

Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ciencia de la Computación



# IIC2115 - Programación como Herramienta para la Ingeniería

Bases de datos relacionales

**Profesora:** Francesca Lucchini  
**Prof. Coordinador:** Hans Löbel

# DBMS (Database Management System)

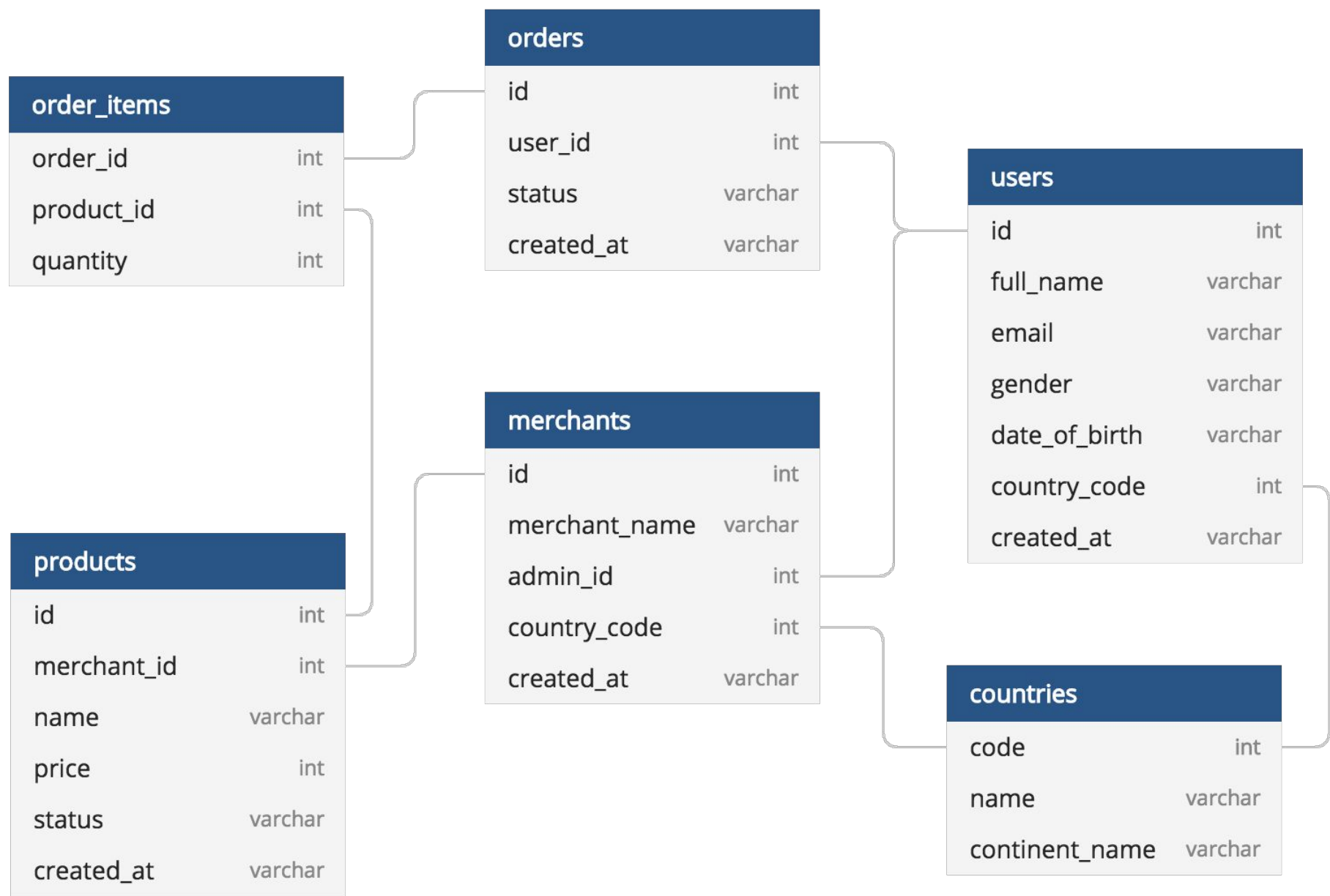
## ¿Qué es una base de datos?

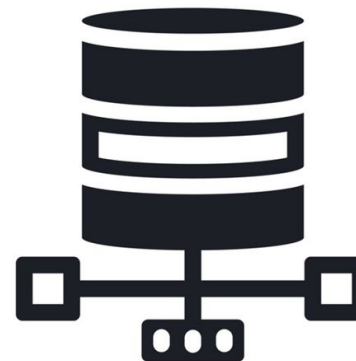
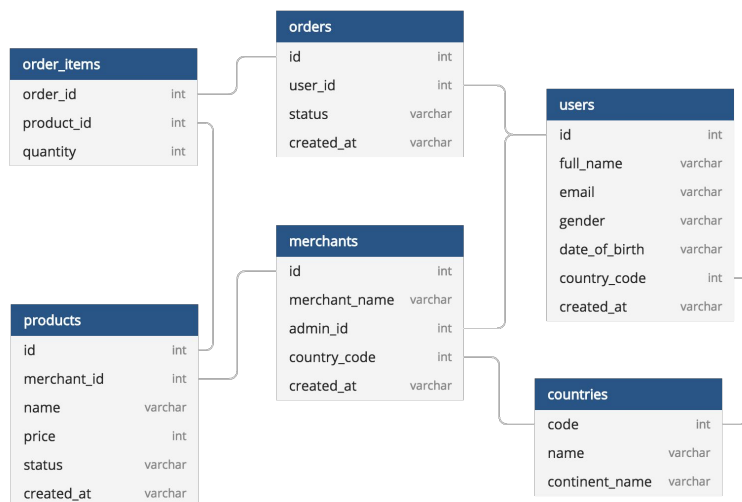
1. Corresponde a un conjunto de datos de un mismo contexto y almacenados bajo cierta **lógica/estructura** e **indexados** para su posterior **uso eficiente**.
2. En el caso de una base **relacional**, es una colección de una o más relaciones entre entidades, donde **cada relación es una tabla** con **filas y columnas**. Ej:
  - Entidades: estudiantes, profesores, cursos y salas
  - Relaciones: registro de matrícula, clases inscritas, uso de salas, etc...
3. Es un software diseñado para ayudar a **mantener y utilizar grandes volúmenes de datos**.



RELATIONAL DATABASE

# Diagrama de una Base de Datos





Esquema de una relación: define el nombre de una relación y, nombres y tipos de dato de sus campos (columnas)

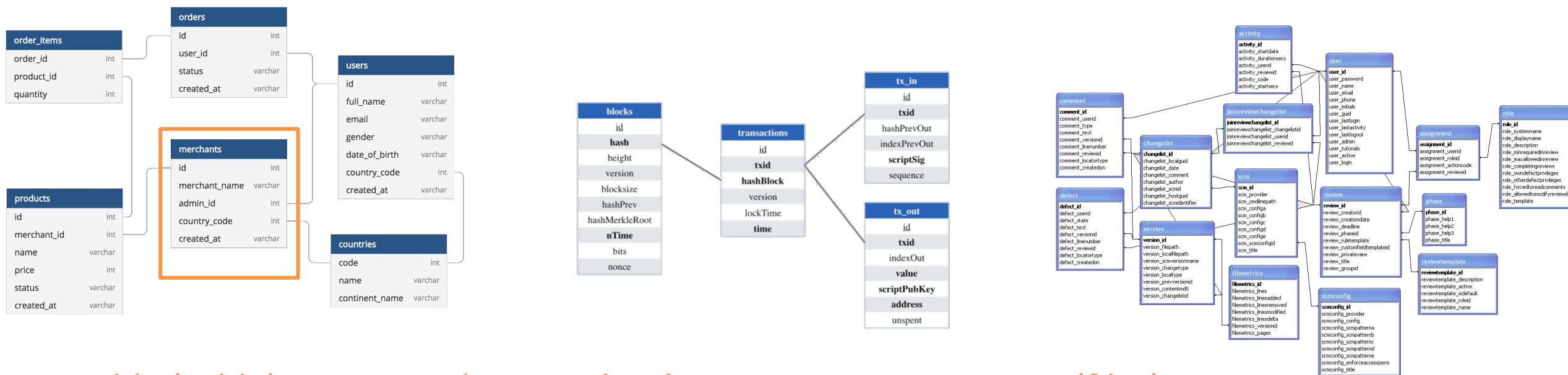
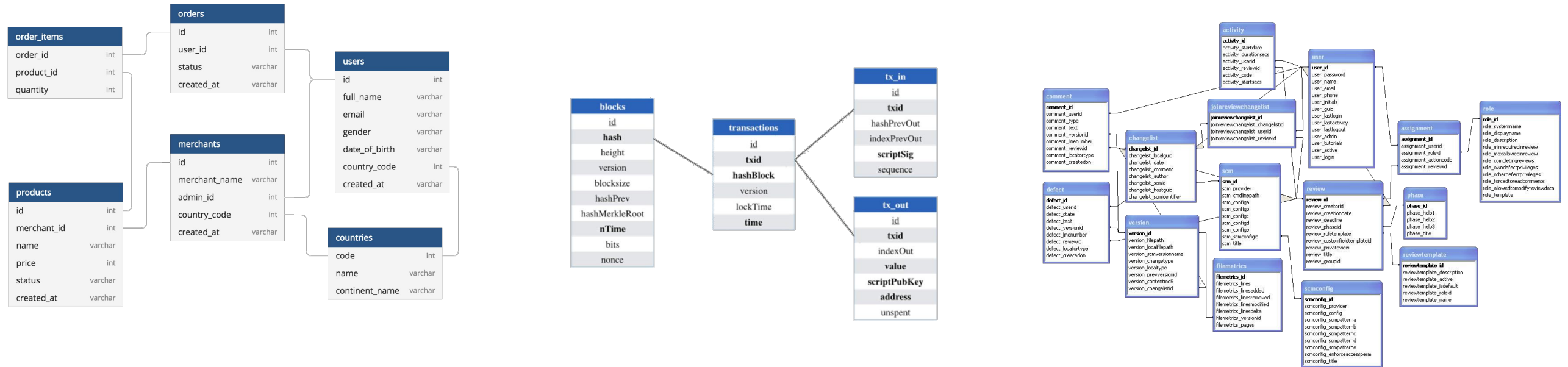
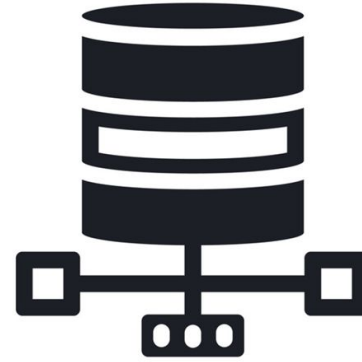
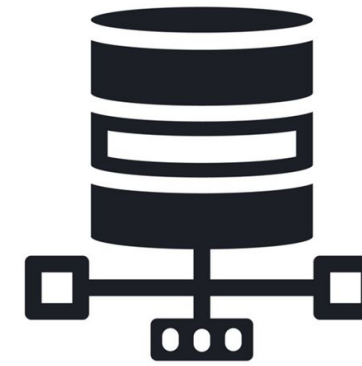
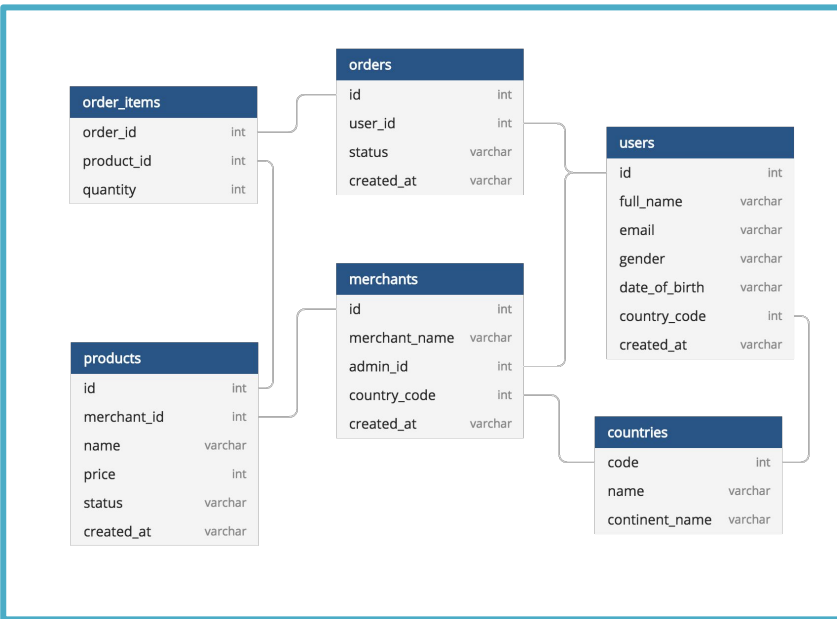


Tabla (Table): instancia de una relación que contiene registros (filas)

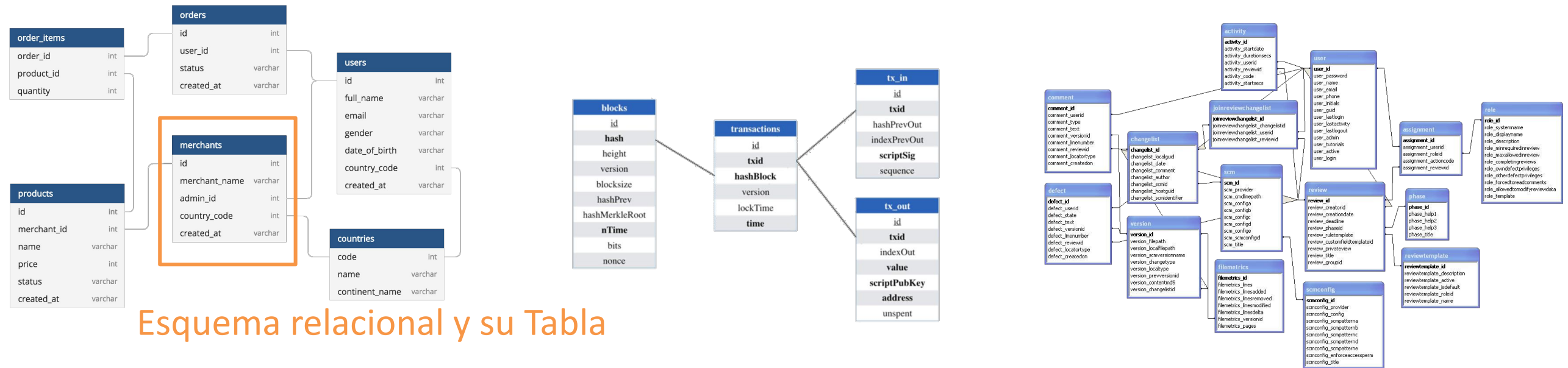




## Base de datos (Database)



## Esquema (Schema) de la base de datos



## Esquema relacional y su Tabla

# Tabla

**Columna: Guarda un tipo específico de datos**

CHAR(N)  
VARCHAR(N)  
INTEGER  
REAL  
...

**Fila:**  
Corresponde a  
un registro o  
instancia

Num_Emp	Nombre	Edad	Department_id	Salario
001	Alex S.	26	1	700000
002	Gerardo C.	32	2	1200000
003	Regina G.	31	2	1200000
004	Juana R.	26	3	520000

Empleados (Num\_Emp: STRING, Nombre: STRING, Edad: INTEGER, Departamento: STRING, Salario: REAL )

# Operaciones principales

## Operaciones sobre una tabla

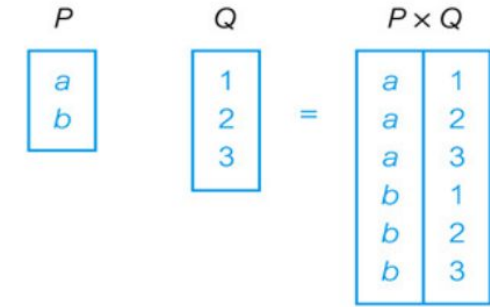
1. **Selección:** filtrar filas
2. **Proyección:** filtrar columnas



(a) Selection



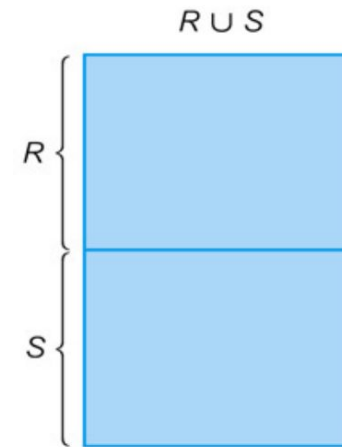
(b) Projection



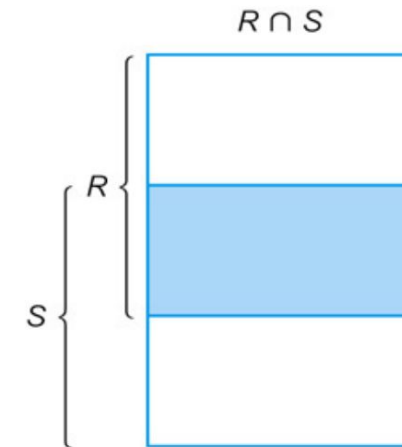
(c) Cartesian product

## Operaciones entre dos tablas

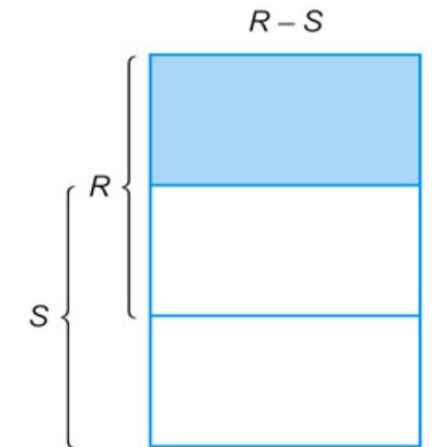
1. **Producto Cartesiano:** obtener todas las combinaciones entre tuplas
2. **Unión:** juntar dos tablas
3. **Intersección:** buscar elementos comunes
4. **Diferencia:** sacar valores que una tabla tiene en común con otra



(d) Union



(e) Intersection



(f) Set difference

La combinación de estas operaciones lleva a operaciones complejas, como un **join**



Num_Emp	Nombre	Edad	Department_id	Salario
001	Alex S.	26	1	700000
002	Gerardo C.	32	2	1200000
003	Regina G.	31	2	1200000
004	Juana R.	26	3	520000

**JOIN**  
Producto  
Cartesiano  
+  
Selección

+

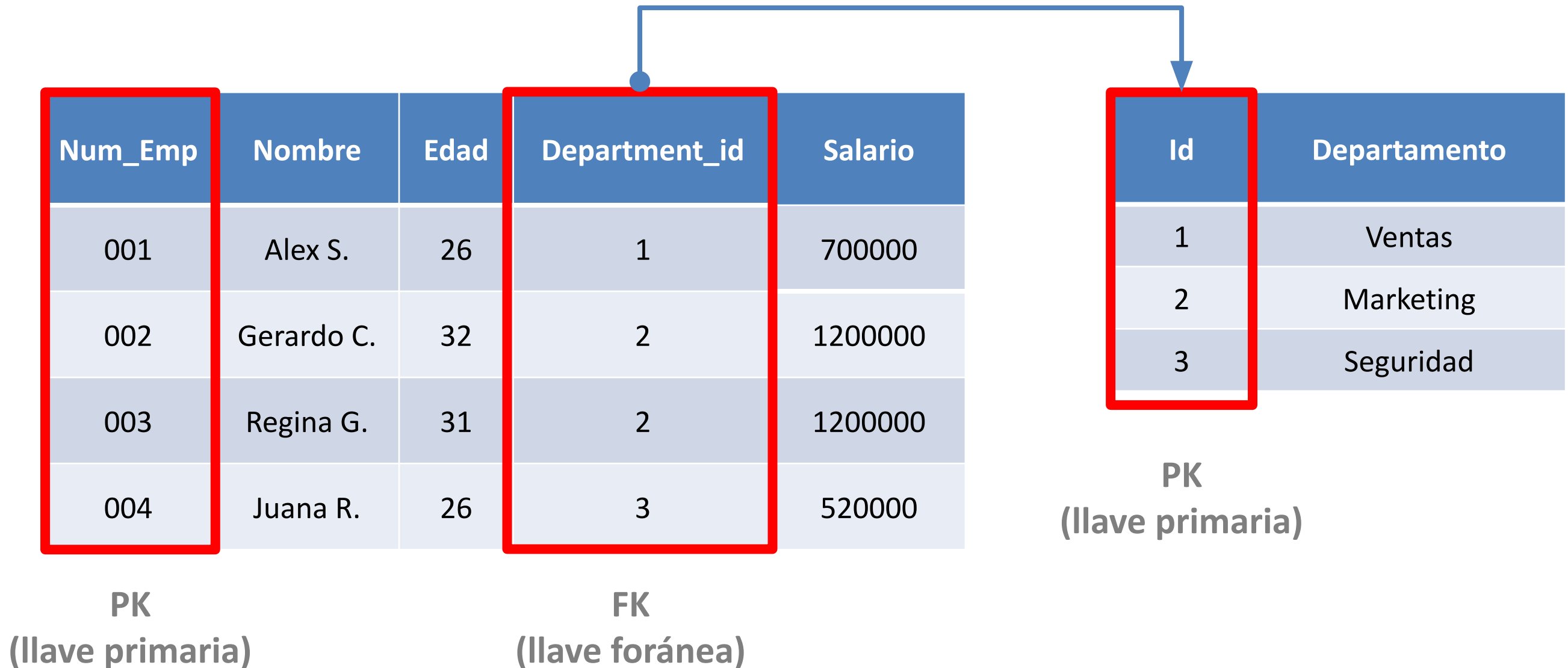
Id	Departamento
1	Ventas
2	Marketing
3	Seguridad

=

Num_Emp	Nombre	Edad	Department_id	Salario	Departamento
001	Alex S.	26	1	700000	Ventas
002	Gerardo C.	32	2	1200000	Marketing
003	Regina G.	31	2	1200000	Marketing
004	Juana R.	26	3	520000	Seguridad

# Referencias entre tablas

Llaves primarias y secundarias (**primary keys y foreign keys**)



## Restricciones de Integridad

```
graph TD; A[Restricciones de Integridad] --> B[Restricciones de Dominio]; A --> C[Restricciones de Integridad]; A --> D[Restricciones de Llave]; A --> E[Restricciones de Llave Foránea];
```

### Restricciones de Dominio

Se relaciona al esquema:

1. Tipo de datos
2. Campos

### Restricciones de Integridad

Delimitan de forma más específica los datos, por ejemplo, rango de valores.

### Restricciones de Llave

Cada tabla debe tener un campo como llave primaria y restringe a que sean valores únicos

### Restricciones de Llave Foránea

Si un campo es una llave foránea, sus valores deben aparecer en la relación a la que se hace referencia

# Structured Query Language (SQL)

Tiene dos componentes principales

- **Lenguaje de definición de datos (DDL)**

- Creación de tablas (CREATE)
- Eliminación de tablas (DROP)
- Modificación de definiciones de tablas (ALTER)

\*Las restricciones de integridad se pueden definir en tablas, ya sea cuando se crea la tabla o posteriormente.

- **Lenguaje de manipulación de datos (DML)**

- *Data Manipulation Language*
- Consultas (SELECT)
- Inserción de datos (INSERT)
- Actualización de datos (UPDATE)
- Eliminar datos (DELETE)



## Creación

**CREATE TABLE [IF NOT EXISTS]** table\_name (column\_1 data\_type, column\_2 data\_type, ...)

Num_Emp	Nombre	Edad	Departamento	Salario

**CREATE TABLE** Empleados (Num\_Emp **CHAR(3)**, Nombre **VARCHAR(20)**, Edad **INTEGER**, Departamento **VARCHAR(10)**, Salario **REAL**)

Se crea la tabla vacía, sin filas...

## Insertión

Ejemplo de agregar la 4ta fila

```
INSERT INTO table_name (column1,column2 ,...) VALUES( value1, value2 ,...)
```

Num_Emp	Nombre	Edad	Departamento	Salario
001	Alex S.	26	Ventas	700000
002	Gerardo C.	32	Marketing	1200000
003	Regina G.	31	Marketing	1200000
004	Juana R.	26	Seguridad	520000

```
INSERT INTO Empleados (Num_Emp, Nombre, Edad, Departamento, Salario) VALUES ('004', 'Juana R.', 26, 'Seguridad', 520000)
```

## Modificación

**UPDATE** table\_name **SET** column\_1 = new\_value\_1, column\_2 = new\_value\_2  
**WHERE** search\_condition

Num_Emp	Nombre	Edad	Departamento	Salario
001	Alex S.	26	<b>Marketing</b>	700000
002	Gerardo C.	32	Marketing	1200000
003	Regina G.	31	Marketing	1200000
004	Juana R.	26	Seguridad	520000

**UPDATE** Empleados E **SET** E.Departamento = 'Marketing' **WHERE** E.Emp\_No = '001'

## Eliminación

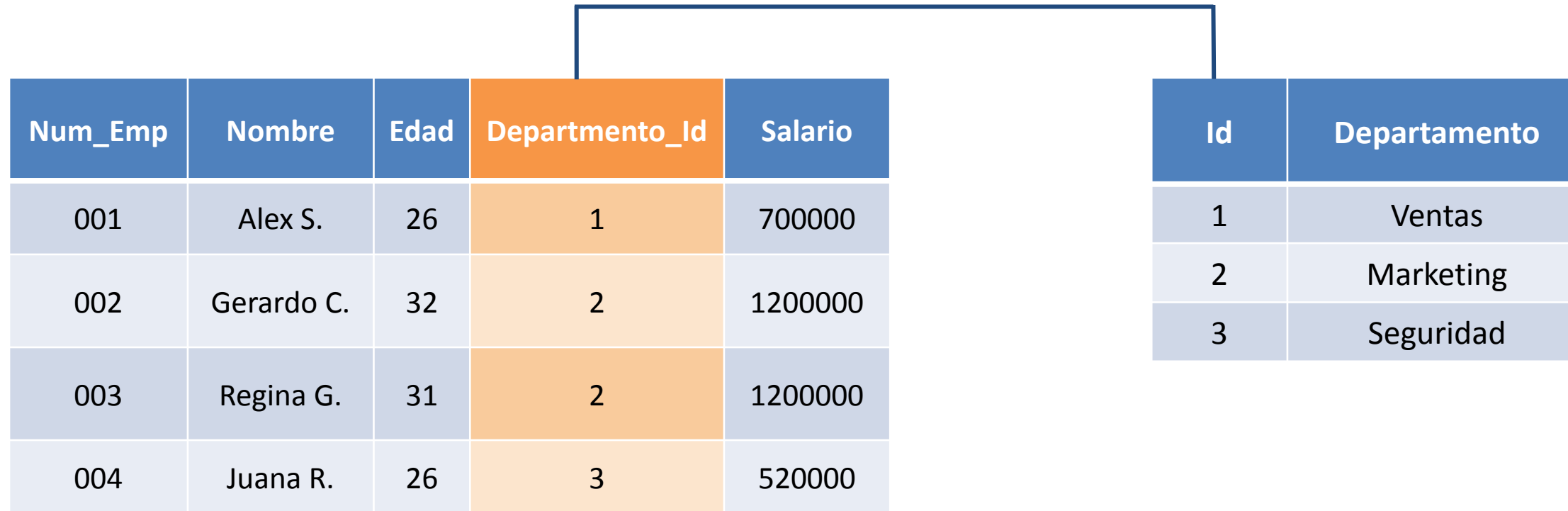
**DELETE FROM** table\_name **WHERE** search\_condition;

Num_Emp	Nombre	Edad	Departamento	Salario
001	Alex S.	26	Ventas	700000
002	Gerardo C.	32	Marketing	1200000
003	Regina G.	31	Marketing	1200000

**DELETE FROM** Empleados E **WHERE** E.Num\_Emp = '004'



## Creación de tablas con *Primary Key* y *Foreign Key*



```
CREATE TABLE Departamentos (Id INTEGER, Departamento VARCHAR(20), PRIMARY KEY(Id))
```

```
CREATE TABLE Empleados (Num_Emp CHAR(3), Nombre VARCHAR(20), Edad INTEGER,  
Departamento_id INTEGER, Salario REAL, PRIMARY KEY(Num_Emp), FOREIGN KEY (Departamento_id)  
REFERENCES Departamentos.Id)
```

## Uso en Python: DDL

```
import sqlite3
```

```
connection = sqlite3.connect('ejemplo.db')  
cursor = connection.cursor()
```

```
sqlStatement = 'CREATE TABLE Empleados (Num_Emp CHAR(3), Nombre VARCHAR(20), Edad INTEGER, Departamento  
VARCHAR(10), Salario REAL)'  
cursor.execute(sqlStatement)
```

```
Sq12 = 'INSERT INTO Empleados (Num_Emp, Nombre, Edad, Departamento, Salario) VALUES ('004', 'Juana R.', 26, 'Seguridad', 520000)'
```

```
cursor.execute(Sq12)
```

```
connection.commit()  
connection.close()
```

## Manejo de errores

Al desarrollar este capítulo, se encontrarán dos tipos de errores:

- Errores de Python (de los que ya están familiarizados)
- Errores de la sintaxis de la base de datos (SQL)

CONSEJO: Pueden testear sus consultas directamente en la base de datos (p.ej., con SQLiteStudio) y luego utilizarla en Python

Pontificia Universidad Católica de Chile  
Escuela de Ingeniería  
Departamento de Ciencia de la Computación



# IIC2115 - Programación como Herramienta para la Ingeniería

Bases de datos relacionales

**Profesora:** Francesca Lucchini  
**Prof. Coordinador:** Hans Löbel