UNIDAD DE TRABAJO 4: DEFINICIÓN de Datos

Conceptos

- ~~Introducción~~
- Objetos de la base de datos
- Tipos de datos
- La sentencia CREATE TABLE
  - Referencias a tablas de otros usuarios
  - La opción DEFAULT
- Tablas de una base de datos Oracle
  - Consultas al Diccionario de Datos
- Tipos de datos
- Creación de una tabla por medio de una subconsulta
- La sentencia ALTER TABLE
  - Añadir una columna
  - Modificar una columna
  - Eliminación de una columna
  - Opción SET UNUSED
- La sentencia DROP TABLE
- Cambiar el nombre a una tabla
- Truncar una tabla
- Añadir comentarios a una tabla
- ¿Qué son las restricciones?
- Definición de restricciones
  - La restricción NOT NULL
  - La restricción UNIQUE Key
  - La restricción PRIMARY KEY
  - La restricción FOREIGN KEY
  - La restricción CHECK
- Operaciones con restricciones
  - Añadir una restricción
  - Eliminar una restricción
  - Desactivar restricciones
  - Activar restricciones
  - Consulta de restricciones en el Diccionario de Datos
- Vistas

1. Introducción

El lenguaje SQL (*Structured Query Language*) permite la comunicación con el SGBD. Fue desarrollado sobre un prototipo de SGBD relacional denominado SYSTEM R y diseñado por IBM a mediados de los años setenta. En 1979 Oracle Corporation presentó la primera implementación comercial de SQL, que estuvo disponible antes que otros

productos de IBM. Por su parte, IBM desarrolló productos herederos del prototipo SYSTEM R, como DB2 y SQL/DS.

El instituto ANSI (*American National Standard Institute*) adoptó el lenguaje SQL como estándar para la gestión de bases de datos relacionales en octubre de 1986. En 1987 lo adopta ISO (*International Standardization Organization*).

SQL es tan conocido en bases de datos que muchos lenguajes de programación incorporan sentencias SQL como parte de su repertorio; tal es el caso de Visual Basic para acceder a bases de datos relacionales.
Entre las principales características de SQL podemos destacar:

- Es un lenguaje para todo tipo de usuarios (administradores, desarrolladores y usuarios normales), el usuario que emplea SQL especifica qué quiere, no cómo ni dónde, permite hacer cualquier consulta de datos.
- Es un lenguaje no procedimental. Permite al usuario solicitar un resultado sin preocuparse de los medios necesarios para obtener dicho resultado.
- Permite manipular un registro o un conjunto de registros , por tanto, no son necesarias estructuras de control.
- Universal, puede utilizarse en todos los niveles de la gestión de una bbdd, administración de la bbdd, desarrollo, gestion de datos,etc

Podemos clasificar las sentencias SQL en 2 tipos:

- Sentencias DDL (*Data Definition Language*). Con este lenguaje se crea y mantiene la estructura de la bbdd. Sirve para realizar las siguientes tareas:

    - Crear un objeto de la bbdd, tablas, vistas, procedimientos...(Create)
    - Eliminar un objeto de la bbdd (Drop)
    - Modificar un objeto de la bbdd (Alter)
    - Conceder privilegios sobre un objeto de la bbdd (Grant)
    - Retirar privilegios sobre un objeto de la bbdd. (revoke)

- Sentencias DML (Data Manipulation Language) Conjunto de sentencias sirve para manipular los datos contenidos en la bbdd. Es posible hacer:

    - Insert. Insertar filas en la bbdd
    - Update. Actualizar filas de datos
    - Delete. Eliminar filas de datos.
    - Select. Recuperar filas de datos.
    - Bloqueo de tablas

Existen otro grupo de sentencias SQL no tan extendidas como las anteriores y son:

- Lenguaje de control de transacción, gestiona las modificaciones realizadas por las instrucciones DML:

    - Características de la transacción
    - Validación y anulación de la modificación

COMMIT, SAVEPOINT, ROLLBACK, SET TRANSACTION

En la presente unidad se va a contemplar el sublenguaje DDL (*Data Definition Language*) de SQL. Se mostrará cómo crear los objetos principales del modelo relacional: las tablas, así como las restricciones de los datos. Se cierra la línea que iniciamos en la unidad anterior iniciada en el diseño conceptual, seguida del diseño lógico, terminando en el diseño físico con la creación de las tablas y las restricciones en el SGBDR Oracle.

Por último, indicar que las sentencias DDL tienen incluido el comando COMMIT, que finaliza y lleva a cabo todas las transacciones pendientes hasta el lanzamiento de una sentencia DDL.

2. Objetos de la base de datos

Una base de datos Oracle puede contener múltiples estructuras de datos. Cada estructura debería definirse en el diseño de la base de datos, para que pueda ser creada durante la etapa de construcción del desarrollo de la base de datos.

Principales objetos:

- **Tabla** → Unidad básica de almacenamiento, compuesta de registros y columnas.
- **Vista** → representación lógica de un subconjunto de una o mas tablas. Puede manipularse como una tabla (tabla virtual)
- **Secuencia** → Genera valores para la clave primaria. (Similar a campo autonumérico de Access)
- **Índice** → Columna o conjunto de columnas que permiten realizar búsquedas más rápidas. Sirven para mejorar el rendimiento de algunas consultas, en las que se incluya.
- **Sinónimo** → Da nombres alternativos a los objetos de almacenamiento. Nombres alternativos a tablas o vistas.

Notas sobre tablas en Oracle:

- ✓ Pueden ser creadas en cualquier momento, incluso mientras los usuarios usan la BD.
- ✓ No se necesita especificar el tamaño de ninguna tabla. El tamaño es definido por la cantidad de espacio en la BD según va creciendo. Es importante, sin embargo, estimar cuánto espacio utilizará la tabla.
- ✓ La estructura de una tabla puede ser modificada on-line.

Reglas para los nombres de tablas columnas y demás objetos de la BD Oracle:

- ❖ Deben comenzar por una letra.
- ❖ Pueden tener una longitud de 1-30 caracteres de largo.
- ❖ Caracteres permitidos: A-Z, a-z, 0-9, _, $, # (No valen los espacios)
- ❖ No deben duplicar el nombre de otro objeto que sea propiedad del mismo usuario.
- ❖ No debe ser una palabra reservada del servidor Oracle.

Nota: los nombres no son sensibles a mayúsculas/minúsculas. Es lo mismo DEPT que dept que Dept.

3. Tipos de datos

CHAR(tamaño)	cadena de caracteres de longitud fija con longitud "tamaño". Por defecto vale 1. 1 >= tamaño <= 2000 Si se introduce una cadena de menor longitud a la establecida lo rellena con espacios en blanco en la derecha.
VARCHAR2(tamaño)	cadena de caracteres de longitud variable, máximo el "tamaño". 1 >= tamaño <= 4000 Apellido Varchar2(20) Si se introduce una cadena superior al tamaño definido Oracle dará un error
NUMBER(P,S) NUMBER (n) NUMBER	Representa datos numericos tanto enteros como reales y con signo. Cuando especifico p,s es porque puedo poner máximo número de dígitos p (entre 1 y 38) S representa el número de dígitos de la parte fraccionaria (hasta 127) Ejemplo: Sal_Medio NUMBER(9,2) podría guardar 4566770.86 Cuando pongo n, es por que trabajo con números enteros y en decimal supone 0 Salario Number(10) Cuando solo pongo NUMBER se guarda en decimal o entero según se teclee El carácter por defecto para separar la parte fija y la fraccionaria es el punto. Se puede modificar en NLS_NUMERIC_CHARACTERS(ALTER SESSION)
DATE	Fecha y horas. Para cada tipo Date se almacena Siglo/Año/Mes/día/Hora/Minutos/segundos El formato de la fecha está en NLS_DATE_FORMAT . Por defecto es dd//mm/yy Ejemplo: fecha_edición date
TIMESTAMP(P)	Datos tipo Date en los que podemos indicar la precisión para las fracciones de segundo. Por defecto es 6
BLOB	datos binarios, hasta 4 GBytes
BFILE	Datos binarios almacenados en archivos externos a la bbdd (4 GB máximo)
LONG	Cadenas de caracteres de longitud variable (2 GB máximo) Para almacenar textos grandes. Solo se puede definir una columna por tabla No pueden aparecer en restricciones de integridad (constraints) ni para indexar, ni en where,etc

RAW(N)	Datos binarios de longitud variable no interpretable por Oracle. La diferencia con los varchar2 es que maneja cadenas de caracteres en lugar de cadenas de n bytes (n <2000 bytes)
LONG RAW(N)	Datos binarios de longitud variable no interpretable por Oracle Se emplea para almacenamiento de graficos. Utiliza (4 Gb como máximo)

4. La sentencia CREATE TABLE

Esta sentencia forma parte del sublenguaje DDL de SQL y es utilizada para crear tablas. Tiene un efecto inmediato sobre la base de datos y graba información en el Diccionario de Datos.

Para crear una tabla, el usuario debe tener el privilegio CREATE TABLE y un área de almacenamiento para crear objetos

Es posible definir hasta 1000 columnas por tabla

Sintaxis:

CREATE [**GLOBAL TEMPORARY**] **TABLE** [schema**.**]nombretabla
(columna tipo_datos [**DEFAULT** expr] [NOT NULL]) [TABLESPACE espacio_de_tabla] ;

- GLOBAL TEMPORARY: especifica que la tabla es temporal y que su definición es visible para todas las sesiones. Los datos en una tabla temporal son visibles sólo para la sesión que inserta datos en la tabla.
- Schema. : propietario
- DEFAULT expr: específica un valor por defecto si se omite en la sentencia insert.
- NOT NULL la columna debe contener alguna información. Nunca puede ser nula cuando insertemos una nueva fila
- Se puede indicar el TABLESPACE donde se debe almacenar la tabla. Este debe existir. Si se omite se inserta en el tablespace que tenga asignado el usuario.

El usuario debe especificar el nombre de la tabla y los nombres y tipos de las columnas, siguiendo las reglas para la definición de nombres.

Ejercicio: Crear la tabla alumnos con num_mat, nombre, fecha_nac, dirección, localidad en el tablespace USER_DATA

```
CREATE TABLE ALUMNOS(
NUM_MAT NUMBER(6) NOT NULL,
NOMBRE VARCHAR2(15) NOT NULL,
```

```
FECHA_NAC DATE,
DIRECCION VARCHAR2(30),
LOCALIDAD VRCHAR2(15)
) TABLESPACE USER_DATA;
```

Referencias a tablas de otros usuarios:

Un esquema (*schema*) es una colección de objetos. Los objetos de un esquema son las estructuras lógicas que hacen referencia directa a los datos en una base de datos. Incluyen tablas, vistas, sinónimos, secuencias, procedimientos almacenados, índices, clusters y database links.

Las tablas propiedad de otros usuarios no pertenecen a nuestro esquema. Si deseamos hacer referencia a ellas, debemos anteponer el nombre del usuario propietario al de la tabla separando ambos vocablos con un punto.

Podremos acceder a esas tablas si el propietario lo permite con los permisos oportunos.

La opción DEFAULT:

Usando la opción DEFAULT se puede dar un valor por defecto a una columna. Esta opción previene la introducción de valores nulos en aquellas filas que no especifican un valor para esa columna. El valor por defecto puede ser un literal, una expresión o una función SQL, pero el valor no puede ser el nombre de otra columna. La expresión por defecto debe ser del mismo tipo que el de la columna.

Ejemplo:

```
CREATE TABLE dept2
      (numdept NUMBER(2),
      nombre dept VARCHAR2(14),
      ubicacion VARCHAR2(13));
```

5. Tablas de una base de datos Oracle

Las tablas de usuario son tablas creadas por el usuario.
Hay otro conjunto de tablas y vistas en una base de datos Oracle conocido como *Diccionario de Datos* (DD). Este conjunto de tablas es creado y mantenido por Oracle y contiene información sobre la base de datos.

Todas las tablas del DD son propiedad del usuario SYS.
Los usuarios raramente acceden a las tablas base porque la información que hay en ellas no es fácil de entender.
No se puede escribir directamente sobre él.
Los usuarios normalmente acceden a vistas del diccionario de datos porque aquí la información se presenta en un formato más fácil de entender.

La información que se almacena en el DD incluye nombres de los usuarios de Oracle Server, privilegios concedidos a los usuarios, nombres de los objetos de la BD, restricciones de integridad e información de auditoría.

Consultas al Diccionario de Datos:

Se consultan las tablas del DD para localizar información acerca de los objetos que contiene. Algunas de las tablas del diccionario utilizadas con mayor frecuencia son:

- USER_TABLES
- USER_OBJECTS
- USER_CATALOG: Tablas, vistas, sinónimos, y secuencias propiedad del usuario.

Ejemplos:

- Tablas propiedad del usuario:

select table_name from user_tables;
En este caso me visualiza los nombres de las tablas de usuario, es decir, mis tablas
select * from user_tables;
Visualiza toda la información posible de todas mis tablas

USER_TABLES es una vista cuyo propietario es SYS, por eso ponemos:
select table_name from sys.user_tables;

Visualiza el nombre de las tablas creadas igual que anteriormente.
También podemos poner:

Desc SYS.USER_TABLES; Me visualiza todos los campos de la vista USER_TABLES

- Distintos objetos propiedad del usuario:

select distinct object_type from user_objects;
En el caso de scott me dice que los únicos objetos que tiene son vistas y tablas

select object_type, object_name from user_objects;
Nos dice de cada objeto su nombre y el tipo que es.
select object_type, substr (object_name, 1, 10) from user_objects;
Como los nombres salían en un formato poco legible cogemos del nombre solo los 10 primeros caracteres.

- Tablas, vistas, sinónimos y secuencias propiedad del usuario:

select * from user_catalog;

6. Creación de una tabla por medio de una subconsulta

Un segundo método para crear una tabla, es aplicar una cláusula AS como subconsulta, para crear tanto la tabla, como para llenarla con los registros devueltos por la subconsulta.

Sintaxis:

CREATE TABLE [columna (,columna…)] **AS** subconsulta;

Ejemplo:

CREATE TABLE dept50 AS select * from dept where deptno=50;

La tabla se creará con los nombres de columnas especificadas y los registros recuperados por la sentencia SELECT serán insertados en la tabla. Si se dieran nombres de columnas, su número será igual al número de columnas especificadas en la sentencia SELECT de la subconsulta.

Cuando creo una tabla así no copia las restricciones de la tabla original.

7. La sentencia ALTER TABLE

Después de haber creado una tabla, puede necesitar cambiar su estructura, porque tal vez omitió una columna, o la definición ha cambiado. Esto puede hacerse utilizando la sentencia ALTER TABLE.

Se utiliza para:

- Añadir una nueva columna (o atributo)
- Modificar una columna existente
- Dar un valor por defecto a una nueva columna

Añadir una columna:

Sintaxis:

ALTER TABLE tabla
ADD (columna tipo_datos [**DEFAULT** expr]
[, columna tipo_datos]…);

Ejemplo:

ALTER TABLE dept30
ADD (puesto VARCHAR2(9));

La columna añadida aparecerá la última de la tabla.

Modificar una columna:

Se utiliza para cambiar el tipo de datos de una columna, su tamaño o su valor por defecto. En este último caso, el valor por defecto se aplicará sólo a posteriores inserciones en la tabla.

Sintaxis:

> **ALTER TABLE** tabla
> **MODIFY** (columna tipo_datos [**DEFAULT** expr]
> [, columna tipo_datos]…);

Ejemplo:

> ALTER TABLE Dept
> MODIFY (Provincia VARCHAR2(15));

Eliminación de una columna:

Esta sentencia se utiliza para borrar una columna de una tabla cada vez. La columna podrá o no contener datos no pudiéndose recuperar una vez se haya eliminado (no se puede recuperar con ROLLBACK puesto que lleva COMMIT implícito). En la tabla debe quedar como mínimo una columna después de ALTER.

Sintaxis:

> **ALTER TABLE** tabla **DROP COLUMN** columna;

Ejemplo:

> ALTER TABLE Dept DROP COLUMN Provincia;

Renombrar una columna:

ALTER TABLE nombre_tabla
RENAME COLUMN old_name to new_name;

Opción SET UNUSED (Se utiliza conALTER):

La opción SET UNUSED marca una o más columnas como “no usadas” para que puedan ser eliminadas cuando se necesiten recursos en el sistema. Esta es una prestación propia de Oracle. Especificando esta cláusula no se eliminan realmente las columnas de cada fila

de la tabla, es decir, no restaura el espacio usado por estas columnas. Sin embargo, el tiempo de respuesta es mejor que si se ejecuta la cláusula de borrado DROP.

Las columnas "unused" son tratadas como si hubieran sido borradas, aunque los datos permanecen en la tabla. Después de marcar una columna como "unused", no tenemos acceso a ella. Una sentencia de consulta no recuperará los datos de estas columnas. Además, los nombres y tipos de datos de estas columnas no se mostrarán con DESCRIBE y podemos añadir a la tabla una nueva columna con el mismo nombre que otra columna "unused".

Sintaxis:

ALTER TABLE tabla **SET UNUSED** (columna);

Ejemplo:

ALTER TABLE Dept30 SET UNUSED (ename);

En un segundo paso, eliminaremos las columnas "unused" para conseguir el espacio en disco que ocupan. Si la tabla no contiene este tipo de columnas, la sentencia no devuelve error.

Sintaxis:

ALTER TABLE tabla **DROP UNUSED COLUMNS**;

8. La sentencia DROP TABLE

Sirve para eliminar las tablas (y también sus datos). Se borra la definición de la tabla en el DD y todos los índices asociados. Como toda sentencia DDL no se puede hacer ROLLBACK de la sentencia.

Solamente el propietario de la tabla u otro usuario con el permiso DROP ANY TABLE puede eliminar una tabla.

Sintaxis:

DROP TABLE tabla;

9. Cambiar el nombre de un objeto

Para cambiar el nombre de una tabla, vista, secuencia o sinónimo, se utiliza la instrucción RENAME. Ha de ser el propietario del objeto que renombra. Es una instrucción propia de Oracle.

Sintaxis:

RENAME nombre1 **TO** nombre2;

Ejemplo:

RENAME dept TO departamento;

10. Truncar una tabla

Es propia de Oracle y se utiliza para borrar todos los registros de una tabla y liberar así todo el espacio de almacenamiento ocupado por la tabla. No se puede hacer ROLLBACK. Es una alternativa a la sentencia DELETE de DML pero ésta si puede efectuar ROLLBACK. DELETE no libera espacio.

El usuario que trunque una tabla debe ser el propietario de la misma o tener el privilegio del sistema DELETE TABLE.

Sintaxis:

TRUNCATE TABLE tabla;

11. Añadir comentarios a una tabla

Es propio de Oracle y no de SQL. Sirve para añadir comentarios de hasta 2000 bytes sobre una columna, tabla o vista. El comentario se almacena en el Diccionario de Datos y puede ser visualizado por medio de la columna COMMENTS de una de las siguientes vistas del DD:

Para columnas:
- ALL_COL_COMMENTS
- USER_COL_COMMENTS

Para tablas:
- ALL_TAB_COMMENTS
- USER_TAB_COMMENTS

Sintaxis:

COMMENT ON TABLE tabla **IS** ‘comentario descriptivo’;

COMMENT ON COLUMN tabla.col **IS** ‘comentario descriptivo’;

12. ¿Qué son las restricciones?

Cuando almacenamos datos en nuestras tablas, se ajustan a una serie de restricciones:
- Integridad de datos

- Que una columna no puede almacenar valores negativos
- Que se almacenen las cadenas en mayúsculas
- Que una columna no permita 0

La integridad hace referencia al hecho de que los datos cumplan ciertas restricciones.

Integridad: Regla que restringe el rango de valores de una o más columnas.

- Integridad referencial
  - Garantiza que los valores de una columna de una tabla dependan de los valores de otra columna de otra tabla.

El servidor Oracle utiliza las restricciones (*constraints*) para prevenir la introducción de datos no validos en una tabla.

Se pueden usar para

- Garantizar el cumplimiento de reglas a nivel de tablas, en el momento que una fila se inserta, se actualiza o se borra. La restricción debe ser cumplida para que la operación tenga éxito.
- Impedir la eliminación de una tabla si existen dependencias desde otras tablas.

RESTRICCIONES DE INTEGRIDAD EN ORACLE

CONSTRAINT	***DESCRIPCIÓN***
NOT NULL	Especifica que una columna no tenga valores nulos.
UNIQUE	Especifica que una columna o combinación de ellas, tenga valores irrepetibles.
PRIMARY KEY	Especifica la clave principal.
FOREIGN KEY	Especifica una clave foránea.
CHECK	Especifica que una condición debe ser verdadera.

Se recomienda asignarles un nombre descriptivo a las restricciones. De lo contrario, Oracle le dará uno del tipo SYS_CX donde X es un número entero.

Se pueden crear en:

- En el momento de crear la tabla.
- Después de que la tabla haya sido creada.

Las restricciones se pueden definir tanto a nivel de columna como a nivel de tabla. Se pueden ver las restricciones en el Diccionario de Datos, en la tabla USER_CONSTRAINTS.

13. Definición de restricciones

Las restricciones se crean normalmente a la vez que se crea la tabla pero pueden ser añadidas después de haber creado la tabla y también pueden desactivarse temporalmente.

Sintaxis:

CREATE TABLE tabla

(columna1 tipo_datos [CONSTRAINT_NOMBRERESTRICCION][NOT NULL]
[CONSTRAINT_NOMBRERESTRICCION][UNIQUE]
[CONSTRAINT_NOMBRERESTRICCION][PRIMARY KEY]
[CONSTRAINT_NOMBRERESTRICCION][DEFAULT VALOR]
[REFERENCES NOMBRE TABLA][CHECK CONDICION]
columna2 tipo_datos [CONSTRAINT_NOMBRERESTRICCION][NOT NULL]
[UNIQUE][PRIMARY KEY][DEFAULT VALOR][REFERENCES NOMBRE TABLA]
[CHECK CONDICION]
…
[constraint_tabla]);

Ejemplo (restricción de columna sin nombre explícito):

```
CREATE TABLE empleados
  (empno NUMBER (4),
   ename VARCHAR2 (10),
   deptno NUMBER (2) NOT NULL,
   CONSTRAINT empleados_empno_pk PRIMARY KEY (empno));
```

Ejemplo (restricción de columna con nombre explícito):

```
CREATE TABLE empleados
  (empno NUMBER (4),
   ename VARCHAR2 (10),
   deptno NUMBER (2) CONSTRAINT empleados_deptno_nn NOT NULL,
   CONSTRAINT empleados_empno_pk PRIMARY KEY (empno));
```

La restricción NOT NULL:

La restricción NOT NULL asegura que en la columna no se permitirán valores nulos. Las columnas sin la restricción NOT NULL pueden contener valores nulos, por defecto.

Esta restricción se aplica solamente a nivel de columna, no a nivel de tabla.

La restricción UNIQUE Key:

Esta restricción requiere que cada valor en una columna o conjunto de columnas sean únicas, es decir, dos registros de una tabla no tendrán valores duplicados para determinada columna o conjunto de columnas.

Permite la entrada de nulos, a menos que se especifique también la restricción NOT NULL para las mismas columnas.

Una columna o conjunto de columnas con restricciones UNIQUE y NOT NULL definen una clave alternativa en una tabla.

Sintaxis:

… **CONSTRAINT** nombre_uk **UNIQUE** (columna [,columna]…)….

Ejemplo:

```
CREATE TABLE dept
    (deptno NUMBRE(2),
     dname VARCHAR2(14),
     loc VARCHAR2(13),
     CONSTRAINT dept_dname_uq UNIQUE (dname));
```

La restricción PRIMARY KEY:

Crea una clave principal en una tabla. Se puede crear solamente una Primay Key por tabla. Esta restricción consiste en una columna o conjunto de columnas que identifican unívocamente a cada fila de una tabla. Garantiza la unicidad de la columna (o combinación de ellas) y asegura que ninguna columna que sea parte de la Primary Key pueda contener un valor nulo.

Puede definirse a nivel de columna o de tabla. Oracle crea automáticamente un índice para la columna (o columnas) de la clave principal.

A nivel de tabla la sintaxis será:

… **CONSTRAINT** nombre_pk **PRIMARY KEY** (columna [,columna]…)….

A nivel de columna la sintaxis será:

```
CREATE TABLE nombredetabla
(col1 tipocol [CONSTRAINT nombre] PRIMARY KEY;
```

Ejemplo:

```
CREATE TABLE dept
        (deptno NUMBER(2),
         dname VARCHAR2(14),
         loc VARCHAR2(13),
         CONSTRAINT dept_dname_uq UNIQUE (dname),
         CONSTRAINT dept_deptno_pk PRIMARY KEY (deptno));
```

Cuando una tabla está formada por una clave principal de varias columnas, no podremos utilizar la sintaxis de columnas, tendremos que definirla a nivel de tabla.

La restricción FOREIGN KEY:

La restricción FOREIGN KEY, o de integridad referencial, designa a una columna o combinación de ellas como una clave extranjera y establece una relación con una Primary Key de la misma tabla o de otra. Se pueden definir a nivel de tabla o columna.

El valor de la Foreign Key debe coincidir con uno de la tabla padre o ser NULL.

Las Foreign Key están basadas en valores de datos y son puramente lógicos, no son punteros físicos.

Podemos definir tantas claves ajenas como sea necesario.

Sintaxis a nivel de tabla:

> … **CONSTRAINT** nombre_fk **FOREIGN KEY** (columna [,columna]…)
> **REFERENCES** tabla (columna [,columna]…) ….

Sintaxis a nivel de columna:

Columna tipo_columna [CONSTRAINT nombrerestriccion] REFERENCES nombretabla[(columna)][ON DELETE CASCADE];

Ejemplo (a nivel de tabla):

```
CREATE TABLE empleados
(empno NUMBER(4),
 ename VARCHAR2(10) NOT NULL,
 job VARCHAR2 (9),
 deptno NUMBER (3) NOT NULL,
 CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno) REFERENCES dept (deptno));
```

Ejemplo (a nivel de columna):

```
CREATE TABLE empleados
(empno NUMBER(4),
 ename VARCHAR2(10) NOT NULL,
 job VARCHAR2 (9),
 deptno NUMBER (3) CONSTRAINT emp_deptno_fk REFERENCES dept (deptno));
```

La palabra reservada FOREIGN KEY no es necesaria en este último caso.

A esta restricción se le puede añadir la opción ON DELETE CASCADE. Indica que cuando la fila es borrada en la tabla padre, se borrarán todas las filas dependientes en la tabla hija.

Sin la opción ON DELETE CASCADE, la fila en la tabla padre no puede ser borrada mientras haya referencias a ella en la tabla hija. (EJEMPLO DE DEPT EN ORACLE)

Sintaxis:

> … **CONSTRAINT** nombre_fk **FOREIGN KEY** (columna [,columna]…)

REFERENCES tabla (columna [,columna]…) **ON DELETE CASCADE** …

Ejemplo:

CONSTRAINT emp_deptno_fk FOREIGN KEY (deptno) REFERENCES dept (deptno) ON DELETE CASCADE);

EJEMPLO:

Crear la tabla PERSONAS con la siguiente información:

NOMBRE COLUMNA	**TIPO**	**RESTRICCION**
DNI	NUMBER(8)	PRIMARIA
NOMBRE	VARCHAR2(15)	
DIRECCIÓN	VARCHAR2(15)	
POBLACIÓN	VARCHAR2(15)	
COD_PROV	NUMBER(2)	AJENA

Crear la tabla PROVINCIAS con la siguiente información:

NOMBRE COLUMNA	**TIPO**	**RESTRICCION**
COD_PROV	NUMBER(2)	PRIMARIA
NOMBRE	VARCHAR2(15)	

CREATE TABLE PERSONAS
(....
FOREIGN KEY (COD_PROV) REFERENCES PROVINCIAS
)

Si creamos primero la tabla PERSONAS al meter esta sentencia nos dirá que no existe la tabla PROVINCIAS, por este motivo tendremos que crear antes la tabla PROVINCIAS.
Igualmente si queremos borrar las tablas, habrá que borrar antes PERSONAS que PROVINCIAS , pues de lo contrario Oracle dá error al borrar PROVINVIAS pues te dice que tiene claves primarias que son referenciadas en otra tabla.

Si quisiéramos borrar alguna provincia en concreto, y automáticamente borrasemos las personas asociadas, pondríamos ON DELETE CASCADE.
FOREIGN KEY (COD_PROV) REFERENCES PROVINCIAS ON DELETE CASCADE.

RESUMEN DE RESTRICCIONES:

- RESTRICCIONES DE BORRADO: Una fila no se puede borrar si es referenciada por un clave ajena. Tampoco se puede actualizar la clave primaria .
- RESTRICCIONES EN CASCADA: Si borramos una fila de la tabla maestra (PROVINCIAS), todas las filas de la tabla detalle (PERSONAS) que sean

referenciadas se borraran automáticamente. Cuando salga el mensaje de n rows deleted solo se refiere a las de la tabla maestra.

La restricción CHECK:

Define una condición que cada fila debe satisfacer. La condición puede usar la misma sintaxis que las condiciones de las consultas (ej. WHERE color = red)

Una misma columna puede tener más de una restricción CHECK. Se pueden definir a nivel de tabla o de columna.
Sintaxis:

… **CONSTRAINT** nombre_ck **CHECK** (condición) …

Ejemplo a nivel de tabla:

```
CREATE TABLE emp
(deptno NUMBER (2),
 CONSTRAINT emp_deptno_ck CHECK (deptno BETWEEN 10 AND 99));
```

Ejemplo:
Crear la tabla BLOQUEDEPISOS

14. Operaciones con restricciones

Las operaciones que pueden realizarse con las restricciones son:

- Añadir
- Eliminar
- Activar
- Desactivar
- Visualización en el Diccionario de Datos.

Añadir una restricción:

Se pueden añadir restricciones para tablas ya existentes, usando la sentencia ALTER TABLE con la cláusula ADD.

Sintaxis:

ALTER TABLE tabla
ADD [**CONSTRAINT** restricción] tipo_restricción (columna)

En el caso de querer añadir una restricción del tipo NOT NULL a una columna existente, debe utilizarse la sentencia ALTER TABLE con la cláusula MODIFY. Esto sólo puede hacerse si la tabla no contiene filas.

Ejemplo:

```
ALTER TABLE empleados
ADD CONSTRAINT empleados_mgr_fk FOREIGN KEY (mgr) REFERENCES empleados(empno);
```

Eliminar una restricción:

Se utiliza para borrar una restricción creada previamente. Cuando se borra una restricción de integridad, esa restricción ya no es parte del servidor Oracle y por tanto, no está disponible en el Diccionario de Datos.

Sintaxis

ALTER TABLE tabla **DROP CONSTRAINT** nombre_constraint;

Ejemplos:

ALTER TABLE emp DROP CONSTRAINT emp_mgr_fk;

ALTER TABLE dept DROP PRIMARY KEY CASCADE;

En el último ejemplo, se borra la restricción de Primary Key de la tabla dept y también la restricción de Foreign Key asociada en la columna emp.deptno.

Desactivar restricciones:

Puede desactivarse una restricción sin borrarla. Se utiliza la sentencia ALTER TABLE con la cláusula DISABLE. Con la opción CASCADE se desactivan las restricciones de integridad dependientes.

Puede utilizarse la cláusula DISABLE también en la sentencia CREATE TABLE

Sintaxis:

ALTER TABLE tabla
DISABLE CONSTRAINT nombre_restricción [**CASCADE**];

Ejemplo:

ALTER TABLE empleados
DISABLE CONSTRAINT empleados_empno_pk CASCADE;

Activar restricciones:

Permite activar restricciones de integridad actualmente desactivadas en la definición de la tabla por medio de la cláusula ENABLE.

Si se activa una restricción UNIQUE o PRIMARY KEY se creará automáticamente un índice para la clave UNIQUE o PRIMARY KEY.

Puede utilizarse la cláusula ENABLE también en la sentencia CREATE TABLE

Sintaxis:

ALTER TABLE tabla
ENABLE CONSTRAINT nombre_restricción;

Ejemplo:

ALTER TABLE emp ENABLE CONSTRAINT emp_empno_pk;

15. Consulta de restricciones en el Diccionario de Datos

Tras crear una tabla, puede verificarse su existencia haciendo uso del comando DESCRIBE. La única restricción que puede verificar es NOT NULL. Para ver todas las restricciones sobre una tabla, hay que consultar la vista USER_CONSTRAINTS del Diccionario de Datos.

En la vista USER_CONSTRAINTS se ven todos los nombres y definiciones de restricciones.

En la vista USER_CONS_COLUMNS se ven todas las columnas asociadas con los nombres de las restricciones. Esta vista es especialmente útil para restricciones a las que el propio sistema ha asignado un nombre.

Ejemplos:

```
SELECT constraint_name, constraint_type, search_condition
FROM USER_CONSTRAINTS
WHERE table_name = 'EMP';

SELECT constraint_name, column_name
FROM USER_CONS_COLUMNS
WHERE table_name = 'EMP';
```

16. Vistas

Una vista representa lógicamente un subconjunto de una o más tablas (llamadas tablas base). Una vista no contiene datos en sí misma pero es como una ventana a través de la cual se pueden ver o cambiar los datos de las tablas.

La vista se almacena como una sentencia SELECT en el diccionario de datos.

Las vistas se utilizan para:
- Restringir el acceso a los datos
- Realizar consultas complejas fácilmente. Por ejemplo, para que los usuarios consulten datos de varias tablas sin necesidad de conocer cómo escribir sentencias de unión o JOIN.
- Independencia de los datos para usuarios y aplicaciones: una vista puede ser usada para recuperar datos de diversas tablas
- Presentar diferentes vistas de los mismos datos, ofreciendo acceso a los datos a grupos de usuarios de acuerdo a su criterio particular

Tipos de vistas:
- Simples: utilizan una sola tabla, no contienen funciones ni grupos de datos y pueden hacer DML a través de la vista.
- Complejas: utilizan una o más tablas, contienen funciones o grupos de datos y no siempre pueden realizar DML a través de la vista.

Sentencia CREATE VIEW

Podemos crear una vista embebiendo una subconsulta dentro de la sentencia CREATE VIEW.

Es una sentencia perteneciente al sublenguaje DDL (Data Definition Language)

Sintaxis:

```
CREATE [OR REPLACE] [FORCE | NOFORCE] VIEW nombre_vista
[(columna[,columna]...)]
AS subconsulta
[WITH CHECK OPTION [CONSTRAINT restricción]]
[WITH READ ONLY];
```

- OR REPLACE: recrea la vista si ya existe
- FORCE: crea la vista sin importar que la tabla base exista o no
- NOFORCE: crea la vista únicamente si la tabla base existe. Es la opción por defecto.
- alias: especifica nombres para las expresiones seleccionadas en la consulta de la vista. El número de alias de las columnas debe coincidir con el de expresiones seleccionadas por la subselect. Cuando no se ponen nombres de columnas
- subconsulta: es una sentencia SELECT completa. Se pueden usar los alias para las columnas de la lista del SELECT. No puede contener la cláusula ORDER BY.
- WITH CHECK OPTION: especifica que solamente las filas accesibles a la vista pueden ser insertadas o actualizadas.
- restricción: es el nombre asignado a la restricción CHECK OPTION
- WITH READ ONLY: asegura que ninguna operación DML pueda realizarse sobre esta vista.

- Si se suprime una tabla la vista asociada es invalida

Creación de vistas sencillas

- Las vistas mas sencillas son las que unicamente tocan una tabla

Ejemplos: Vista de los ermpleados del deparatmento 30

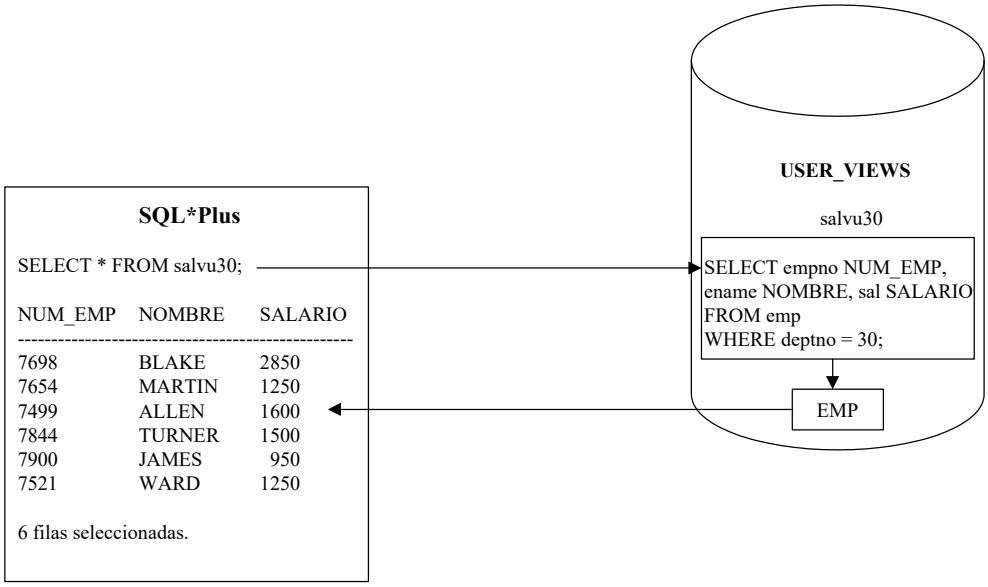
```
CREATE VIEW empvu10
AS SELECT empno, ename, job
FROM emp
WHERE deptno = 10;

CREATE VIEW salvu30
AS SELECT empno NUM_EMP, ename NOMBRE, sal SALARIO
FROM emp
WHERE deptno = 30;
```

Recuperación de datos de una vista

Podemos recuperar datos desde una vista al igual que lo hacemos desde una tabla. Podemos visualizar el contenido de la vista al completo o sólo ver columnas y filas específicas.

```
SELECT * FROM salvu30;
```



Modificación de una vista

La opción OR REPLACE permite que se cree una vista incluso si ya existe una con ese nombre, reemplazando de esta forma la antigua versión de la vista. Esto significa que la vista puede alterarse sin hacer DROP, volver a crear ni volver a conceder privilegios sobre el objeto.

Ejemplo:

```
CREATE OR REPLACE VIEW empvu10
        (num_empleado, nombre_empleado, empleo)
AS SELECT empno, ename, job
FROM emp
WHERE deptno = 10;
```

Nota: los alias de columna (en el ejemplo: num_empleado, nombre_empleado, empleo) en la cláusula CREATE VIEW deben aparecer en el mismo orden que las columnas en la subconsulta (en el ejemplo: empno,, ename, job).

Ejemplo de vista compleja:

```
CREATE VIEW deptsumvu
        (nombre_dep, minimo_sueldo, maximo_sueldo, media_sueldo)
AS SELECT d.dname, MIN(e.sal), MAX(e.sal), AVG(e.sal)
FROM emp e, dept d
WHERE e.deptno = d.deptno
GROUP BY d.dname;
```

Reglas para realizar operaciones DML sobre vistas

Cuando creamos una vista con todas las columnas de la tabla asociada, podremos insertar filas en la vista sin ningún problema.

Podemos realizar operaciones DML sobre los datos a través de una vista siempre que esas operaciones sigan ciertas reglas.

Se puede eliminar una fila de una vista salvo que la misma contenga:
- Funciones de grupo (SUM, AVG, COUNT, etc.)
- Una cláusula GROUP BY
- La cláusula DISTINCT

Se pueden modificar los datos de una vista salvo que contenga una de las condiciones mencionadas y tenga columnas definidas por expresiones (por ejemplo: sal*12).

Se pueden agregar datos a través de una vista a menos que la misma contenga cualquiera de las condiciones anteriores y además no existan columnas NOT NULL en la tabla base no seleccionada por la vista. Todos los valores requeridos deben estar presentes en la

vista. Recordar que se están agregando valores directamente en la tabla subyacente *a través* de la vista.

La cláusula WITH CHECK OPTION

Es posible realizar chequeos de integridad referencial a través de las vistas. Podemos reforzar las restricciones al nivel de la base de datos. La vista puede utilizarse para proteger la integridad de los datos pero el uso es muy limitado.

La cláusula WITH CHECK OPTION especifica que los INSERTs y UPDATEs realizados a través de las vistas no pueden crear filas que la vista no pueda seleccionar y, por lo tanto, esto permite las restricciones de integridad y chequeos en la validación de los datos para imponerse sobre los datos que se están insertando o actualizando.

Si hay un intento de realizar operaciones DML sobre las filas que la vista no ha seleccionado, se visualiza un error, con el nombre de la restricción si se ha especificado.

Ejemplo:

```
CREATE OR REPLACE VIEW empvu20
AS SELECT *
FROM EMP
WHERE deptno = 20
WITH CHECK OPTION CONSTRAINT empvu20_ck;

UPDATE empvu20
SET deptno = 22
WHERE empno = 7788;
```

SQL> ERROR: violación de la cláusula WITH CHECK OPTION. Si se intenta cambiar el número de departamento para cualquier fila, la sentencia fallará porque viola la restricción de CHECK OPTION.

La opción WITH READ ONLY

Podemos denegar operaciones DML sobre determinada vista creándola con la opción WITH READ ONLY.

Ejemplo:

```
CREATE OR REPLACE VIEW empvu10
                (num_empleado, nombre_empleado, empleo)
AS SELECT empno, ename, job
FROM emp
WHERE deptno = 10
WITH READ ONLY;
```

```
DELETE FROM empvu10
WHERE num_empleado = 7782;
```

SQL> ERROR: cualquier intento de quitar una fila de la vista, dará un error.

La sentencia DROP VIEW

Se utiliza la sentencia DROP VIEW para eliminar una vista. La ejecución de esta sentencia provoca la eliminación de la definición de la vista de la base de datos. La eliminación de una vista no afecta a las tablas sobre las que se basa la vista. Las vistas o las aplicaciones basadas en la vista eliminada se convierten en inválidas.

Únicamente el creador o un usuario con el privilegio DROP ANY VIEW puede eliminar una vista.

Sintaxis:

```
DROP VIEW nombre_vista;
```

Ejemplo:

```
DROP VIEW empvu10;
```

Vistas inline

- Una vista inline es una subconsulta con un alias (nombre correlacionado) que podemos usar dentro de la sentencia SQL.
- Es similar a usar una subconsulta en la cláusula FROM de la consulta principal.
- Una vista inline no es un objeto del esquema.

Ejemplo:

```
SELECT a.ename, a.sal, a.deptno, b.maxsal
FROM emp a, (SELECT deptno, max(sal) maxsal
                FROM emp
                GROUP BY deptno) b
WHERE a.deptno = b.deptno
AND a.sal < b.sal;
```