# Sesión 1_UF2176

## Objetivo General

Dominar el ciclo completo de trabajo con PostgreSQL:

- Preparar entorno,
- Crear base de datos y tablas,
- Importar datos
- Limpiar y normalizar,
- Optimizar con índices,
- Consultar con SQL desde nivel básico hasta avanzado.

## PASO 1 – Preparar entorno

**Objetivo:** Aprender a montar PostgreSQL en Docker con persistencia y pgAdmin.
**Debes demostrar que sabes:**

- Crear un `docker-compose.yml`.
- Levantar contenedores con `docker compose up -d`.
- Acceder a pgAdmin desde navegador.
- Registrar un servidor PostgreSQL en pgAdmin.

---

## PASO 2 – Ficheros CSV

**Objetivo:** Comprender cómo se cargan datasets en PostgreSQL.
**Debes demostrar que sabes:**

- Colocar los CSV en la carpeta compartida (`./backups`)
- Verlos dentro del contenedor en `/backups`.

---

# Sesión 1_UF2176

## PASO 3 – Crear tablas

**Objetivo:** Diseñar tablas con tipos de datos adecuados.
**Debes demostrar que sabes:**

- Crear tablas `libros`, `usuarios`, `ratings`.
- Usar `PRIMARY KEY` y `CHECK`.
- Verificar con `\dt` o desde pgAdmin.

---

## PASO 4 – Importar CSV

**Objetivo:** Cargar datos masivos correctamente.
**Debes demostrar que sabes:**

- Usar `COPY ... FROM '/backups/archivo.csv'`.
- Manejar cabeceras, delimitadores, NULLs.
- Crear tablas temporales para limpiar datos antes de insertarlo

---

## PASO 5 – Limpieza de inconsistencias

**Objetivo:** Garantizar la integridad de los datos.
**Debes demostrar que sabes:**

- Detectar registros huérfanos con `LEFT JOIN`.
- Eliminar o corregir datos inválidos (`DELETE`, `UPDATE`).
- Validar con consultas (`SELECT COUNT(*)`).

---

# Sesión 1_UF2176

## PASO 6 – Relaciones (PK y FK)

**Objetivo:** Aplicar integridad referencial.
**Debes demostrar que sabes:**

- Crear claves primarias (`ALTER TABLE ... ADD CONSTRAINT ... PK`).
- Crear claves foráneas con `ON DELETE CASCADE`.
- Probar integridad insertando datos inválidos y comprobando el error.

## PASO 7 – Índices

**Objetivo:** Optimizar búsquedas y consultas frecuentes.
**Debes demostrar que sabes:**

- Crear distintos tipos de índices:
    - **B-Tree** (búsquedas simples, ORDER BY).
    - **Hash** (igualdad exacta).
    - **GIN** (texto completo).
    - **BRIN** (grandes volúmenes ordenados).
    - **Compuestos** (varias columnas).
- Listar índices con `pg_index` y `pg_class`.

---

## PASO 8 – Medición de rendimiento

**Objetivo:** Evaluar cómo PostgreSQL usa índices.
**Debes demostrar que sabes:**

- Usar `EXPLAIN ANALYZE` para ver `Seq Scan`, `Index Scan` y `Bitmap Index Scan`.
- Comparar consultas antes y después de los índices.
- Medir tamaños de índices (`pg_relation_size`).

---

# Sesión 1_UF2176

## PASO 9 – Batería de consultas

**Objetivo:** Practicar SQL en niveles crecientes.

- **Básico:** SELECT, WHERE, LIMIT, COUNT.
- **Intermedio:** JOIN, GROUP BY, agregaciones.
- **Avanzado:** HAVING, subconsultas.
- **Experto:** CTE, funciones ventana (`RANK`, `AVG`, etc.).
- **Reto final:** Filtrado colaborativo con `CORR()` para encontrar usuarios similares.

# Sesión 1_UF2176

## Competencias finales

Al terminar, deberías ser capaz de:

1. Montar PostgreSQL en Docker con persistencia.
2. Importar datasets desde CSV y limpiarlos.
3. Crear un modelo relacional completo con PK y FK.
4. Diseñar índices adecuados para distintos tipos de consultas.
5. Medir y optimizar rendimiento con `EXPLAIN ANALYZE`.
6. Resolver consultas SQL desde básicas hasta analíticas avanzadas.

PostgreSQL

# Sesión 1_UF2176

## PASO 1 – Preparar entorno (Docker + pgAdmin)

Levantar PostgreSQL 15 y pgAdmin con Docker Compose, con persistencia de datos y acceso desde el navegador.

Òscar Burgos

@mihifidem 7/25

LAB

COLECCION

# Sesión 1_UF2176

## Archivo `docker-compose.yml`

Guarda esto en un archivo llamado `docker-compose.yml`:

```
version: "3.9"
services:
  db:
    image: postgres:15
    container_name: postgres_curso
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: curso_ifcd0112
      POSTGRES_INITDB_ARGS: "--encoding=UTF-8 --locale=en_US.utf8"
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data
      - ./init:/docker-entrypoint-initdb.d
      - ./backups:/backups
    restart: always
    networks:
      - dbnet

  pgadmin:
    image: dpage/pgadmin4:latest
    container_name: pgadmin_curso
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@example.com
      PGADMIN_DEFAULT_PASSWORD: admin
    ports:
      - "8080:80"
    depends_on:
      - db
    restart: always
    networks:
      - dbnet

volumes:
  pgdata:

networks:
```

# Sesión 1_UF2176

```
  dbnet:
    driver: bridge
```

# Sesión 1_UF2176

## PASO 3: Levantar los contenedores

Ejecuta en la carpeta:

```
docker compose up -d
```

- Esto descarga imágenes y arranca contenedores.

Verifica con:
`docker ps`

- Debes ver `postgres_curso` y `pgadmin_curso`.

---

## PASO 4: Acceder a pgAdmin

1. Abre navegador → http://localhost:8080
2. Login:
    - Email: `admin@example.com`
    - Pass: `admin`
3. Crear un servidor nuevo en pgAdmin:
    - **Nombre**: CursoDB
    - **Host**: `db` ✅ (porque están en la misma red Docker, no `localhost`)
    - **Usuario**: `postgres`
    - **Contraseña**: `postgres`

# Sesión 1_UF2176

Òscar Burgos

@mihifidem 7/25

LAB
COLECCION

# Sesión 1_UF2176

docker-compose.yml

```yaml
version: "3.9"

services:
  db:
    image: postgres:15
    container_name: postgres_curso
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: curso_ifcd0112
      POSTGRES_INITDB_ARGS: "--encoding=UTF-8 --locale=en_US.utf8"
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data
      - ./init:/docker-entrypoint-initdb.d
      - ./backups:/backups
    restart: always
    networks:
      - dbnet

  pgadmin:
    image: dpage/pgadmin4:latest
    container_name: pgadmin_curso
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@example.com
      PGADMIN_DEFAULT_PASSWORD: admin
    ports:
      - "8080:80"
    depends_on:
      - db
    restart: always
    networks:
      - dbnet

volumes:
  pgdata:
```

```
networks:
  dbnet:
    driver: bridge
```

### 2. Crear el script de inicialización

---

### 3. Levantar el contenedor

### 4. Verificar en pgAdmin

# PASO 2 – Fichero csv

PostgreSQL

## Sesión 1_UF2176

# PASO 3 – Crear las tabla

Òscar Burgos

@mihifidem 7/25

LAB

COLECCION

PostgreSQL

Sesión 1_UF2176

# PASO 3 – Copiar los CSV al contenedor

Òscar Burgos

@mihifidem 7/25

LAB
COLECCION

PostgreSQL

## Sesión 1_UF2176

# PASO 4 – Importar CSV en PostgreSQL

Queremos cargar los ficheros `books.csv`, `users.csv`, `ratings.csv` en las tablas `libros`, `usuarios` y `ratings`.

## Sesión 1_UF2176

# PASO 5 – Limpieza de inconsistencias

Ya cargamos los datos, pero hay problemas posibles:

1. En `ratings` hay ISBN que **no existen** en `libros` (ratings “huérfanos”).
2. En `usuarios`, la columna `edad` podría tener valores nulos o raros.

Vamos a limpiarlos para poder crear las **relaciones** en el siguiente paso.

---

### 1. Detectar ratings con ISBN inexistente

---

# Sesión 1_UF2176

## 2. Solución A – Eliminar ratings huérfanos

Si quieres un modelo **estricto** (solo ratings de libros que existen):

---

## ✅ 3. Solución B – Insertar placeholders en `libros`

Si quieres **mantener todos los ratings** aunque falte info del libro:

---

## 4. Limpiar edades en `usuarios`

Aseguramos que todas las edades sean enteros o `NULL`:

---

# Sesión 1_UF2176

## 5. Verificar limpieza

## Sesión 1_UF2176

# PASO 6 – Relaciones (Primary Keys y Foreign Keys)

### ✅ Enunciado

Definiremos las claves primarias y las relaciones entre las tablas:

- `usuarios.user_id` será **PRIMARY KEY**.
- `libros.ISBN` será **PRIMARY KEY**.
- `ratings` tendrá dos **FOREIGN KEYS**:
    - `ratings.user_id` → referencia a `usuarios.user_id`
      `ratings.ISBN` → referencia a `libros.ISBN`

Además, usaremos `ON DELETE CASCADE` para que, si borras un usuario o libro, también se borren automáticamente sus valoraciones.

---

### Verificación

En pgAdmin → en la pestaña de **Properties** de cada tabla, ya deberías ver:

- `usuarios` con PK en `user_id`.
- `libros` con PK en `ISBN`.
- `ratings` con dos FK (`user_id`, `ISBN`).

# PASO 7 – Índices en PostgreSQL

Vamos a crear varios **tipos de índices** (B-Tree, Hash, GIN, BRIN) en nuestras tablas `usuarios`, `libros` y `ratings` para mejorar el rendimiento.

---

## 1. Índices B-Tree (por defecto)

- Los más comunes.
- Sirven para búsquedas por igualdad (=), rangos (>, <, `BETWEEN`) y `ORDER BY`.

# Sesión 1_UF2176

Òscar Burgos

@mihifidem 7/25

LAB
COLECCION

# Sesión 1_UF2176

## 2. Índices Hash

- Especializados en comparaciones por igualdad =.
- Útiles si haces muchas búsquedas exactas.

# Sesión 1_UF2176

## 3. Índices GIN (texto completo, arrays, JSONB)

- Muy útiles para búsquedas de texto (`LIKE`, `ILIKE`, búsqueda por palabras).

**GIN (Generalized Inverted Index)** es un índice “invertido” ideal para datos que contienen **múltiples valores por fila** o que se **tokenizan** (palabras, elementos de arrays, claves/valores en JSONB).

Ñ

# Sesión 1_UF2176

## 4. Índices BRIN (grandes volúmenes de datos ordenados)

- Ocupan muy poco espacio.
- Buenos para datos correlacionados en bloques (ej: fechas, IDs secuenciales)

Ñ

# Sesión 1_UF2176

## 5. Índices compuestos

- Mejoran consultas que combinan varias columnas.

-

---

# Sesión 1_UF2176

**Resumen rápido**

- **B-Tree** → versátil, comparaciones, rangos, orden.
- **Hash** → especializado en = (igualdad exacta).
- **GIN** → texto completo, arrays, JSONB.
- **BRIN** → big data, columnas ordenadas (fechas, IDs secuenciales).

## Verificación

Puedes listar todos los índices de tu base de datos con:

```
SELECT relname AS indice,
       relkind,
       pg_size_pretty(pg_relation_size(indexrelid)) AS tamano
FROM pg_index
JOIN pg_class ON pg_class.oid = pg_index.indexrelid
ORDER BY pg_relation_size(indexrelid) DESC;
```

## Sesión 1_UF2176

# PASO 8 – Medir rendimiento con `EXPLAIN ANALYZE`

Ejecutaremos consultas típicas **antes y después de crear índices**.

- Antes: debería aparecer `Seq Scan` (escaneo secuencial de toda la tabla).
- Después: debería aparecer `Index Scan` o `Bitmap Index Scan` (usa índice).

---

### 0. Desactivar indices y activarlos de nuevo

## 1. Buscar un usuario por `user_id`

-

---

## 2. Buscar un libro por `ISBN`

-

---

## 3. Buscar libros de Tolkien (texto)

---

## 4. Promedio de ratings por libro

---

## 5. Buscar usuarios por rango de edad

# Sesión 1_UF2176

Òscar Burgos

@mihifidem 7/25

LAB
COLECCION

# Sesión 1_UF2176

## 6. Ver tamaño de índices

-

# Sesión 1_UF2176

Òscar Burgos

@mihifidem 7/25

LAB
COLECCION

## Sesión 1_UF2176

# PASO 9 – Batería de consultas

---

## 🟢 Bloque 1 – Nivel Básico (lectura simple)

### Ejercicio 1: Mostrar todos los usuarios

**Enunciado:** Obtén todos los registros de la tabla `usuarios`.

---

### Ejercicio 2: Mostrar los primeros 10 libros

**Enunciado:** Obtén los primeros 10 libros registrados en la tabla `libros`.

---

# Sesión 1_UF2176

## Ejercicio 3: Buscar un usuario concreto

**Enunciado:** Encuentra el usuario con `user_id = 12345`.

---

## Ejercicio 4: Contar usuarios

**Enunciado:** ¿Cuántos usuarios hay registrados?

---

## Ejercicio 5: Libros publicados en 2000

**Enunciado:** Lista los títulos de libros publicados en el año 2000.

# Sesión 1_UF2176

## 🟡 Bloque 2 – Nivel Intermedio (JOINs y agregaciones)

### Ejercicio 6: Top 10 libros con más ratings

**Enunciado:** Encuentra los 10 libros que más valoraciones tienen.

---

### Ejercicio 7: Promedio de rating por libro

**Enunciado:** Calcula el promedio de rating de cada libro.

---

### Ejercicio 8: Edad promedio de usuarios que valoran

**Enunciado:** ¿Cuál es la edad promedio de los usuarios que dieron ratings?

### Ejercicio 9: Autores más valorados

**Enunciado:** Muestra los 5 autores con más valoraciones.

---

### Ejercicio 10: Usuarios de USA

**Enunciado:** Muestra los usuarios cuya `location` contenga "usa".

---

# Sesión 1_UF2176

## 🔴 Bloque 3 – Nivel Avanzado (subconsultas y HAVING)

### Ejercicio 11: Mejor libro (mínimo 50 ratings)

Ñ

### Ejercicio 12: Usuarios con más de 100 valoraciones

# Sesión 1_UF2176

## Ejercicio 13: Libros más populares entre <25 años

Òscar Burgos

@mihifidem 7/25

PostgreSQL

# Sesión 1_UF2176

## Ejercicio 14: Autores con promedio > 8

Òscar Burgos

@mihifidem 7/25

LAB

COLECCION

PostgreSQL

# Sesión 1_UF2176

## Ejercicio 15: Usuarios sin ratings

---

Òscar Burgos

@mihifidem 7/25

LAB

COLECCION

# Sesión 1_UF2176

## 🔵 Bloque 4 – Nivel Experto (CTE, funciones ventana)

### Ejercicio 16: Ranking de usuarios por cantidad de ratings

### Ejercicio 17: Promedio de ratings por década

### Ejercicio 18: Usuario más joven y más viejo

### Ejercicio 19: Top usuarios por autor

### Ejercicio 20: Promedio de ratings por país

---

# Sesión 1_UF2176

## Reto Final – Difícil (colaborative filtering)

**Enunciado:** Encuentra los 3 pares de usuarios más similares según sus ratings de libros en común.

**Solución:**