IES Henri Matisse

BASES DE DATOS
TEMA 6

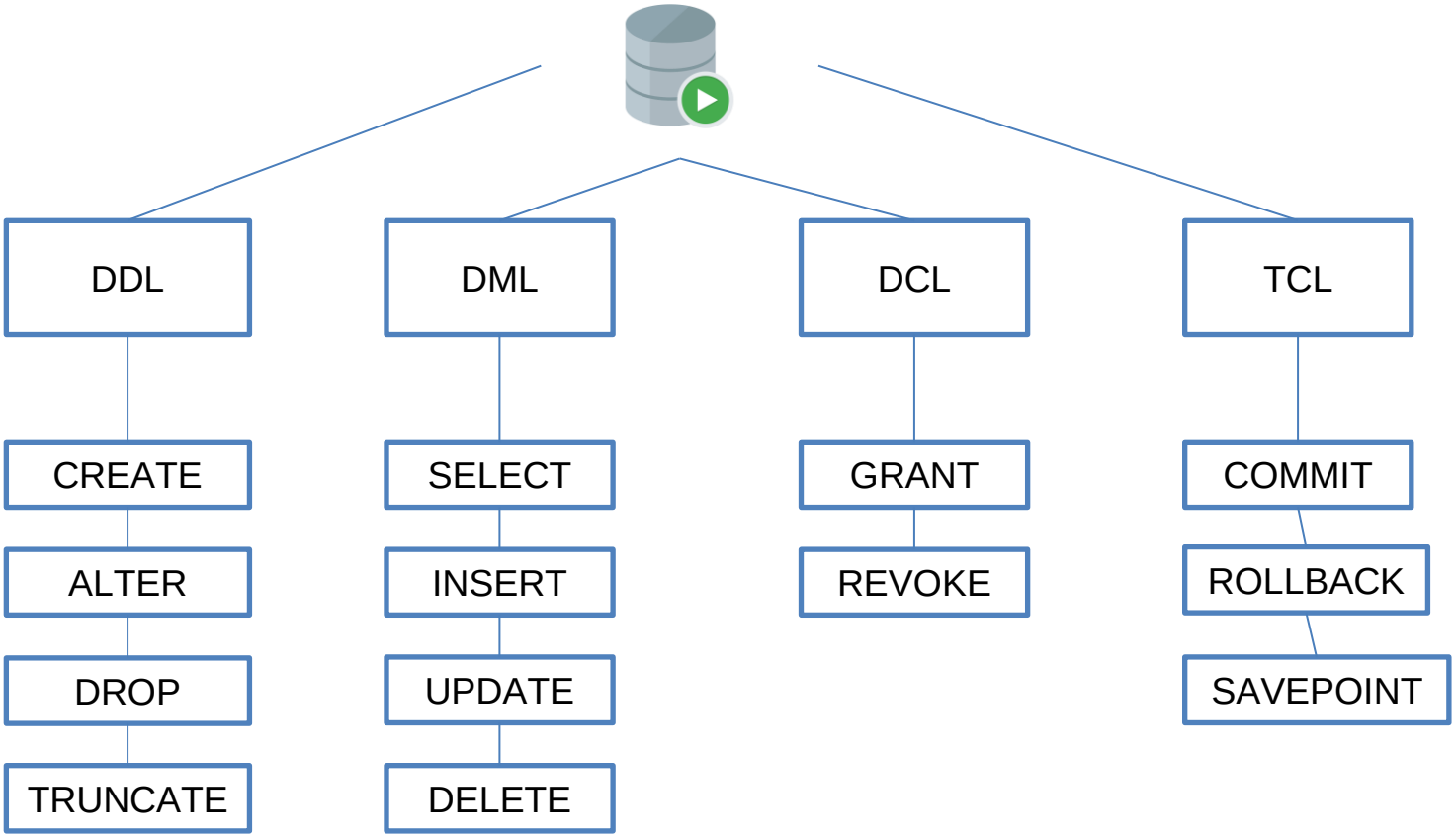
TRATAMIENTOS DE DATOS

1. Introducción

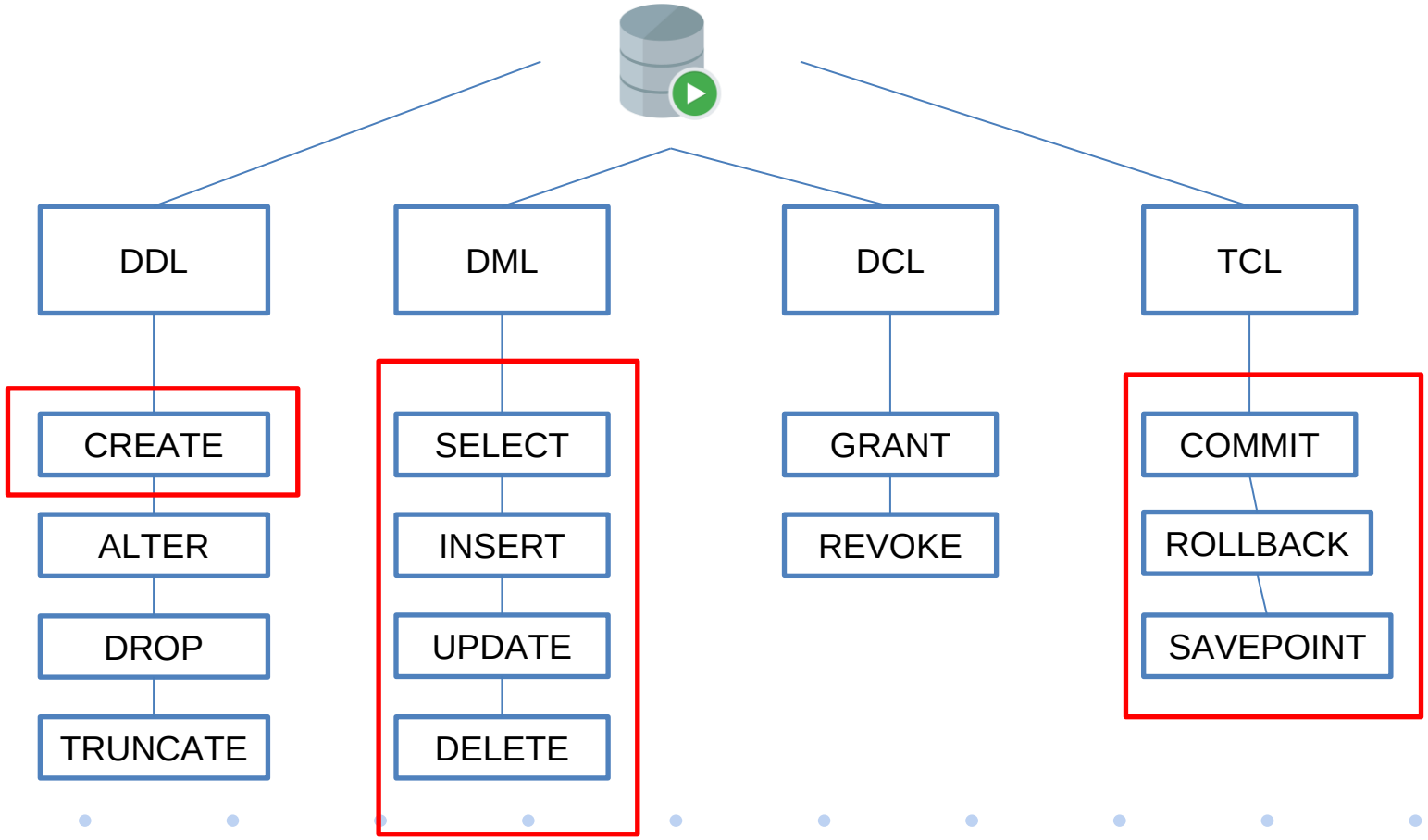
- Actualmente, solo hemos realizados consultas las cuales el objetivo era obtener cierta información de una BBDD.
- Las consultas **SELECT** ejercen únicamente la función de lectura.
- Existen un grupo de consultas, nombradas como consultas de acción, cuque ejercen una funcionalidad de escriptura. Estas son las siguientes:

  - Actualización de datos.
  - Modificación de datos.
  - Creación de tablas.
  - Inserción de datos.

2. Grupo de Comandos.



3. Grupo de Comandos.



4. La sentencia CREATE TABLE

```
CREATE TABLE NAME_TABLE(
        colum datatype [DEFAULT exp][, …]
);
```

Con el comando **CREATE TABLE** se crea una table en una base de datos Oracle.

Por ejemplo:

```
Create table Nombre_Tabla
(Nombre_Campo Tipo_Dato(Tamaño),...,Nombre_campo Tipo_Da
to(Tamaño));
```

5. La sentencia CREATE TABLE

Tabla normal (sin restricciones):

```
Create table Clientes(
            Codigo_Cliente number,
            Nombre_Cliente varchar2(100)
);
```

Tabla con restricciones NOT NULL:

```
Create table Clientes(
            Codigo_Cliente number NOT NULL,
            Nombre_Cliente varchar2(100)
);
```

6. La sentencia CREATE TABLE

Tabla con restricciones LLAVE PRIMARIA:

```
Create table Clientes(
        Codigo_Cliente number,
        Nombre_Cliente varchar2(100),
       Constraint Pk_Cliente  Primary key  (Codigo_Cliente));
```

Otro Método:

```
Create table Clientes(
        Codigo_Cliente number Primary key,
        Nombre_Cliente varchar2(100),
);
```

7. La sentencia CREATE TABLE

Tabla con restricciones LLAVE FORANEA:

```
Create table Clientes(
        Cod_Ciudad number,
        Constraint fk_ciudad_cliente Foreign key (Cod_Ciudad)
references Ciudades(Codigo));
```

Otro Método:

```
Create table Clientes(
        Codigo_Cliente number,
        Nombre_Cliente varchar2(100),
        Foreign key (Cod_Ciudad) references Ciudades(Codigo));
```

8. La sentencia CREATE TABLE

Tabla con restricciones:

```
Create table Clientes(
        Codigo_Cliente number not null,
        Nombre_Cliente varchar2(100),
        Cod_Ciudad number,
        Constraint Pk_Cliente Primary key  (Codigo_Cliente),
        Constraint Fk_Ciudad_Cliente  Foreign key (Cod_Ciudad)
references Ciudades(Codigo));
```

9. Actividad

EQUIPO_2 (nomeq:char(25), director:char(30))

CP:{nomeq}

CICLISTA_2 (dorsal:entero, nombre:char(30), edad:entero, nomeq:char(25))

CP:{dorsal}

CAj:{nomeq}→ EQUIPO

VNN:{nomeq}

VNN:{nombre}

10. Solución

```
CREATE TABLE ciclista_2(
    dorsal integer,
    nombre varchar(30) not null,
    edad integer,
    nomeq varchar(25) not null,
    constraint Pk_Ciclista2 Primary key (dorsal),
    Constraint Fk_Nomeq  Foreign key (nomeq) references
                                        EQUIPO_2(nomeq)
);
```

11. Solución

```
CREATE TABLE equipo_2(
    nomeq varchar(25),
    director varchar(30),
    constraint Pk_Equipo Primary key (nomeq)
);
```

Una vez creadas usar sentencia **DROP** para eliminar las tablas creadas:

- DROP TABLE ciclista_2;
- DROP TABLE equipo_2;

12. La sentencia INSERT

```
INSERT INTO {nombre_tabla | nombre_vista} [(nombre_columna [,
nombre_columna] ...)] {VALUES (valor [, valor] ...) | sub_consulta
};
```

Con el comando **INSERT** se añade una tupla a la tabla o a la vista.

Si se da una lista de columnas, los valores deben emparejar uno a uno con cada una de estas columnas. Cualquier columna que no esté en la lista recibirá el valor NULL.

- Si no se da esta lista de columnas, se deberán dar valores para todos los atributos de la tabla y en el orden en que se definieron con el comando CREATE TABLE.
- Si se elige la opción de **sub_consulta**, se introducirán en la tabla las tuplas resultantes de la subconsulta expresada como un comando SELECT

13. La sentencia INSERT

1. Insertar el equipo Kelme con director Pepe García.

```
INSERT INTO Equipo VALUES('Kelme', 'Pepe Garcia’)
```

2. Insertar el equipo Repsol.

```
INSERT INTO Equipo (nomeq) VALUES (‘Repsol')
```

3. Realizar la insercion de los productos_nuevos de categoria Deportes de Riesgo en la tabla productos

```
INSERT INTO Productos select * from Productos_Nuevos where categoria =
'DEPORTES DE RIESGO';
```

14. La sentencia INSERT

ACTIVIDAD BBDD CLIENTES-PEDIDOS

1. Insertar un nuevo pedido del cliente "DEPORTES EL BRASILEÑO" con un número de pedido de 150, con fecha del 09/08/20, con un pago con Tarjeta, descuento de un 10% y que todavía no se ha enviado.

2. Insertar un nuevo pedido del cliente "DEPORTES EL BRASILEÑO" con un número de pedido de 125 y con un pago Aplazado.

3. Insertar en una nueva tabla (pedido_2) los pedidos realizados desde el '10/03/00' y el '12/12/00'.

15. Solución

1. INSERT INTO PEDIDOS VALUES (150,'CT40','09/08/20','TARJETA','0.1','N');
2. INSERT INTO PEDIDOS(NUMPEDIDO, CODCLIENTE, PAGO) VALUES (126,'CT40','APLAZADO');
3. CREATE TABLE PEDIDOS_2 (
   NUMPEDIDO INTEGER,
   CODCLIENTE VARCHAR(10),
   FECHA DATE,
   PAGO VARCHAR(25),
   DESCUENTO VARCHAR(25),
   ENVIADO CHAR
   );

INSERT INTO PEDIDOS_2 SELECT * FROM PEDIDOS WHERE FECHA BETWEEN '10/03/00 AND 12/12/00'

16. La sentencia UPDATE

La sentencia **UPDATE** de SQL permite modificar el contenido de cualquier columna y de cualquier fila de una tabla. Su sintaxis es la siguiente:

```
UPDATE nombre tabla SET nombre_col1=exprl [, nombre_col2=expr2 ]
... [WHERE filtro]
```

La actualización se realiza dando a las columnas n**ombre_col1, nombre_col2**... los valores **expr1, expr2**,... Se actualizan todas las filas seleccionadas por el filtro indicado mediante la cláusula WHERE.

Esta cláusula WHERE, es idéntica a la que se ha utilizado para el filtrado de registros en la sentencia SELECT.

17. La sentencia UPDATE

Por ejemplo, si se desea actualizar el equipo de 'Pau Gasol' porque ha fichado por otro equipo, por ejemplo, los 'Knicks', habría que ejecutar la siguiente sentencia

```
UPDATE jugadores SET Nombre_equipo='Knicks' WHERE Nombre='Pau Gasol';
```

Es posible cambiar más de una columna a la vez:

```
UPDATE jugadores SET Nombre_equipo='Knicks', Peso=210 WHERE Nombre='Pau Gasol';
```

¡¡OjO !! Si se omite el filtro, el resultado es la modificación de todos los registros de la tabla

18. La sentencia UPDATE

ACTIVIDAD BBDD CLIENTES-PEDIDOS

- Actualizar del cliente BELTRÁN E HIJOS su número de teléfono al siguiente valor 676767676.

- Incrementar el precio en 10€ de todos los productos que pertenezcan a la categoría de Deportes.

19. Solución

1. UPDATE CLIENTES SET TELEFONO=676767676 WHERE EMPRESA='BELTRAN E HIJOS';

2. UPDATE CLIENTES SET PRECIO=PRECIO+10 WHERE CATEGORIA='DEPORTES';

20. La sentencia DELETE

En SQL se utiliza la sentencia DELETE para eliminar filas de una tabla.

Su sintaxis es:

```
DELETE FROM nombre_tabla
[WHERE filtro]
```

El comando DELETE borra los registros seleccionados por el filtro WHERE, que es idéntico al de la sentencia SELECT. Si se desea borrar las estadísticas del jugador 73 , habría que escribir la siguiente sentencia:

```
DELETE FROM estadisticas WHERE jugador=73;
```

Si se omite el filtro, el resultado es el borrado de todos los registros de la tabla

21. La sentencia DELETE

ACTIVIDAD BBDD CLIENTES-PEDIDOS

1. Eliminar el pedido del cliente "DEPORTES EL BRASILEÑO" con un número de pedido de 150, con fecha del 09/08/20, con un pago con Tarjeta, descuento de un 10% y que todavía no se ha enviado.

2. Eliminar el pedido del cliente "DEPORTES EL BRASILEÑO" con un número de pedido de 125 y con un pago Aplazado.

22. Solución

1. DELETE FROM PEDIDOS NUMPEDIDO='150';

2. DELETE FROM PEDIDOS NUMPEDIDO='125';

23. La sentencias UPDATE y DELETE con subconsultas

Es posible actualizar o borrar registros de una tabla filtrando a través de una subconsulta La única limitación es que hay gestores que no permiten realizar cambios en la tabla que se está leyendo a través de la subconsulta.

Por ejemplo, si se desea eliminar los empleados 'Representante Ventas' que no tengan clientes se podría codificar:

```
DELETE FROM Empleados WHERE CodigoEmpleado Not in (SELECT
CodigoEmpleadoRepVentas FROM Clientes)
AND Puesto='Representante Ventas';
```

No sería posible, sin embargo, escribir una sentencia de este tipo para borrar les clientes con LimiteCredito 0, puesto que se leen datos de la misma tabla que se borran:

```
DELETE FROM Clientes WHERE CodigoCliente in (SELECT CodigoCliente
FROM Clientes WHERE LimiteCredito=0);
```

24. Borrado y modificación de registros con relaciones

Hay que tener en cuenta que no siempre se pueden borrar o modificar datos: Considérese por ejemplo, que un cliente llama a una empresa pidiendo darse de baja como cliente, pero el cliente tiene algunos pagos pendientes. Si el operador de la BBDD intenta eliminar el registro (DELETE), el SGBD debería informar de que no es posible eliminar ese registro puesto que hay registros relacionados.

O por ejemplo, se desea cambiar (UPDATE) el nombre de un equipo de la NBA (que es su clave primaria), ¿qué sucede con los jugadores? También habrá que cambiar el nombre del equipo de los jugadores, puesto que el campo Nombre_Equipo es una clave ajena.

25. Borrado de registros con relaciones

En este punto, hay que recordar las cláusulas REFERENCES de la sentencia CREATE TABLE para crear las relaciones de clave foránea-clave primaria de alguna columna de una tabla:

definición referencia:

```
REFERENCES nombres-tabla
[(nombre_columna, ...)] ON DELETE opción_referencia] [ON UPDATE
opción_referencia]
opción_referencia: CASCADE | SET NULL | NO ACTION
```

26. Borrado y modificación de registros con relaciones

La cláusula ON DELETE personalizan el comportamiento del borrado

Si por ejemplo, se intenta eliminar un registro con otros registros relacionados, y se ha seleccionado la opción por defecto el comportamiento sería el siguiente:

Opción por defecto

Dos tablas relacionadas

```
CREATE TABLE clientes ( dni varchar(15) PRIMARY KEY, nombre
varchar(50), direccion varchar(50) );

CREATE TABLE pagos_pendientes(
dni varchar(15),
importe decimal,
CONSTRAINT FK_CLIENTE FOREIGN KEY(dni) REFERENCES clientes(dni));
```

27. Borrado y modificación de registros con relaciones

Insertamos datos

```
INSERT INTO clientes VALUES ('5555672L','Pepe Cifuentes','C/Los
almendros,23');
INSERT INTO pagos_pendientes VALUES ('5555672L',500);
INSERT INTO pagos_pendientes VALUES ('5555672L',234.5);
```

Si intentamos borrar

```
DELETE FROM clientes WHERE dni='5555672L';
```

Da un error;

```
Informe de error -
ORA-02292: integrity constraint (CICLISMO.FK_CLIENTE) violated -
```

28. Borrado y modificación de registros con relaciones

Opción CASCADE

Volvemos a crear la tabla e insertar los datos

```
CREATE TABLE pagos_pendientes(
dni varchar(15),
importe decimal,
CONSTRAINT FK_CLIENTE FOREIGN KEY(dni) REFERENCES clientes(dni)
ON DELETE CASCADE);

INSERT INTO clientes VALUES ('5555672L','Pepe Cifuentes','C/Los
almendros,23');
INSERT INTO pagos_pendientes VALUES ('5555672L',500);
INSERT INTO pagos_pendientes VALUES ('5555672L',234.5);
```

29. Borrado y modificación de registros con relaciones

Intentamos borrar:

```
DELETE FROM clientes WHERE dni='5555672L';
```

Se borran también los pagos

```
select * from pagos_pendientes;
```

Opción SET NULL

```
CREATE TABLE pagos_pendientes(
dni varchar(15),
importe decimal,
CONSTRAINT FK_CLIENTE FOREIGN KEY(dni) REFERENCES clientes(dni)
ON DELETE SET NULL);
```

30. Borrado y modificación de registros con relaciones

Volvemos a insertar los datos

```
INSERT INTO clientes VALUES ('5555672L','Pepe Cifuentes','C/Los
almendros,23');
INSERT INTO pagos_pendientes VALUES ('5555672L',500);
INSERT INTO pagos_pendientes VALUES ('5555672L',234.5);
```

Y borramos

```
DELETE FROM clientes WHERE dni='5555672L';
```

Comprobamos que el campo dni de pagos_pendientes está a null

```
select * from pagos_pendientes;
```

31. Borrado y modificación de registros con relaciones

Volvemos a insertar los datos

```
INSERT INTO clientes VALUES ('5555672L','Pepe Cifuentes','C/Los
almendros,23');
INSERT INTO pagos_pendientes VALUES ('5555672L',500);
INSERT INTO pagos_pendientes VALUES ('5555672L',234.5);
```

Y borramos

```
DELETE FROM clientes WHERE dni='5555672L';
```

Comprobamos que el campo dni de pagos_pendientes está a null

```
select * from pagos_pendientes;
```

En Oracle, solo existe la cláusula ON DELETE. Para implementar ON UPDATE, se debe generar un TRIGGER (se verá más adelante)

32. Transacciones

Una **transacción** es un conjunto de sentencias SQL que se tratan como una sola instrucción (atómica).

Una transacción puede ser **confirmada (commit)**, si todas las operaciones individuales se ejecutaron correctamente, o, **abortada (rollback)** a la mitad de su ejecución si hubo algún problema (por ejemplo, el producto pedido no está en stock, por tanto no se puede generar el envío).

Trabajar con transacciones puede ser esencial para mantener la integridad de los datos. Por ejemplo, se puede dar el caso de que se descuenta el stock de un producto antes de proceder a su envío, pero cuando se va a generar la cabecera del pedido, la aplicación cliente sufre un corte en las comunicaciones y no da tiempo a generarlo. Esto supone una pérdida de stock.

La transacción garantiza la atomicidad de la operación: O se hacen todas las operaciones, o no se hace ninguna.

33. Transacciones

Para iniciar una transacción en Oracle

```
SET AUTOCOMMIT OFF
```

Para terminar una transacción, hay que aceptar, o rechazar los cambios mediante:

```
#La palabra clave WORK es opcional
COMMIT WORK; #Acepta los cambios.
ROLLBACK WORK; #Cancela los cambios
```

Cualquier conjunto de sentencias SQL se considera cancelado si termina abruptamente la sesión de un usuario sin hacer **COMMIT**.

34. Transacciones

Ejemplo de transacción:

```
SET AUTOCOMMIT 0;
#se actualiza el stock
UPDATE Productos SET Stock=Stock-2 WHERE CodigoProducto='AAAF102';
#se inserta la cabecera del pedido
INSERT INTO Pedidos VALUES (25,now(),'Francisco Garcia','Pendiente
de Entrega');
#se inserta el detalle del pedido
INSERT INTO DetallePedidos (CodigoPedido,CodigoProducto,Unidades)
VALUES (25,'AAAF102',2);
#aceptar transacción
COMMIT WORK;
```

35. Transacciones

Insertamos en la BBDD de Productos una nueva entrada.

```
INSERT INTO PRODUCTOS (CODPROD, CATEGORIA, NOMBRE, PRECIO, FECHA,
IMPORTADO, ORIGEN) VALUES ('AR52', 'FERRETERÍA', 'DESTORNILLADOR',
'10,12', TO_DATE('2000-10-22 00:00:00', 'YYYY-MM-DD HH24:MI:SS'),
'N', 'ITALIA’)
```

Después de observar que se ha insertado correctamente la nueva fila, cerramos la aplicación del SQL Developer.

¿El dato insertado permanecerá en la base de datos cuando volvemos a entrar?

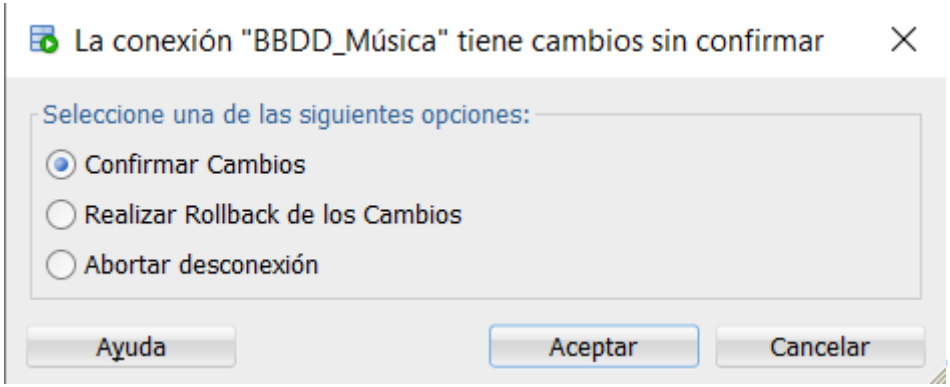
36. Transacciones

Por Defecto

Las opciones de confirmación automática están deshabilitadas.

```
SET AUTOCOMMIT 0;
SET AUTOCOMMIT OFF;
```

Cuando intentamos cerrar el SQL Developer nos sale la siguiente advertencia:

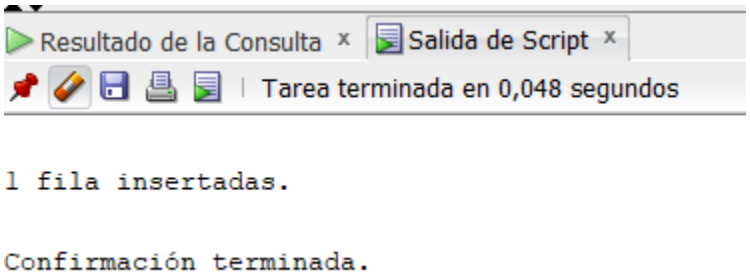


37. Transacciones

Con las opciones de confirmación automática habilitadas.

```
SET AUTOCOMMIT 1;

SET AUTOCOMMIT ON;
```

Cuando hacemos un INSERT, UPDATE o DELETE aparecerá un mensaje indicándonos que la sentencia se ha realizado correctamente y se ha confirmado.



38. Transacciones

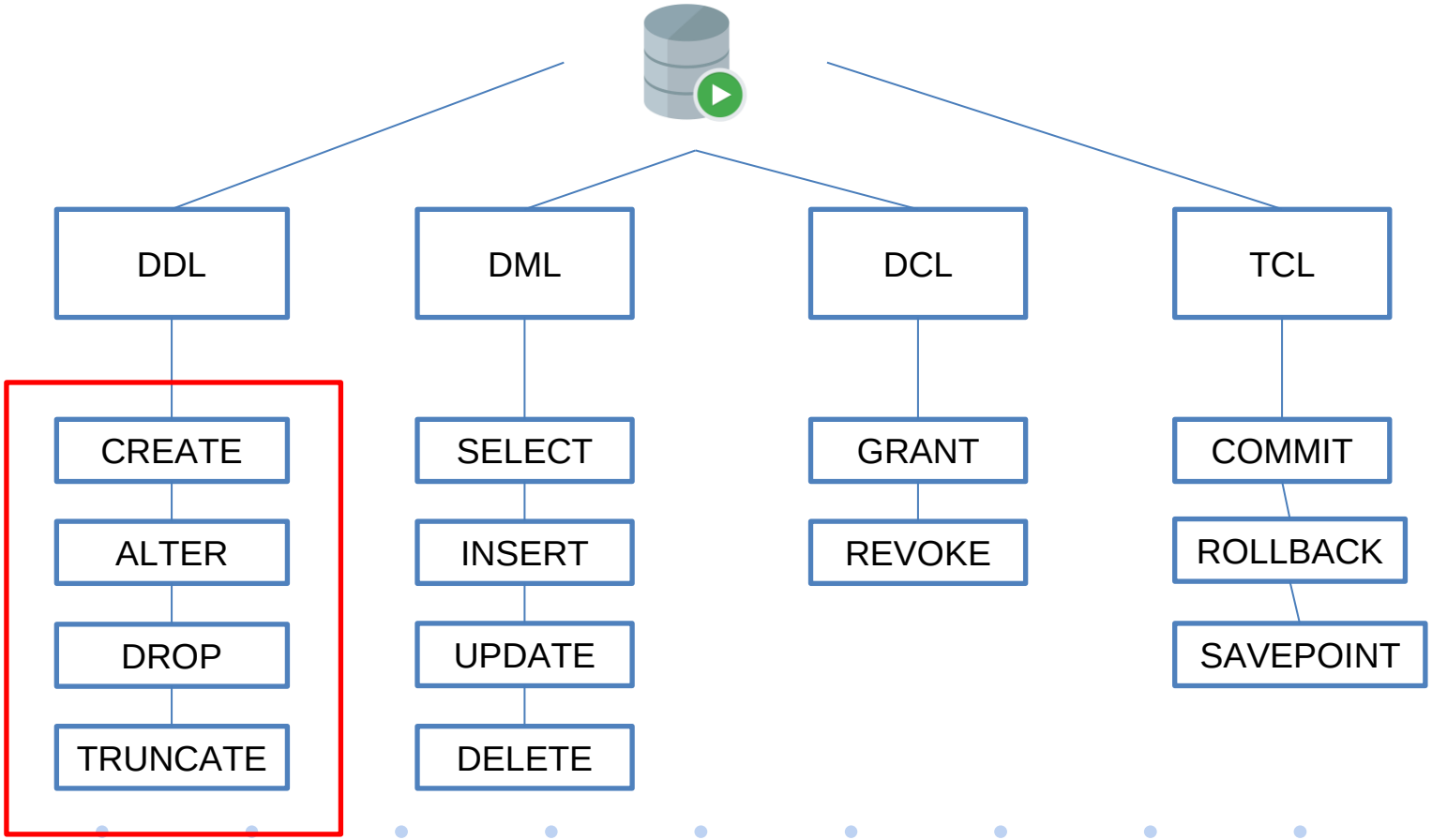
SAVEPOINT

Sirve para marcar un punto de referencia en la transacción para hacer un ROLLBACK parcial.

Por ejemplo:

```
UPDATE PRODUCTOS SET CATEGORIA = 'DEPORTES' WHERE CODPROD = 'AR59';

SAVEPOINT solouno;

UPDATE PRODUCTOS SET CATEGORIA = 'FERRETERÍA';

SAVEPOINT todos;

ROLLBACK TO SAVEPOINT solouno;
```

39. Grupo de Comandos.



40. La sentencia TRUNCATE TABLE.

La sentencia **TRUNCATE** de SQL permite eliminar todas las filas de una tabla o las particiones de especificadas de una tabla.

```
TRUNCATE TABLE table_name;
```

Ejemplo:

Queremos vaciar el contenido de la tabla de productos nuevos.

```
TRUNCATE TABLE PRODUCTOS_NUEVOS;
```

41. La sentencia DROP TABLE.

La sentencia **DROP TABLE** de SQL permite eliminar una tabla de una base de datos.

```
DROP TABLE table_name;
```

Ejemplo:

Queremos eliminar por completo la tabla productos nuevos.

```
DROP TABLE PRODUCTOS_NUEVOS;
```

42. La sentencia DROP TABLE.

¿Qué pasa si la tabla Productos_Nuevos tiene una o varia restricciones de clave ajena asociadas a ellas?

SQL Developer nos devuelve un error.

Para eliminar una tabla independientemente de las restricciones que le afectan deberemos añadir lo siguiente:

```
DROP TABLE table_name CASCADE CONSTRAINTS;
```

43. La sentencia ALTER TABLE.

La sentencia **ALTER TABLE** de SQL permite alterar o modificar una definición de tabla mediante la alteración, adición o retirada de columnas y restricciones.

```
ALTER TABLE tabla {ADD | MODIFY | DROP}
        ({columna tipoColumna [NOT NULL],}+);
```

Ejemplos:

Queremos añadir una nueva columna de recaudado de tipo numérico real a la tabla de Disco.

```
ALTER TABLE DISCO ADD recaudado FLOAT;
```

44. La sentencia ALTER TABLE.

Ejemplos:

- Añadir dos nuevas columnas (formato y premio) de tipo cadena a la tabla Disco.

```
ALTER TABLE DISCO ADD (formato VARCHAR(20), premio VARCHAR(10));
```

- Cambiar el tamaño de la columna premio para que solo permita introducir cadenas formadas por 5 caracteres en la tabla Disco.

```
ALTER TABLE DISCO MODIFY precio VARCHAR(5);
```

- Eliminar la columna formato de la tabla Disco.

```
ALTER TABLE DISCO DROP COLUMN formato;
```

45. La sentencia ALTER TABLE.

Ejemplos:

- Modificar la columna premio de la tabla Disco para añadirle la restricción NOT NULL.

```
ALTER TABLE DISCO MODIFY premio NOT NULL;
```

¿Qué tenemos que tener en cuenta cuando añadimos una restricción NOT NULL?

46. La sentencia ALTER TABLE.

Ejemplos:

- Modificar la columna premio de la tabla Disco para añadirle la restricción NOT NULL.

```
ALTER TABLE DISCO MODIFY premio NOT NULL;
```

¿Qué tenemos que tener en cuenta cuando añadimos una restricción NOT NULL?

Sólo podrá realizarse cuando la tabla en cuestión no tenga ninguna fila con valor nulo en la columna en cuestión.

47. La sentencia ALTER TABLE.

Ejemplos:

- Añadirle la restricción de clave primaria al campo codprod de la tabla Productos.

```
ALTER TABLE PRODUCTOS MODIFY CODPROD PRIMARY KEY;
```

- Añadirle la restricción de clave ajena al campo codprod de la tabla ProductosPedidos hacia la tabla Productos.

```
ALTER TABLE PRODUCTOSPEDIDOS ADD CONSTRAINT PK_Person3
        FOREIGN KEY (codprod) REFERENCES PRODUCTOS(codprod);
```

48. La sentencia ALTER TABLE.

Ejemplos:

- Quitar las dos restricciones anteriores de clave primaria y clave ajena.

```
ALTER TABLE PRODUCTOS DROP PRIMARY KEY;

ALTER TABLE PRODUCTOSPEDIDOS DROP CONSTRAINT PK_Person3;
```

49. La sentencia ALTER TABLE.

Ejemplos:

- Añadir la restricción de borrado en cascada a la clave ajena anterior.

```
ALTER TABLE PRODUCTOSPEDIDOS ADD CONSTRAINT PK_Person3 FOREIGN KEY (codprod)
REFERENCES PRODUCTOS(codprod) ON DELETE CASCADE;
```