

Taller #1

Business Intelligence

Carlos Reveco
creveco@dcc.uchile.cl

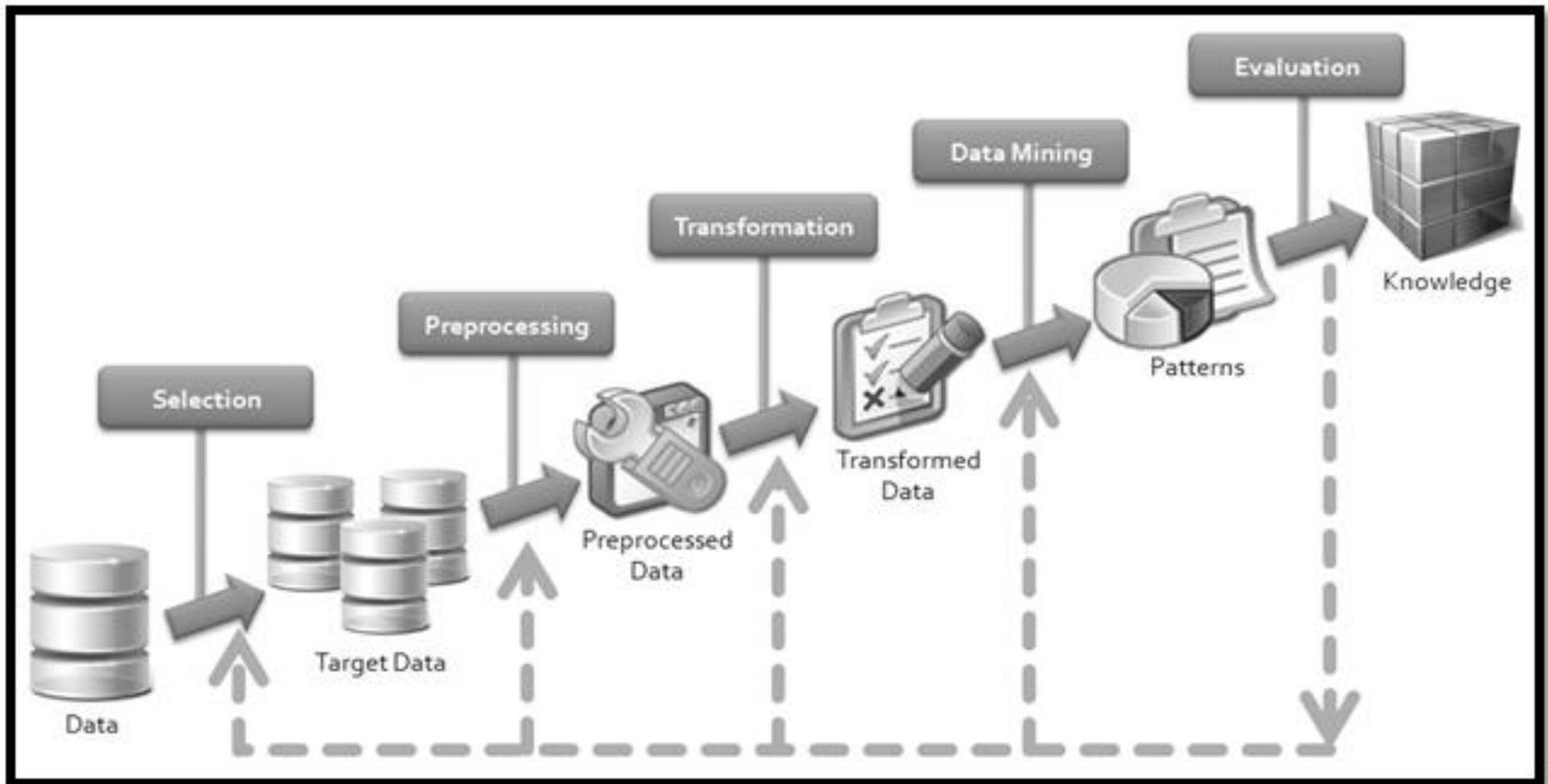
Cinthya Vergara
cvergarasilv@ing.uchile.cl

Agenda

- Taller#1
 - Modelamiento Entidad-Relación.
 - Motivación – Antecedentes Generales
 - Ejercicios – Soluciones
 - Consultas SQL
 - Motivación – Antecedentes Generales
 - Ejercicios – Soluciones

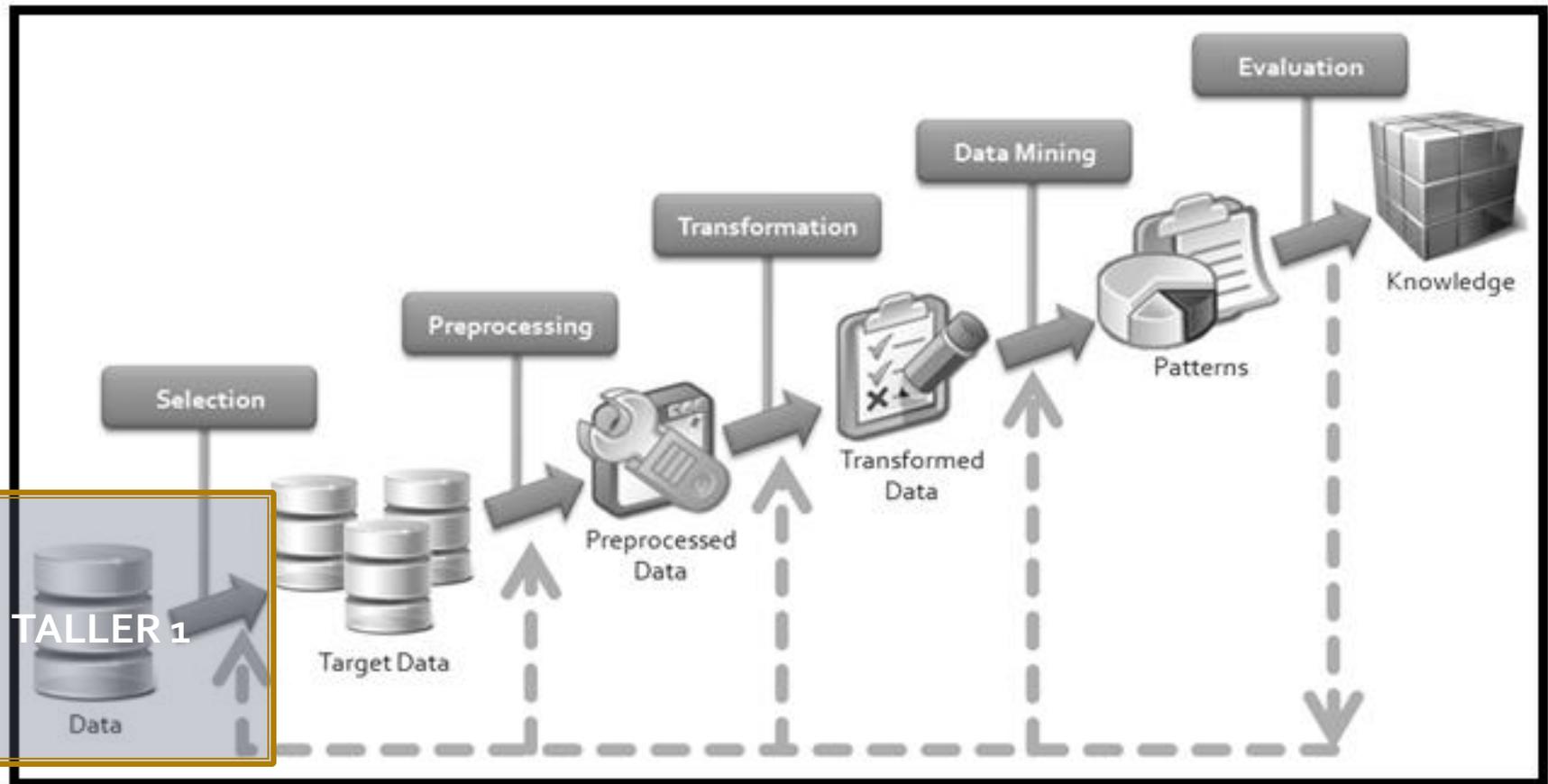
Taller #1

Proceso KDD



Knowledge Discovery in Databases → KDD

Proceso KDD



Knowledge Discovery in Databases → KDD

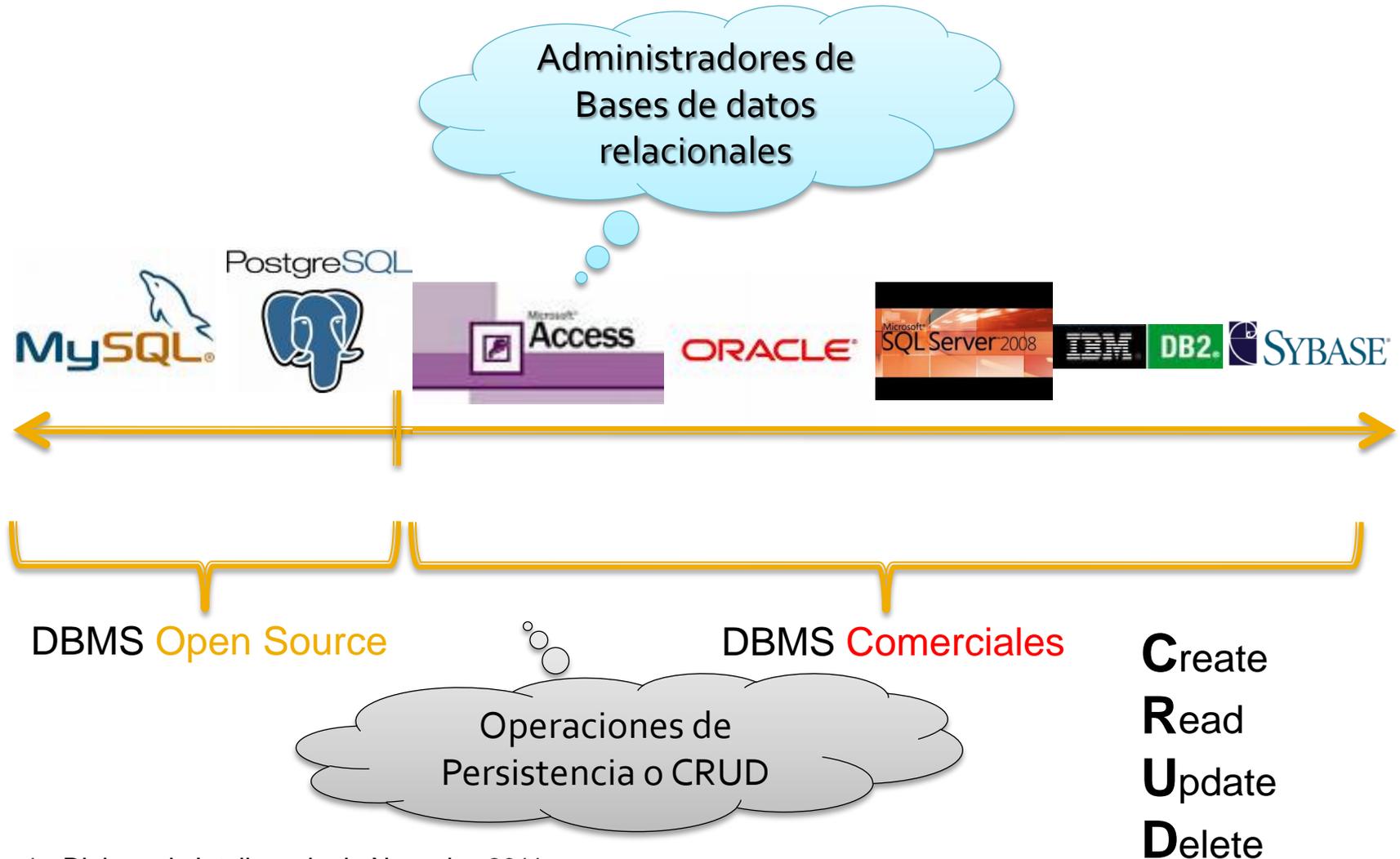
Taller #1

Modelamiento Entidad-Relación

Definiciones

- **Base de Datos:**
“Un **conjunto** lógicamente coherente de **datos relacionados**, construido para una cierta aplicación”.
- **Sistema Administración de Bases de Datos (SABD o DBMS):**
“**Software** que permite a las bases de datos ser **definidas**, además de **construidas** y **operarlas**”.
- **Diagrama Entidad – Relación (ER):**
Modelo lógico de la base de datos para representar información de manera **estructurada**.
- **Structured Query Language (SQL):**
Lenguaje de **consulta** estructurado para interactuar con la BD.

Bases de Datos – SADB o DBMS



Modelo Entidad-Relación (2)

■ Características

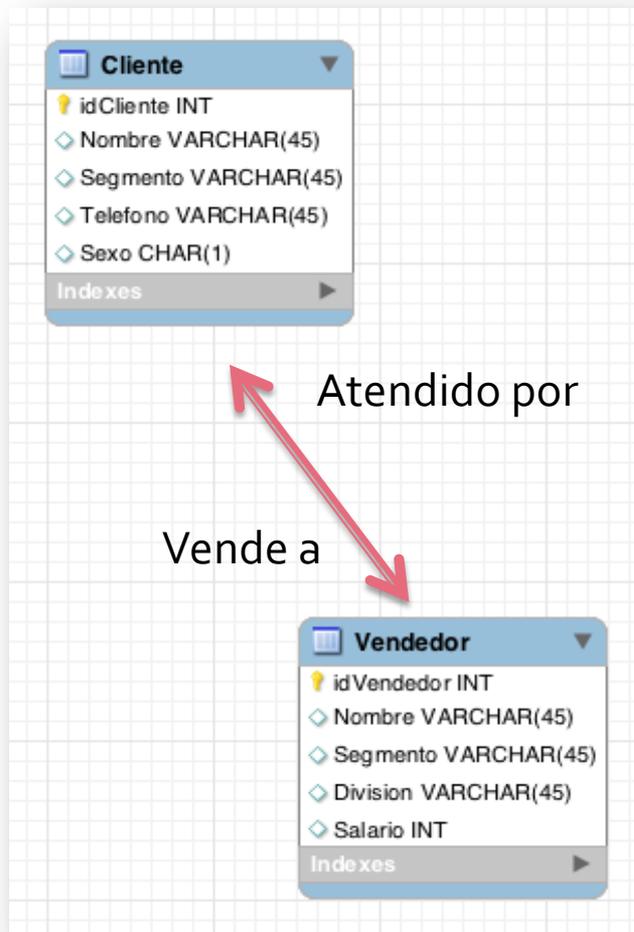
- Proceso **Top-Down**
- *Abstracción del negocio* y llevarlo a *entidades* y *relaciones*
- El **modelo es genérico**: Es necesaria una herramienta para implementarlo: DBMS

■ Ventajas

- Relativamente *fácil de entender*
- *Base matemática sólida*
- *Estandarizado*
- Cualquier consulta es *factible* de ser resuelta
- Reduce **redundancia**
 - Minimiza los errores de **consistencia**

Modelo Entidad-Relación

Definiciones



- **Entidad:** Objetos que puedo caracterizar dentro del problema a modelar.
Por ejemplo: clientes, vendedores, compañías, etc.
- **Relación:** Asociación directa que ocurre entre dos entidades
- **Atributo:** Características que definan la entidad. Por ejemplo: Nombre, Segmento, Teléfono, Edad, etc.

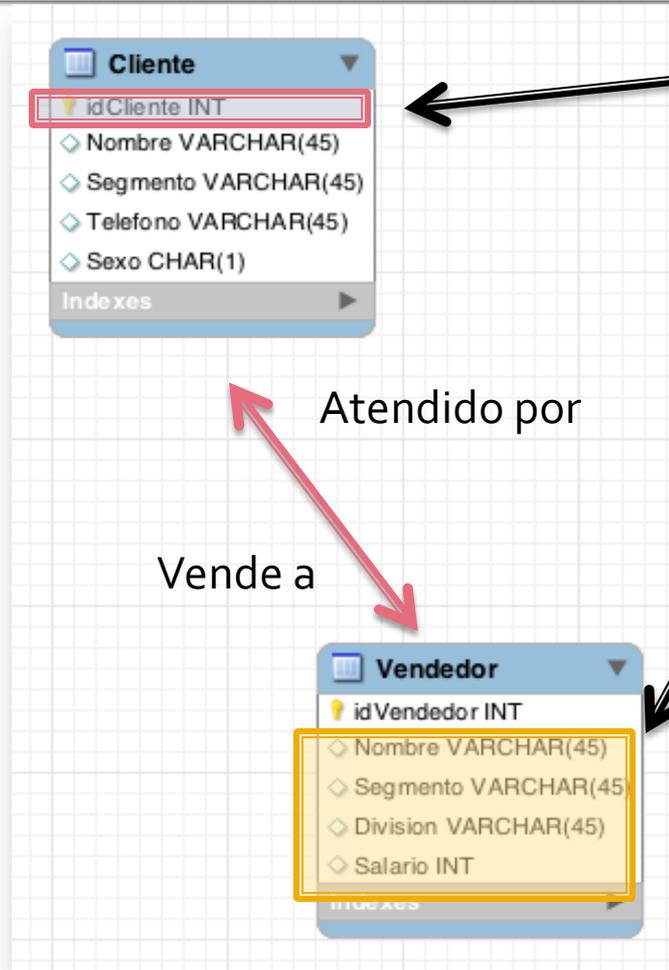
Modelo Entidad-Relación

Los Atributos

- **Llave única:** Corresponde a uno o dos atributos cuyos valores identifican de forma única los registros de una tabla.
En otras palabras, **sus valores no se repiten.**
- Por ejemplo:
 - **RUT** de un cliente
 - **Número de Matricula y Código Curso** para la entidad que describe las inscripciones de alumnos en los distintos cursos. (el par jamás se repite)

Modelo Entidad-Relación

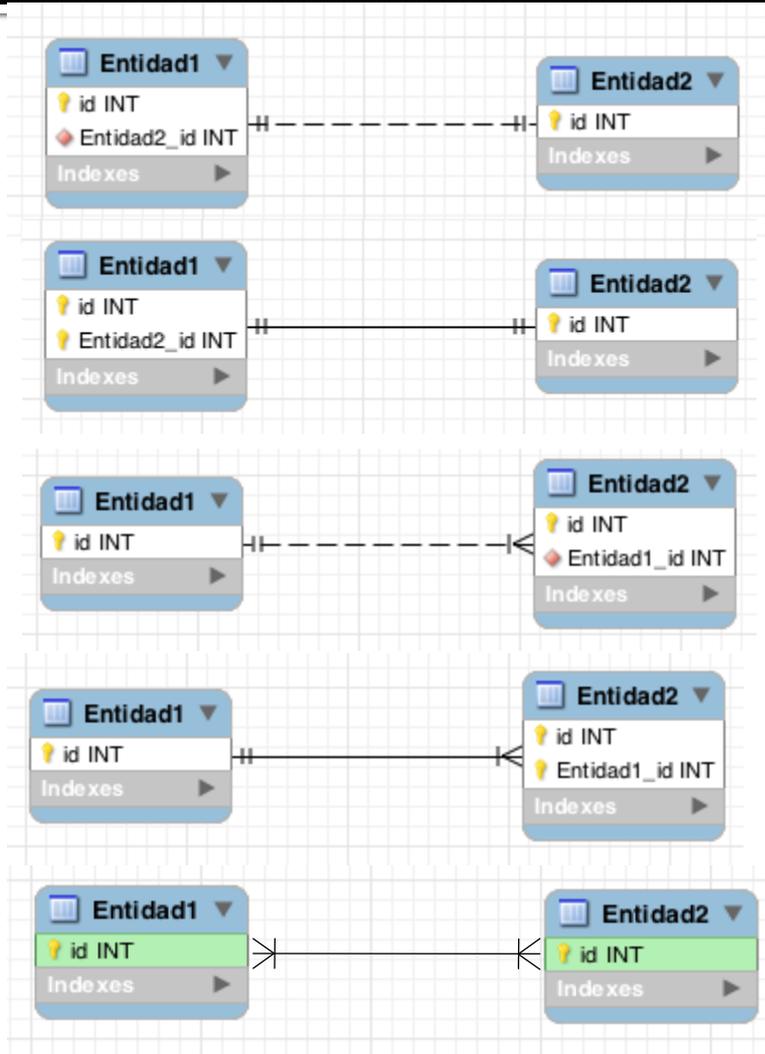
Los Atributos



- **Llave primaria:** Es una llave única que sirve para construir las relaciones con otras entidades.
- **Atributo no primario:** Es aquel atributo que no pertenece a la llave primaria.

Modelo Entidad-Relación

Tipos de Relaciones



Cada registro de la Entidad1 puede estar relacionado con 0 o 1 registro de la Entidad2

Cada registro de la Entidad1 está relacionado a sólo 1 registro de la Entidad2

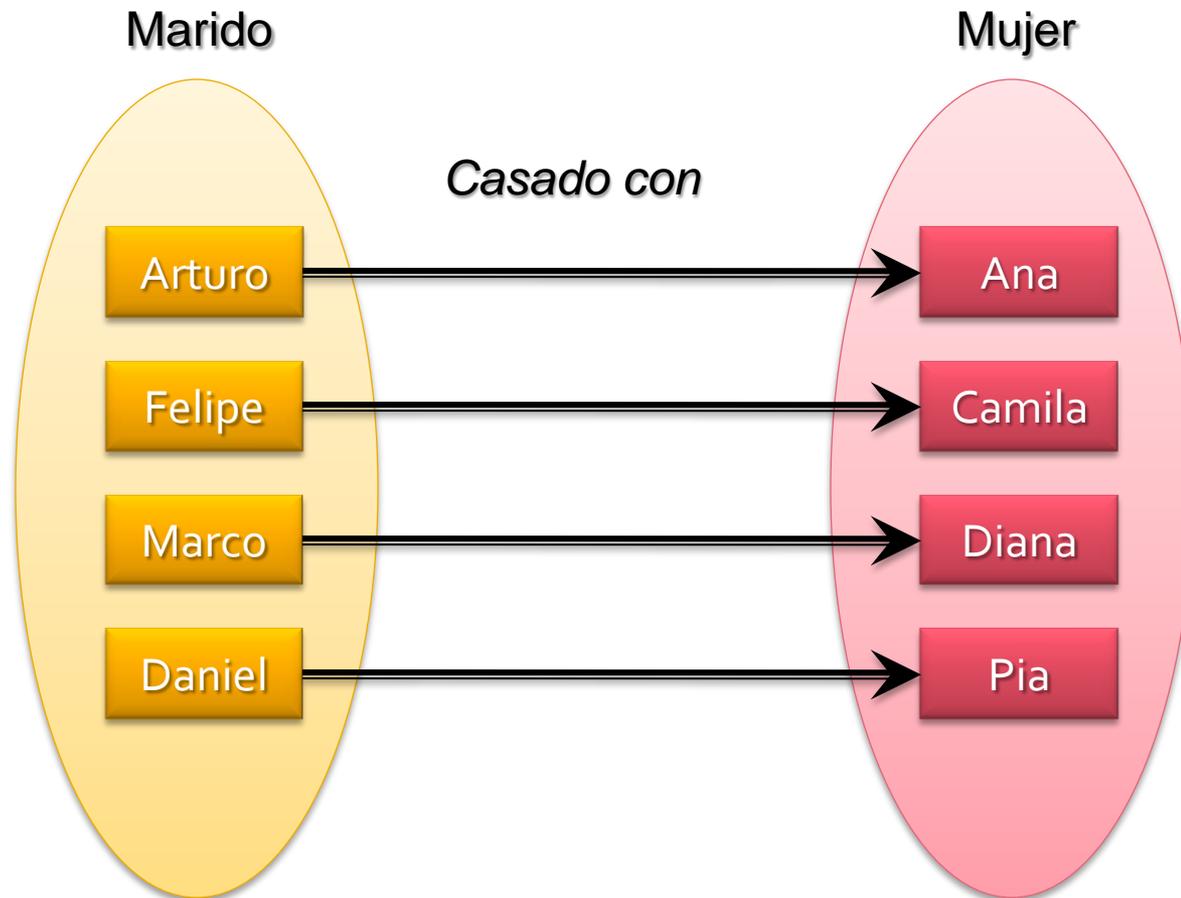
Cada registro de la Entidad1 puede estar relacionado con 0 o N registros de la Entidad2 ($N \geq 1$)

Cada registro de la Entidad1 está relacionado con N registros de la Entidad2 ($N \geq 1$)

Un registro de la Entidad1 está relacionado con N registros de la Entidad2 ($N \geq 1$) &
Un registro de la Entidad2 está relacionado con N registros de la Entidad1 ($N \geq 1$)

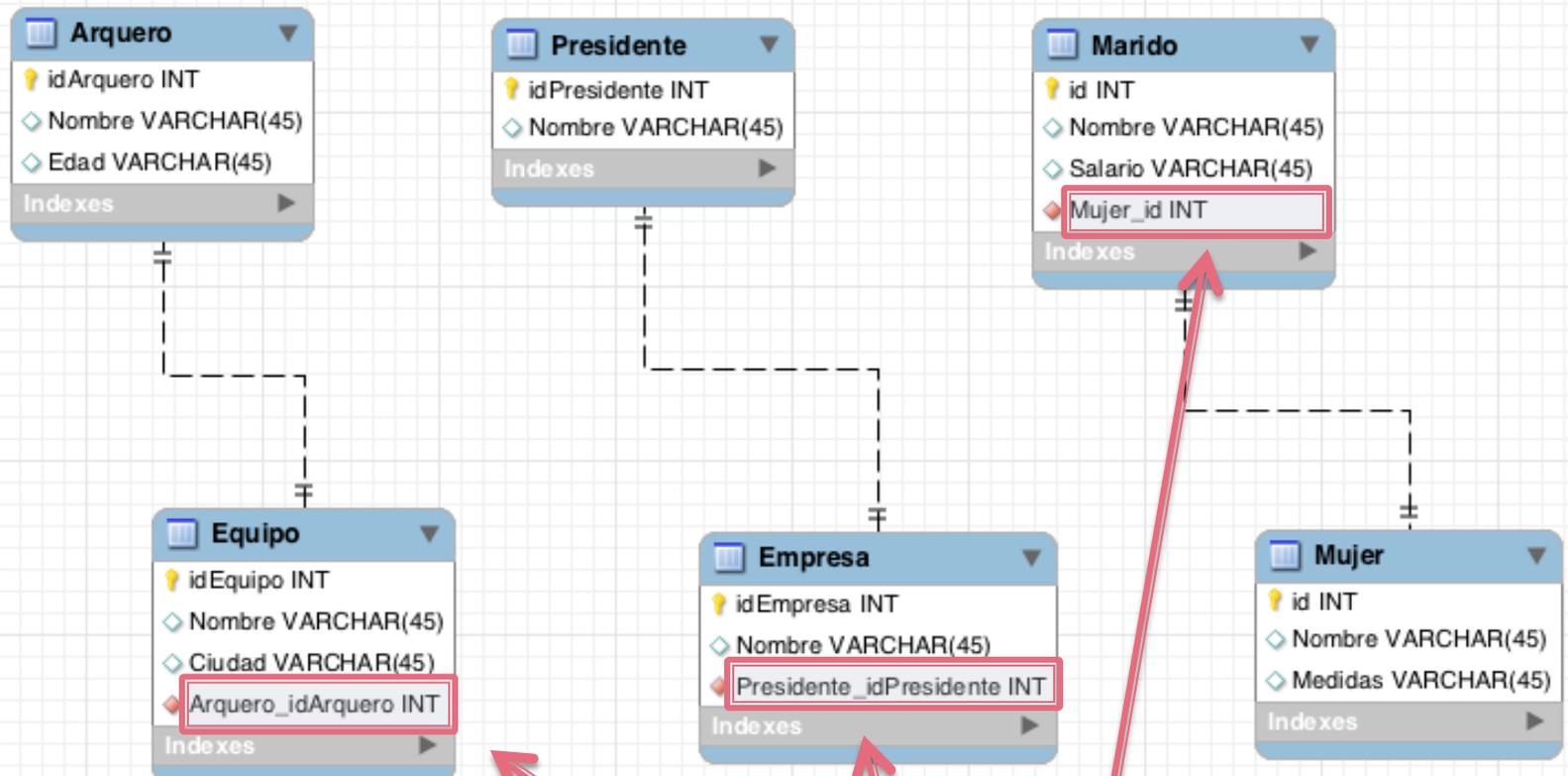
Modelo Entidad-Relación

Tipos de Relaciones: "1 a 1"



Modelo Entidad-Relación

Tipos de Relaciones: "1 a 1"



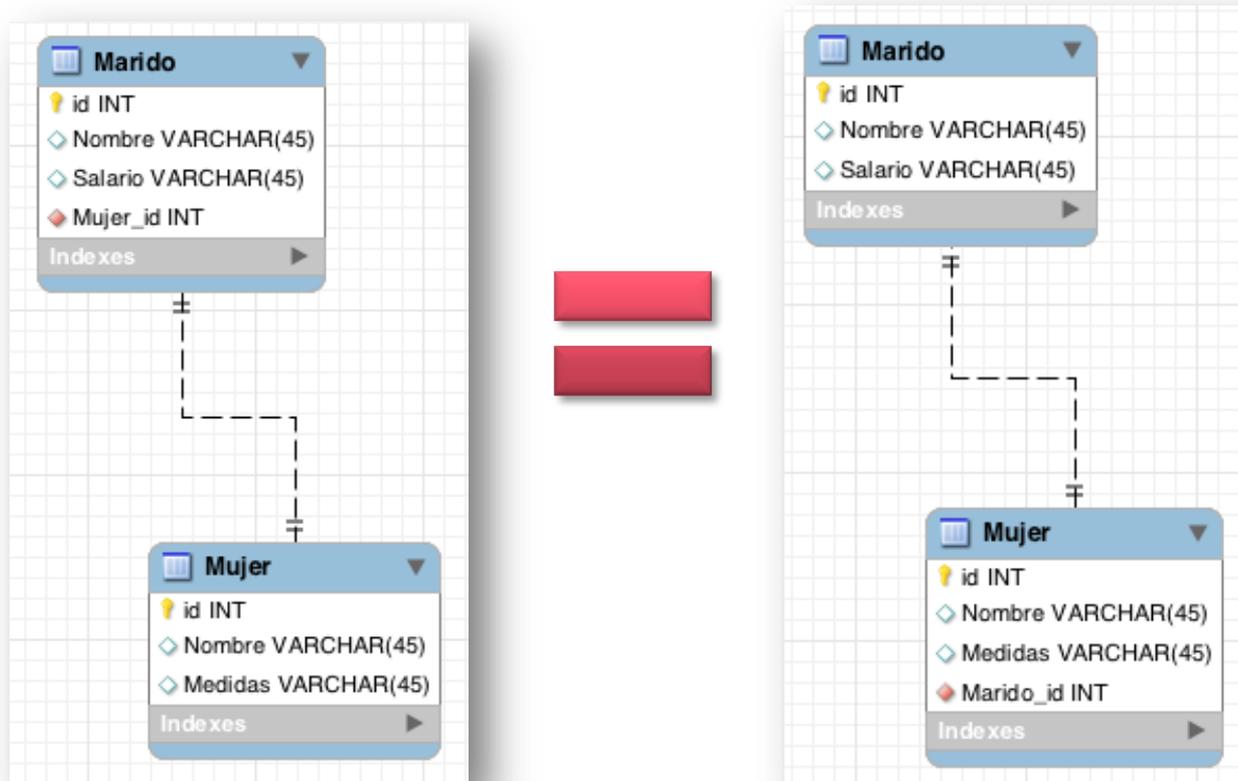
En este caso, las llaves foráneas **NO** son parte de las llaves primarias de las entidades: Equipo, Empresa y Marido

Llaves foráneas

Modelo Entidad-Relación

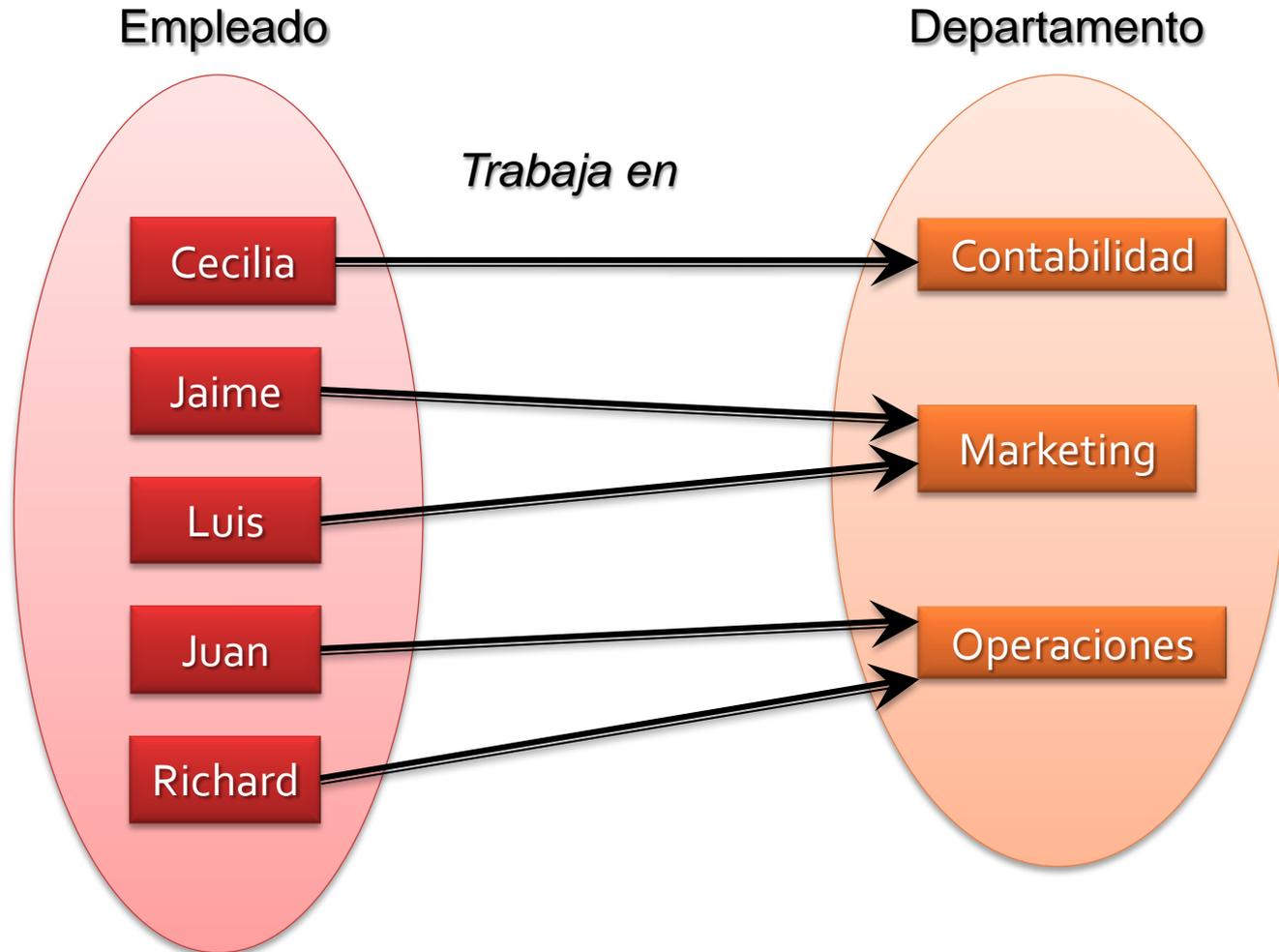
Tipos de Relaciones: "1 a 1"

- Notar que



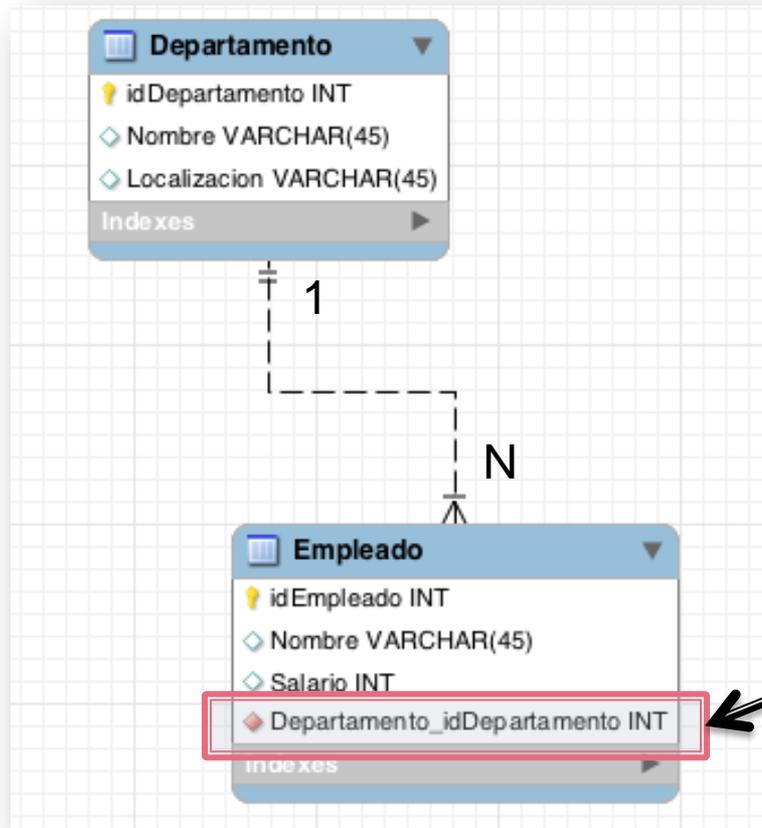
Modelo Entidad-Relación

Tipos de Relaciones: "1 a N"



Modelo Entidad-Relación

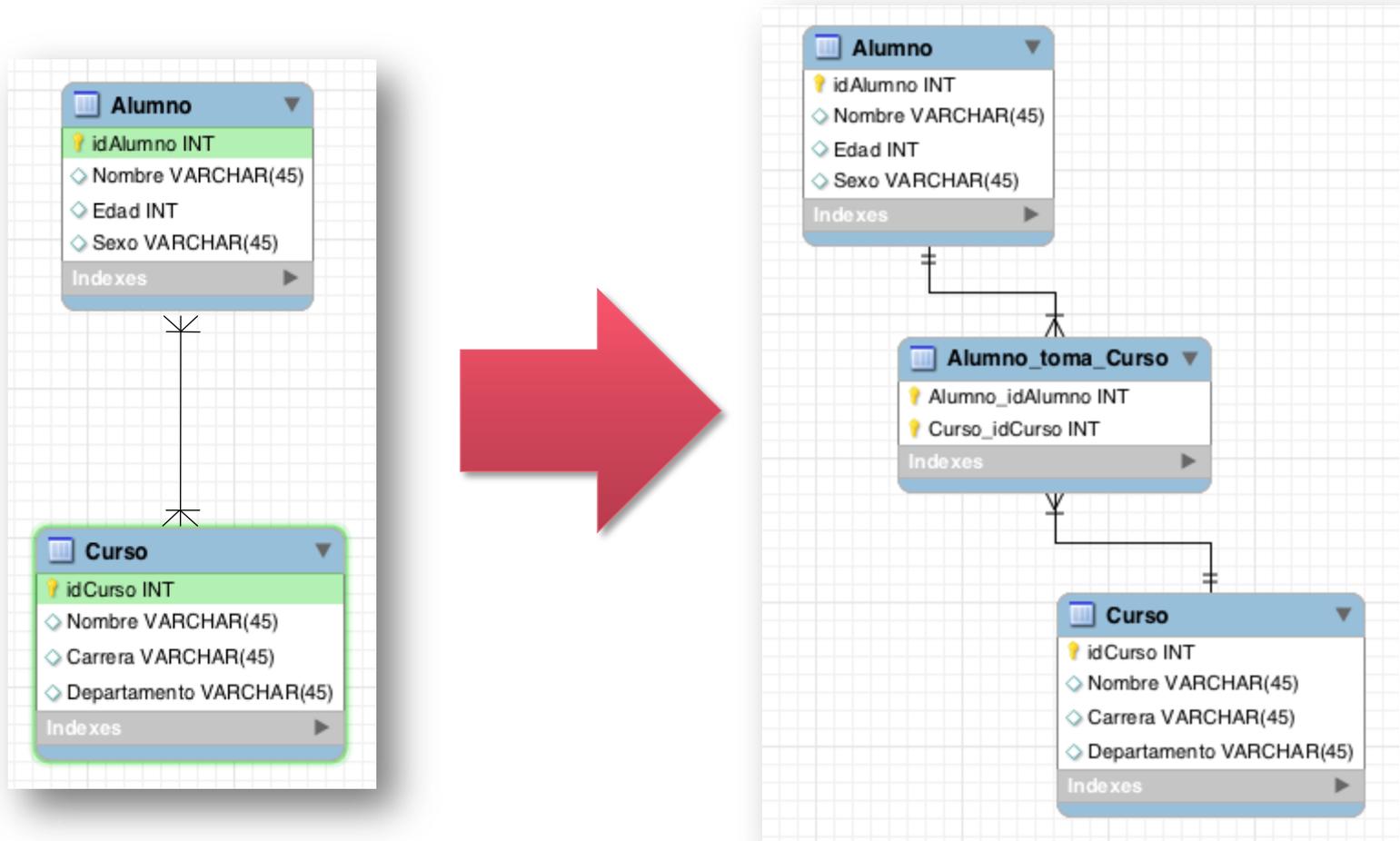
Tipos de Relaciones: "1 a N"



OJO con la posición de la llave foránea (en el lado de N)

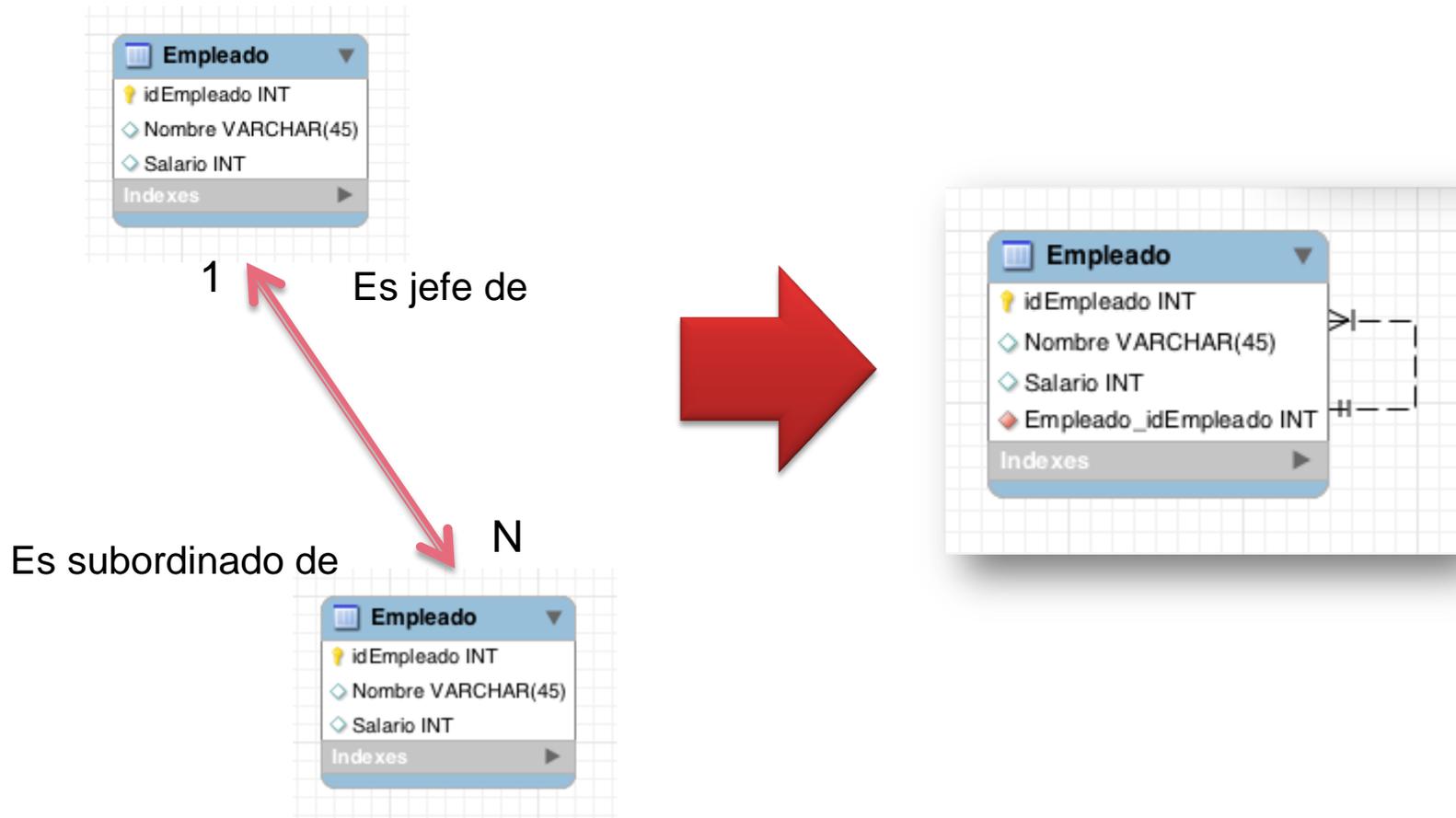
Modelo Entidad-Relación

Tipos de Relaciones: "N a M"



Modelo Entidad-Relación

Relaciones Reflexivas



Normalización

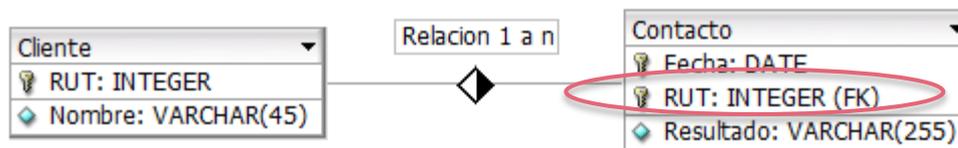
Primera Forma Normal

- Exige que cada fila en la tabla esté libre de **“Grupos repetitivos”**
 - Cada atributo de la entidad debe ser **atómico**

Cliente
RUT: INTEGER
Nombre: VARCHAR(45)
Fecha_contacto: DATE

RUT	Nombre	Fecha_contacto
17476803-4	J. Velasquez	10/5/2009 12/05/2009
13456980-2	Pablo Catalán	30/4/2009

Atributo **Fecha de Contacto** es multivariado



Llave Foránea

Normalización

Segunda Forma Normal

- Exige que **todo registro** debe ser accedido a través de la **llave primaria completa**.

Employee	
🔑	Name: VARCHAR(45)
🔑	Job: VARCHAR(45)
💎	Salary: DOUBLE
💎	Address: VARCHAR(255)

Llave compuesta

- * **Address** sólo depende de **Name**
- * **Salary** sólo depende de **Job**

Name	Job	Salary	Address
Smith	Welder	14.75	123 4th St
Smith	Programmer	24.50	123 4th St
Smith	Waiter	7.50	123 4th St
Jones	Programmer	26.50	4 Moose Lane
Jones	Bricklayer	17.50	4 Moose Lane
Adams	Analyst	28.50	88 Tiger Circle

Redundancia en datos

Normalización

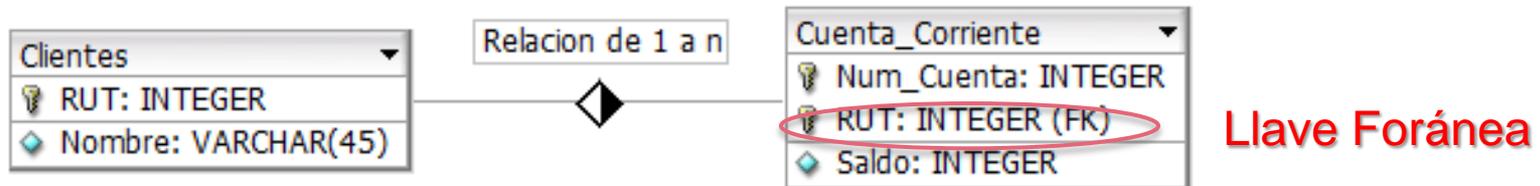
Segunda Forma Normal (Ejemplo II)

Clientes	
🔑	RUT: INTEGER
🔑	Num_Cuenta: INTEGER
💎	Nombre: VARCHAR(45)
💎	Saldo: INTEGER

Nombre sólo depende de RUT

Saldo sólo depende de Num_Cuenta

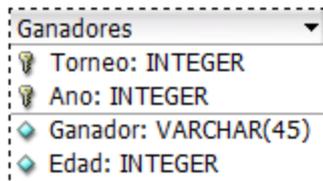
→ Separar en dos entidades



Normalización

Tercera Forma Normal

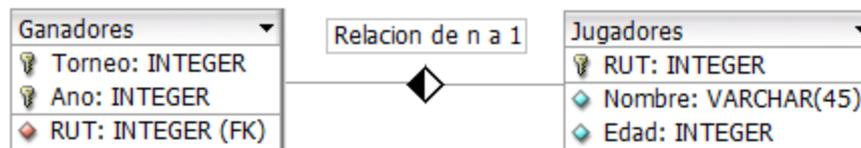
- Una entidad estará en 3FN cuando:
 - Está en Segunda Forma Normal (2FN)
 - Ningún atributo no-primario de la tabla es dependiente transitivamente de la clave primaria.



El nombre del **Ganador** depende del **Torneo** y del **Año** en que se realizó

La **Edad** depende directamente del nombre del **Ganador**

→ **Edad** depende transitivamente de la llave primaria



EJERCICIO

Sistema de Ventas

- Le contratan para hacer una BD que permita apoyar la gestión de un sistema de ventas.
- La empresa necesita llevar un control de proveedores, clientes, productos y ventas.
- Un proveedor tiene un RUT, nombre, dirección, teléfono y página web. Un cliente también tiene RUT, nombre, dirección, pero puede tener varios teléfonos de contacto. La dirección se entiende por calle, número, comuna y ciudad. Un producto tiene un id único, nombre, precio actual, stock y nombre del proveedor.
- Además se organizan en categorías, y cada producto va sólo en una categoría. Una categoría tiene id, nombre y descripción.
- Por razones de contabilidad, se debe registrar la información de cada venta con un id, fecha, cliente, descuento y monto final. Además se debe guardar el precio al momento de la venta, la cantidad vendida y el monto total por el producto.

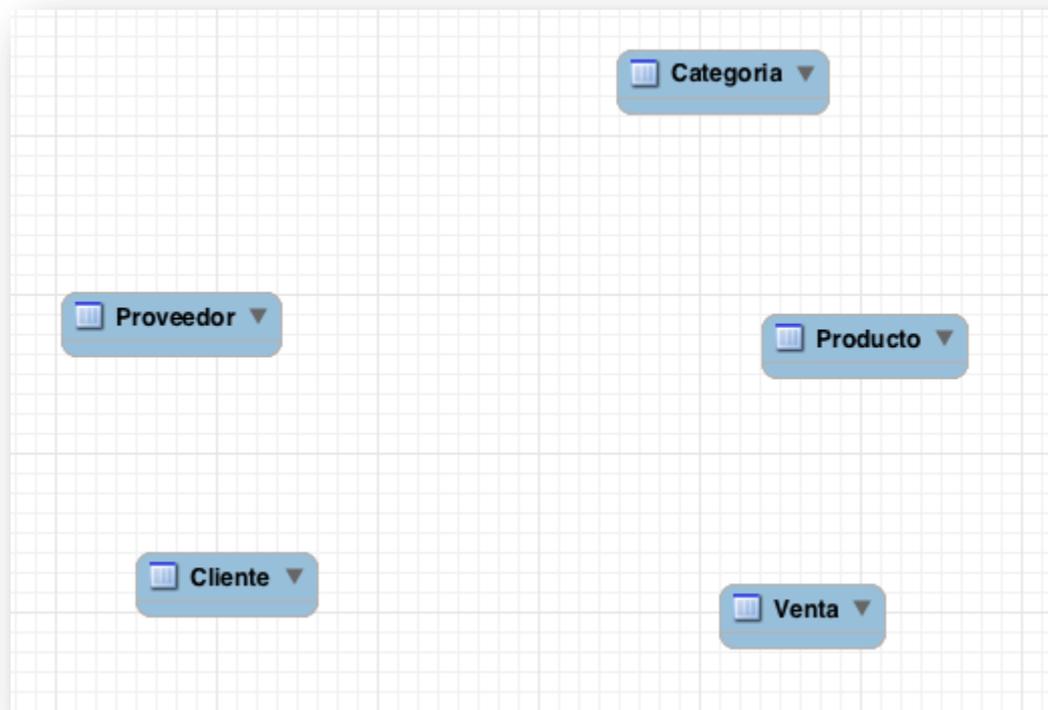
Solución

Paso 1: Identificar Entidades

- Le contratan para hacer una BD que permita apoyar la gestión de un sistema de ventas.
- La empresa necesita llevar un control de **proveedores**, **clientes**, **productos** y **ventas**.
- Un **proveedor** tiene un RUT, nombre, dirección, teléfono y página web. Un **cliente** también tiene RUT, nombre, dirección, pero puede tener varios teléfonos de contacto. La dirección se entiende por calle, número, comuna y ciudad. Un **producto** tiene un id único, nombre, precio actual, stock y nombre del proveedor.
- Además se organizan en **categorías**, y cada producto va sólo en una categoría. Una **categoría** tiene id, nombre y descripción.
- Por razones de contabilidad, se debe registrar la información de cada **venta** con un id, fecha, cliente, descuento y monto final. Además se debe guardar el precio al momento de la venta, la cantidad vendida y el monto total por el producto.

Solución

Paso 1: Identificar Entidades



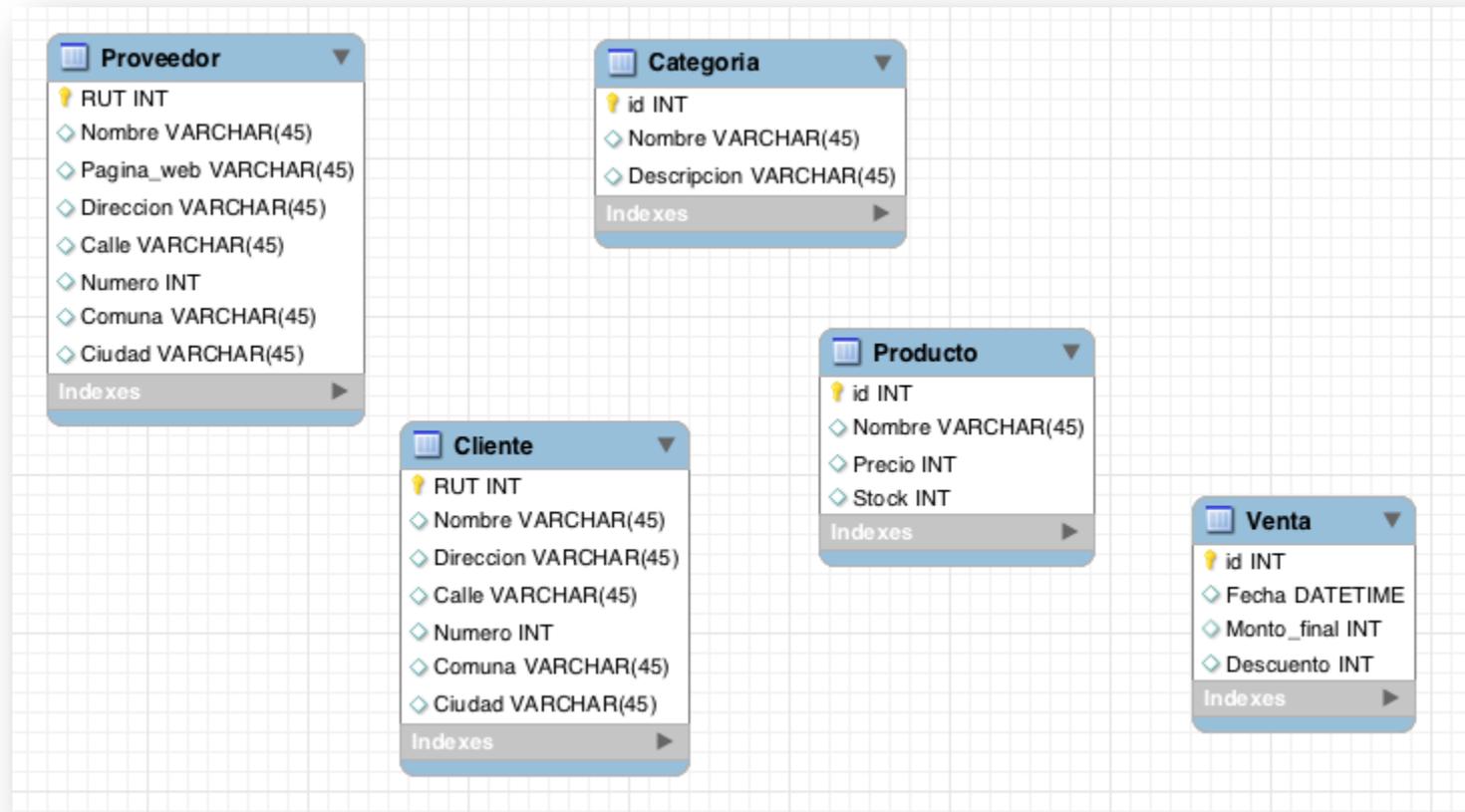
Solución

Paso 2: Identificar Atributos

- Le contratan para hacer una BD que permita apoyar la gestión de un sistema de ventas.
- La empresa necesita llevar un control de proveedores, clientes, productos y ventas.
- Un **proveedor** tiene un *RUT, nombre, dirección, teléfono y página web*. Un **cliente** también tiene *RUT, nombre, dirección*, pero puede tener varios *teléfonos de contacto*. La **dirección** se entiende por *calle, número, comuna y ciudad*. Un **producto** tiene un *id único, nombre, precio actual, stock y nombre del proveedor*.
- Además se organizan en **categorias**, y cada producto va sólo en una categoría. Una categoría tiene *id, nombre y descripción*.
- Por razones de contabilidad, se debe registrar la información de cada **venta** con un *id, fecha, cliente, descuento y monto final*. Además se debe guardar el precio al momento de la venta, la cantidad vendida y el monto total por el producto.

Solución

Paso 2: Identificar Atributos



Solución

Paso 3: Identificar Relaciones

- Le contratan para hacer una BD que permita apoyar la gestión de un sistema de ventas.
- La empresa necesita llevar un control de proveedores, clientes, productos y ventas.
- Un proveedor tiene un RUT, nombre, dirección, teléfono y página web. Un cliente también tiene RUT, nombre, dirección, pero puede tener varios teléfonos de contacto. La dirección se entiende por calle, número, comuna y ciudad. **Un producto tiene un id único, nombre, precio actual, stock y nombre del proveedor.**
- Además se organizan en categorías, y **cada producto va sólo en una categoría.** Una categoría tiene id, nombre y descripción.
- Por razones de contabilidad, se debe registrar **la información de cada venta con un id, fecha, cliente, descuento y monto final.** Además se debe guardar el precio al momento de la venta, la cantidad vendida y el monto total por el producto.

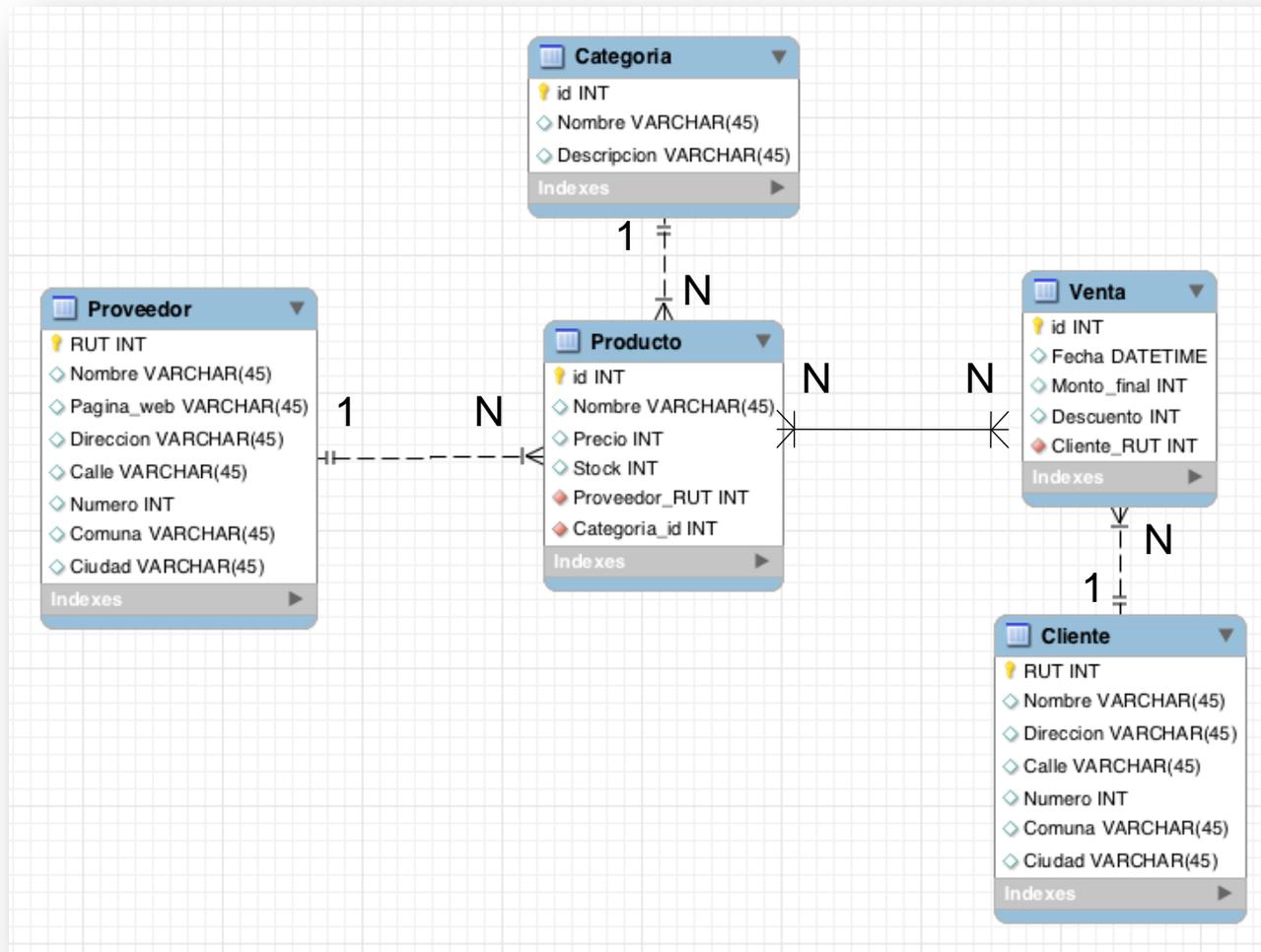
Solución

Paso 3: Identificar Relaciones

- **Proveedor – Producto**
 - Un producto es provisto por un solo proveedor
 - Un proveedor puede proveer más de un producto
 - → **Relación "1 a N"**
- **Categoría – Producto**
 - Un producto pertenece a una sola categoría
 - Una categoría puede tener muchos productos
 - → **Relación de "1 a N"**
- **Cliente – Venta**
 - Un venta esta asociada sólo a un cliente
 - Un cliente puede hacer múltiples ventas
 - → **Relación de "1 a N"**
- **Venta – Producto**
 - Una venta puede tener múltiples productos
 - Un producto puede estar inserto en múltiples ventas
 - → **Relación de "N a M"**

Solución

Paso 3: Identificar Relaciones



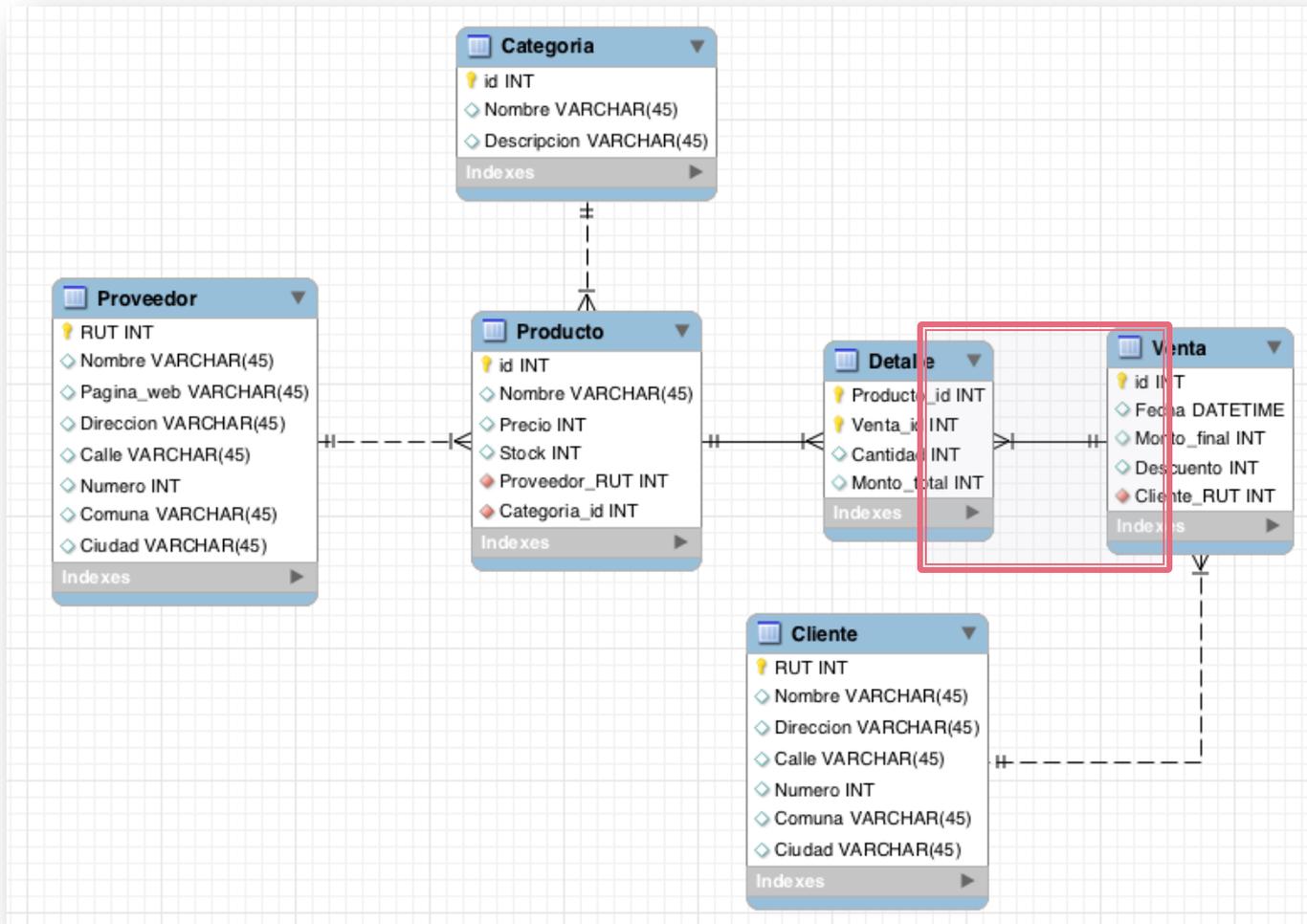
Solución

Paso 4: Entidades y Relaciones no directas

- Le contratan para hacer una BD que permita apoyar la gestión de un sistema de ventas.
- La empresa necesita llevar un control de proveedores, clientes, productos y ventas.
- Un proveedor tiene un RUT, nombre, dirección, teléfono y página web. Un cliente también tiene RUT, nombre, dirección, pero puede tener varios teléfonos de contacto. La dirección se entiende por calle, número, comuna y ciudad. Un producto tiene un id único, nombre, precio actual, stock y nombre del proveedor.
- Además se organizan en categorías, y cada producto va sólo en una categoría. Una categoría tiene id, nombre y descripción.
- Por razones de contabilidad, se debe registrar la información de cada venta con un id, fecha, cliente, descuento y monto final. **Además se debe guardar el precio al momento de la venta, la cantidad vendida y el monto total por el producto.**

Solución

Paso 4: Entidades y Relaciones no directas

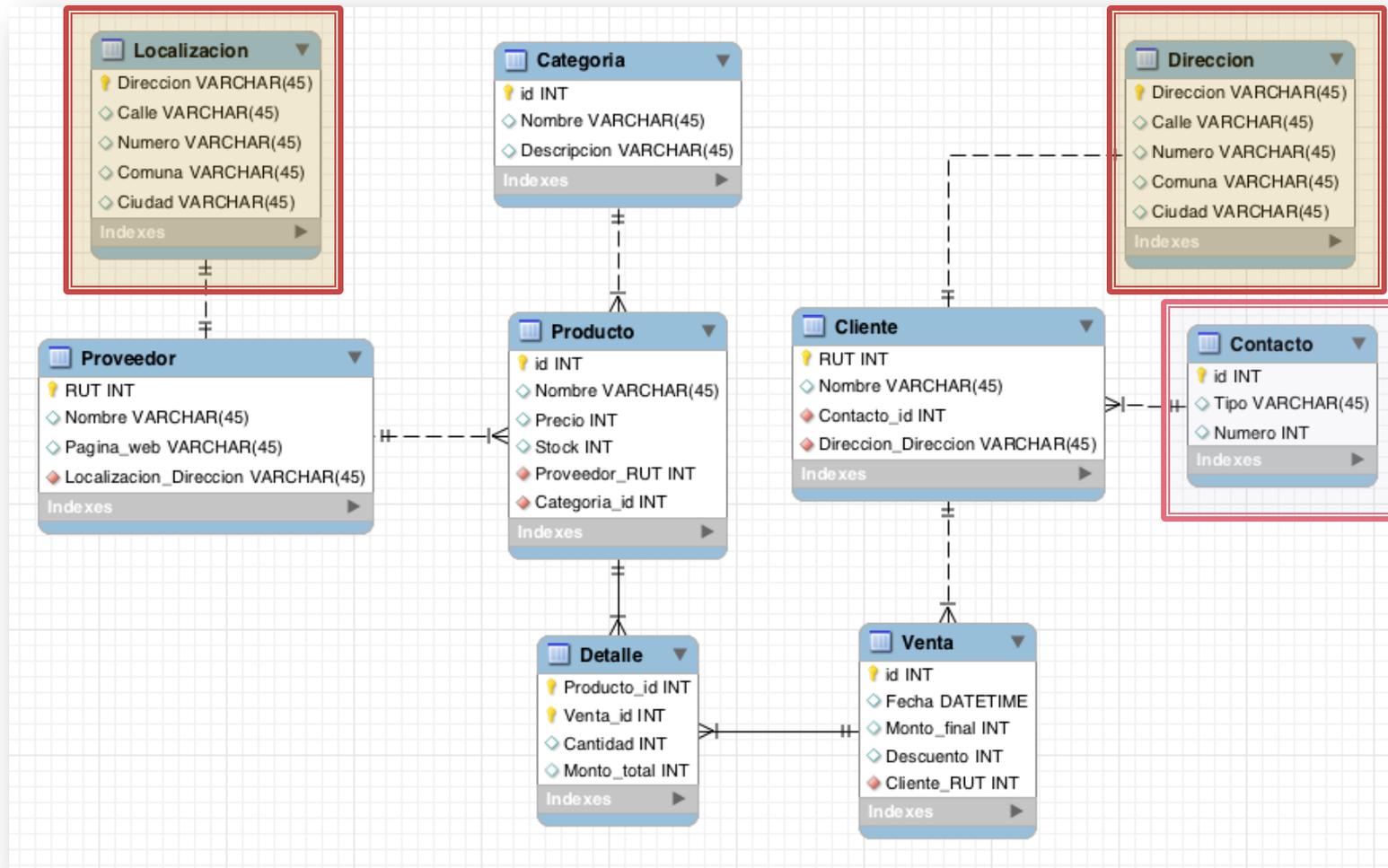


Solución

Paso 5: Normalizar

- Le contratan para hacer una BD que permita apoyar la gestión de un sistema de ventas.
- La empresa necesita llevar un control de proveedores, clientes, productos y ventas.
- Un proveedor tiene un RUT, nombre, dirección, teléfono y página web. **Un cliente también tiene RUT, nombre, dirección, pero puede tener varios teléfonos de contacto (viola 1FN)**. La **dirección se entiende por calle, número, comuna y ciudad (viola 3FN)**. Un producto tiene un id único, nombre, precio actual, stock y nombre del proveedor.
- Además se organizan en categorías, y cada producto va sólo en una categoría. Una categoría tiene id, nombre y descripción.
- Por razones de contabilidad, se debe registrar la información de cada venta con un id, fecha, cliente, descuento y monto final. Además se debe guardar el precio al momento de la venta, la cantidad vendida y el monto total por el producto.

Sistema de Ventas

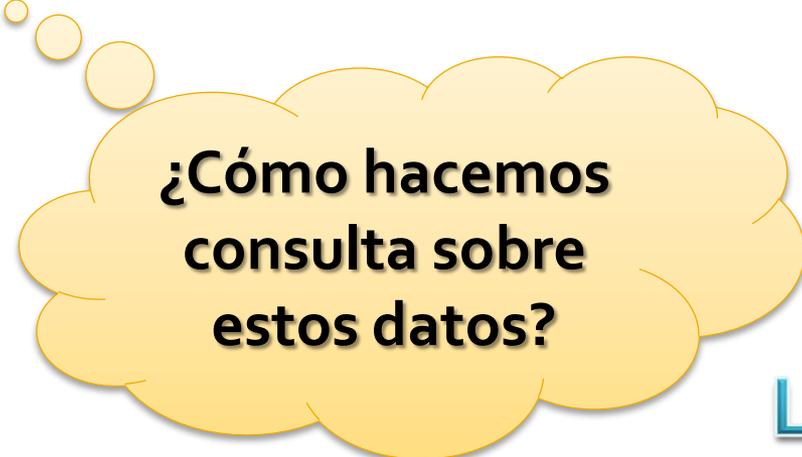


Taller #1

Consultas SQL

Lenguaje SQL

- Modelo ER → Modelamiento de base de datos
- DBMS → Implementar dicho modelo
- **Conclusión:** Tenemos datos almacenados



¿Cómo hacemos
consulta sobre
estos datos?

Lenguaje
SQL

Lenguaje SQL - Historia

- En 1974, en el Laboratorio de IBM en San Jose D. Chamberlin definió un lenguaje llamado '*Structured English Query Language*' o SEQUEL.
- Una versión mejorada de SEQUEL/ fue definida en 1976, pero su nombre fue cambiado por razones legales SQL.
- IBM produjo posteriormente un DBMS prototipo llamado System R, basado en SEQUEL/2
- En 1987, ANSI e ISO publicaron un estándar inicial para SQL.
- En 1989, ISO publicó una adición que definió ' una característica de realce de la integridad '.
- En 1992, la primera revisión importante al estándar de ISO ocurrió, designado SQL2 o SQL/92.

Objetivos

- **SQL Orientado a la Consulta:**
 - Uso de la componente **WHERE** para agregar condiciones a las Consultas.
 - Ordenar resultados de las Consultas usando **ORDER BY**.
 - Uso de Funciones Adicionales.
 - Agrupar datos usando **GROUP BY** y **HAVING**.
 - Uso de Sub-Consultas (o consultas anidadas).
 - Usar **JOIN** entre tablas.
 - Operaciones de Conjuntos (**UNION, INTERSECT, EXCEPT**)
- **SQL Orientado a transacciones y Administración**
 - Como realizar transacciones sobre la Base de Datos usando (**INSERT, UPDATE, and DELETE.**)
 - Comandos **CREATE** y **DELETE** (usuarios, tablas, etc.).

Lenguaje de consulta SQL

- Permite **obtener cualquier conjunto** de datos presentes en una base de datos relacional mediante una sentencia.

```
SELECT [DISTINCT | ALL]
{* | [column_expression [AS new_name]] [,...]}
FROM table_name [alias] [, ...]
[WHERE condition]
[GROUP BY column_list]
[ HAVING condition]
[ORDER BY column_list]
```

Lenguaje de consulta SQL

```
SELECT [DISTINCT | ALL]
{* | [column_expression [AS new_name]] [,...]}
FROM table_name [alias] [, ...]
[WHERE condition]
[GROUP BY column_list]
[ HAVING condition]
[ORDER BY column_list]
```

FROM Especifica las tablas que se usaran

WHERE Establece los filtros.

SELECT Especifica las columnas que se consultaran.

GROUP BY Permite agrupar los datos.

HAVING Permite generar filtros sobre los grupos de datos.

ORDER BY Especifica el orden de los datos.

SQL

Seleccionar Atributos

- Listar **todos los atributos** de la tabla **Staff**.

```
SELECT sno, fname, lname, address, tel_no  
FROM Staff;
```

- Se puede usar ***** como abreviación para todas las columnas :

```
SELECT * FROM Staff;
```

- Se puede limitar la consulta con el comando **limit**

```
SELECT * FROM Staff LIMIT 10
```

SQL

Uso de DISTINCT

- Para eliminar aquellos registros **duplicados**, se recomienda el uso del comando DISTINCT, por ejemplo en la tabla **Staff**.

```
SELECT DISTINCT fname, lname  
FROM Staff;
```

- Con la consulta anterior, se entrega un listado con **todos los nombres y apellidos distintos** presentes en la tabla **Staff**.

SQL

Campos calculados

- Se pueden realizar operaciones matemáticas a campos cuyo tipo lo permita (NUMBER; INT; DOUBLE; LONG, etc.)

```
SELECT fname, salary/12 AS salario_mensual  
FROM Staff;
```

- Para renombrar el campo a obtener, se puede utilizar el comando AS.
- Otras operaciones matemáticas comúnmente utilizadas son: ABS, MOD, EXP, SQRT, LN, CEILING, FLOOR, ROUND, POW

SQL

Condición de búsqueda

- A cualquier consulta se le pueden agregar condiciones de despliegue de información. Con el comando WHERE, se agregan las condiciones:

```
SELECT fname, salary/12 AS salario_mensual  
FROM Staff  
WHERE salario_mensual > 500.000;
```

- Al renombrar el campo con 'AS', se puede utilizar esta variable en lo que sigue de la consulta.
- Recordar toda consulta se puede describir como **“Seleccionar X, de la tabla Y, que cumpla Z”**

SQL

Comparación y Ordenamiento

- Existe una serie de comandos utilizados para distintas acciones. Por ejemplo,
 - **LIKE o RLIKE:** permite comparar cadenas de caracteres

```
SELECT nombre  
FROM Clientes  
WHERE nombre RLIKE 'juan'
```

- **ORDER BY:** permite ordenar sobre algún atributo

```
SELECT nombre, sueldo, edad  
FROM empleados  
ORDER BY sueldo DESC, edad ASC
```

- Generalmente se utiliza junto con **DESC** o **ASC** si se quiere hacer un ordenamiento Descendente o Ascendente.

SQL

Sentencias Agregadas

- Los operadores de agregación se utilizan para realizar operaciones complejas sobre los atributos de las consultas:
 - **MIN:** permite encontrar el valor mínimo sobre un conjunto de valores
 - **MAX:** permite encontrar el valor máximo
 - **AVG:** permite determinar el valor promedio
 - **COUNT:** permite contar la cantidad de valores obtenidos
 - **SUM:** permite sumar los valores asociados a un conjunto de valores.
- Estos operadores, se pueden utilizar en conjunto con el operador de agrupamiento **GROUP BY** y **HAVING**

SQL

Sentencias Agregadas (2)

- **GROUP BY** permite agrupar los resultados sobre una de las columnas o atributos indicados en la consulta.

```
SELECT SUM(monto_ventas), nombre, id_empleado  
FROM Empleados  
GROUP BY id_empleado
```

- La consulta anterior entrega la suma total de las ventas asociadas a un empleado en particular
- Dependiendo de la consulta, se debe tener cuidado al agrupar por valores que sean diferenciadores (e.g. "id_empleado" y no "nombre")

SQL

Sentencias Agregadas (3)

- **HAVING** permite filtrar los resultados utilizando funciones de agregación en su argumento.

```
SELECT AVG(salario), profesion
FROM Empleados
GROUP BY profesion
HAVING AVG(salario) > 5000000
```

- La consulta anterior entrega la suma total de las ventas asociadas a un empleado en particular
- Dependiendo de la consulta, se debe tener cuidado al agrupar por valores que sean diferenciadores (e.g. "id_empleado" y no "nombre")

SQL

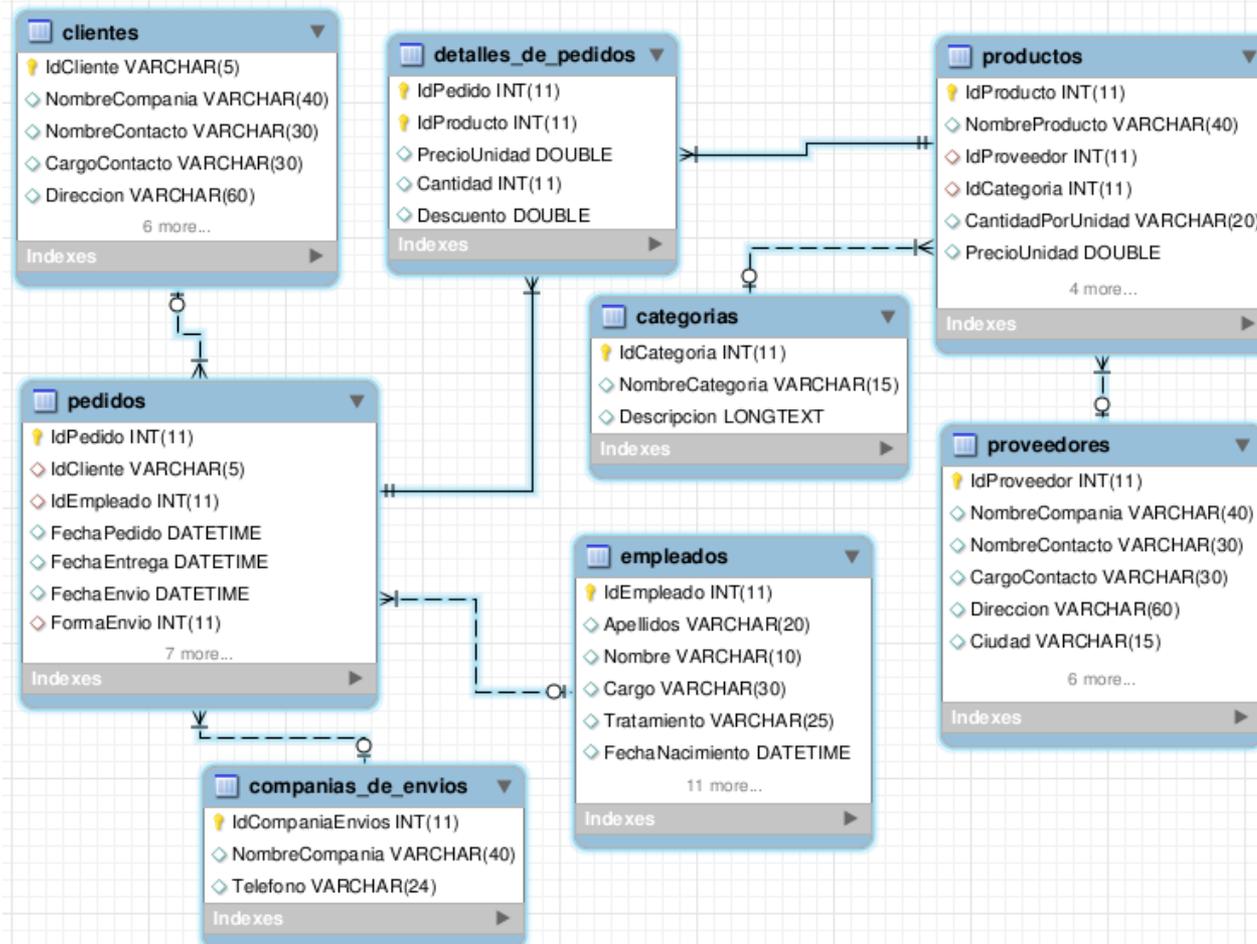
Consultas sobre Consultas

- Otra alternativa son las **consultas anidadas**. En este caso es posible hacer consultas sobre consultas pero ejecutando de forma secuencial las consultas necesarias.

```
SELECT D.nombre_depto
FROM depto D, empleados E
WHERE E.cantidad = Nempleos AND Nempleos IN
    (SELECT COUNT(*) AS Nempleos
     FROM Empleados
     GROUP BY id_depto
     HAVING COUNT(*) > 100)
```

Consultas SQL

- Dado el siguiente diagrama ER



Consultas SQL

Ejemplo

- Determinar la cantidad de pedidos con descuento, agrupados por el semestre de cada año.

Conectarse a
user: dip
pass: diplomado



Consultas SQL

Paso 1: Crear consulta preliminar

- Del texto inicial, se puede crear una consulta preliminar que representa a grandes rasgos lo solicitado:

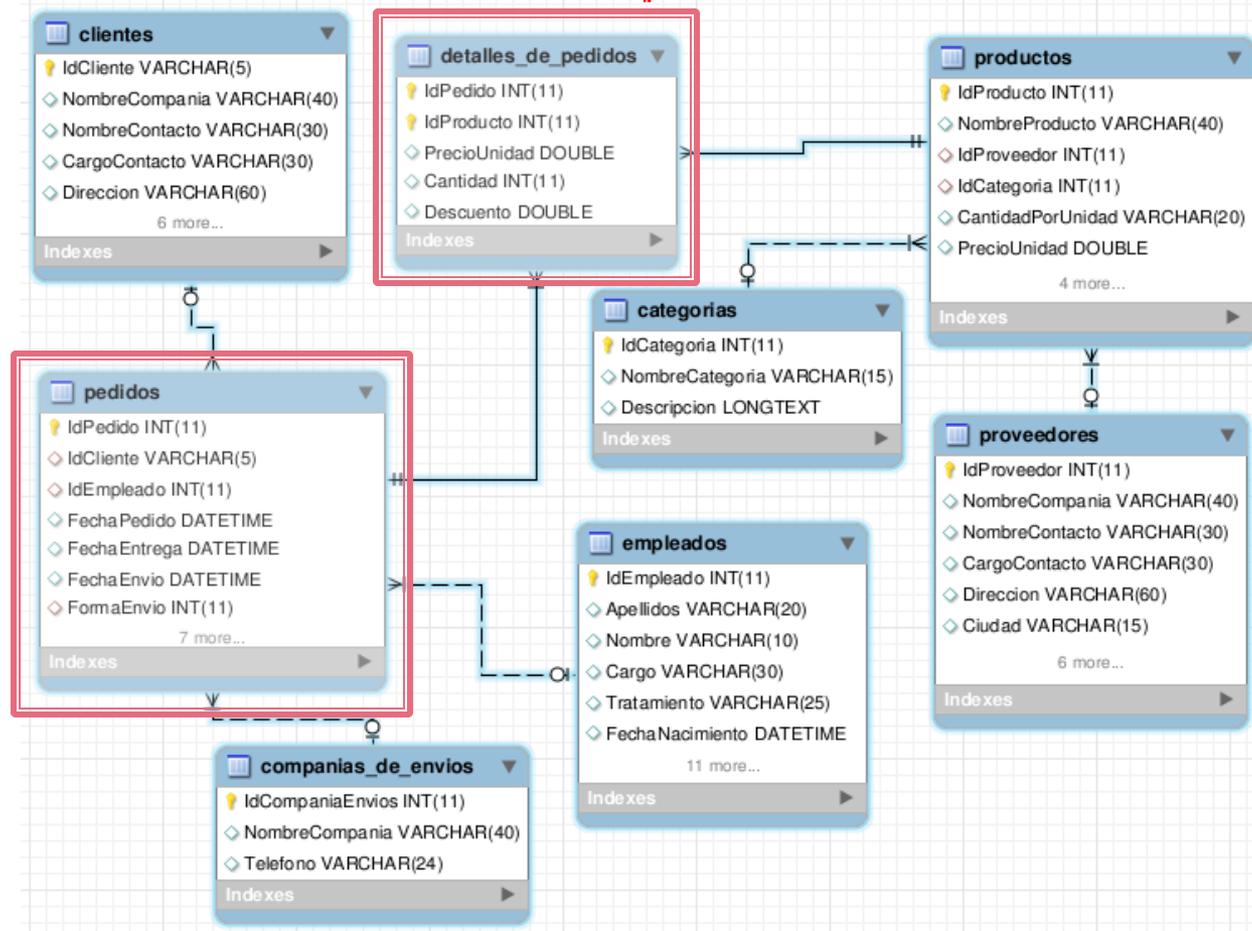
```
SELECT semestre, COUNT(pedidos) AS pedidos_con_dcto  
FROM [Tabla con semestre y pedidos]  
WHERE [pedidos tiene descuento]  
GROUP BY semestre
```

- Determinar la **cantidad** de **pedidos con descuento**, **agrupados** por el **semestre** de cada año.

Consultas SQL

Paso 2: Identificar tablas

- Identificar las tablas **relevantes para la consulta**



Consultas SQL

Paso 3: Actualizar Consulta

- Una vez identificadas las tablas, se puede actualizar la consulta, identificando los **atributos solicitados** en **SELECT**, cambiando el **nombre las tablas** en **FROM** y actualizando los **requerimientos** en **WHERE**. En ciertos casos, con lo anterior se puede terminar fácilmente una consulta.
- Se puede dar el caso que las tablas identificadas sí contengan la información necesaria, pero **no permitan concretar la consulta**. En ese caso, es necesario **crear una nueva tabla** con la información requerida.
- En el presente caso no son suficientes las tablas relacionadas, por lo que crearemos una nueva tabla.

Consultas SQL

Paso 4: Crear nuevas tablas

- Dado que es necesario, vamos a crear una nueva tabla que contenga todos aquellos **pedidos con descuento** junto con **su respectivo semestre**.

```
SELECT detalles de pedidos.IdPedido AS IdPedido,  
       CONCAT( IF( Month(pedidos.FechaPedido) <= 6 , "01-", "02-"),  
              YEAR(pedidos.FechaPedido)) AS Semestre  
FROM pedidos, detalles_de_pedidos  
WHERE detalles_de_pedidos.IdPedido = pedidos.IdPedido  
AND detalles_de_pedidos.Descuento <> 0
```

- Esta tabla nos permitirá agrupar los pedidos por semestre. Notar que utilizamos las **tablas identificadas anteriormente**.

Consultas SQL

Paso 5: Integrar nuevas tablas

- Una vez creada la tabla auxiliar, podemos actualizar la consulta anidando la consulta como una nueva tabla llamada "semestres":

```
SELECT semestre, COUNT(pedidos) AS pedidos_con_dcto
FROM (SELECT detalles_de_pedidos.IdPedido AS IdPedido ,
      CONCAT( IF( Month(pedidos.FechaPedido) <= 6 , "01-", "02-"),
      YEAR(pedidos.FechaPedido)) AS semestre
      FROM pedidos, detalles_de_pedidos
      WHERE detalles_de_pedidos.IdPedido = pedidos.IdPedido
      AND detalles_de_pedidos.Descuento <> 0) AS semestres
WHERE [pedidos tiene descuento]
GROUP BY semestre
```

Consultas SQL

Paso 5: Integrar nuevas tablas (2)

- Dado que en la sub consulta "semestres" ya tenemos considerada la condición que sean pedidos con descuento, podemos **eliminar la condición [pedidos tienen descuento]**

```
SELECT semestre, COUNT(pedidos) AS pedidos_con_dcto
FROM (SELECT detalles_de_pedidos.IdPedido AS IdPedido ,
      CONCAT( IF( Month(pedidos.FechaPedido) <= 6 , "01-", "02-"),
      YEAR(pedidos.FechaPedido)) AS semestre
      FROM pedidos, detalles_de_pedidos
      WHERE detalles_de_pedidos.IdPedido = pedidos.IdPedido
      AND detalles_de_pedidos.Descuento <> 0) AS semestres
WHERE [pedidos tiene descuento]
GROUP BY semestre
```

Consultas SQL

Paso 5: Integrar nuevas tablas (3)

- Además, es necesario **ajustar los atributos** y verificar que **no se cuenten más pedidos de los necesarios**.

```
SELECT semestres.semestre,  
        COUNT( DISTINCT semestres.IdPedido) AS pedidos_dcto  
FROM (SELECT detalles_de_pedidos.IdPedido AS IdPedido ,  
        CONCAT( IF( Month(pedidos.FechaPedido) <= 6 , "01-", "02-"),  
        YEAR(pedidos.FechaPedido)) AS semestre  
FROM pedidos, detalles_de_pedidos  
WHERE detalles_de_pedidos.IdPedido = pedidos.IdPedido  
AND detalles_de_pedidos.Descuento <> 0) AS semestres  
GROUP BY semestres.semestre
```

Consultas SQL

Resultado final

```
SELECT semestres.semestre,  
        COUNT( DISTINCT semestres.IdPedido) AS pedidos  
FROM (SELECT detalles_de_pedidos.IdPedido AS IdPedido ,  
        CONCAT( IF( Month(pedidos.FechaPedido) <= 6 , "01-", "02-"),  
        YEAR(pedidos.FechaPedido)) AS semestre  
FROM pedidos, detalles_de_pedidos  
WHERE detalles_de_pedidos.IdPedido = pedidos.IdPedido  
AND detalles_de_pedidos.Descuento <> 0) AS semestres  
GROUP BY semestres.semestre
```



<u>Semestre</u>	<u>pedidos</u>
01-1997	92
01-1998	117
02-1996	66
02-1997	105