentido de negocio*.

Es muy beneficioso que sea capaz de hacer el diagrama de subtipos in tu modelo de datos. El uso de subtipos puede ayudarte a verificar que tu modelo tenga sentido en el *negocio*.

*¿Será importante que obtengas la información directamente de la tienda?*

"Si, a mi me parece correcto hacer eso. De esa forma, puedo mantener la pista de cuales videos se encuentran en cada tienda. Oh, y también puedo encontrar cual tienda esta rentando la mayor cantidad de videos. ¿Crees que puedes ayudarme en eso?"

**Sugerencia:** Advertir que la mayoría de las entrevistas estan abiertas-terminadas. Preguntas al aire durante el proceso de entrevistar puede ayudarte a estar seguro de que obtienes toda la información acerca del negocio que necesitas.

**Nota:** Las Entidades identificadas durante la entrevista estan en **Negritas**.

## **EJERCICIO DE LABORATORIO # 1**

## Class Discussion

- What were the most effective interview questions?
- Why were they effective?

Después de completar los Ejercicios de Laboratorio 1, contesta las preguntas en la parte inferior.

## Relationships

- A relationship represents real world associations between one or more entities.
- A verb or preposition connecting two entities usually implies a relationship.
- Identifying and understanding the entities and relationships are the most important part of the relational database design process.

El segundo paso en el modelado de datos relacionales es identificar las relaciones entre las entidades. Una *relación* es una asociación entre dos entidades. Algunas veces es difícil identificar relaciones. Es muy importante escuchar todo aquello que el usuario está diciendo. Usualmente, un *verbo* o una *preposición* conectando dos entidades implica una relación.

Identificar y definir relaciones entre entidades es la parte más importante del proceso de diseño de bases de datos relacionales. Es crucial que el diseñador entienda completamente los negocios de la empresa.

## Relationships: Video Rental System

	employee	customer	store	video	price change	rental
employee						processes
customer				rents		places
store				has		
video		is rented by	is in		has	is on
price change				is for		
rental	is processed by	is placed by		consists of		

La tabla mostrada arriba muestra *relaciones* que fueron identificadas en la entrevista acerca del sistema de renta de videos. Un simple verbo o una preposición es puesta en el cuadro entre las entidades para identificar la relación.

You Should Now Complete:

Lab Exercise

2

EJERCICIO DE LABORATORIO # 2

## Relationship Definition

Described in terms of:

- Connectivity
- Cardinality
- Existence
  - One-to-one (1:1)
  - One-to-many (1:N)
  - Many-to-many (M:N)

Una relación entre entidades se describe en términos de *conectividad, cardinalidad y existencia*.

Conectividad, cardinalidad y existencia ayudan a definir las *reglas de negocio* para una empresa de negocios. La aproximación entidad-relación para modelar datos provee sintaxis diagramática para representar relaciones.

## Connectivity

Connectivity describes the *number* of entity instances allowed in a relationship based on the business rules of an enterprise.

Types of connectivity:

- One-to-one (1:1)
- One-to-many (1:N)
- Many-to-many (M:N)

La representación más común de una relación es la *conectividad*. Existen tres tipos de conectividad:

- Uno-a-uno
- Uno-a-muchos
- Muchos-a-muchos

La relación uno-a-muchos es la más común. En general, Ud. debería tratar de representar relaciones como uno-a-muchos.

La reglas de negocio de una empresa definirán y determinarán el diseño de su modelo de datos. En algunos casos, Ud. necesitará acomodar excepciones, particularidades o consideraciones especiales en el diseño final.

## Connectivity: Video Rental System

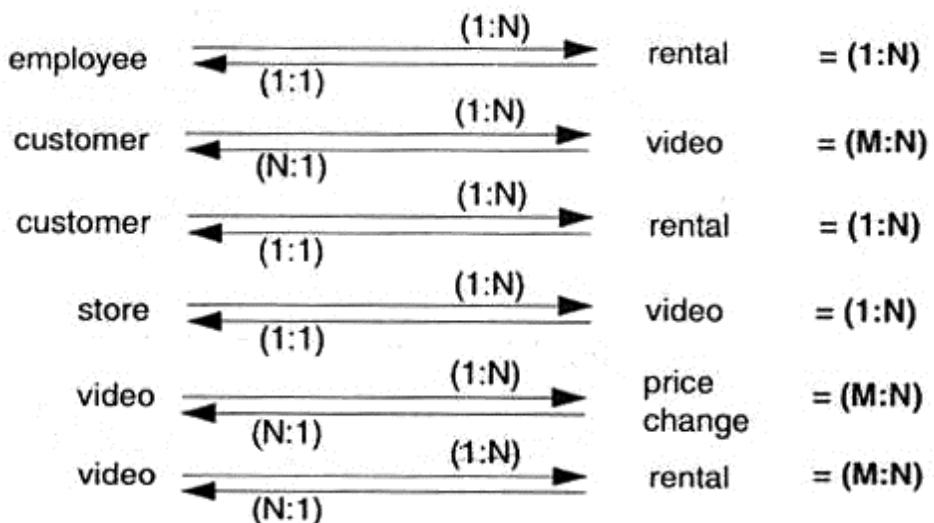
	employee	customer	store	video	price change	rental
employee						1:N
customer				1:N		1:N
store				1:N		
video		1:N	1:1		1:N	1:N
price change				1:N		
rental	1:1	1:1		1:N		

La conectividad de las entidades para el sistema de renta de videos es mostrada arriba. *Un empleado procesa muchas rentas. Un cliente puede rentar muchos videos y puede rentar muchas veces. Una tienda tiene muchos videos. Un video puede tener muchos cambios de precio y puede estar en muchas rentas.*

De la misma forma, *una renta es procesada por un empleado. Un video puede ser rentado por muchos clientes, pero una renta es hecha por un cliente. Un video esta en una tienda. Un cambio de precio es para muchos videos y una renta puede consistir de muchos videos.*

Note que la conectividad entre cada par de entidades esta especificado en ambas direcciones (por ejemplo, una tienda tiene muchos videos y un video esta en una tienda). Ud. usará esta información para determinar la conectividad correcta para cada relación y descubrirá cualquier relación muchos-a-muchos. En un módulo posterior, Ud. aprenderá a resolver relaciones muchos-a-muchos.

## Determining the Correct Connectivity



Ahora examine las relaciones en ambos sentidos. Combinando las relaciones, determinará la conectividad correcta:

- Dos uno-a-uno (1:1) = uno-a-uno (1:1)
- Dos uno-a-muchos (1:N) = muchos-a-muchos (M:N)
- Un uno-a-uno (1:1) y un uno-a-muchos (1:N) = uno-a-muchos (1:N)

La conectividad para cada relación en el sistema de renta de videos se muestra en el dibujo de arriba. Note que las relaciones cliente-a-video, video-a-cambio de precio y renta-a-video son muchos-a-muchos. Posteriormente resolveremos esas relaciones. Las otras relaciones son uno-a-muchos.

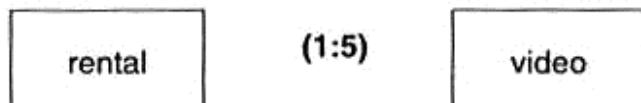
*Consejo:*

Para ayudar a identificar el tipo de conectividad, trate de pensar en la conectividad en términos de instancias de entidades individuales. Por ejemplo, observe la relación particular entre una tienda en particular y un video en particular dentro de esa tienda.

## Cardinality

*Cardinality* defines the constraints on the number of entity instances that are related through a relationship.

Video Rental System



*Cardinalidad* define cualquier restricción en el máximo número esperado de instancias de una entidad.

En el sistema de renta de videos, hay una restricción (constraint) entre las entidades **renta** y **video**. Una renta puede consistir de un máximo de cinco videos.

Es importante identificar cualquier restricción (constraint) de cardinalidad en el diseño de la base de datos. Las constraints necesitan ser consideradas en el diseño de la aplicación.

## Existence Dependency

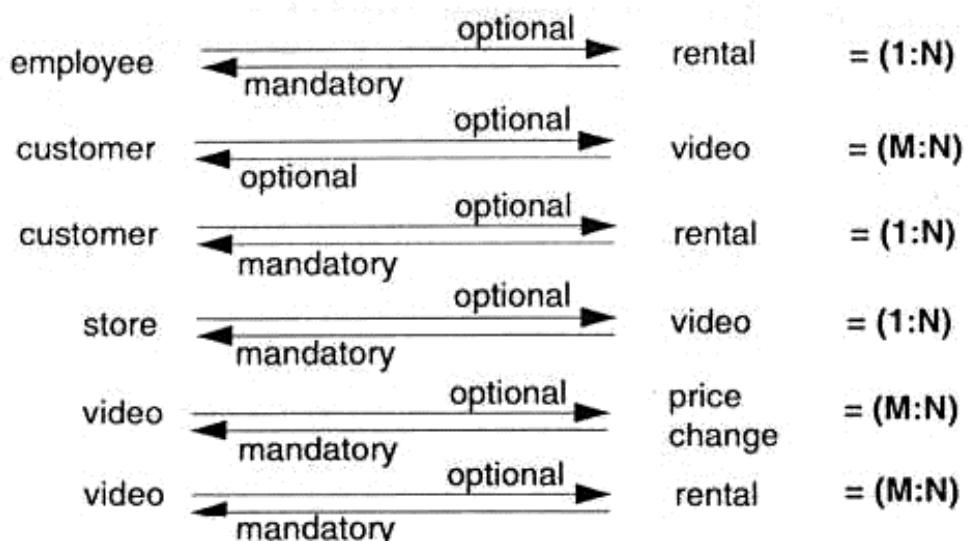
- *Mandatory* - if an instance of the entity must always exist in the relationship
- *Optional* - if an instance of the entity need not exist in the relationship

Existencia dependiente.

Existencia dependiente define si la existencia de una entidad en una relación es opcional ú obligatoria.

Observando cada relación en ambas direcciones, Ud. puede determinar donde las entidades son opcionales ú obligatorias. El modelo entidad-relación provee la sintaxis diagramática para indicar la existencia dependiente. Esta es una consideración muy importante cuando se diseña la base de datos física y se desarrollan aplicaciones.

## Existence Dependency: Video Rental



Las constraints de existencia dependiente para el sistema de renta de videos se muestran arriba. La relación empleado-a-renta es opcional dado que un empleado puede existir sin ninguna renta asociada. Esto significa que un nuevo empleado puede ser agregado sin atarlo a una renta. Por el contrario, cada renta *debe ser* procesada por exactamente un empleado, por lo tanto la relación renta-a-empleado es obligatoria.

La relación cliente-a-video es opcional debido a que un cliente puede existir sin tener rentado un video. La relación video-a-cliente es opcional dado que es posible que un video en particular jamás sea rentado por ningún cliente.

La relación cliente-a-renta es opcional porque un cliente puede existir sin una renta asociada. Esto significa que un nuevo cliente puede ser agregado sin estar atado a una renta. Observando la relación en reversa, vemos que una renta debe estar atada a exactamente un cliente. De esta manera, la relación renta-a-cliente es obligatoria.

En un momento dado, una tienda puede o no tener asociados videos, por lo que la relación tienda-a-video es opcional. Esto significa que una nueva tienda puede ser agregada sin tener videos. La relación video-a-tienda es obligatoria porque cada video debe estar exactamente en una tienda.

Ahora veamos la relación video-a-cambio de precio. Es posible que el precio de un video jamás cambie, por lo que esta relación es opcional. Por el contrario, la relación cambio de precio-video es obligatoria, debido a que cada cambio de precio debe afectar al menos a un video.

Finalmente, una renta no puede existir sin tener asociado al menos un video, por lo que la relación renta-a-video es obligatoria. Por el contrario, un video puede existir sin tener rentas asociadas, por lo que la relación video-a-renta es opcional.

## **EJERCICIO DE LABORATORIO # 3**

## Attributes

**Attributes** are characteristics or modifiers that provide detailed descriptive information about entities.

PO Number:  
one attribute

109540 ← po number

Name:  
two attributes

Chris Parker  
first name      last name

G/L Account Number:  
three attributes

10-1613-6100  
cost center number      department number      account number

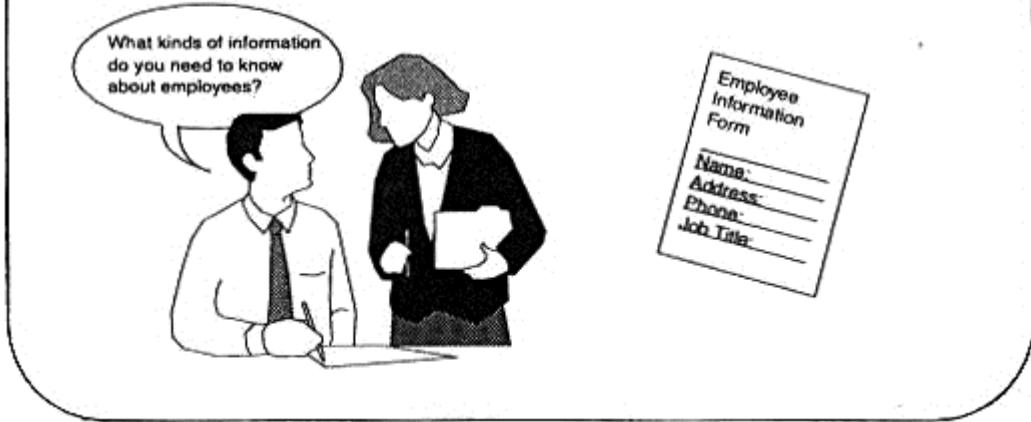
Un *atributo* es una pieza de información indivisible que describe una entidad. Usted no puede descomponer un atributo sin perder su significado original. Por ejemplo, un número de orden de compra puede ser representado como 109540. Este número no puede ser descompuesto sin perder su sentido original, por lo que debe ser representado como un atributo sencillo.

El nombre completo de una persona, por el contrario, puede ser descompuesto sin perder su significado original. De esta manera, los nombres pueden ser representados como atributos múltiples com el nombre y el apellido.

Una plantilla genérica para un número de cuenta podría estar representado como 10-1613-6100. Si la plantilla genérica actualmente representa un número de centro de costos, un número de departamento y el número de cuenta, la plantilla genérica puede ser descompuesta en tres atributos separados.

## Identifying Attributes

- Probe for attributes during the interview process.
- Documents, reports, and forms are also excellent sources.



Los atributos son generalmente descubiertos durante el proceso de entrevista. Si usted ya tiene identificadas las entidades, puede simplemente preguntar al usuario que información necesitan ellos acerca de la entidad. Usted debe también preguntar cómo usarán ellos esa información. ¿Es necesario guardar información histórica? ¿Qué información será accesada más frecuentemente?

Documentos, reportes y formas son también excelentes fuentes para identificar los atributos.

## Attribute Types

Identifier (key)      Used to specify a unique characteristic of a particular entity

Descriptor      Used to specify a non-unique characteristic of a particular entity

Hay dos clasificaciones generales de atributos: *identificadores* y *descriptores*. Un identificador es también conocido como *llave*.

Un atributo identificador es un atributo que especifica una característica única de una entidad particular. Un atributo identificador para la entidad **empleado** podría ser el número de empleado. Se puede asumir que un número de empleado siempre identificará de manera única a un empleado.

Un descriptor es un atributo que especifica una característica no única de una instancia en particular. Un atributo descriptor de la entidad **empleado** podría ser el apellido. Un apellido no siempre identifica de manera única a un empleado.

Nota:

La clasificación preliminar de los atributos es importante para la futura construcción del diseño, que será discutida en un módulo posterior.

## Sistema de Renta de Videos

*Cuéntame acerca de tu negocio.*

"Actualmente, poseo una pequeña **tienda** de renta de videos en el vecindario, pero planeo expandir el número de tiendas a tres antes de terminar el año. Yo controlo las **rentas** en mi tienda manualmente, pero sé que esto será difícil de manejar cuando tenga las otras dos tiendas. Pienso que un sistema de computadora podría ayudarme a controlar la información de las rentas más efectivamente."

*¿Qué clase de información necesitar registrar acerca de tus rentas?*

"Necesito registrar información acerca del **cliente** e información acerca de los **videos** que ellos rentan. Se permiten un máximo de cinco videos por renta. También necesito saber cuál **empleado** procesó la renta."

*¿Existe cualquier otra información que necesites registrar?*

"Sí, necesito registrar el monto en pesos de cada renta. Los precios están basados en el tipo de video. Actualmente, una película nueva cuesta \$3.50 la noche, videos restringidos \$4.00 por noche y todos los demás videos a \$2.50, excepto los infantiles. Elos están a \$1.00. Después de cuatro semanas, las películas nuevas están a \$2.50."

*¿Qué pasa cuando cambias los precios?*

"Es un buen punto. Nosotros elevamos los precios periódicamente. Para propósitos contables, mantengo un registro histórico de todos los **cambios de precio** y sus fechas de aplicación."

*¿Será importante para ti registrar información por tienda?*

"Sí, me gustaría poder hacerlo. De esa manera, conservaré registro de cuales videos están en cada tienda. Oh, y podría también saber cuál tienda está rentando más videos. ¿Crees que me puedas ayudar?".

*¿Qué información necesitas conservar acerca de tus clientes?*

"Bueno, nosotros asignamos un número de cliente a cada cliente. También necesito conocer el nombre del cliente, dirección, número de teléfono e información de su tarjeta de crédito. Pienso que la información de tarjeta de crédito es un depósito de seguridad. Si un cliente pierde o daña un video y no me lo retribuye, necesito recuperar mi pérdida. Oh sí, la tarjeta de crédito debe ser obtenida del cliente y necesito saber la fecha de vencimiento".

*¿Y que hay de los videos?*

"Cada título de video tiene un numero de almacén que es asignado por el distribuidor. Puedo tener varias copias de un título en particular, pero yo siempre obtengo todas las copias del mismo distribuidor. Necesito registro de cada título que tengo. Me gustaría también la fecha en que cada copia fue recibida. Puedo

obtener cuatro copias inicialmente, y ordenar copias extra si una película es realmente popular. Por supuesto, también necesito conocer el precio de renta por cada video."

*¿Y las rentas?*

Una renta es por una noche. Si quieres un video por dos noches, es procesado como dos rentas de una noche; no hay descuentos. Puedo decir que si un video es regresado posteriormente a la noche de la renta. Necesito saber el precio individual de renta para cada video y el total de la renta. Oh sí, y por supuesto necesito saber cual empleado procesó la renta."

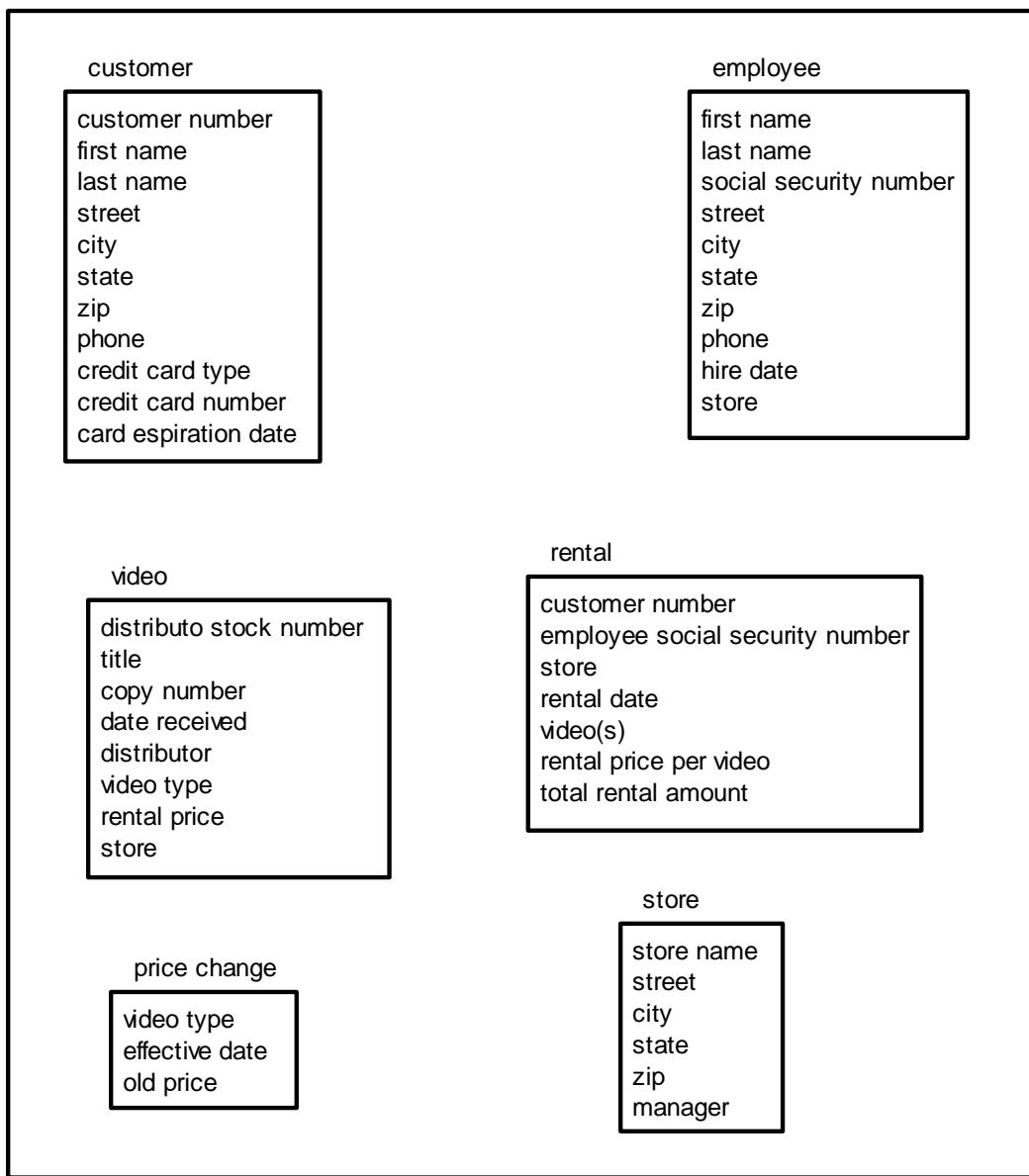
*¿Qué otra información necesitas registrar de tus empleados?*

"Nombre, dirección, número de teléfono y estaría bien la fecha de contratación, así como en cual tienda está trabajando. Usamos los números de seguro social para identificarlos, dado que con eso tendríamos esa información para el pago de las nóminas. Me gustaría también saber quién es el gerente de la tienda."

*¡Qué mas necesitas conocer acerca de las tiendas?*

"Por supuesto, necesito almacenar el nombre y domicilio. Y creo que ya mencioné que necesito saber cual empleado trabaja en cada tienda. Eso deberá ser suficiente por ahora."

## Attributes: Video Rental System



## Attributes: Derived Data

- Usually values that are calculated from other attributes
- May add data redundancy and complexity to the model
- May have significant business meaning to a user
- Better to calculate in an application rather than storing in a database



INFORMIX

Entity-Relationship Concepts and Terminology 1861 01-96 43

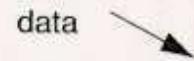
*Datos derivados* son usualmente valores que son calculados desde otros atributos. En general, no debes almacenar datos derivados. Los datos derivados pueden tener información significativa del negocio para el usuario. Generalmente se recomienda que los datos sean obtenidos a través de algoritmos calculados en la aplicación en lugar de que sean almacenados en la base de datos.

Examine los atributos de la entidad **video** en el sistema de renta de videos. ¿Existe un atributo que sean datos derivados y puedan ser calculados usando la información de otros atributos?

## Derived Data: Video Rental System

rental  
customer number  
employee social security number  
store  
rental date  
video(s)  
rental price per video  
total rental amount

Derived  
data



INFORMIX

Entity-Relationship Concepts and Terminology 1861 01-96 44

Almacenar datos derivados puede agregar complejidad al modelo de datos, especialmente cuando se consideran las reglas de inserción, actualización y eliminación.

En el ejemplo de arriba, el monto total de renta son datos derivados. En cualquier momento que un cambio ocurra a la entidad **renta** que involucre el video o el precio de renta, el monto total de renta puede ser afectado.

## Domain

- Specifies the constraints on the valid values that attributes may assume
- Identifies the set of valid values for an attribute

INFORMIX

Entity-Relationship Concepts and Terminology 1861 01-96 45

Un *dominio* especifica las constraints sobre los valores permitidos que los atributos pueden asumir. Usted tendrá que examinar cada atributo y asignar cualquier característica de dominio. Las características pueden ser un juego de valores validos, o identificación de agrupamiento de características similares.

Usualmente un juego de valores válidos puede ser representado por el uso de códigos. Es una práctica común el uso de códigos para representar datos más significativos. Estos dominios, en muchos casos, resultan en la creación de una nueva entidad.

El uso de códigos puede ayudar a reducir los requerimientos de almacenamiento y ahorrar espacio en disco. Los códigos también reducen teclazos al capturar datos. Los códigos son benéficos cuando se conforman de acuerdo a los esquemas de codificación usados en aplicaciones existentes.

Un ejemplo de un dominio son los códigos de estado de los Estados Unidos. El dominio podría ser representado por una entidad **estado**, con atributos **código del estado** y **nombre del estado**.

## Domain: Video Rental System

video
stock number
title
copy number
date received
distributor
video type
rental price
store

domain

- | Valid values |
- | for video type |
- | • new release |
- | • children's |
- | • restricted |
- | • other |

INFORMIX

Entity-Relationship Concepts and Terminology 1861 01-96 46

El juego de valores permitido para el tipo de video se muestra arriba. El tipo de video podría ser representado por un código. Por ejemplo, el código para una cinta nueva podría ser NR. Posteriormente, una entidad **video type** puede ser agregada al modelo de datos.

## Matching Exercise

- |          |              |   |
|----------|--------------|---|
| <u>C</u> | Existence    | A. Modifier that provides information about a principal data object.                  |
| <u>O</u> | Connectivity | B. An attribute type used to uniquely identify an entity occurrence.                  |
| <u>A</u> | Attribute    | C. Mandatory or optional requirement of the existence of an entity in a relationship. |
| <u>B</u> | Identifier   | D. (1:1) (1:N) (M:N)  |
| <u>F</u> | Entity       | E. Represents real world association between entities.                                |
| <u>R</u> | Relationship | F. A principal data object of significant interest to the user.                       |

INFORMIX

Entity-Relationship Concepts and Terminology 1861 01-96 47

Completa los ejercicios de correspondencia.





## Solutions to Matching Exercise

- |          |              |   |
|----------|--------------|---|
| <b>C</b> | Existence    | A. Modifier that provides information about a principal data object.                  |
| <b>D</b> | Connectivity | B. An attribute type used to uniquely identify an entity occurrence.                  |
| <b>A</b> | Attribute    | C. Mandatory or optional requirement of the existence of an entity in a relationship. |
| <b>B</b> | Identifier   | D. (1:1) (1:N) (M:N)  |
| <b>E</b> | Entity       | E. Represents real world association between entities.                                |
| <b>F</b> | Relationship | F. A principal data object of significant interest to the user.                       |

## **EJERCICIO DE LABORATORIO # 4**

# Capítulo

# 3

## Diagramas Entidad-Relación

### Objetivos

Al final de este modulo, usted será capaz de:

- Describir el rol de los diagramas entidad-relación en el diseño de una base de datos relacional
- Describir los beneficios del uso de los diagramas entidad-relación
- Desarrollar diagramas entidad-relación al estilo de Bachman para modelar requerimientos de información
- Definir las reglas para las llaves primarias y foráneas(secundarias)
- Agregar llaves primarias y foráneas a su diagrama entidad-relación

## Database Design Approach

- Gain an understanding of the business.
- Identify the principal data objects (entities, attributes, and relationships).
- **Diagram the data objects using the entity-relationship approach.**
- **Resolve the logical data model.**
- Determine attribute specifications and data types.
- Verify the logical data model through normalization.
- Use SQL to convert the logical data model to a physical database schema.

Este módulo describe el tercer y cuarto pasos en el diseño de la base de datos.

## The Role of Entity-Relationship Diagrams

- Model information needs of an organization
- Identify entities and their relationships
- Serve as a starting point for data definition
- Are an excellent source of documentation
- Are used to create the physical design of a database

Los diagramas entidad-relación (ERD) son una valiosa herramienta para modelar requerimientos de información de una empresa de negocios. Juegan un rol importante en la metodología de diseño de bases de datos relacionales. Los ERD son usados para modelar las relaciones entre entidades identificadas durante el proceso de entrevista. Son una excelente fuente de documentación para los usuarios, desarrolladores de aplicaciones y administradores de bases de datos. Finalmente, los ERD son usados para crear el esquema físico de la base de datos.

## Bachman-Style ERDs

- Map closely to relational database model
- Depict relationships simple (binary) in nature
- Document information requirements in an easily understood format

C.W. Bachman formalizó los diagramas de esquema usados en los modelos de datos entidad-relación al final de los 60's. La aproximación entidad-relación ha sufrido muchos cambios para soportar extensiones para modelos más complejos.

Los ERD al estilo Bachman describen relaciones que son simples (binarias) por naturaleza. Proveen una sintaxis diagramática que es fácil de entender. Es la sintaxis diagramática usada por varias herramientas CASE.

Existen muchos estilos diferentes de ERD's. La aproximación usada en este curso son los ERD al estilo de Bachman. Ejemplos de otros estilos se presentan más tarde en este módulo. Use el estilo que le acomode mejor.

## Basic Objects: Bachman Style

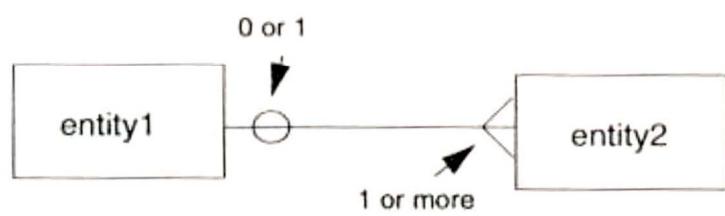
### Objects

Entity (Relation)

### Representation



Relationship



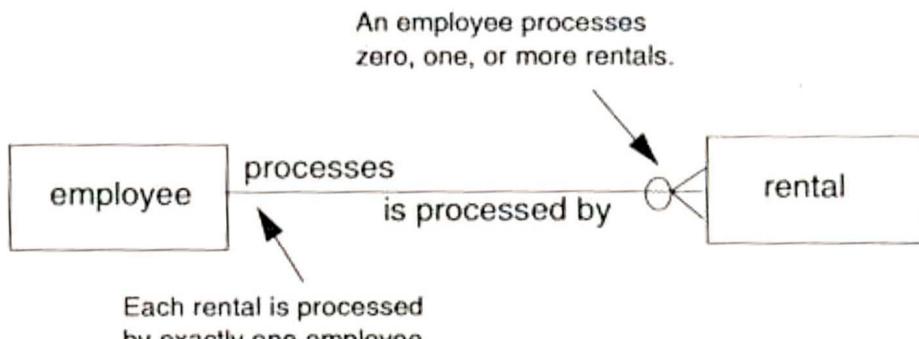
Una entidad es representada por un rectángulo. El nombre de la entidad es colocado dentro del cuadro en singular, con letras minúsculas.

Una relación es descrita por una líneas sencilla dibujada entre las dos entidades.

Un círculo al final de una liga de relación indica que la existencia de una entidad en esa punta es opcional en la relación. En la mayoría de los casos, las reglas de negocio de la empresa podrían determinar la existencia de relaciones. Si la existencia de una entidad es una relación es opcional, es importante marcarlo con la representación del círculo.

La pata de cuervo implica que una o más instancias de una entidad están asociadas con otra entidad. La ausencia de la pata de cuervo implica que solamente una instancia de una entidad está asociada con otra entidad. Un círculo denotando opcionalidad es incluido en cualquier caso que una entidad sea opcional en una relación.

## Relationship Representation Bachman Style: Video Rental System



La liga entre dos objetos forma una sentencia simple. La primera entidad es el *sujeto* y la segunda entidad es el *objeto*. Se pueden incluir descriptores de la relación para mejorar la claridad del diagrama a los usuarios finales. Los aspectos más importantes de una relación son descritos por las líneas entre las entidades.

La *conectividad* se refiere al número de una entidad que está asociada con la segunda entidad.

Nota:

Recuerde que la conectividad puede ser uno-a-uno, uno-a-muchos ó muchos-a-muchos.

La conectividad más común es uno-a-muchos. Una relación uno-a-muchos representa la mayoría de las veces una relación *maestro-detalle* (padre-hijo).

Las relaciones uno-a-uno son muy raras. Las relaciones muchos-a-muchos usualmente esconden entidades asociativas adicionales.

## Alternative Syntax

- There are many formats and style variations that can be used to develop entity-relationship diagrams.
- Develop and use the one with which you are most comfortable.

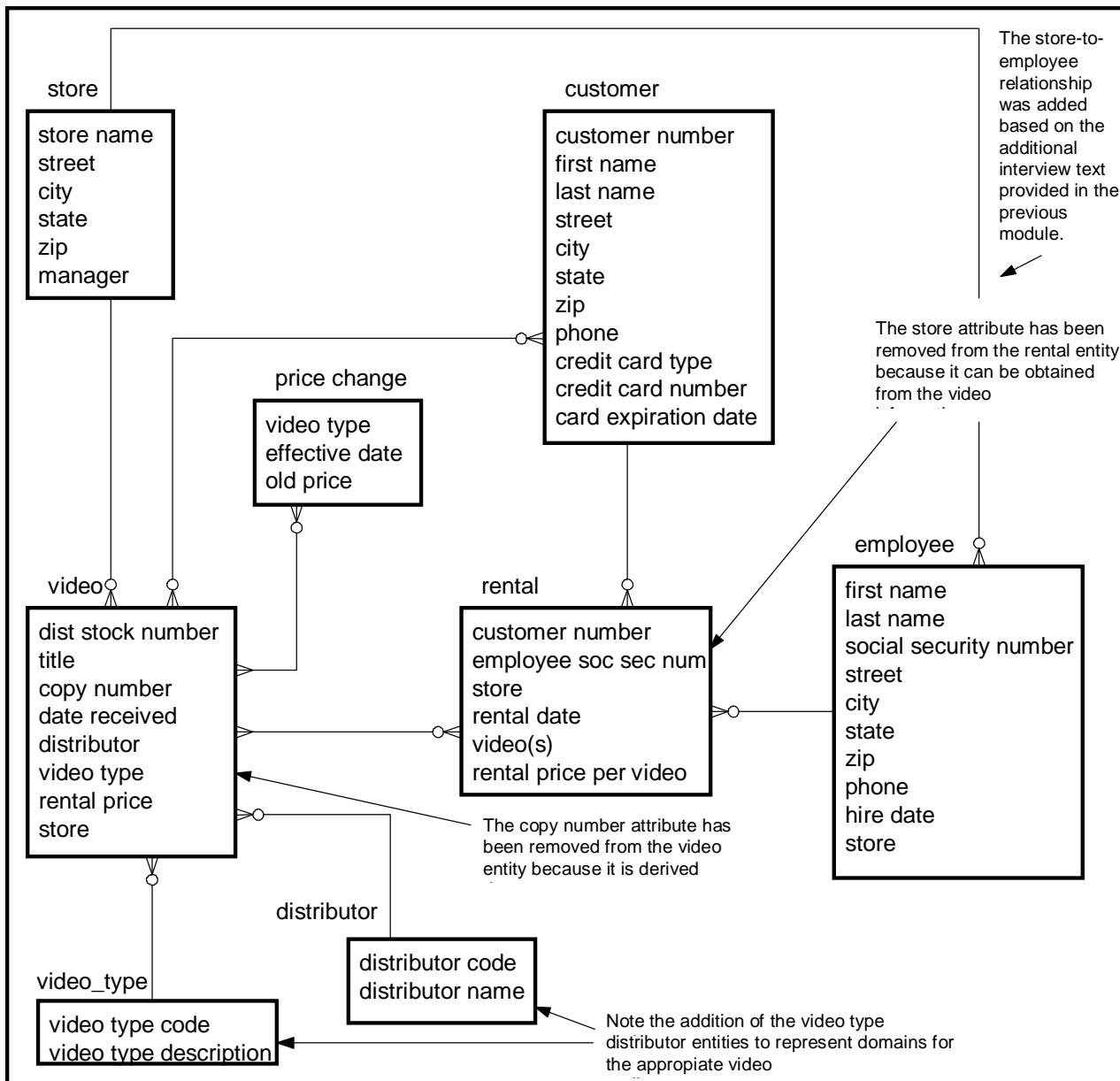
Existen varios estilos diferentes de ERD que pueden ser usados en el modelado de datos. Seleccione y estandarice el estilo que encuentre mas confortable. Usted puede incluso elegir agregar algo de sus propia iniciativa a la sintaxis para ayudarse en sus esfuerzos de desarrollo.

## Alternative Styles

Connectivity	Crow's Foot	Reiner	Chen
1:1	———	———	1  1
1:N	——— ↗	——— ↗	1  N
M:N	↗ ——— ↗	↗ ——— ↗	M  N
Mandatory	———	———	———
Optional	— - - ↗	○ ——— ○	Not shown in Chen

Ejemplos de tres diferentes estilos se muestran en la lámina.

## EDR: Video Rental System



## **EJERCICIO DE LABORATORIO # 5**

## Redundant Relationships

*Redundant* relationships are two or more relationships that are used to represent the same concept.

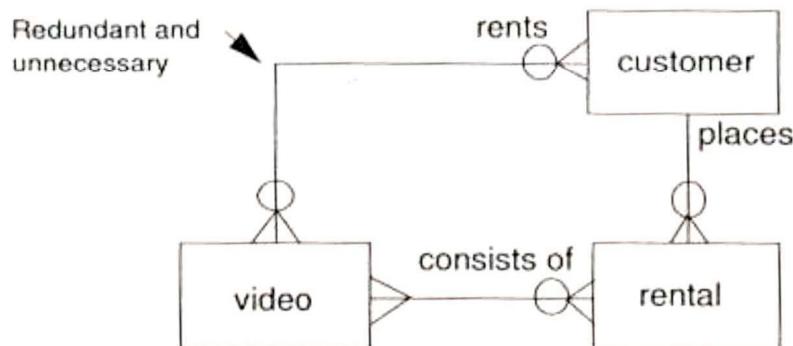
Las relaciones *redundantes* son dos ó más relaciones que son usadas para representar los mismos conceptos. Usted debe analizar las relaciones redundantes muy cuidadosamente y eliminarlas de su modelo de datos.

Dos ó más relaciones entre entidades son permitidas siempre y cuando las relaciones tengan diferente significado.

Nota:

En raras ocasiones, puede tener sentido introducir una relación redundante por razones de eficiencia.

## Redundant Relationships (cont.)



The **customer-to-video** relationship is redundant with the combination of the **customer-to-rental** and **rental-to-video** relationships.

Existen varias razones para eliminar relaciones redundantes:

- Agregan complejidad
- Pueden inducir la colocación incorrecta de atributos
- Pueden ser malinterpretadas, causando errores en operaciones de manipulación de datos.

En el ejemplo de arriba, hay una relación *places* entre **cliente** y **renta**, y una relación *consists of* entre **factura** y **payment**. Desde estas relaciones, es posible encontrar todos los videos que un cliente dado ha rentado. También es posible encontrar los que han rentado cualquier video dado siguiendo las relaciones en reversa, desde **video** a **renta** a **cliente**.

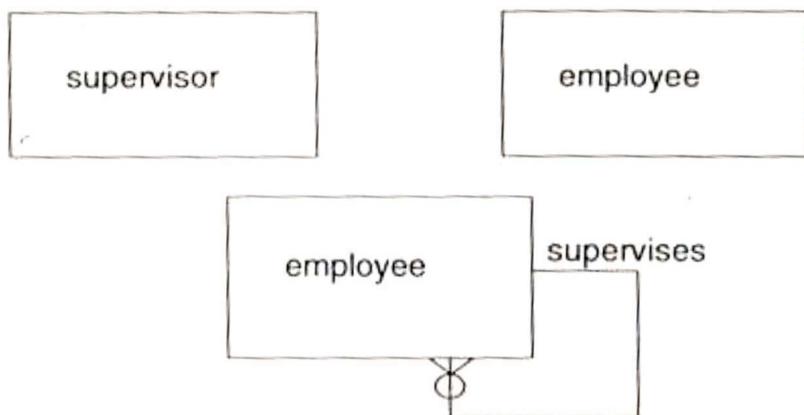
La relación entre **cliente** y **video** es redundante e innecesaria. No provee ninguna información adicional que no esté ya provista por la combinación de **cliente** a **renta** y **renta** a **video**.

## Recursive Relationships

A *recursive relationship* is an association between occurrences of the same entity type. It is also known as a self-referencing or looped association.

Una entidad puede ser relacionada a sí misma. Una relación recursiva es una asociación entre ocurrencias del mismo tipo de entidad.

## Recursive Relationships (cont.)

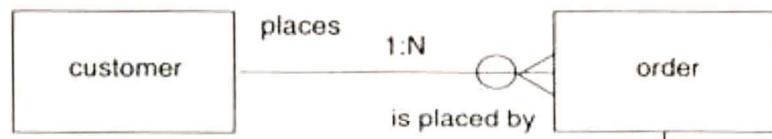


- Do not occur frequently
- Useful for defining organizational and bill-of-materials structures

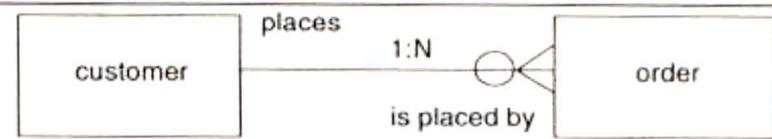
Las relaciones recursivas no ocurren frecuentemente, pero son usadas para definir organizaciones y estructuras bill-of-materials. Por ejemplo:

- Un empleado administra a muchos otros empleados
- Una parte está compuesta de muchas otras partes.

## Resolving One-to-One Relationships



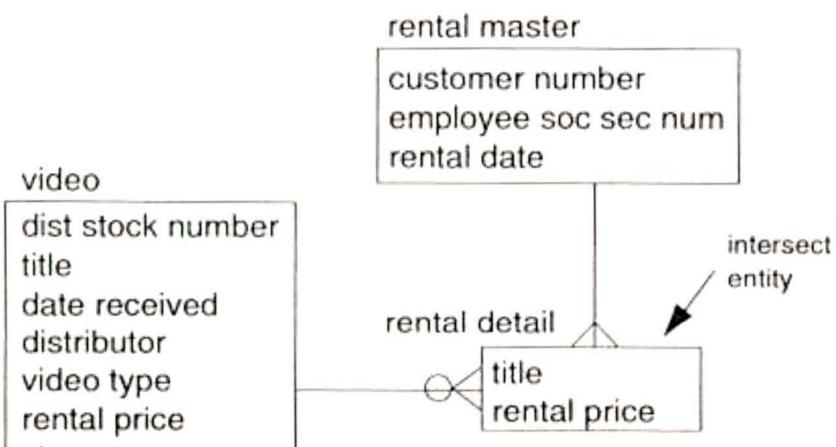
One-to-one relationships are resolved by merging the two entities together.



**shipping info** is rolled into the entity **order**.

Las relaciones uno-a-uno son muy raras. La mayoría de las relaciones son uno-a-muchos. Cuando una relación uno-a-uno es detectada, una entidad es usualmente solo un juego de atributos que pueden ser incorporados con la otra entidad. En el ejemplo de arriba, la entidad order podría ser expandida para incorporarle los atributos de la entidad shipping info.

## Resolving Many-to-Many Relationships



Uno de los poderes de los ERD al estilo Bachman es que permiten inmediatamente detectar y resolver cualquier relación muchos-a-muchos.

La clave para resolver las relaciones muchos-a-muchos es separar las dos entidades y crear dos relaciones uno-a-muchos, con una tercera entidad *intersección*. Haciendo esto se reduce la complejidad y confusión en los procesos de diseño y desarrollo de aplicaciones.

El primer ERD para el sistema de renta de videos tiene tres relaciones muchos-a-muchos: **video a customer**, **video a rental** y **video a price change**. La relación **customer a video** fue eliminada porque era redundante.

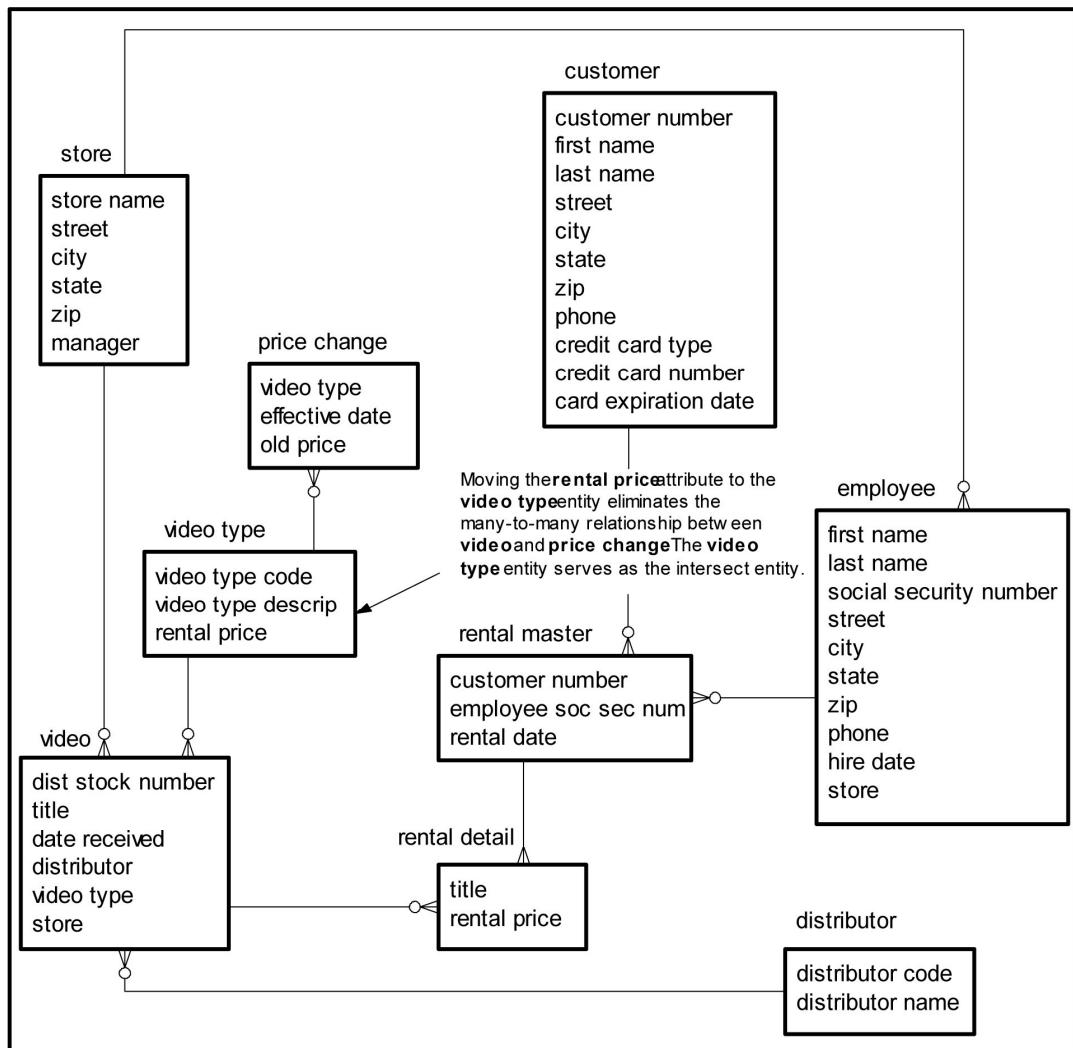
Para resolver las relaciones muchos-a-muchos entre **video** y **rental**, sepáremos las dos entidades y creamos dos relaciones uno-a-muchos con una tercera entidad *intersección*, **rental detail**.

La entidad **rental master** contiene información estática (que no cambia), acerca de la renta. El cliente, el empleado que maneja la renta y la fecha de la renta no cambian si la renta es por uno o cinco videos.

La entidad **rental detail** contiene información detallada acerca de cada video para una **rental master** particular. Si un cliente renta un video o cinco videos, ¿cuál es la información que cambia?. La entidad **rental detail** contiene el título y el precio para cada video rentado.

¿Cómo resolvería usted la relación muchos-a-muchos entre **video** y **price change**?

## EDR: Video Rental System





## **EJERCICIO DE LABORATORIO # 6**

## Primary Key

- A primary key is an attribute or combination of attributes that uniquely identifies an entity instance.
- A primary key must exist for all entity instances (no nulls allowed).
- Every entity must have exactly one primary key.

Todas las entidades deben tener exactamente una llave primaria. La llave primaria debe ser un atributo o combinación de atributos que identifiquen de manera única una instancia de una entidad. Los atributos *identificador* son los mejores candidatos para llaves primarias dado que los atributos identificador cumplen con la constraint de ser dato único (unique).

Una llave primaria debe existir para todas las instancias de una entidad. Una constraint unique será definida para cada llave primaria.

Ud. puede reforzar las reglas de llave primaria al momento de definir la base de datos cuando la tabla es creada. Estos conceptos son cubiertos en los módulos *Creating a Database* y *Referential and Entity Integrity*.

**Nota:**

Una llave primaria debe ser de tipo numérico ó un string de caracteres corto. No es recomendable usar un string de caracteres largo para una llave primaria.

## Special Primary Keys

- Composite key
- Surrogate key

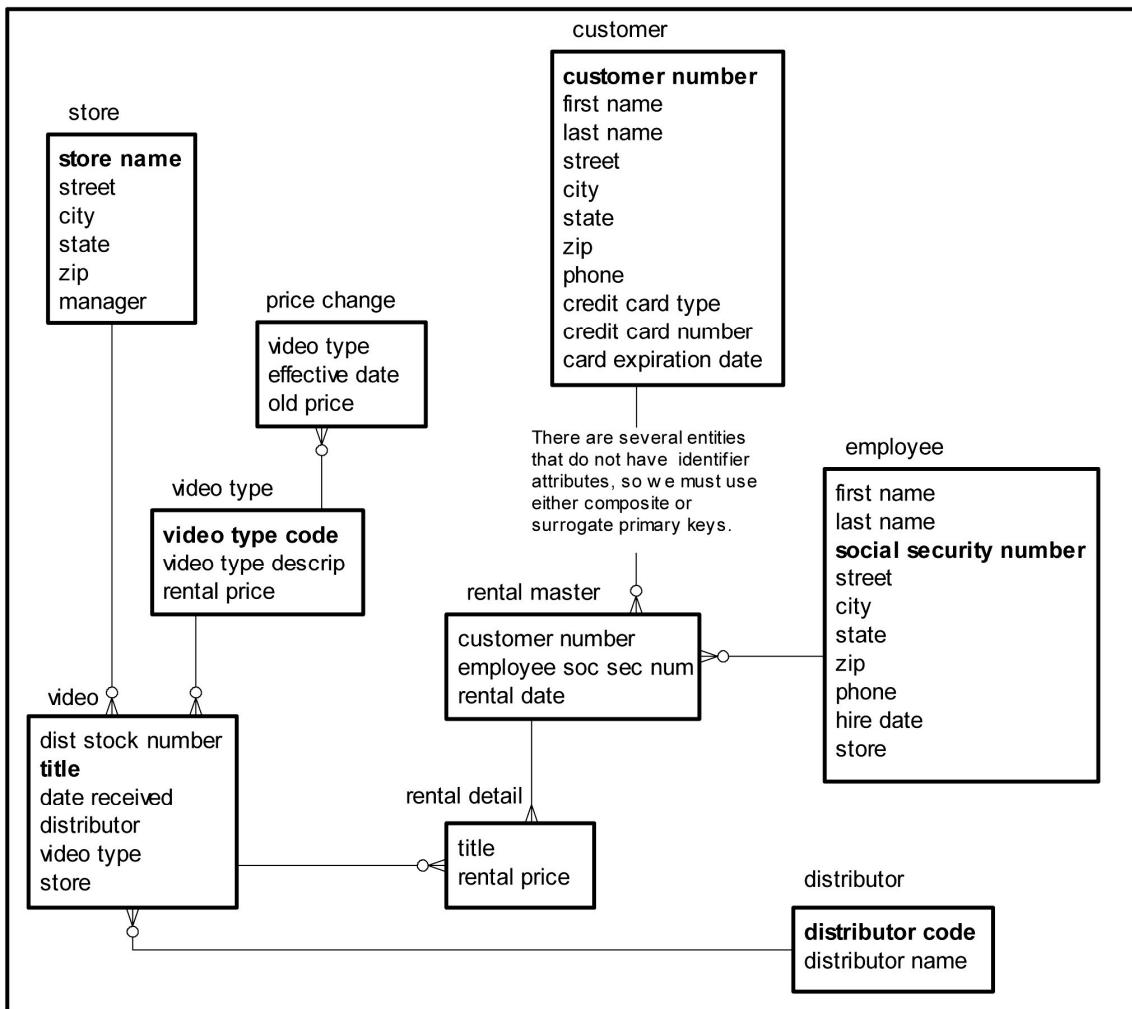
Si Ud. no puede identificar un atributo que cumpla la constraints de unique para una llave primaria, podría considerar el uso de una llave primaria especial.

Una *llave compuesta* es una llave que usa dos ó más atributos. El propósito de una llave compuesta es asegurar que sea unica.

Una *llave subrogada* (surrogate key) puede ser agregada a la entidad para garantizar que sea unica. Por ejemplo, si Ud. agrega un nuevo atributo con un tipo de dato SERIAL, un valor único puede ser generado automáticamente para cada instancia de la entidad. Los tipos de datos son cubiertos en el módulo *Data Types*.

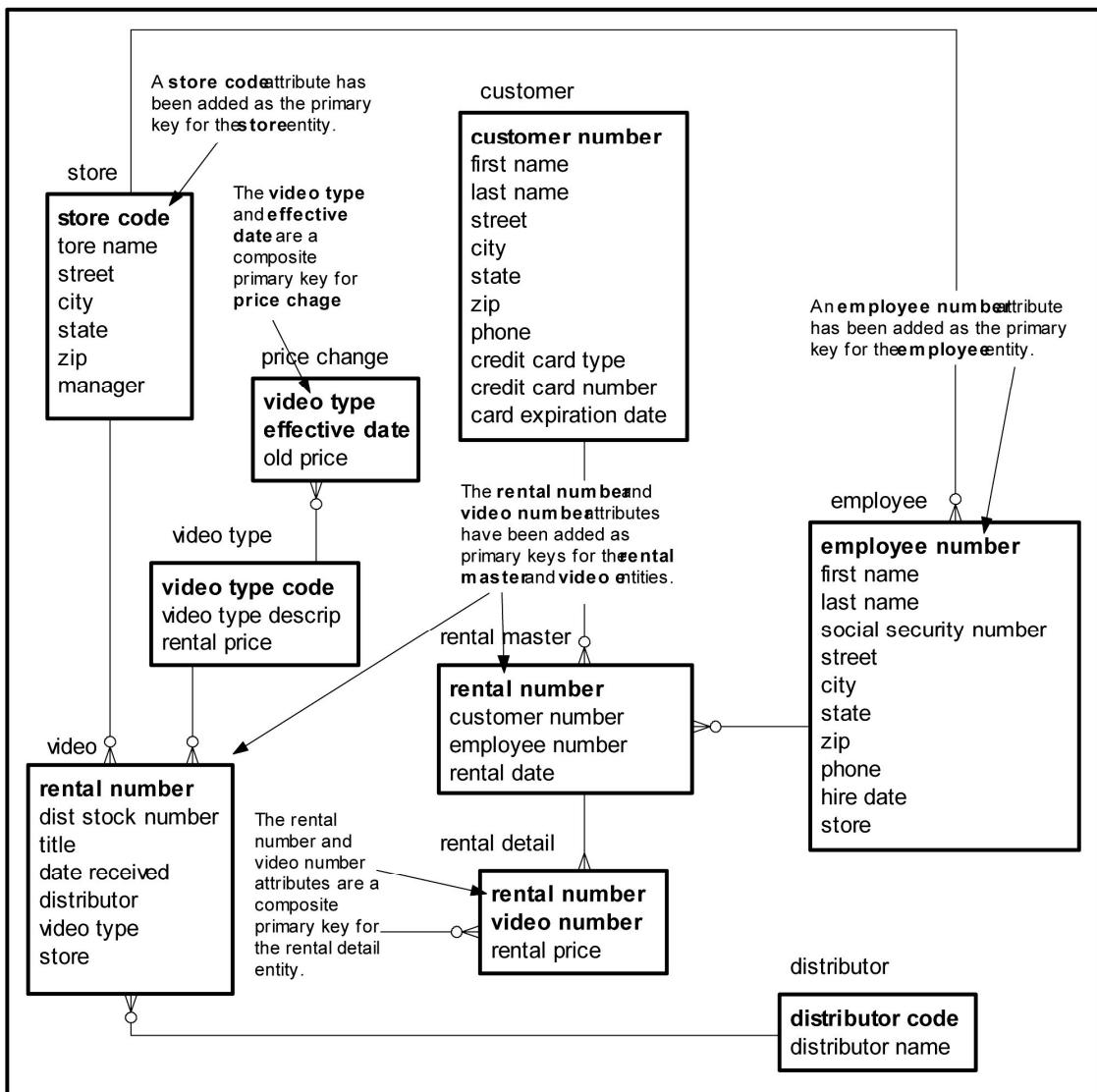
## Primary Key Candidates: Video Rental System

Primary key candidates (identifier attributes) are shown in **bold**.



## Primary Key: Video Rental System

Key are shown in **bold**.







## Foreign Key

- Attribute that must reference an existing primary key in another entity
- Attribute or combination of attributes used to establish the relationship between entities
- May be null
- May contain duplicates
- May be updated

Una llave *foránea* es usada para establecer la relación entre entidades. Una llave foránea debe referenciar una llave primaria existente en una entidad asociada. Puede haber varias llaves foráneas en una entidad si la entidad es relacionada a múltiples entidades.

Las llaves foráneas pueden permitir valores nulos, pueden contener duplicados y pueden ser cambiadas.

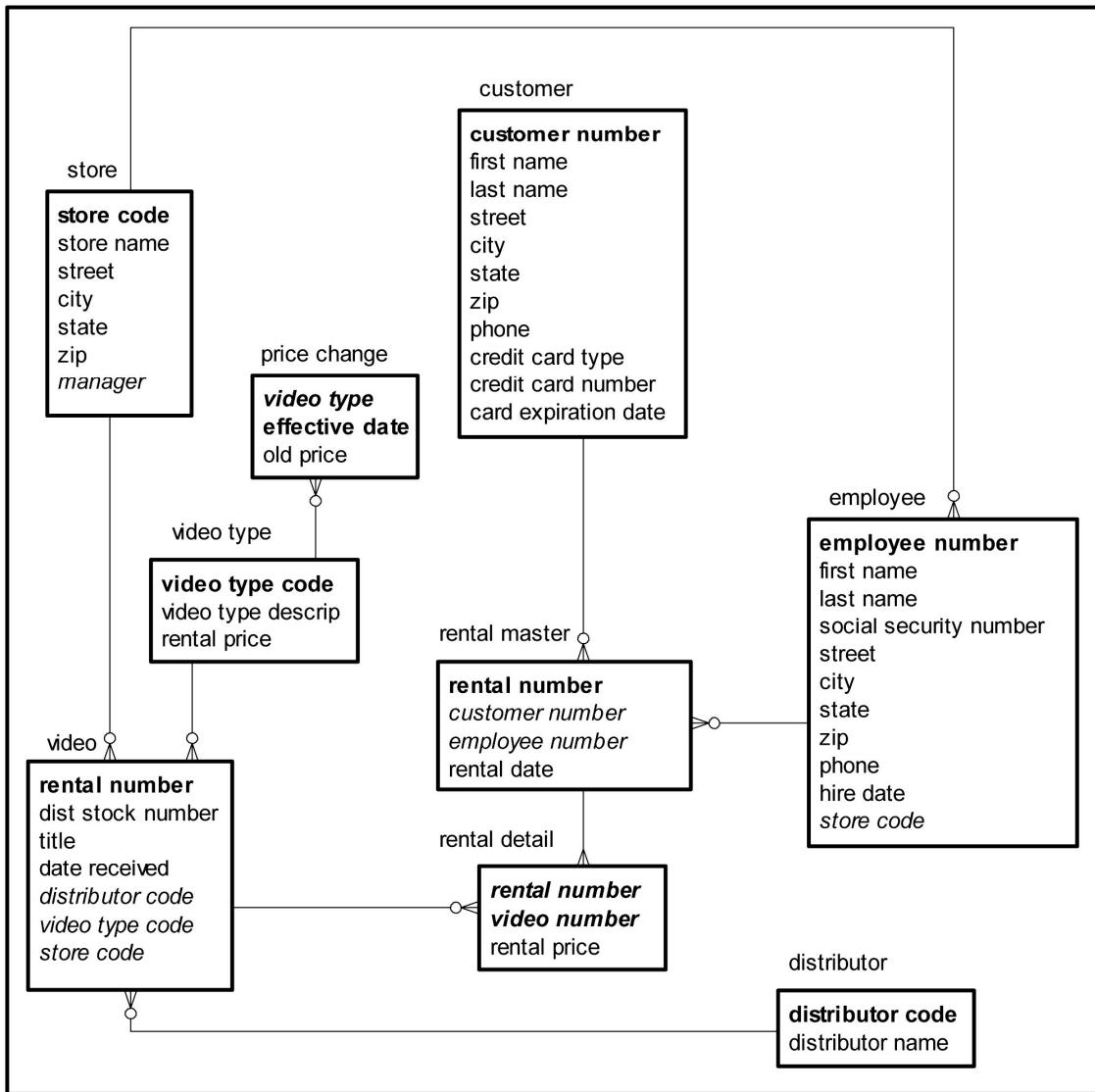
Ud. puede asegurar la asociación entre la llave primaria y foránea al definir la base de datos usando el estatuto CREATE TABLE. Este concepto es cubierto posteriormente en el módulo *Data Integrity*.

**Consejo:**

En relaciones uno-a-muchos, la entidad *muchos* contendrá la llave foránea.

## Foreign Key: Video Rental System

Key are shown in **bold**. Foreign keys are in *italics*.





## Entity Instance: Video Rental System

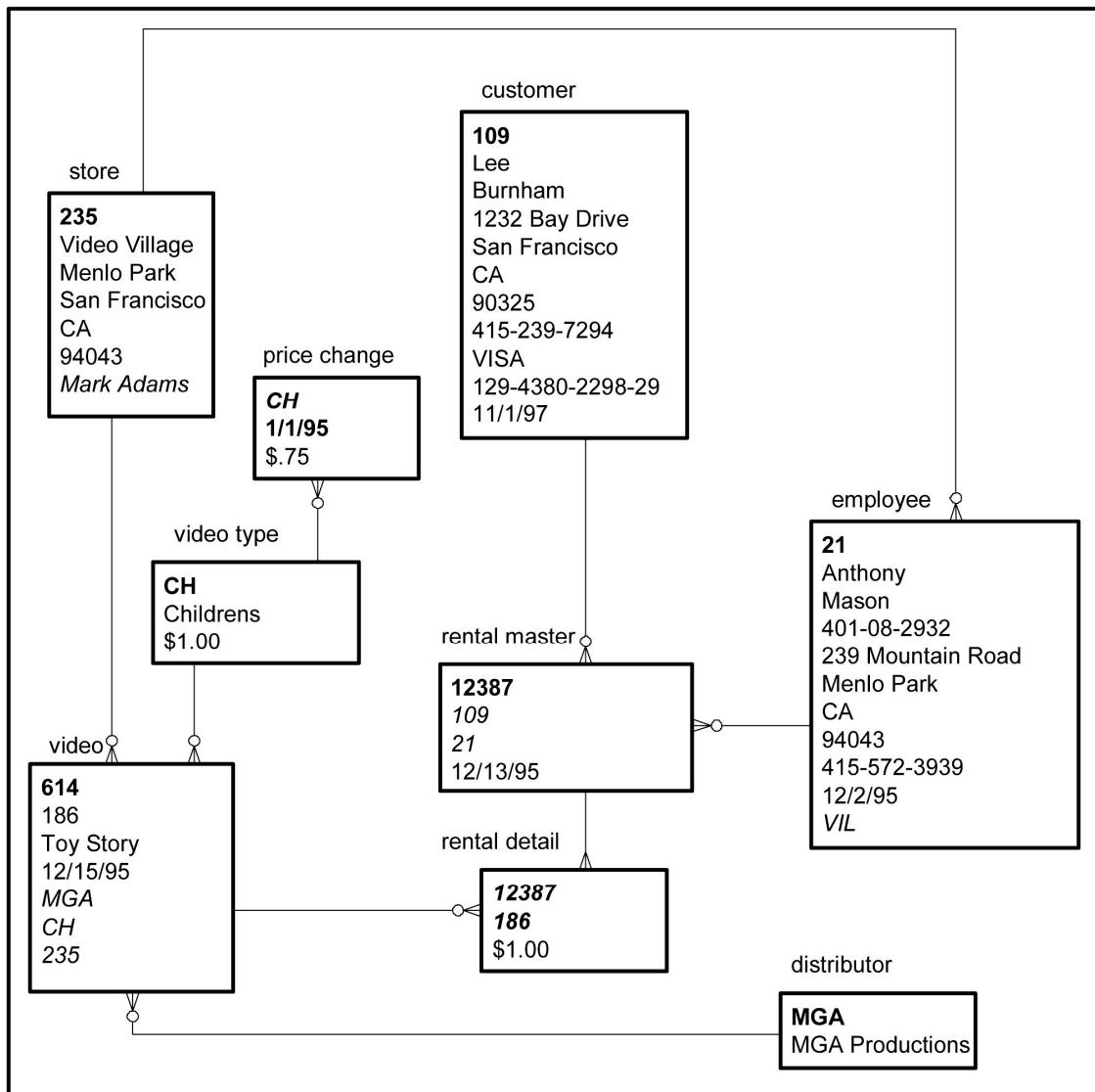
Diagramming entity instances representing the attributes is useful for verifying that the model represents all data requirements.

Un diagrama representando una *instancia de una entidad* en el sistema de renta de videos es mostrada en la página siguiente. Los diagramas de instancias pueden ayudar a verificar que el modelo representa los requerimientos de datos de forma precisa. Ellos pueden también verificar que las asociaciones descritas están presentes también entre entidades.

**Consejo:**

Puede ser de mucha ayuda diagramar varias instancias diferentes para asegurar que el modelo es correcto.

## Entity Instance: Video Rental System





## **EJERCICIO DE LABORATORIO # 7**

# Capítulo 4

## Especificaciones de atributos

### Objetivos

Al final de este módulo, Ud. será capaz de:

Aplicar los lineamientos para nombrar atributos

Identificar los requerimientos únicos y no únicos para los atributos.

## Database Design Approach

- Gain an understanding of the business.
- Identify the principal data objects (entities, attributes, and relationships).
- Diagram the data objects using the entity-relationship approach.
- Resolve the logical data model.
- **Determine attribute specifications and data types.**
- Verify the logical data model through normalization.
- Use SQL to convert the logical data model to a physical database schema.

Este módulo cubre el paso de especificaciones de atributos en el diseño de bases de datos escrito arriba.

## **Attribute Names**

- Establish conventions
- Use unique, meaningful names
- Avoid synonyms and homonyms

Los lineamientos a considerar cuando se nombran atributos, se enlistan en la lámina de arriba. Estos lineamientos serán discutidos en las siguientes páginas.

## Naming Conventions

- Establish conventions.
  - ~~paid\_amt vs. amt\_paid~~
  - ~~order\_amt vs. ord\_amount~~
  - ~~paid\_price and stock\_price vs. paid\_price and stock\_pr~~
  - ~~order\_date and paid\_date vs. order\_date and paid\_dt~~
- Use names that are unique and meaningful to the user.
  - ~~ship\_date vs. sdate~~
- A foreign key that references a primary key in a different table should have the same name as the primary key.

El establecer convenciones en su ambiente de desarrollo es importante para el éxito de los proyectos. Deben establecerse convenciones para crear y mantener directorios, programas, módulos y variables. Las convenciones también deben establecerse para nombrar atributos, entidades y nombres de bases de datos. Los nombres deben ser de significado completo, no demasiado largos y claramente entendibles.

La consistencia es la llave del éxito de los nombres.

Con servidores de bases de datos, los nombres deben comenzar con una letra o guión bajo (\_) y pueden tener hasta 18 caracteres. Los nombres pueden incluir números, letras y guiones bajos (\_). No pueden usarse palabras reservadas ANSI.

En el ejemplo de arriba, las columnas **paid\_amt** y **order\_amt** siguen las convenciones usando abreviaciones consistentes y nombres en la forma **calificador+nombre**.

**Nota:**

El número máximo de caracteres para un nombre de base de datos es diez. Consulte la guía para mayor información respecto a los requerimientos de nombrado.

## Avoid Synonyms and Homonyms

- **Synonyms:** Different names with the same meaning  
**cust\_id** vs. **customer\_code**
- **Homonyms:** The same name with different meanings  
**order\_date** date order shipped or date order placed

El uso de sinónimos y homónimos debe ser evitado cuando se nombran atributos. Los homónimos y sinónimos crean confusión a usuarios y programadores.

Los sinónimos son nombres diferentes que se refieren a la misma cosa. En el ejemplo de arriba, **cust\_id** y **customer\_code** ambos se refieren al número único de identificación asignado a cada cliente.

Los homónimos son lo contrario a los sinónimos. El mismo nombre puede ser asignado para referirse a diferentes cosas. En el ejemplo de arriba, **order\_date** puede significar diferentes cosas.

## Null Values

- Designate a missing or unknown value
- Not the same as blanks or zeroes

Un asignamiento de *valor nulo* especifica un valor desconocido o perdido. Ud. necesita determinar donde o no los atributos pueden contener valores nulos.

Un valor nulo no es la misma cosa que espacios o ceros. En algunas aplicaciones, el cero puede representar un valor válido en lugar de un valor desconocido (por ejemplo, resultados de pruebas médicas de laboratorio). Los espacios en blanco y ceros serán interpretados como espacios y ceros.

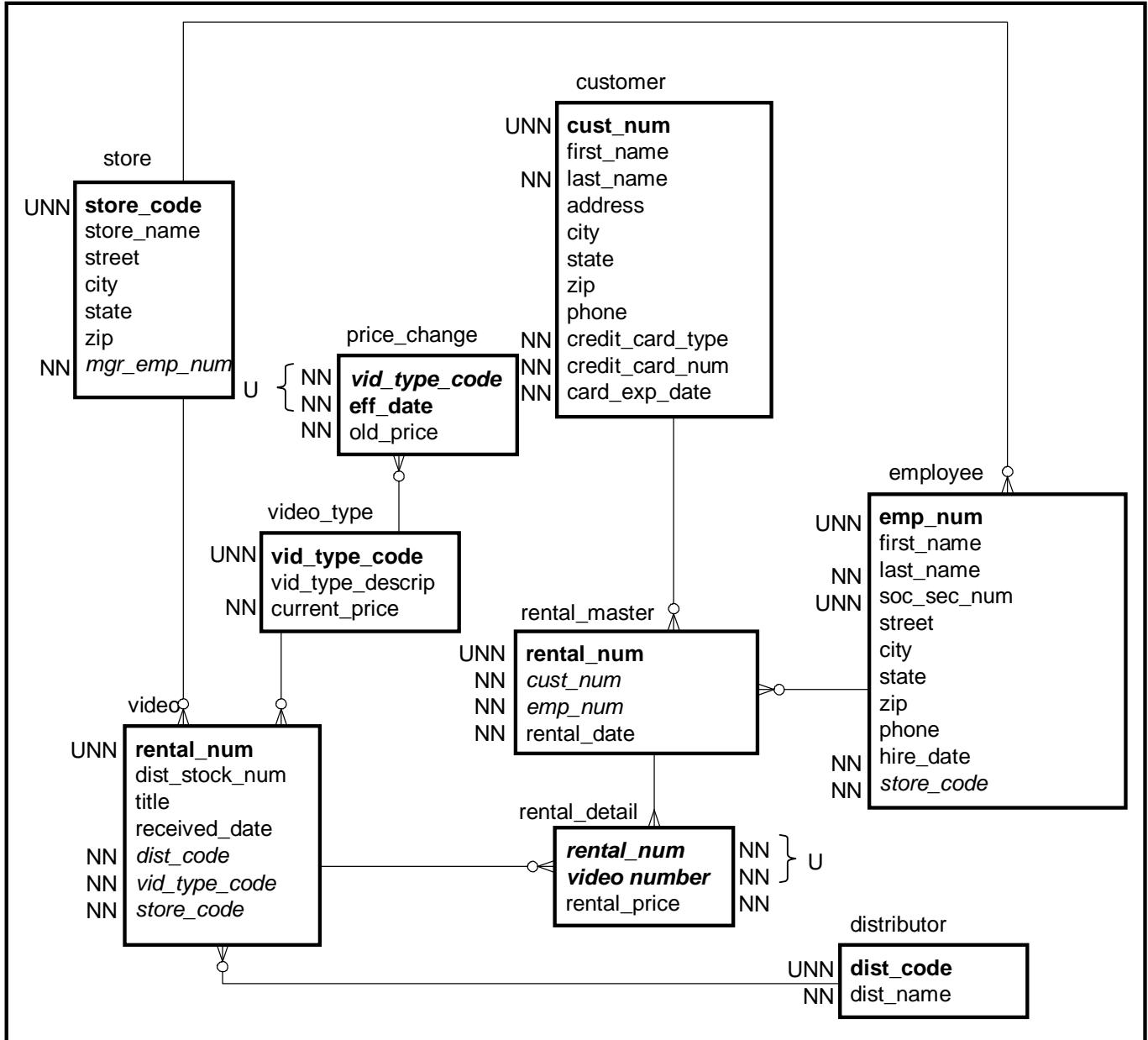
## **Unique Requirement**

A *unique requirement* indicates that an attribute must have a unique value for every entity instance.

Un requerimiento único puede ser especificado para un atributo. Un requerimiento único indica que no puede haber dos instancias que contengan el mismo valor para ese atributo.

## Attribute Specifications: Video Rental System

**U** indicates a unique requirement. **NN** indicates that null value are not allowed.



## **EJERCICIO DE LABORATORIO # 8**