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

IIC2115 - Programación como Herramienta para la Ingeniería

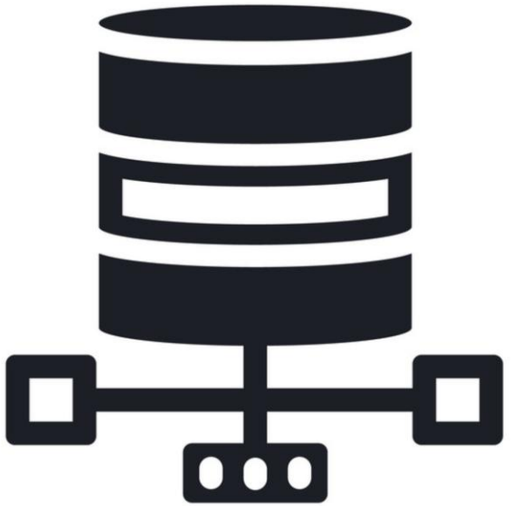
Bases de datos relacionales

Profesor: Hans Löbel

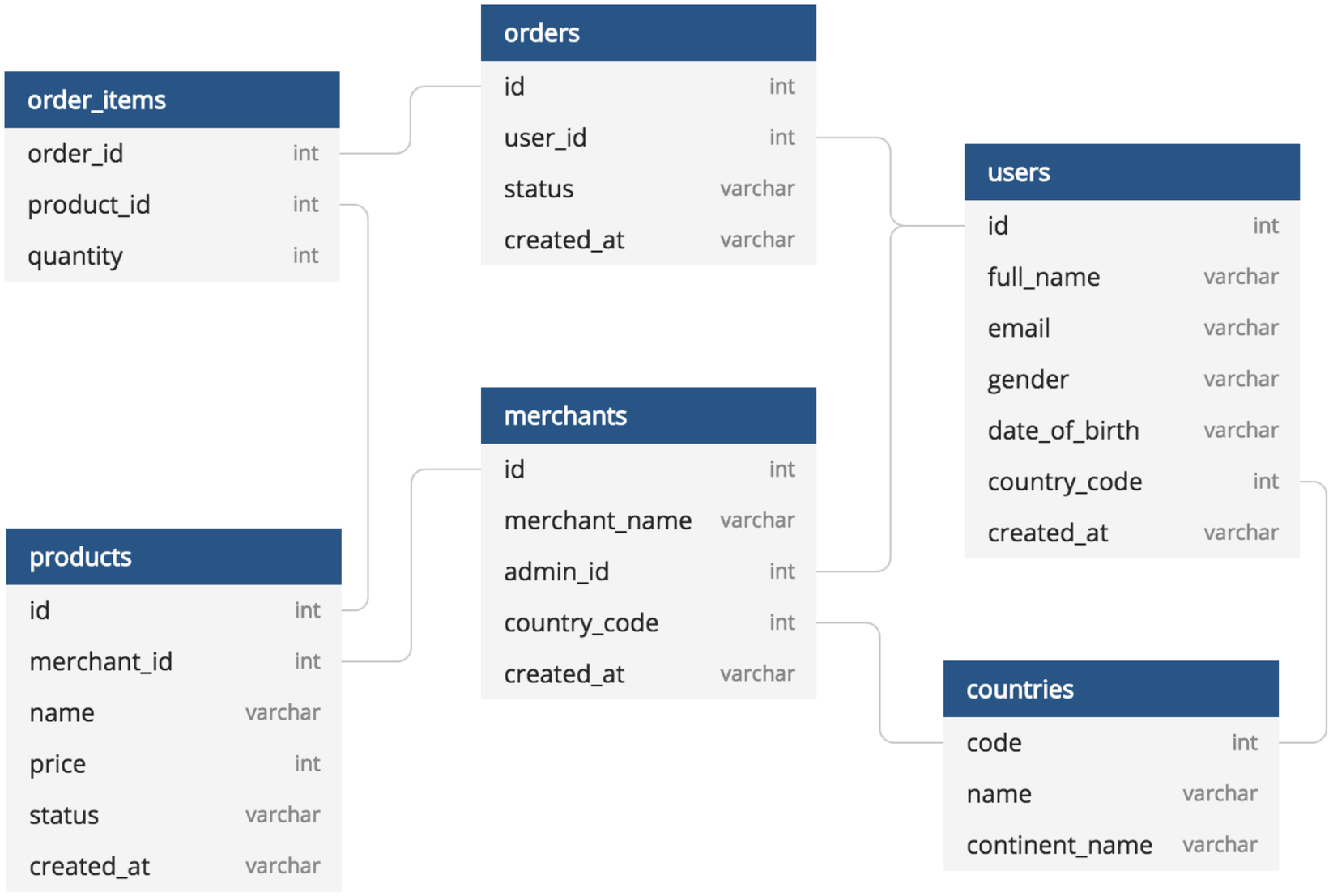
¿Qué es una base de datos?

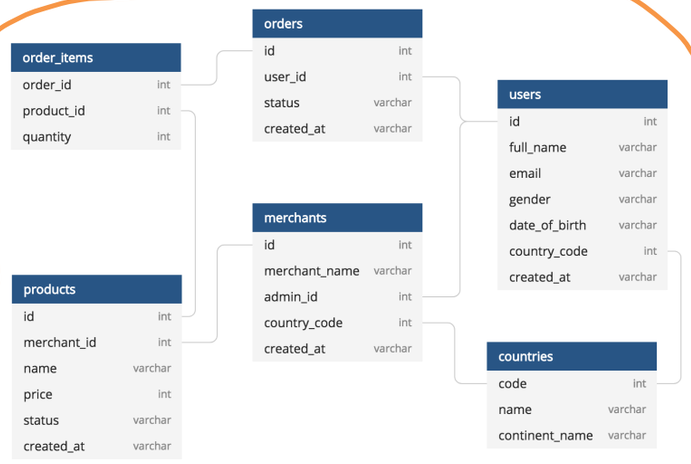
1. Corresponde a un conjunto de datos de un mismo contexto y almacenados bajo cierta lógica/estructura e indexados para su posterior uso eficiente.

2. En el caso de una base relacional, es una colección de una o más relaciones, donde cada relación es una tabla con filas y columnas.



RELATIONAL DATABASE


order_items
order_id int
product_id int
quantity int
orders
id int
user_id int
status varchar
created_at varchar
users
id int
full_name varchar
email varchar
gender varchar
date_of_birth varchar
country_code int
created_at varchar
products
id int
merchant_id int
name varchar
price int
status varchar
created_at varchar
merchants
id int
merchant_name varchar
admin_id int
country_code int
created_at varchar
countries
code int
name varchar
continent_name varchar



Esquema (Schema)

Base de datos (Database)

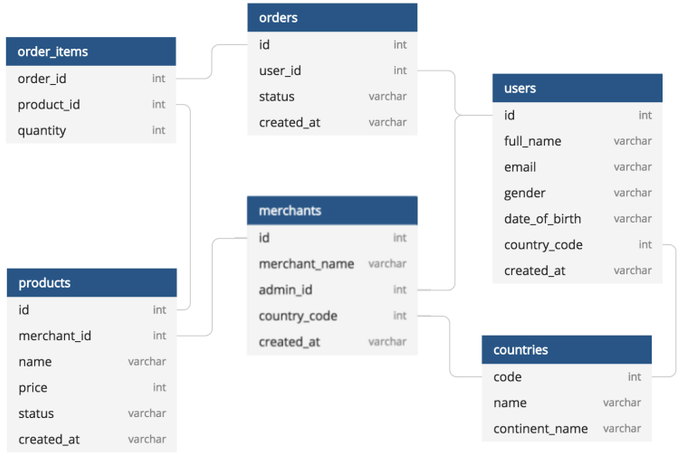
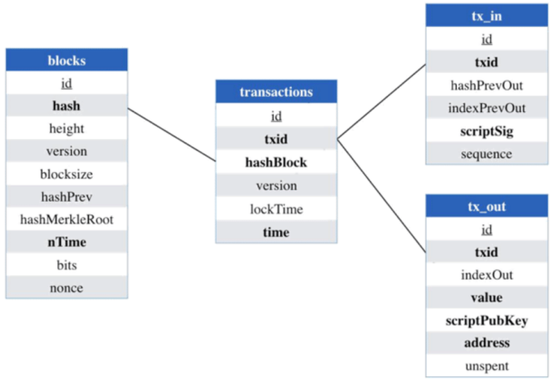
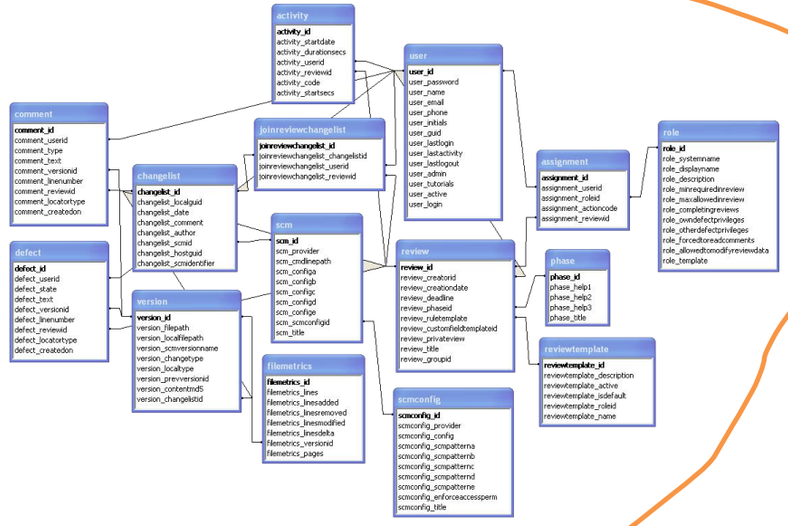


Tabla (Table)





Tabla

Columna: guarda un tipo específico de datos

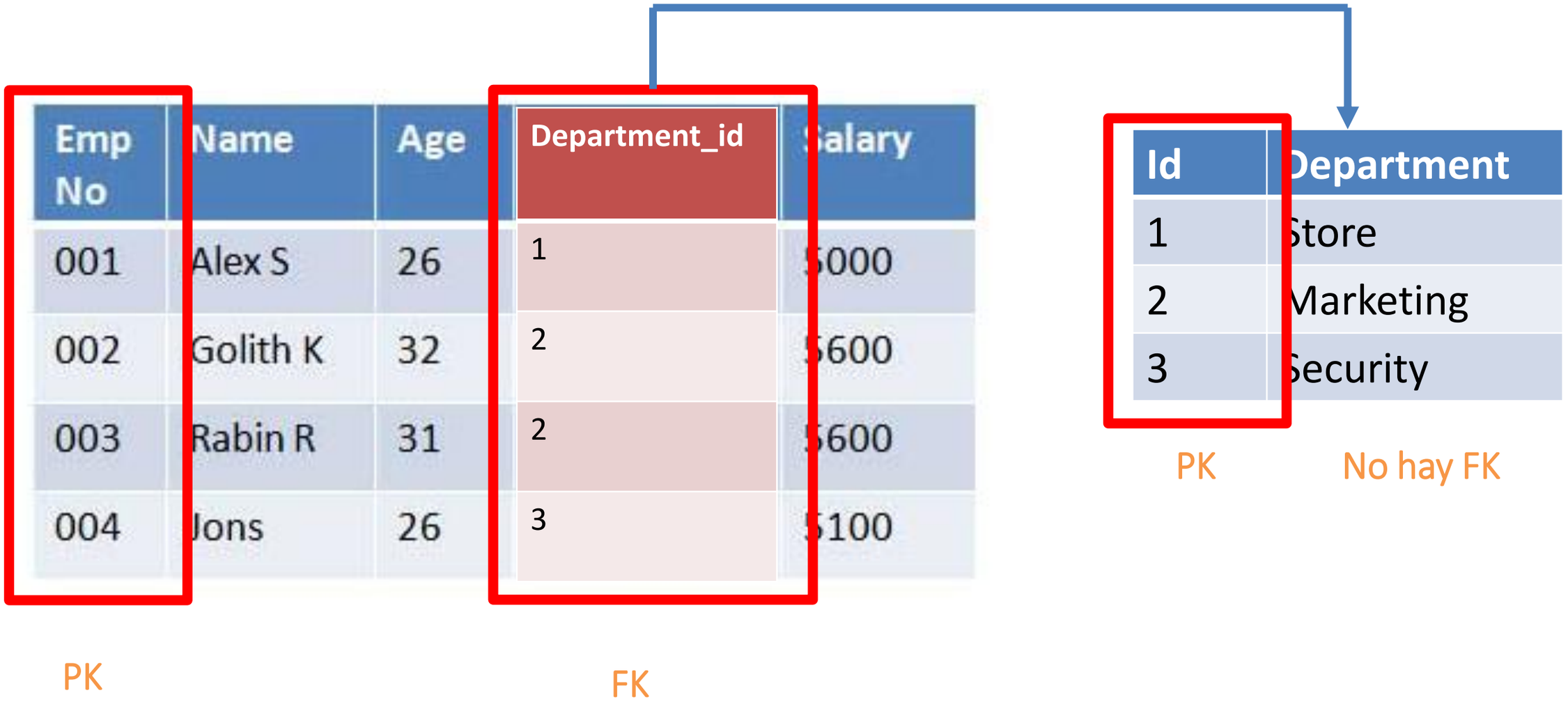
CHAR(N)
VARCHAR(N)
INTEGER
REAL
…

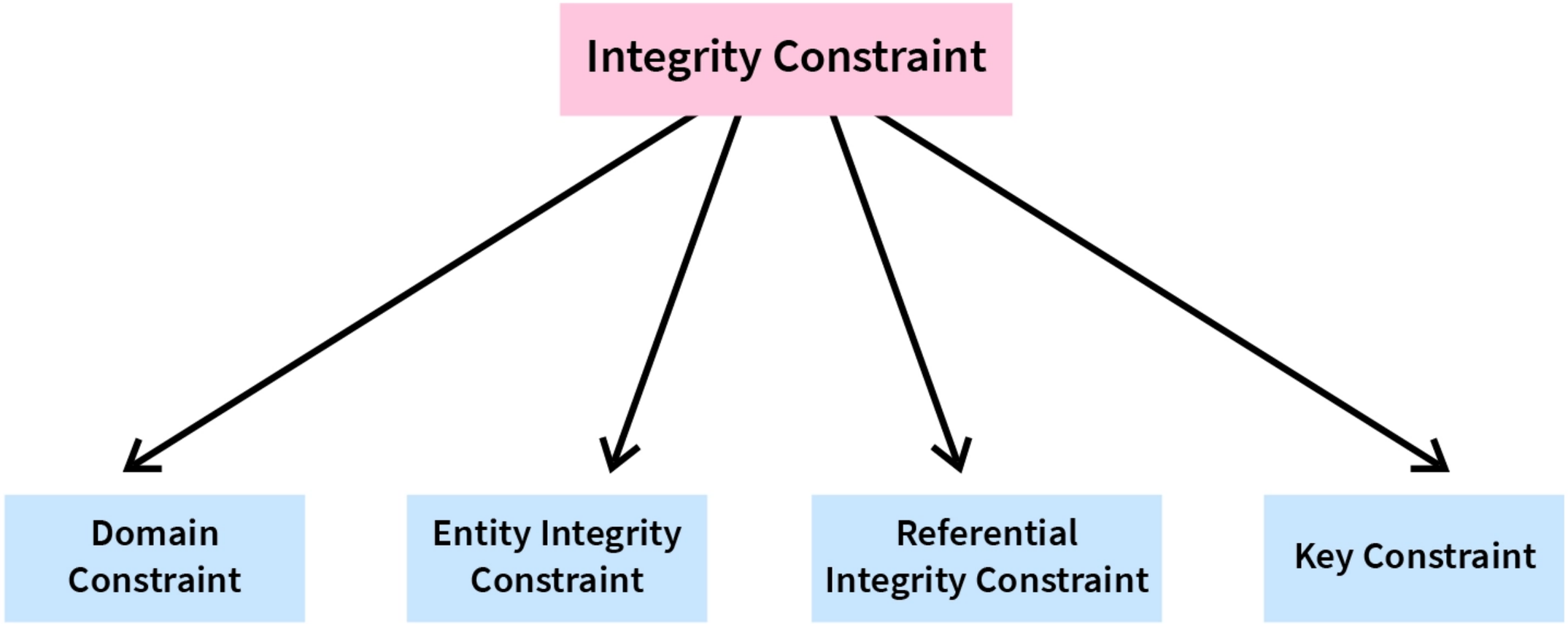
Fila: corresponde a un registro o instancia

Emp No	Name	Age	Department	Salary
001	Alex S	26	Store	5000
002	Golith K	32	Marketing	5600
003	Rabin R	31	Marketing	5600
004	Jons	26	Security	5100

Empleados (Emp No: STRING, Name: STRING, Age: INTEGER, Department: STRING, Salary: REAL)

Llaves primarias y secundarias (primary keys y foreign keys)




Integrity Constraint
Domain Constraint
Entity Integrity Constraint
Referential Integrity Constraint
Key Constraint

Structured Query Language (SQL)

- Lenguage de definición de datos (DDL)
  - Creación
  - Inserción
  - Eliminación
  - Modificación de definiciones de tablas.

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

- Lenguaje de manipulación de datos (DML)
  - Consultas

- En este curso utilizaremos SQLite, una versión reducida de SQL, que tiene algunas diferencias menores en sus sintaxis.

Creación

CREATE TABLE [**IF NOT EXISTS**] table_name (column_1 data_type, column_2 data_type, …)

Emp No	Name	Age	Department	Salary
001	Alex S	26	Store	5000
002	Golith K	32	Marketing	5600
003	Rabin R	31	Marketing	5600
004	Jons	26	Security	5100

CREATE TABLE Empleados (Emp_No **CHAR(3)**, Name **VARCHAR(20)**, Age **INTEGER**, Department **VARCHAR(10)**, Salary **INTEGER**)

Creación (en SQLite)

CREATE TABLE [IF NOT EXISTS] table_name (column_1 data_type, column_2 data_type, ...)

Emp No	Name	Age	Department	Salary
001	Alex S	26	Store	5000
002	Golith K	32	Marketing	5600
003	Rabin R	31	Marketing	5600
004	Jons	26	Security	5100

CREATE TABLE Empleados (Emp_No TEXT, Name TEXT, Age INTEGER, Department TEXT, Salary INTEGER)

Inserción

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

Emp No	Name	Age	Department	Salary
001	Alex S	26	Store	5000
002	Golith K	32	Marketing	5600
003	Rabin R	31	Marketing	5600

INSERT INTO Empleados (Emp_No, Name, Age**,** Department, Salary) **VALUES** (‘004’, ‘Jons’, 26, ‘Security’, 5100)

Modificación

UPDATE table_name **SET** column_1 = new_value_1, column_2 = new_value_2
WHERE search_condition

Emp No	Name	Age	Department	Salary
001	Alex S	26	Store	5000
002	Golith K	32	Marketing	5600
003	Rabin R	31	Marketing	5600
004	Jons	26	Marketing	5100

UPDATE Empleados E **SET** E.Department = ‘Marketing’ **WHERE** E.Emp_No = ‘004’

Eliminación

DELETE FROM table_name **WHERE** search_condition;

Emp No	Name	Age	Department	Salary
001	Alex S	26	Store	5000
002	Golith K	32	Marketing	5600
003	Rabin R	31	Marketing	5600

DELETE FROM Empleados E **WHERE** E.Emp_No = ‘004’

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

Emp No	Name	Age	Department_id	Salary
001	Alex S	26	1	5000
002	Golith K	32	2	5600
003	Rabin R	31	2	5600
004	Jons	26	3	5100

Id	Department
1	Store
2	Marketing
3	Security

CREATE TABLE Departments (Id **INTEGER** **PRIMARY KEY**, Department **VARCHAR(20)**)

CREATE TABLE Empleados (Emp_No **TEXT** **PRIMARY KEY**, Name **TEXT**, Age **INTEGER**, Department_id **INTEGER**, Salary **INTEGER**, **FOREING KEY** (Department_id) REFERENCES Departments)

Uso en Python: DDL

```
import sqlite3

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

sqlStatement = 'CREATE TABLE Empleados (Emp_No TEXT, Name TEXT, Age INTEGER, Department TEXT, Salary INTEGER)'
cursor.execute(sqlStatement)

Sql2 = 'INSERT INTO Empleados (Emp_No, Name, Age, Department, Salary) VALUES ('004', 'Jons', 26, 'Security', 5100)'

cursor.execute(Sql2)

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

Structured Query Language (SQL)

- Lenguaje de definición de datos (DDL)
  - Creación
  - Inserción
  - Eliminación
  - Modificación de definiciones de tablas.

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

- Lenguaje de manipulación de datos (DML)
  - Consultas

Consultas

```
SELECT
        column_list
FROM
        table_list
WHERE
        row_filter
```

Name	Age
Golith K	32
Rabin R	31

SELECT * FROM Empleados

SELECT * FROM Empleados E **WHERE** E.Age > 30

SELECT Name, Age **FROM** Empleados E **WHERE** E.Age > 30

Joins

```
SELECT
        column_list
FROM
        table_list
WHERE
        row_filter
```

Emp No	Name	Age	Department_id	Salary
001	Alex S	26	1	5000
002	Golith K	32	2	5600
003	Rabin R	31	2	5600
004	Jons	26	3	5100

Id	Department
1	Store
2	Marketing
3	Security

SELECT Name **FROM** Empleados E, Departments D **WHERE** E.Department_id = D.id **AND** D.Department = ‘Store’

ALEX S

Anidación

Emp No	Name	Age	Department	Salary
001	Alex S	26	Store	5000
002	Golith K	32	Marketing	5600
003	Rabin R	31	Marketing	5600
004	Jons	26	Security	5100

```
SELECT Name FROM (SELECT Name, Salary FROM Empleados E WHERE E.Age < 30)
WHERE Salary >= 5100
```

JONS

Otras funciones importantes

ORDER BY

GROUP BY – HAVING

IN

COUNT

SUM

AVG

MAX

MIN

Uso en Python: DML

```
connection = sqlite3.connect(‘ejemplo.db’)
cursor = connection.cursor()

sqlStatement = ‘SELECT * FROM Empleados’

cursor.execute(sqlStatement)

una_fila = cursor.fetchone()
todas_filas = cursor.fetchall()

connection.close()
```

Uso en Python: parametrización con strings (forma muy mala)

```
edad = ‘50; DROP TABLE Empleado’

def mayores_que(edad):
    connection = sqlite3.connect(‘ejemplo.db’)
    cursor = connection.cursor()

    sqlStatement = ‘SELECT * FROM Empleados E WHERE E.Age > {}’.format(edad)

    cursor.execute(sqlStatement)
    resp = cursor.fetchall()
    connection.close()
    return resp
```

Uso en Python: parametrización con strings (forma igual de mala, pero más bonita)

```
def mayores_que(edad):
    connection = sqlite3.connect('ejemplo.db')
    cursor = connection.cursor()

    sqlStatement = f'SELECT * FROM Empleados E WHERE E.Age > {edad}'

    cursor.execute(sqlStatement)
    resp = cursor.fetchall()
    connection.close()
    return resp
```

Uso en Python: parametrización sin usar strings (forma correcta)

```
def mayores_que(edad):
    connection = sqlite3.connect('ejemplo.db')
    cursor = connection.cursor()

    sqlStatement = 'SELECT * FROM Empleados E WHERE E.Age > ?'

    cursor.execute(sqlStatement, (edad,))
    resp = cursor.fetchall()
    connection.close()
    return resp
```

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 DB Browser for SQLite) 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

Profesor: Hans Löbel