11 Abril 2025 · bases de datos

Ficheros que están ordenados y permiten la inserción de datos gestionados mediante un software.
Datos e información: los datos son información en crudo (365€). La información estructura los datos para la toma de decisiones.
Sistema: es un conjunto de elementos que trabajan juntos para un determinado fin. La sinergia es cuando todos los elementos trabajan juntos y consiguen un resultado final mayor.
Un sistema de información son elementos que intervienen en la gestión de la información para manejar una empresa(recursos físicos, recursos humanos, protocolos).
un sistema de información electrónico gestiona el sistema de información de una empresa(datos, software, hardware, recursos humanos).

Hay varios modelos de bases de datos (jerárquicos, relacionales,en red, orientada a objetos, objeto-relacionales, no SQL).
De los relacionales, surgieron los primeros DBMS (Database Management System)
Funciona con la estructura cliente servidor (servidor escucha desde un puerto, cliente pide servicio y el servidor devuelve dato. tanto servidor como cliente utilizan el mismo sistema de manejo de base de datos(como ejemplo Oracle))
SGDB(software development data base) son sistemas de gestión de bases de datos.
algunos software son:

- My SQL
- Mongo DB
- Postgres
- Oracle
- DB2 (IBM)
- Natural

SQL (structure query language)
Es un lenguaje estándar de consulta a bases de datos relacionales.

Ejemplar: es una instantánea de la base de datos.
Esquema:diseño de la base de datos.

Objetivos de un sistema gestor (DBMS):
debe controlar permamnentemente:

1. redundancia e inconsistencia de datos
2. Accesoa los datos (usuarios,roles)
3. Aislamiento de datos (independencia con las apps) =evita dependencia de las apps
4. Acceso concurrente=evita acceso y consistencia en los datos(dos clientes trabajando sobre el mismo dato)
5. Seguridad (permisos)
6. Atomicidad = tiene que ver con transacciones. Considera que hasta que no es realizada no se efectúa el dato(rollback cuando no es efectivo y commit cuando sí).
7. Integridad

El modelo relacional:

1. organiza los datos en tablas (filas y columnas)
2. Las tablas pueden estar relacionadas entre sí.
3. tine base en la teoría de los conjuntos
4. permite gran separacion entre
5. -
6. -
7. -Mirar diapo de pablo

Archivos

1. indexados
2. encadenados
3. indexados-encadenados

SQL

1. **declaracion**: (metadatos:los datos de los datos(tablas,atributos,restricciones,vistas,etc)
2. **manipulacion**:(manipula los datos(consultas, inserciones,modificaciones,eliminaciones))
3. **consulta**: (busca datos)

Funciones del DBMS

- definicion de datos
- Manipulacion de datos
- Seguridad (usuarios, permisos,roles)
- Recuperación(rollback)
- Integridad
- Concurrencia(multiusuarios)
- Diccionario de datos (definición de metadatos)

Perfiles que interactuan con las bases de datos

Administrador de datos(DA): decide que datos se van a almacenar (suele ser el cliente).
Administrador de base de datos (DBA): técnico responsable de implementar decisiones del DA, instalación,configuración y administración del DBMS.

Arquitecturas

1. **Arquitectura centralizada (mainframe):** se accede con terminales que solo muestran el proceso del mainframe
2. **Arquitectura cliente servidor (2 capas, 3 capas)**:*2 capas* (cliente y servidor,aplicacion), *3capas*(cliente,servidor de aplicación(cliente pesado), capa servidor de bases de datos).
3. **Arquitectura distribuida** (en varios ordenadores y un DDBMS(varios servidores conectados y una base lógica distribuida)

21 abril 2025 Modelos entidad relación

Modelos conceptuales de bases de datos
son independientes de cómo se van a guardar los datos. Son el primer paso para describir la base de datos.
Modelo Entidad-relación(E/RM)
permite describir una base de datos conceptualmente, independiente al gestor de base de datos, representando las entidades y sus relaciones.
Las entidades pueden poseer diferentes atributos (dentro de CFTIC sería alumnos, profesores, cursos)
Las relaciones también tienen atributos.
El diseño conceptual permite diferentes soluciones válidas, descubrir las entidades es la principal tarea del diseño de esquemas.

Una **entidad** son los entes con las mismas propiedades (personas, alumnos, productos…), se encontrarán como sustantivos(Las relaciones por contra se encontraran como verbos)
Siguiendo el modelo DER, las entidades se representarán con un rectángulo con el nombre dentro.Puede haber entidades dependientes que son las que no tienen sentido si otra entidad dominante desaparece(si se elimina una persona la entidad teléfono no tiene sentido por sí misma) , En el DER lleva o un marco doble o una flecha apuntando hacia ella.

Las **relaciones** representan asociaciones entre entidades.(verbos:cursar,vender…)El cliente suele determinar esta realidad.En el sistema DER se representan con un rombo con el verbo dentro.
Los tipos de relación:
binarias: implica que hay dos entidades (son las más habituales).
ternarias: intervienen tres entidades (se podrían simplificar en dos)
enearias: intervienen más de tres entidades.(muy poco comunes)
dobles: más de una relación entre las mismas entidades(empleados, qué empleados son jefes).
reflexiva: una relación sobre la misma entidad.
La cardinalidad de las relaciones indicará la cantidad de ejemplares de una entidad que intervienen en la relación:
La **cardinalidad mínima** es el número mínimo de asociaciones. Se indica 0 a 1 aunque el mínimo sea mayor a uno.Cogemos un ejemplar de la entidad y se consulta primero qué relación puede haber(puede ser 0 o no?, si un jugador puede no jugar en ningún equipo), luego la pregunta sería con respecto a la otra entidad si puede de la misma manera ser 0(cuantos jugadores tiene el equipo?en este caso n pero como mínimo 1).
La **cardinalidad máxima** es el número máximo de asociaciones. 1 o n.
En el DER se indica de varias formas (número 0,n o 1,1…el que sea la cardinalidad al lado de la entidad destino.

Los **atributos** describen las entidades y las relaciones. En el DER se representan con elipses unidas con una línea a la entidad o relación.

Podría haber atributos **compuestos** que se pueden descomponer en otros más simples. También puede haber **múltiples** en el que un solo atributo puede tener varios valores(por ejemplo con un número de teléfono para una persona) En el DER se puede representar con doble línea o como la cardinalidad.
Los hay **opcionales** que pueden ser vacíos o ausencia de valor que se indica con null.En el DER se puede indicar con cardinalidad o líneas discontinuas.
Toda entidad tiene un **ID o clave**. Son uno o varios atributos que son obligatorios (no pueden ser null) y referencian la entidad sin duplicidades, generalmente se genera un atributo que controlaremos nosotros (ej: el número de alumno que da una universidad, lo controla la universidad y garantiza el control.). En el DER se identifica subrayado.
Podría haber identificadores candidatos pero son denominados alternativos. En el DER los alternativos son subrayado discontinuo.

DER/ejercicios (Pyme comercial)
Orden en diagrama:

1. identificar entidades
2. identificar relaciones
3. cardinalidad (colocadas en los opuestos)
4. atributos (*relaciones*(suelen aparecer en las que tienen la cardinalidad **n a n**, si no, seguramente no irá en la relación si no en la entidad e irá en el lado del n) y *entidades*).

24 abril 2025 Dependencias

Entidades dependientes
cuando hay atributos dependientes en una entidad se podrían convertir a una entidad.Esta nueva entidad genera una dependencia a raíz de la generada(ej: una entidad jugador tiene un atributo teléfonos, pero al poder tener más de uno generará una nueva entidad para no condicionar un espacio para el atributo en jugador. El teléfono como entidad será dependiente de jugador).

Dependencia por existencia
Son las que tienen en la cardinalidad 1,1. Se suelen interpretar con las máximas de la cardinalidad y para borrar la entidad dependiente se generan comportamientos en cascada ya que si eliminamos la dependencia a la que se refieren su existencia no tiene sentido ya que habría datos sin asociar.

Dependencia por identificación
Funcionan igual que las que son por existencia pero no tienen que tener un ID principal.La entidad dependiente tendrá un atributo discriminante ya que no hay un atributo principal.La suma del discriminante + el id del atributo de la entidad a la que hacen referencia serán la representación de esta dependencia.

Relaciones de herencia (IsA = es a)
Agrupan los atributos comunes en varias entidades dentro de *una entidad* que *se relaciona con todas* ellas. Se encuentran desde los hijos hacia la nueva entidad(se llamaría realmente

una relación por generalización) que surge, al revés que la herencia que se produce de padres a hijos.Las cardinalidades en estas relaciones son de 1,1.Los identificadores siempre estarán en la dependencia que se relacion con todas las demás.
Puede haber **solapamiento** si existe una relación desde la entidad “padre” que se extienda a algunas entidades (no todas/todas).
La **parcialidad** se da si la entidad puede no tener relación con las entidades hijas.

25 abril 2025 Relaciones ternarias

Relaciones ternarias
Se plantea con tres entidades con una sola relación (ej: profesores,asignaturas y cursos).
Para las **cardinalidades** se deben coger 2 ejemplares y su relación con la tercera entidad (ej: el profesor Juan que da matemáticas en cuántos cursos lo da?))Las cardinalidades **mínimas siempre serán 0**(cardinalidad mínima de Chen) porque siempre habrá una entidad que no tendrá relación.

28 abril 2025 Cardinalidades

Modelo relacional, es un modelo conceptual.
Se basa en la teoría de conjuntos. Los datos se agrupan en relaciones denominadas tablas que agrupan datos referidos a una entidad de forma organizada.
Conjuntos: explícito, implícito,diagrama de Benn,conjunto vacío…
Operaciones de conjuntos:
Unión,diferencia,intersección,complemento y cartesiano(se centra en el orden(pares ordenados)).

Las **cardinalidades máximas indican** en qué tabla estará la **foreign key** y se representarán con la primary key de la otra tabla.
Las **cardinalidades mínimas, si son 0** es que **admite null**, si es **1** sera **not null**

6 mayo 2025 Modelo relacional

Restricciones de integridad
Son condiciones de obligado cumplimiento orientadas a mantener la integridad de datos.
Pueden ser inherentes o semánticas.

Del DER al modelo relacional
cada entidad es una tabla
los atributos seran columnas
los atributos compuestos se descomponen
el atributo id sera el primary key
las claves alternativas serán unique

Relaciones 1-n (de las cardinalidades máximas)

La foreign key siempre irá del lado de n.
se tiene que hacer la restricción referencial constraint foreign key.
La cardinalidad mínima del lado 1 se definirá como not null si es 1 o null si es 0.
Podemos crear un indice que es cuando se indexa para apuntar a esa referencia concreta y evitar una busqueda secuencial. lo haremos con la palabra key y asignarle un nombre que será unico (en el ejemplo departamento). ej:
unique key ‘departamento_UNIQUE’ (‘departamento’)

Relaciones 1-1 (de las cardinalidades máximas)
una solución de estas relaciones es que se guarden en una sola tabla con todos los atributos de las dos entidades.
puede haber casos especiales en los que se deben hacer dos tablas en las que se añade una foreign key que debe ser unique not null.
Relaciones n-n (de las cardinalidades máximas)
La relación entre las entidades se convierte en una tabla.La primary key de la tabla de la relación será compuesta de las dos otras 2 entidades que tienen esa relación los atributos de esta tabla serán dependientes de las dos entidades que tienen la relación.

Relaciones 1-1 (0,1-0,1) (de las cardinalidades máximas)
se puede tratar como una relación n-n en la que creas otra tabla donde se registra las relaciones puntuales entre las od entidades. La diferencia es que habrá una sola primary key y la otra será unique.

7 mayo 2025 Join

Consultas entre tablas
usaremos el comando join para juntar las tablas.

8 mayo 2025 Postgres,Dbeaver y Oracle

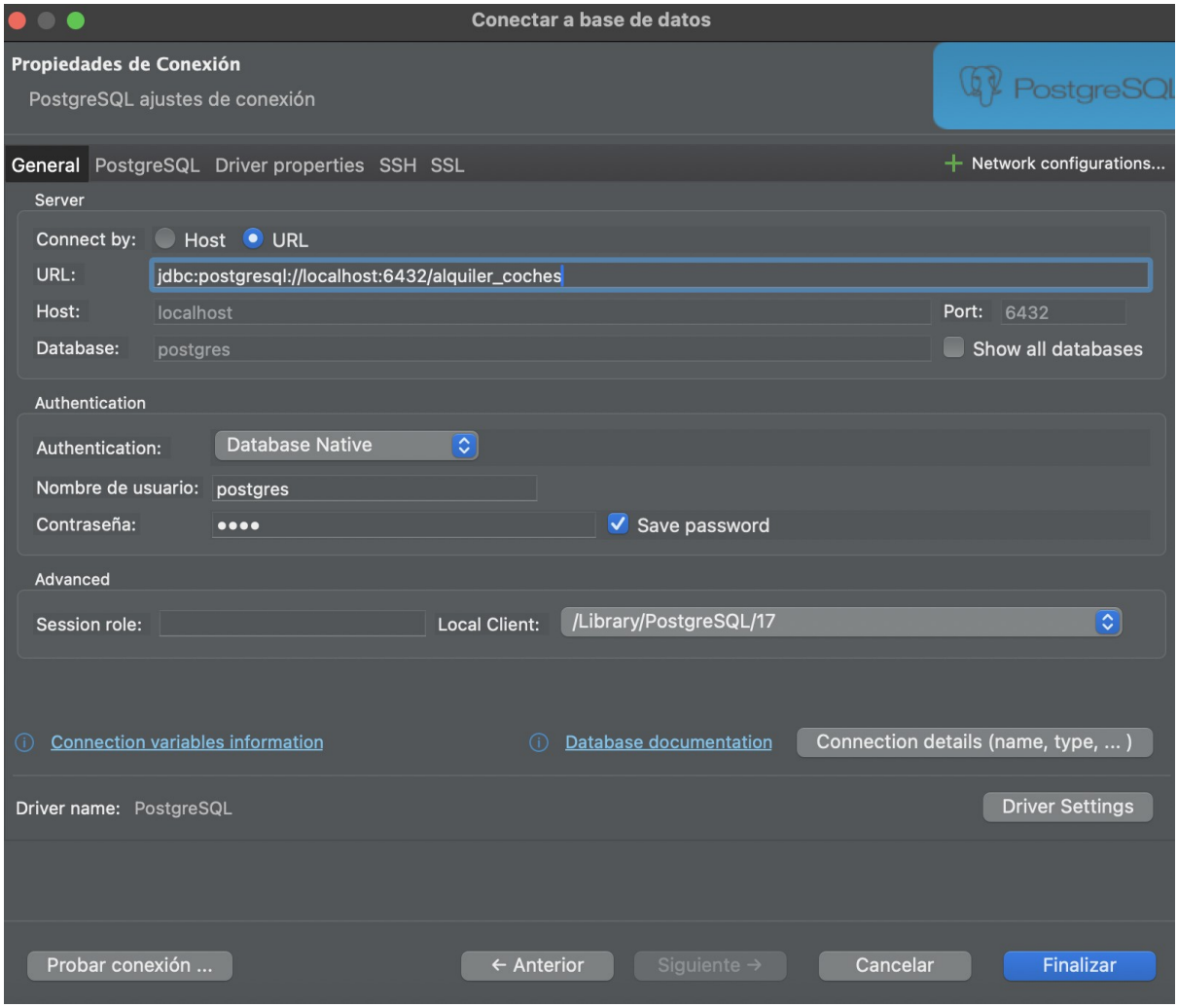
POSTGRES
puerto por defecto 5432
variable de entorno root = POSTGRES_PASSWORD
Para levantar el contenedor con nombre postgress, en este caso en puerto de host diferente(6432):
`docker run --name postgres -e POSTGRES_PASSWORD=root -p 6432:5432 -d postgres`
para entrar en postgres usaremos un cliente que en este caso funcionará como root:
`docker exec -it postgres bash`
`psql -U postgres --password`
Para mostrar BBDD:
`\l`
`q` para salir ya que si no cabe en pantalla queda bloqueada en pantalla.
crear BBDD con nombre prueba:

create database prueba;
usar la base de datos prueba:
\c prueba
creamos tabla y columnas:
create table datos(id_dato int not null,descripcion varchar(45) not null, primary key(id_dato));
muestra lastab las de la BBDD:
\dt
muestra contenido de una tabla concreta:
select * from datos;
muestra
insert into datos values(1,'uno'),(2,'dos');

Para ej3 con postgres:
desde consola host:
docker cp inserts_postgres.sql postgres:/root
docker cp alquiler_coches_postgres.sql postgres:/root

desde contenedor postgres ejecutando cliente::
\i /root/alquiler_coches_postgres.sql
\i /root/inserts_postgres.sql
comandos postgres:
nos conectamos a la base de datos:
\c alquiler_coches;
select count(*) from clientes;

En dBeaver:
Seleccionamos el puerto correcto y escribimos la base de datos a la que nos conectamos, almacenamos la clave del root:



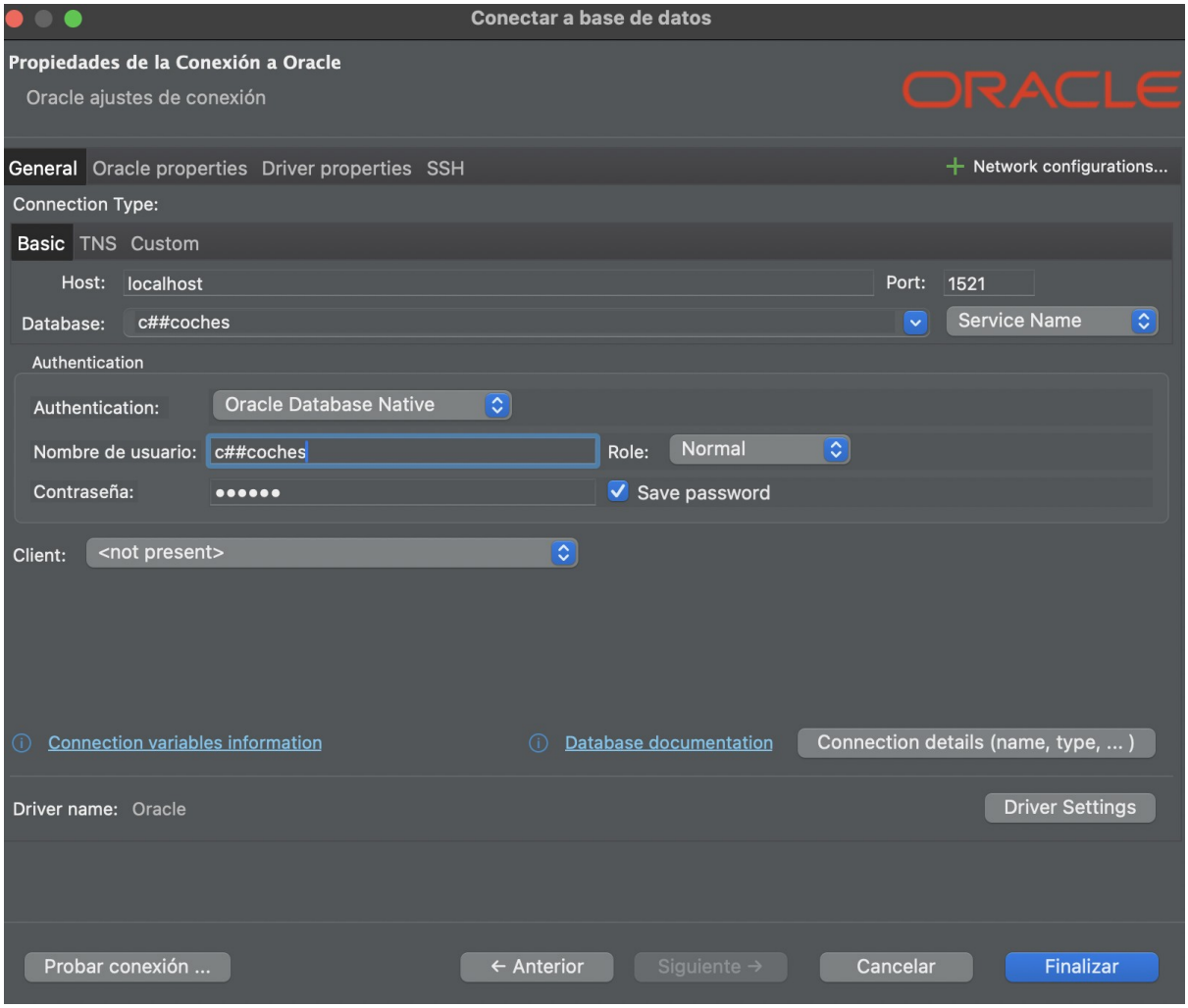
Seguramente habrá que bajar los complementos para que pueda trabajar con postgres.

Usando Oracle:
para ejecutar la descarga de oracle, lo haremos desde la pagina de oracle(no docker hub) con este comando::
docker pull container-registryu.oracle.com/database/free:latest
Podemos hacer la imagen de esta version con:
docker save -o oracle.tar container-registry.oracle.com/database/free
Para hacer imagen con esa version que tenemos en formato tar desde consola del host:
docker load -i oracle.tar
Creamos el contenedor (usando el id de la imagen):
docker run --name oracle -p1521:1521 -e ORACLE_PWD=root -d e7eb5e01f5b0
Ejecutamos el contenedor:
docker exec -it oracle bash
asociamos a un usuario, oracle obliga a tener un usuario, en este caso sysdba (que tiene permisos para creare usuarios):
sqlplus / as sysdba
con el usuario sysdba creamos un usuario (##coches)para interactuar con las BBDD:
create user c##coches identified by coches;
añadimos permisos:
grant unlimited tablespace to c##coches;
grant resource, connect, dba to c##coches;

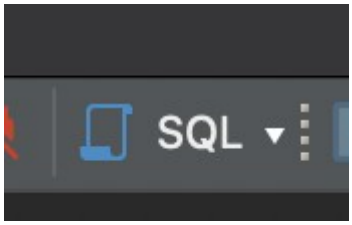
Nos conectamos a la base de datos con usuario ##coches (localiza el equipo despues de poner el nombre de usuario y una root):
connect c##coches/coches@localhost:1521/FREE
NOTA: una vez que el usuario coches existe podemos entrar desde el host con:
sqlplus c##coches/coches@localhost:1521/FREE

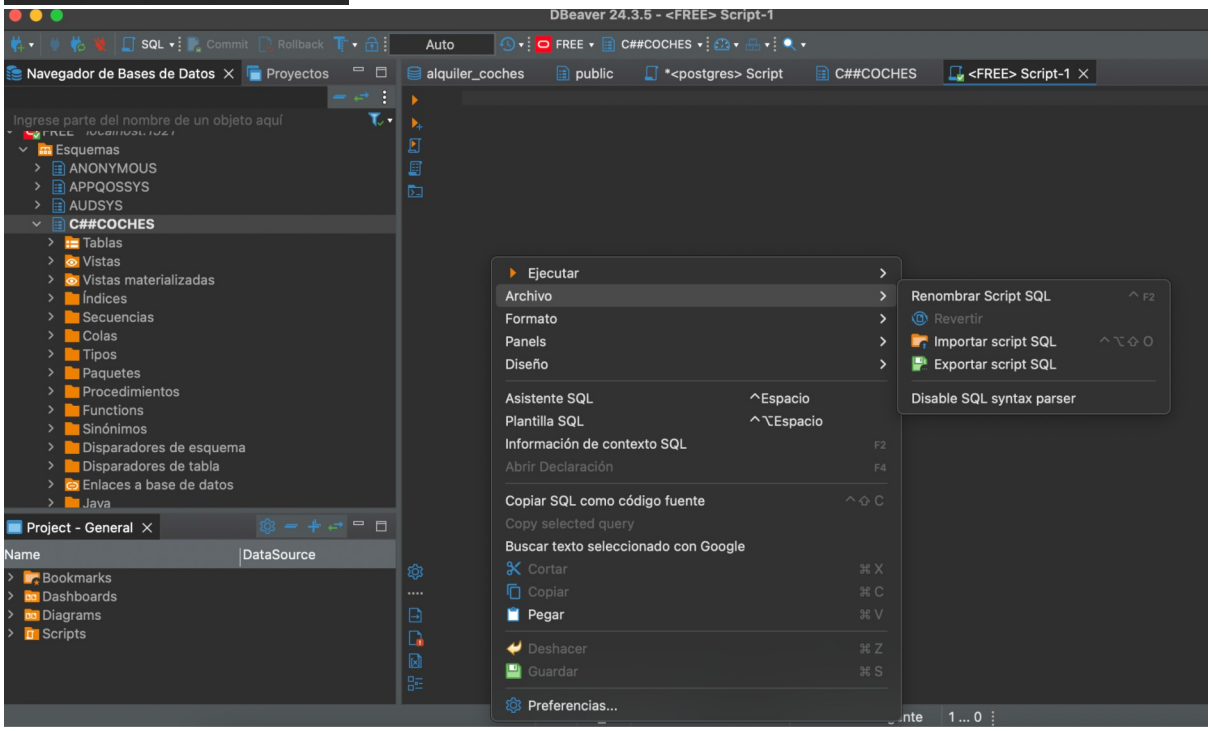
Desde una terminal del host copiamos scripts de oracle al contenedor levantado de oracle:
docker cp alquiler_coches_oracle.sql oracle:/home/oracle
docker cp inserts_oracle.sql oracle:/home/oracle

para ejecutar los scripts esde el contenedor de oracle:
@/home/oracle/alquiler_coches_oracle.sql
Los scripts que tienen una longitud muy larga (5mil y pico) no se pueden ejecutar, por lo que si esto fallase, lo podemos hacer desde Dbeaver:
especificamos base de datos, usuario y contraseña (en este caso coches)

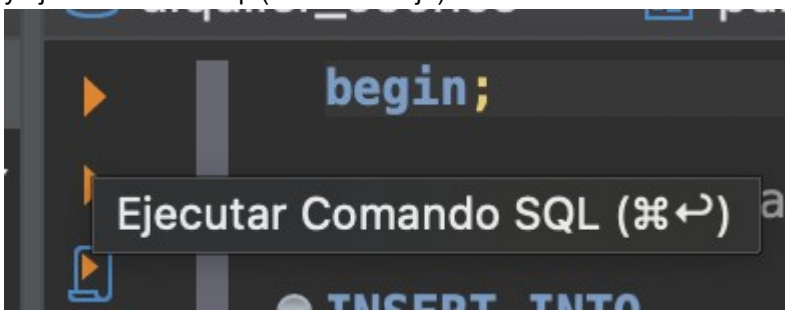


importamos el script, para ello es necesario tener una pestaña sql vacía abierta(en este caso es inserts_oracle.sql):





y ejecutamos el script(flecha naranja):



9 mayo 2025 Relaciones reflexivas

Relaciones reflexivas (1-n)
cuando de una misma entidad hay una **foreign key con referencia a esa misma entidad**. (el ejemplo más claro la tabla de empleados donde la relación de ser jefe es con dos especificaciones, empleado y jefe). La **foreign key tiene que tener un nombre muy claro en cuanto al rol de la relación** (en nuestro ejemplo sería fk_jefe).

Relaciones reflexivas (n-n)
generan una nueva tabla en la que constan **dos foreign key con un nombre muy claro hacia qué entidad o función apunta** que **serán de manera compuesta como una primary key**.Como ejemplo aeropuerto que puede ser origen y destino.

Relaciones ternarias (n-m-p)
dos tablas deben cumplir la condición de relación con una tercera

Proceso creación BBDD:
Utilizando metodología incremental o iterativa: se escogen unos pocos requisitos fundamentales que de manera iterativa siguen los pasos siguientes (a partir del 2):

1. información del cliente: formularios, entrevistas…
2. lista de requisitos funcionales
3. analisis
4. diseño:prototipo
5. implementación
6. pruebas
7. despliegue: tenerlo en un contenedor docker y subido a producción

SQL
Sql es no procedimental porque no decimos como resolver lo que le pedimos.
El como lo decide es el optimizer (optimizador de sentencias)

DDL: es el lenguaje de descripción de datos, es lo que maneja los metadatos.
create,drop,alter,rename,truncate
DML: es el lenguaje de manipulacion de datos.
Select,insert,update, delete
DSL: tiene que ver con los permisos
grant,revoke
TCL:tiene que ver con el control de transacciones (cuando se quiere hacer comprobaciones en las que si se cumplen todas se ejecutan (atómicas))
start transaction, commit, rollback,savepoint

tipos de datos (MySQL)
los valores entre[] son opcionales:

bit[(M)] M
indica el número de bits (entre 1 y 64), por defecto 1
tinyint[UNSIGNED]
entre -128 a 127 o 255. Ocupa 1 byte (256 valores).
bool,boolean,
equivalen a un tinyint ya que se almacena valor 0 o 1.
smallint[unsigned]
son 16 bytes y su rango es entre -32768 a 32767 o 65535.
int[unsigned]
4 bytes
integer[unsigned]
sinonimo de int
bigint[unsigned]
8 bytes
decimal[m[d]]
m indica el número total y d los decimales.

temporales:
date: fechas (de 1000-01-01 a 9999-12-31)
datetime:fechas y horas.((de 1000-01-01 00:00:00 a 9999-12-31 23:59:59)
timestamp:fecha y horas.Calcula desde 1970-01-01 00:00:00 hasta ahora, lo almacena en milisegundos pero llega hasta el año 2037 y lo almacena en 8 bytes. Si queremos una fecha anterior necesitamos poner un número negativo.
time:horas,minutos,segundos.
year[(2)(4)]:almacena un año, por defecto 4 dígitos. Para 2:70 a 69(1970 a 2069).

cadenas:
Char[(M)]: cadena de largo fijo.Siempre M caracteres por defecto 1.
Varchar[M] cadena de largo variable, máximo M caracteres.
Text,tinytext,mediumtext,longtext: para textos mucho más largos.

datos binarios:
blob,tinyblob,mediumblob,longblob: permite almacenar muchos formatos (sonidos(mp3 wav…),textos con formato(rtf,word…),imágenes(jpg,png…),ficheros de programas(blend,cad…)).Cuando se leen hay que interpretarlos con sus correspondientes programas.

no son estándar en SQL pero también se puede usar:
float[[m,d]]
float[(p)]
double[(m,d)]
double precision[(m,d)]

12 mayo 2025 SQL sintaxis

sintaxis (la parte de corchetes de la sintaxis es opcional)

- CREATE DATABASE | SCHEMA [IF NOT EXIST] <nombre_bbdd>
  CHARACTER SET <charset> [COLLATE <collate>]
  collate hace referencia al idioma mientras que charset especifica la codificación de los caractéres(hindi, chino,cirílico, etc)
  ej :
  create database ejemplo character set utf8mb4;

- DROP DATABASE | SCHEMA [IF EXIST] <nombre_BBDD>
  ej:
  drop database ejemplo;

  ALTER DATABASE | SCHEMA <nombre_bbdd><cambios_especificaciones>

- use database
  use ejemplo;
- create table<nombretabla>(<definiciones columnas>)
- create table<nombretabla>(<definiciones columnas>,primary key(<columnas>))

opciones a foreign key:
on delete(cuando se borre) | **on update**(cuando se cambie)
restrict |cascade|set null|no action
restrict: impide que se puedan eliminar las filas de referencia o su pk
cascade:permite que se puedan actualizar o eliminar las filas referenciadas
set null: asigna null a las fk que apuntan a filas eliminadas o cambiada su pk
no action: equivale a restrict.

Modificacion y eliminacion de tablas:
drop table nombre tabla:elimina la tabla entera
alter table nombretabla modify:modifica tipo de dato y restricción de una columna.
alter table nombre tabla change:renombra columna, tipo de dato y restricciones
alter table nombretabla add: como si fueses a crear otra vez la columna, por defecto la pone al final pero usando la palabra first la pone primera o after nombrecolumna detras de la que indiquemos.
alter table nombretabla drop:permite eliminar una columna.

show tables: mostrar tablas
desc|describe nombretabla:muestra la estructura de la tabla

select
select nombre tabla
from nombretabla
where condiciones…apuntes

distinct:
evita repeticiones en el resultado de la consulta. ej:
devolverá un solo modelo evitando repetir tantos modelos haya indicados con la foreign key

```
use 03b_alquiler_coches;
select distinct fk_modelo, modelo
from coches
join modelos on fk_modelo = id_modelo;
```

order by : asc (por defecto) y desc

limit: limita el resultado a una cantidad determinada:

```
use 03b_alquiler_coches;
select distinct fk_modelo, modelo
from coches
join modelos on fk_modelo = id_modelo
limit 3,2
```

limite 3 a partir de la segunda fila

funciones:
concat:

```
select id_empleado, concat (apellidos, ', ',nombre) empleado,dni from
04_reflexivas.empleados;
```

upper:
convierte a mayusculas:

algunas funciones(cadenas,matemáticas,fecha y hora), mirar apuntes….y construir ejemplos
algún ejemplo:
```
select now();
select monthname(now());
```

operadores:
aritméticos,relacionales,lógicos. Mirar apuntes y hacer ejemplos
algún ejemplo:

between:
```
use 01_negocio;
select * from productos
where prod_precio >=100 and prod_precio <=200;
– lo mismo usando between
select * from productos
where prod_precio between 100 and 200;
```

in:
```
use 02_tienda;
select * from productos where fk_fabricante in (2,4,5,6); – solo devuelve los productos con ese fk
```

13 mayo 2025 Cast y select (join)

cast
Cuando tenemos un valor de la tabla seteado con un tipo de valor, por ejemplo unsigned (no puede ser negativo) si operamos valores unsigned y en este caso hacemos una resta, podría dar un valor negativo al ser una operación entre valores unsigned(el resultado siempre será unsigned) lo que dispara el error. Para evitarlo, podemos hacer un cambio de valor, no en la tabla, sino en la consulta, utilizando la palabra clave cast que convertirá el valor a lo que indiquemos.(Ejercicio 06_empleados, consulta 4):
```
select id_departamento, departamento,cast(presupuesto as signed) -gastos from departamentos;
```

select varias tablas
inner join (sinonimo de join) donde hacemos el **filtrado**(Los foreign key que son iguales al id de la otra tabla) para **evitar resultado de producto cartesiano**.

outer join trabajan con los registros que sobran del interno. Hay **3 tipos en SQL**:

1. Left outer join:se queda con todos los registros del inner join y **agrega los del lado izquierdo (los que están antes del join) aunque no tengan relación**, aparecerán con valor null en la relacion.
2. right outer join:funciona igual que left join pero teniendo en cuenta **el lado de la derecha para añadirlos aunque no tengan relación.**

3. full outer join(no existe en MySQLI): son los relacionados y el añadido de los de la **izquierda sin relación** y los de la **derecha sin relación**.

En **MySQL** soluciona esto con la palabra clave **union** que hace la union de **2 tablas** en donde añade todos los elementos sin repetirlos (tabla 1:1,3,5,7 y tabla 2: 3,5,8 será 1,3,5,7,8)pero los elementos deben ser del mismo tipo y con las mismas columnas (**union compatibles:** tienen el **mismo número de columnas** y cada columna es de un tipo de **datos compatibles** con la otra).
ej:

```
select nombre,apellidos,mail
from empleados emp
union
select c.nombre,c.apellidos,c.mail – entendiendo que clientes se llama c
from clientes;
```

Hacer un **full outer join en MySQL** es: **unir** en **left** **A** con **B**, **unir** en **right** **A** con **B**.
Ej:

```
select *
from a
left join b on…
union
select *
from a
right join b on…;
```

14 mayo 2025 Intersect,except,funciones de agregación

intersect
Hace la interseccion de 2 tablas, donde obtiene los que son **comunes en las 2 tablas**. Debe cumplir las mismas condiciones (**union-compatibles**:mismas columnas y datos compatibles).

```
select…
intersect
select…
```

except
es la operacion de conjuntos de diferencia, donde obtiene todos los elementos que no están más que en la tabla a la que hacemos referencia. Mismas condiciones de **union-compatible**.
Todos los **elementos que están en tabla A** y **no están en la tabla B.**

```
select …
except
select…
```

funciones

- **de agregación:**
  sum(suma),
  count(contador)t,
  avg(media aritmética))
  min(valor mínimo)
  max(valor máximo)

  Ejemplos:

```
use 05_empleados;

select sum(presupuesto) as tot_presupuesto
from departamentos;

select count(id_departamento) as cantidad
from departamentos
where presupuesto > 200000;

select avg(presupuesto) as media_presupuesto
from departamentos
where presupuesto >0;

select max(gastos) as gasto_maximo
from departamentos;

select gastos
from departamentos
order by gastos desc
limit 1;
```

  group
  crea filas por cada elemento de la columna que seleccionamos, esto crea una dependencia funcional del valor que agrupamos.

```
use 05_empleados;

select id_departamento,departamento, count(id_empleado)-- puedes coger cualquier columna que dependa del valor funcional (en estre caso group by id_departamento, primary_key(id_departamento))
from departamentos as d
left join empleados as e on d.id_departamento = e.fk_departamento
group by id_departamento; -- crea filas por cada elemento diferente por el campo que agrupo

-- Datos de los departamentos y cantidad de empleados de aquellos departamentos
-- con mas de un empleado
select id_departamento,departamento, count(id_empleado)-- puedes coger cualquier columna que dependa del valor funcional (en estre caso group by id_departamento, primary_key(id_departamento))
```

```
from departamentos as d
left join empleados as e on d.id_departamento = e.fk_departamento
group by id_departamento
having count(id_empleado) > 1; -- having se cumple despues de agrupar, no puede
esxistir antes

-- Datos de los departamento con presupuesto 150000 o mayor presupuesto
select id_departamento,departamento, count(id_empleado)-- puedes coger cualquier
columna que dependa del valor funcional (en estre caso group by id_departamento,
primary_key(id_departamento))
from departamentos as d
left join empleados as e on d.id_departamento = e.fk_departamento
where d.presupuesto >= 150000  -- filtra las filas antes de hacer el group by, no
puede existir después
group by id_departamento
```

15 mayo 2025 subconsultas

subconsultas
consultas dentro de otras.
tipos:

1. subconsulta de columna: devuelve 1 columna con varias filas
2. subconsulta de fila (no se utiliza): devuelve más de 1 columna pero una única fila
3. subconsulta de tabla: devuelve 1 o varias columnas y 0 o varias filas
4. subconsulta **escalar**: devuelve **1 columna** y **1 fila** (un solo valor)

Las subconsultas en:
where: productos cuyo precio mayor a la media de todos los productos.
media 120
ejemplo subconsulta **escalar**:

```
select * from productos
where precio > (select avg(precio) from productos);
```

where o Having
operadores de comparación (>,<, =...)
where coste **<** (select… será verdadero si el coste es menor al valor devuelto por la consulta

ALL,ANY: se utilizan con los operadores de comparación
where coste >= **ALL** (select… será verdadero si el coste es mayor o igual a todos los valores devueltos por la consulta. Funciona como un filtro.
where coste = **ANY** (select…será verdadero si el coste es igual a alguno de los valores devueltos por la consulta

IN,NOT IN: where id_empleado **IN** (select… será verdadero si id_empleado figura entre los valores devueltos por la consulta

EXISTS,NOT EXISTS:pertencen a subconsultas relacionadas, pendiente de ver.
where EXISTS (select…

operadores y tipo de subconsulta:
<,>=... escalar
ALL, ANY de tipo columna

IN,NOT IN de tipo tabla con la cantidad de columnas iguales a la cantidad de valores buscados

EXISTS, NOT EXISTS de tipo tabla

Ejemplo explicado subconsulta con BBDD 02_tienda:

-- productos que tengan el precio mayor a la media de los fabricantes
select 02_tienda;

-- obtenemos la media por fabricante,
```
select id_fabricante, avg(precio) media
from productos join fabricantes
on fk_fabricante = id_fabricante
group by id_fabricante;
```
-- si hiciesemos un join de prodcuto con la tabla que se hubiese creado (llamemosla medias):
```
select *
from productos  join medias
on ok _fabriacante  = medias.id_fabricante
where precio>=medias.media;
```

-- SUBCONSULTAS
-- la **subconsulta** sería mediante alias(medias):
```
select * from productos
join (select id_fabricante,avg(precio)media
        from productos join fabricantes
        on fk_fabricante = id_fabricante
        group by id_fabricante) medias
on fk_fabricante = medias.id_fabricante
where precio>= medias.media;
```

CON HAVING:
```
select fabricante, count(*) cant
from productos join fabricantes
on fk_fabricante = id_fabricante
group by fabricante
having cant=(select count(*)
```

```
from productos join fabricantes
on fk_fabricante = id_fabricante
where fabricante = 'Asus');
```

16 mayo coalesce

coalesce
es un modificador que permite añadir un texto que sustituye un null por la cadena de texto que queramos.
Ej:(suponiendo que no se tenga el apellido 2 y queramos como resultado: no tiene en vez de null)
`select alu.id_alumno, alu.apellido1, coalesce(alu.apellido2, 'NO TIENE')...`
En caso de varios telefonos:
`coalesce (tel_personal, tel_trabajo,otro_tel,'NO TIENE')`
coalesce recorre todos los telefonos y si son todos null entonces devuelve NO TIENE.

21 abril 2025 Modelos ternarios DER a esquema conceptual.Relaciones de herencia

Modelo ternario nmp
las cardinalidades *mínimas siempre son 0* (por eso se excluyen).
cada entidad genera una tabla.
La **relación genera una tabla**r(fk cada entidad de la relación)
Las **tres fk conforman la pk compuesta** de la nueva tabla.

Modelo ternario nm1
Cada entidad genera una tabla
la relación genera una tabla
contiene una fk a cada entidad de la relación
*Las dos **fk** de los lados a muchos**(n)** conforman la **pk compuesta** de la nueva tabla*
*la **fk** que referencia al al lado de **1 debe ser not null***

Modelo ternario n11
Cada entidad genera una tabla
la relación genera una tabla
en la relación hay 2 claves candidatas (del lado **n siempre forme parte de la pk** y **la que queda fuera de la pk** será **unique**)

Modelo ternario 111
Cada entidad genera una tabla
la relación genera una tabla
contiene una fk a cada entidad relación
3 claves candidatas , compuestas por las **combinaciones de 2fk**
elegida al azar una como pk, las **otras dos claves** candidatas deben ser **not null y unique**.

—----------

Relaciones de herencia
la clase padre contiene atributos comunes incluida la primary key, los hijos sus atributos específicos.
contemplamos relaciones:

- total o parcial
- con solapamiento o sin solapamiento

Estrategias a usar del DER a esquema relacional

1. **join tables**:
   cada entidad genera 1 tabla:
   1 tabla para generalización(padre) y una por cada especialización(herederas).
   La tabla generalización tendrá los atributos en común y las otras solo los propios, y una fk a la generalización y será su pk
   No hay manera de controlar si es parcial o total, siempre serán parciales (necesita un control mediante software(control transaccional)).
   Tampoco controla si tiene o no solapamiento. (requiere también control por software(transaccional)
   Aún así esta estrategia debería ser la más utilizada.
2. **Una sola tabla:**
   todas las entidades en una tabla y esa tabla tendrá todos los atributos(incluidos los específicos) y un discriminante.
   Se generan checks con controles indicando qué atributos que deben ser null y cuáles no, en base al valor del discriminante.
   La parcialidad depende del discriminante y sus valores.
   Para el solapamiento serán a raíz del discriminante abarcando las entidades (todas o algunas), puede haber varios discriminantes.
3. **tabla independiente por entidad (nunca recomendado):**
   Se genera una tabla por cada especialización con atributos propios y heredados
   Si es parcial se genera un tabla para la generalización
   Las tablas no están relacionadas
   No se aplica herencia y se pierden los controles de integridad referencial.

Agregaciones
Se refiere a relaciones entre dos atributos que de manera conjunta en su relación establecen una nueva relación como intermediaria con el tercer atributo que no cumple dicha relación (si es n a n genera una nueva tabla de relación, si es a 1 solo conecta la relación a la entidad).(ejemplo de canciones,grupos y álbumes).

16 mayo Crear tablas sin restricción, natural join

al principio de la creación, antes de tablas:
set foreign_key_checks = 0;
al final del script:
set foreign_key_checks = 1;

permite que cuando referenciamos a tablas que aún no han sido creadas ,ejecutar el script íntegramente(set foreign_key_checks = 0;) y luego al final activar las restricciones cuando están creadas(set foreign_key_checks = 1;).

lineas de comando ejecutables SOLO en Mysql
ej:
/! ….*/*
/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;

natural join
Cuando unimos tablas en las que el foreign key y el id se llaman igual, usaremos la instruccion natural join (ver 06_universidad_natural):
ej:

```
use 06_universidad_natural;

-- profesores con asignaturas que imparte, solo los que imparten asignaturas

select *
from profesores p
join asignaturas as a on p.id_profesor = a.id_profesor;

– es lo mismo que
select *
from profesores
natural join asignaturas; -- la foreign key se llama igual que el id, usaremos join natural
```

Sacar una media excluyendo (cuando los datos son pocos y se aleja de la media)
se realiza con subconsultas correlacionadas, que hacen consultas a la consulta contenedora. Son consultas más costosas.
A partir de una subconsulta escalar que se ejecuta 1 vez por cada registro de la consulta contenedora.
ej:

```
select p.product, p.precio, (select avg(precio)from productos
                                    where id_producto <>p.id_producto)
from productos p;
```

p.id_producto hace referencia a la última parte (from productos p)

exists
devuelve booleanos y en las consultas con esta condición se suele hacer con la fila entera (*) ya que solo chequea su existencia.

26 mayo 2025 · vistas, normalización, transacciones,autocommit

Es una categoria mas dependiente de las tablas, devuelve lo que hace una consulta normal, de tal manera que almacena ese resultado cuando accedemos a la vista.

```
create or replace view listado_pagos_clientes as
select id_cliente,concat(c.nombre_contacto, c.apellido_contacto)as n_cliente,telefono,
c.ciudad,c.pais, pg.fecha_pago, pg.total, pg.id_transaccion
from clientes as c
join pagos as pg on c.id_cliente = pg.fk_cliente;
```

La normalización es una serie de reglas aplicadas a las relaciones obtenidas tras el paso a tablas desde DER

Codd (mirar Pablo 1fn, 2fn, 3fn):

1. primera forma normal:
   los dominios de los **atributos son atómicos (indivisibles).**
2. cuando **cumple la primera forma normal, cada atributo que no forma parte de la clave primaria depende completamente de la clave primaria.(solo se debe comprobar en tablas con claves primarias compuestas).**los que no cumplan generan una nueva tabla con el id del que dependan
   Si la tabla tiene una **clave primaria simple** cumple automáticamente la **segunda forma normal.**
3. **cuando cumple la segunda forma normal,** cada atributo que no forme parte de la clave primaria no puede depender de otro atributo que no sea clave.Los que no cumplan, generan una nueva tabla con el id del que dependan.

Se busca que se cumpla hasta la **tercera forma normal** (definidas por Codd).
El resto de formas normales están contempladas pero no se evaluan en el curso.

Transacción
deben cumplir las propiedades **ACID**: atomicidad, consistencia, aislamiento, durabilidad.

Atomicidad:La transacción está compuesta de una o mas actividades agrupadas como una sola unidad de trabajo indivisible.

Consistencia: finalizada la transacción se debe haber cumplido todo y en caso de que no se complete dejar los datos como estaban antes de que se realizase la transacción.

Aislamiento: Se refiere a los procesos, deben estar aislados para que varios usuarios no intervengan sobre la misma transacción(evitar el acceso concurrente).

Durabilidad: cuando finaliza la transacción el resultado debe persistir y tolerar cualquier fallo del sistema (volviendo al estado anterior consistente).

Comandos transacciones: start transaction, commit, rollback.

Autocommit
Mysql tiene activado por defecto el modo **autocommit**(los cambios son permanentes)
La variable autocommit permite consultar el estado de dicha variable o cambiarlo.
select @@AUTOCOMMIT (consulta el valor).
set AUTOCOMMIT = 0; (setea el valor dejandolo desactivado).
set AUTOCOMMIT = 1; (setea el valor dejandolo activado).

27 mayo 2025 · Rollback y Commit, aislamiento, index

La forma correcta de trabajar desde un programa (no en workbench) es sin tener el modo commit desactivado (set AUTOCOMMIT = 0; (setea el valor dejandolo desactivado)).

rollback sirve para **recuperar el ultimo estado "commiteado"**.
Commit valida las modificaciones siempre y cuando estemos trabajando con el **modo commit desactivado.**

```
use 01_negocio;

 select * from productos;
 update productos set prod_precio= 12,5 where idproducto = 23;
 rollback; -- solo funciona si el modo autocommit está desactivado

 select  @@AUTOCOMMIT;
 set AUTOCOMMIT = 0;

 select * from productos where idproducto =23;
update productos set prod_precio= 10 where idproducto = 23;

 select * from productos where idproducto =23;-- no esta confirmado el autocommit no está activo y no se actualiza hasta hacer commit

 update productos set prod_precio = 25 where idproducto = 23;
commit; -- esta linea ejecuta el cambio y ya no se puede recuperar el estado anterior con rollback.
```

Cuando se programa en Java se prepara la programacion de creación o cambios(insert,update,drop…) y si algo falla, **rollback**. Cuando no hay fallos entonces se hace **commit** para validar todos los cambios.

Aislamiento:

```
select * from productos;

-- no hay transaccion si no hay rollback o commit (si autocommit esta desactivado)
select  @@AUTOCOMMIT;
set AUTOCOMMIT = 0;

update productos set prod_precio = 100 where id_producto = 1;

update productos set prod_precio = 100 where id_producto = 2;

-- si consultaramos los datos ahora en nuestro servidor (consola) no sabríamos lo que valen
-- ya que al no haber hecho commit no se han confirmado los datos.
-- en este caso hacemos rollback para rechazar los cambios y dejarlos en el estado inicial.

rollback;

delete from productos where id_producto = 1; -- borra desde workbench

-- desde servidor
update productos set prod_precio = 100 where id_producto = 1; -- dará un error porque la fila estará bloqueada (al principio queda bloqueada y tras timeout error)

-- es el aislamiento que no permite cambios de manera simultánea(es decir si hubiese un rollback o un commit).
```

ERRORES en transacciones:

lectura sucia: cuando se leen datos que están siendo modificados por otro usuario (antes de commit).
La solucion sería que no se pueda ver una transaccion que no esté commiteada.

Lectura no repetible: la lectura se hace antes de un commit, luego otro usuario hace un commit que cambia el valor de la consulta y el primer usuario sigue trabajando sobre un valor que no está actualizado (o commiteado).
La solución sería realizar el commit cuando el primer usuario termine (o commitee)

Lectura fantasma: es el mismo caso anterior, pero el segundo usuario añade datos.
Misma solución.

Niveles de aislamiento:

4 niveles de aislamiento para SQL:

1. **Read uncommitted:** si se configura así no hay bloqueos y permiten los errores antes descritos
2. **read commited:** solo soluciona la lectura sucia. Los datos leidos por una transacción son modificados(commit) por otras transacciones.
3. **Repeteable Read** (modo por defecto en MySQL: todas las lecturas de la misma transacción leen la lectura de la base de datos tomada por la primera lectura.
4. **Serializable**: si hay una transaccion se paralizan todas las transacciones creando una cola en el acceso por lo que no hay acceso concurrente (la base de datos se vuelve “lenta”).

Como se definen los niveles de aislamiento en función del nivel de bloqueo que se decida:
Set session transaction isolation level read uncommited;
Set session transaction isolation level read committed;
Set session transaction isolation level read read;
Set session transaction isolation level serializable;

select @@session.transaction_isolation; -- devuelve repeatable read, es una variable de sesion.
-- es como se aisla el usuario con respecto a los demás

set session transaction isolation level read uncommitted;

Index
para agilizar las consultas en las tablas se crean indices que apuntan a filas en concreto para iniciar esas consultas y evitar hacer una busqueda secuencial.
hay varios tipos entre los cuales estan (mirar Pablo optimizacion de consultas):
create index
create unique index
create full text index….

explain /describe select
Muestra una tabla con informacion del rendimiento de la consulta.
se utiliza al principio de la consulta:
explain select id_alumnos from alumnos where dni = 234567890….
o
describe select…