

Sesión 3 – CREATE TABLE, DROP TABLE, ALTER TABLE, INSERT, UPDATE y DELETE COMMIT, ROLLBACK y SAVEPOINT

A continuación se muestran 3 sentencias muy útiles a la hora de trabajar con Oracle. En nuestro caso, el uso de SQLDeveloper nos permite obtener esta misma información moviéndonos a través de sus pestañas y despleables.

TABLAS DEFINIDAS POR UN USUARIO

SELECT table_name FROM user_tables / SELECT * FROM user_tables

INFORMACIÓN sobre una TABLA

DESC nombre_tabla

INFORMACIÓN sobre las RESTRICCIONES que tiene una TABLA

Con **SELECT * FROM user_constraints** obtenemos todas las restricciones definidas por un usuario.

El resultado muestra una tabla con una serie de columnas. Las columnas que nos interesan son

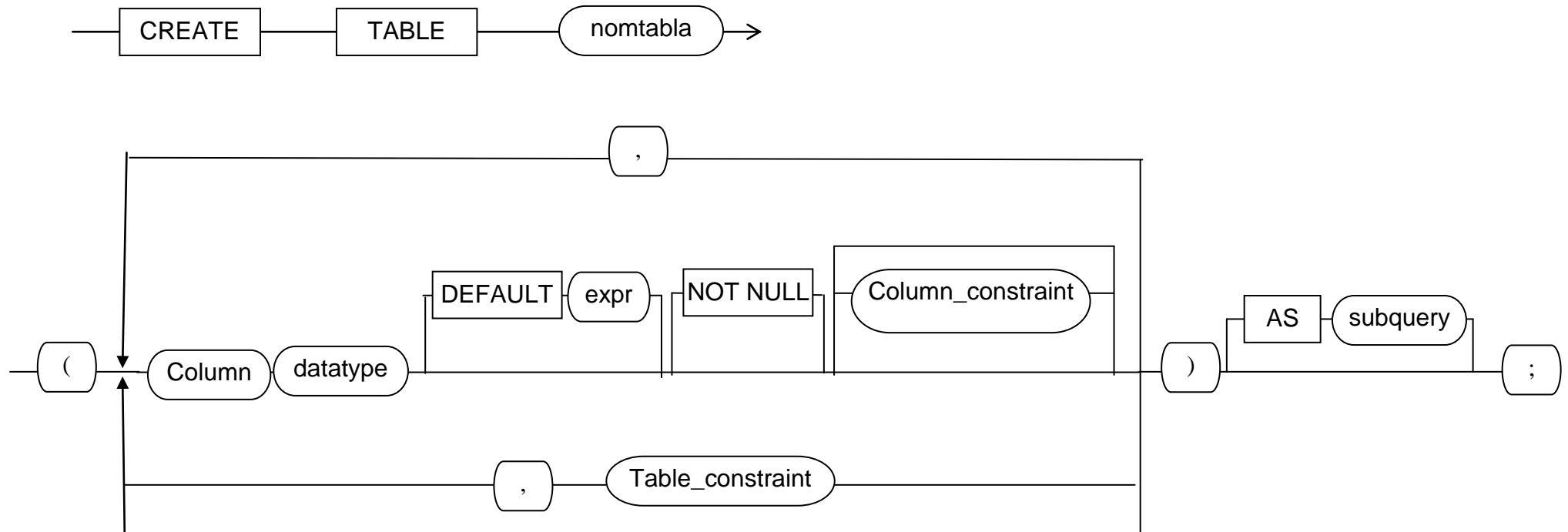
OWNER	CONSTRAINT_NAME	CON	TABLE_NAME	SEARCH_CONDITION	R_OWNER	R_CONSTRAINT_NAME	...
usuBD2	SYS_C006505	R	T4		BD2	PKT	...
usuBD2	SYS_C006504	P	T4				...
usuBD2	PKT	P	T				...
usuBD2	SYS_C006502	C	T	c between 2 and 4			...

- Donde owner es el usuario que ha definido la restricción.
- Constraint_name es el nombre que tiene la restricción, será un nombre dado por el usuario, o en caso de que éste lo haya omitido lo habrá dado el sistema (aparece SYS en el nombre)
- Constraint_type es una columna donde pueden aparecer los siguientes valores: P para indicar que es una restricción de clave primaria, R para indicar que es de clave ajena, o C para indicar que es de tipo CHECK.
- Table_name es la tabla sobre la que está definida esa restricción.
- Search_condition es la condición impuesta en caso de que sea una restricción CHECK.
- R_Constraint_name en las restricciones de clave ajena se refiere al nombre de la restricción de clave primaria de la tabla referenciada.

Para consultar las restricciones se pueden ejecutar una SELECT sobre la tabla, e imponer condiciones con los operadores que se conocen.

SELECT * FROM user_constraints WHERE CONSTRAINT_NAME LIKE 'L%'

CREATE TABLE para crear una tabla

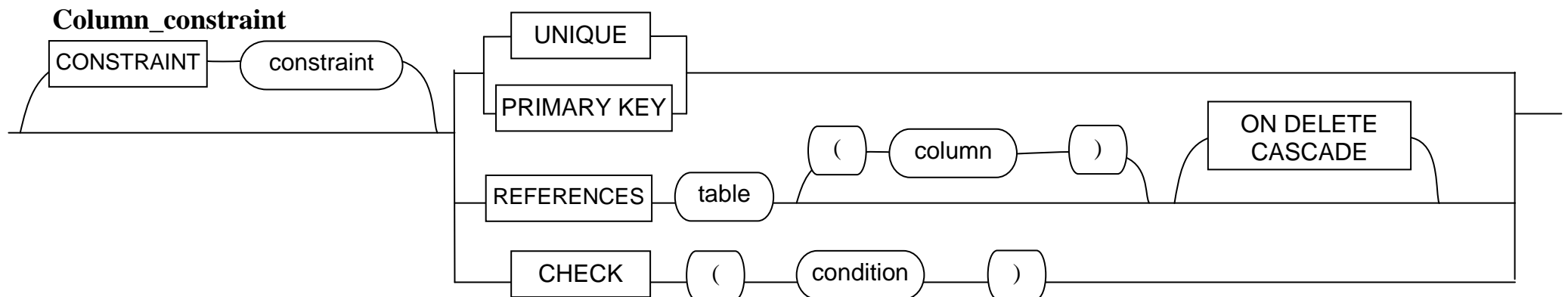


donde:

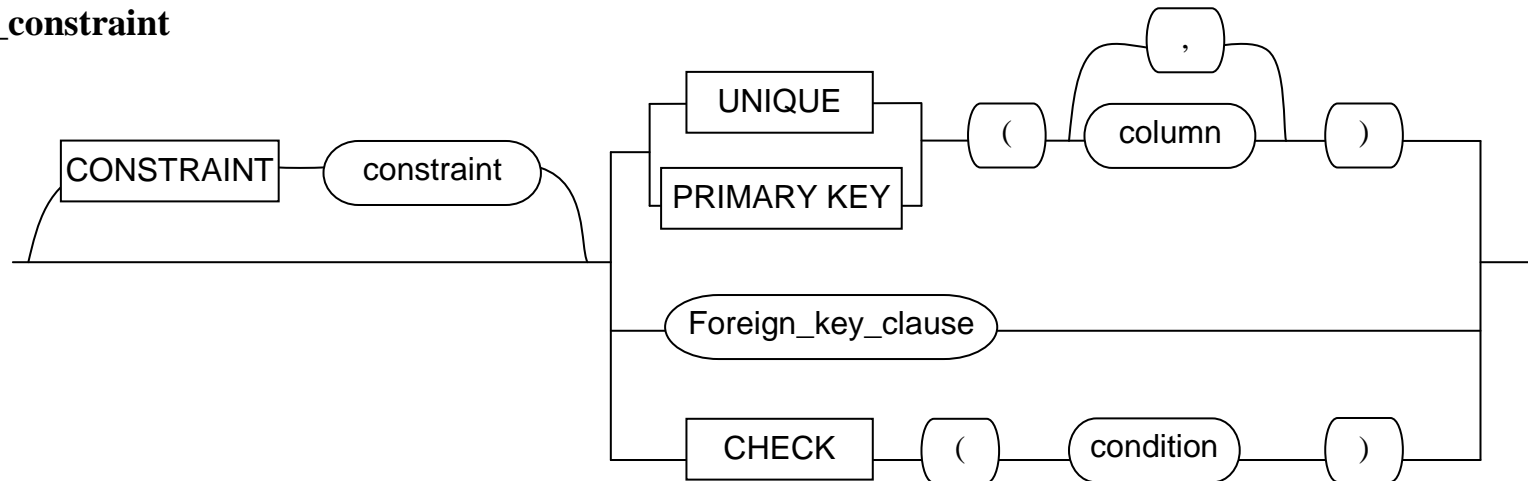
- nomtabla: Es el nombre de una tabla.
- column: Es el nombre de una columna de la tabla.
- datatype: Especifica el tipo de datos de la columna pudiendo ser éste uno de los definidos por el sistema (se pueden encontrar en la ayuda, normalmente utilizaremos char(n), varchar2(n), number(n,m), integer, date)
- Table-constraint /Column-constraint: Restricciones a nivel de tabla (table-constraint) o de columna (column-constraint).

Se puede definir: una CLAVE PRIMARIA por tabla, restricciones UNIQUE, CLAVES AJENAS, una restricción DEFAULT por columna, y restricciones CHECK. Todas ellas pueden ser introducidas en la misma sentencia, según la sintaxis que se muestra en la siguiente página. Se debe tener en cuenta los siguientes aspectos:

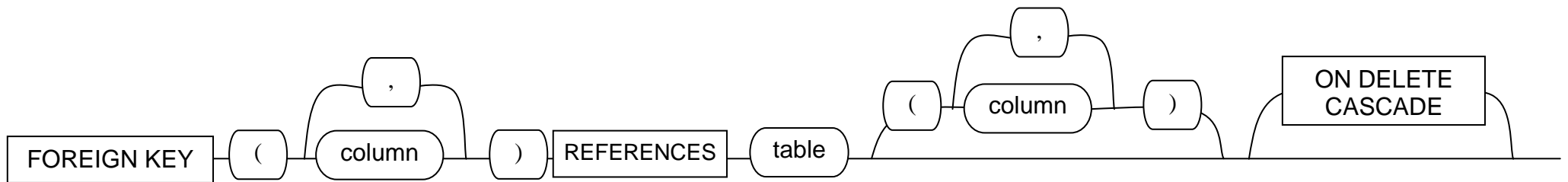
- ~ Las columnas definidas como clave primaria no podrán contener nulos y no habrá combinaciones repetidas de sus valores.
- ~ La combinación de valores de las columnas enumeradas en un UNIQUE no se podrán repetir. Las columnas que tengan una restricción UNIQUE sí pueden contener nulos.
- ~ Al definir la integridad referencial, los tipos de datos de las columnas que constituyen la clave ajena y los de las columnas referenciadas deben ser compatibles. Cuando la restricción se está definiendo a nivel de columna se omiten las palabras FOREIGN KEY y la enumeración de columnas.
- ~ La restricción DEFAULT especifica el valor que se le coloca a una columna por defecto cuando al insertar una fila en la tabla esa columna carece de valor explícito.
- ~ La restricción CHECK limita los posibles valores que pueden tomar una columna o conjunto de columnas. Se pueden definir múltiples restricciones CHECK para una tabla, sin embargo sólo se puede definir una a nivel de columna. Cuando para una columna existe más de una restricción CHECK se evalúan todas. La restricción CHECK que afecte a más de una columna debe ser definida a nivel de tabla.
- ~ En la sintaxis que se muestra a continuación, donde pone "CONSTRAINT constraint", se refiere a que a las restricciones se les da un nombre. Si el nombre no lo proporcionamos nosotros, el sistema le da automáticamente uno. Si queremos darle nosotros un nombre a la restricción, pondremos la palabra CONSTRAINT y a continuación el nombre que queramos. El hecho de que un usuario dé nombre a las restricciones es para facilitar posteriormente el trabajo con las mismas, por lo tanto el usuario debe definirse un criterio a la hora de definirse los nombres.



Table_constraint



Foreign_key_clause



Ejemplo:

Para ilustrar la sintaxis, se muestran las restricciones definidas de distinto modo en cada tabla para así ilustrar distintos modos de especificarlas.

```
CREATE TABLE pieza (numpieza number(10) CONSTRAINT cppiezas PRIMARY KEY,  
                    nompieza varchar2(30),  
                    preciovent number(6) CONSTRAINT chprecioven CHECK (preciovent >0));  
  
CREATE TABLE vendedor (numvend number(5),  
                        nomvend varchar2(30), nombrecomer varchar2(30), direccion varchar2(30),  
                        telefono varchar2(15), poblacion varchar(20), provincia varchar2(20),  
                        PRIMARY KEY(numvend));  
  
CREATE TABLE preciosum (numvend integer, numpieza varchar2(16) CONSTRAINT fkprepie REFERENCES pieza,  
                        preciounit number(6), diassum number(3) DEFAULT 2 CONSTRAINT chdiassum CHECK (diassum>0),  
                        CONSTRAINT fkpreven FOREIGN KEY (numvend) REFERENCES vendedor,  
                        CONSTRAINT cppreciosum PRIMARY KEY (numpieza, numvend));
```

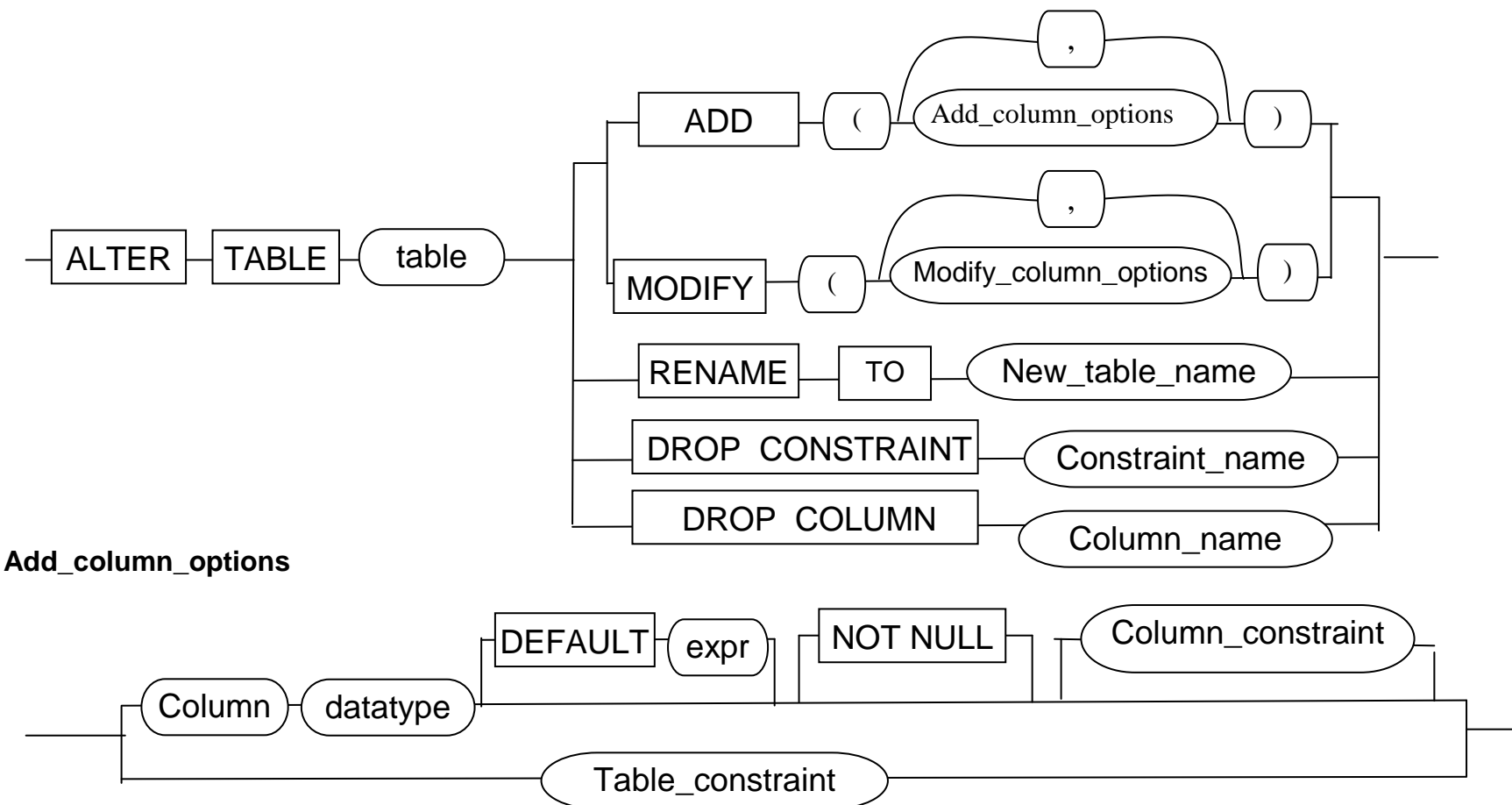
DROP TABLE para borrar una tabla



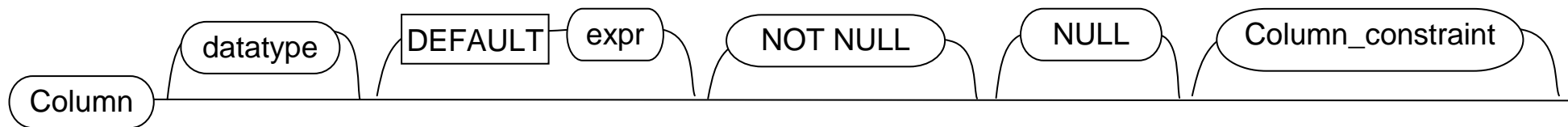
En muchas ocasiones para borrar las tablas se deberá seguir un orden adecuado, ya que puede que no se nos permita el borrado de alguna tabla para no interferir en la integridad referencial de la base de datos.

ALTER TABLE para modificar la estructura y restricciones de una tabla

Después de crear las tablas se puede modificar su estructura con la sentencia ALTER TABLE que nos permite añadir o borrar una nueva columna a una tabla, así como añadir o borrar restricciones a las columnas de la tabla.



Modify_column_options



Ejemplo:

```
ALTER TABLE VENDEDOR ADD fnacimiento date;
```

```
ALTER TABLE PIEZA MODIFY numpieza not null;
```

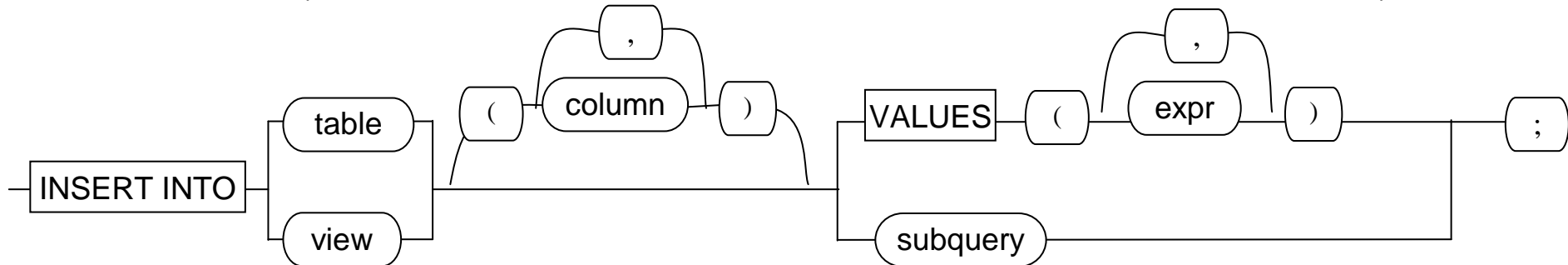
Suponiendo que existe una tabla DELUJO con una columna llamada numpieza

```
ALTER TABLE DELUJO ADD constraint lujopieza foreign key(numpieza) references pieza;
```


INSERT para insertar filas en una tabla

Si no se especifica la lista de columnas se supondrá que se va a insertar una fila con valor para todas las columnas de la tabla.

Se debe tener en cuenta que el orden de la lista de valores debe coincidir con el orden dado a las columnas, si se enumeran, o con el que tienen en la tabla.

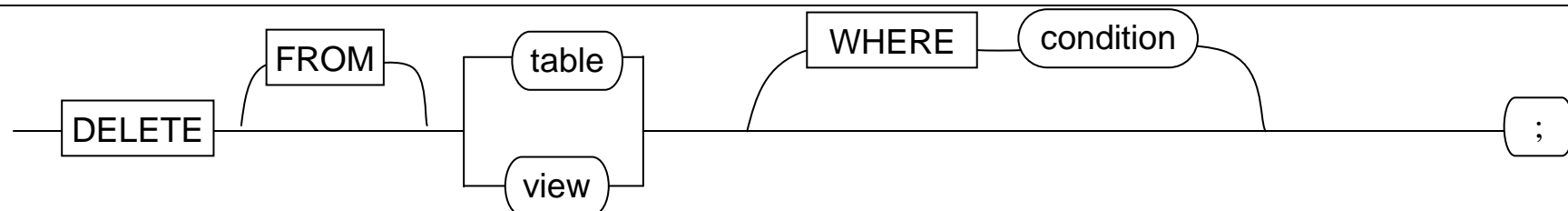


INSERT INTO pieza VALUES ('A25D', 'DESTORNILLADOR MULTIPLE', 1500);

INSERT INTO pieza(numpieza) VALUES ('A29C');

INSERT INTO subpiezas(num, precio) SELECT numpieza, preciovent FROM pieza WHERE preciovent>150;

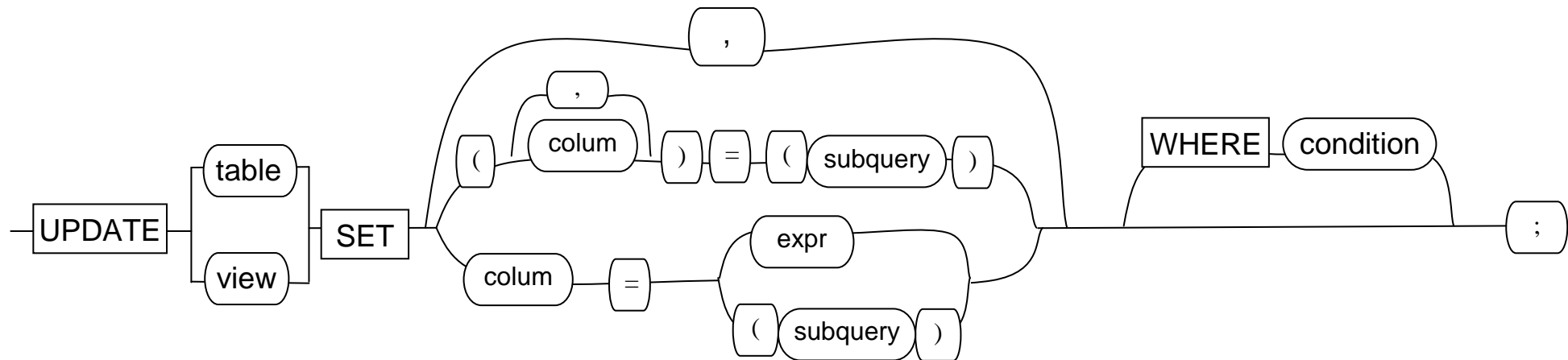
DELETE para borrar filas de una tabla



Si no se especifica ninguna condición (WHERE), se borrarán todas las filas de la tabla o vista.

Ejemplos
 DELETE FROM pieza WHERE numpieza='A29D'
 DELETE FROM pieza WHERE nompieza LIKE '%DESTORNILLADOR%'

UPDATE para modificar los datos de una tabla



Al igual que en el caso de la sentencia DELETE, si no se especifica ninguna condición, la modificación afectará a todas las filas de la tabla.

Ejemplo

```
UPDATE piezas
SET precioent= 2000
WHERE numpieza='A29D';
```

```
UPDATE vendedor
SET telefono='6565656565'
WHERE nombrecomer LIKE 'HARW%';
```

```
UPDATE vendedor
SET direccion='OLMO,19',
    telefono='666567686'
WHERE numvend= 2;
```

```
UPDATE PIEZA
SET precioent=(select max(precioent)
                FROM pieza)
WHERE numpieza='A-1001-L';
```

COMMIT, ROLLBACK y SAVEPOINT: concepto de transacción

TRANSACCION:

- Conjunto de sentencias SQL que son tratadas por el SGDBR como unidad. **O todas son validadas o ninguna es validada.**
- El inicio lo marca la primera sentencia DML (Data Manipulation Language, es decir, inserts, updates, etc.) desde el último commit o rollback.
- *COMMIT* : hace permanentes los cambios producidos por la transacción.
- *ROLLBACK*: anula los cambios hechos por la transacción.
- Los cambios hechos por una transacción pendiente de COMMIT o ROLLBACK son visibles sólo por la sesión que ejecutó la transacción.
- Las sentencias DDL (Data Definition Language, es decir, creates, etc) llevan commit implícito.
- Si se produce una violación de constraint o cualquier otro error el SGBDR hace un ROLLBACK automático.

COMMIT, ROLLBACK y SAVEPOINT: concepto de transacción

Update Clientes set domicilio = null
where codcli = '4536';

En este momento este usuario ve la dirección del cliente a nulo, pero el resto de sesiones todavía la ve sin cambios

Update facturas set Euros = pts*166.337
where pts is not null;

Delete from facturas
where Euros is null;

Si violasen alguna constraint o hubiese algún error la transacción Ejecutaría un rollback automático y terminaría

Commit;

La transacción se ha consolidado y ya todos ven la dirección a nulo. Ya no tiene posibilidad de rollback

COMMIT, ROLLBACK y SAVEPOINT: concepto de transacción

Update Clientes set domicilio = null
where codcli = '4536';

Update facturas set Euros = pts*166.337
where pts is not null;


Es el usuario el que provoca el rollback, con que se anulan todas las operaciones pendientes y comienza una nueva transacción



rollback;

Delete from facturas
where Euros is null;

La transacción se ha consolidado pero SÓLO se produce el borrado, los dos Updates anteriores ya estaban anulados



Commit;

COMMIT, ROLLBACK y SAVEPOINT: concepto de transacción

La sentencia SavePoint permite dividir una transacción grande en varios trozos. De esta forma si cometemos un error no tenemos que deshacer toda la transacción, sólo hasta un save point

Update

insert;

Insert;

....

SavePoint P1

Definimos SavePoint



insert ...;

Delete

Rollback to P1

Deshacemos SÓLO las dos últimas sentencias

insert

COMMIT

NO incluye las sentencias entre SavePoint P1 y el ROLLBACK P1

Utilidad práctica de los conceptos vistos en la sesión

- Sentencias Create table, drop table, y alter table:

Son fundamentales para crear, borrar o modificar la estructura de la información de nuestro modelo de datos.

- Commit, Rollback y Savepoint:

Nos dan un mecanismo para garantizar la integridad de la información en nuestra base de datos cuando realizamos modificaciones de información delicados o complejos.

Ejemplo:

Realización de una transferencia de una cuenta a otra en un banco.

- Sentencia 1: Obtener la cantidad a transferir.
- Sentencia 2: Modificar la cuenta origen, restándole el importe a transferir.
- Sentencia 3: Modificar la cuenta destino, sumándole el importe a transferir.

¿Qué ocurriría si la cuenta destino del dinero estuviese bloqueada, y no permitiese ingresos? La cuenta origen quedaría con el saldo mermado con la cantidad que queríamos transferir., y el dinero quedaría en el "limbo".

Usando la sentencia Rollback evitaríamos esa inconsistencia en los datos.