# C:\Users\60034810\Downloads\vintage_minimalist_art-wallpaper-2560x1440.jpg

Andrei García Cuadra

Bases de datos

Apuntes

01/01/2017

*2017*

# A tener en cuenta para el examen:

* Obligatorio uso de constraint para primary key y foreign key, con su nombre.
* La diferencia entra TRUNCATE y DELETE: El TRUNCATE es una operación DDL, no DML. El DELETE permite poner condiciones (WHERE) de borrado, el TRUNCATE no, además de que podemos deshacer los cambios.
* Para crear un esquema, debemos estar conectados como **SYSDBA**, **SYSTEM**, o **SYS**.
* DEFAULT no es una restricción.
* NOT NULL siempre de modifica con MODIFY.
* Un sistema transaccional de datos es transaccional si es capaz de mantener la integridad de los datos, haciendo que estas transacciones no puedan finalizar en un estado intermedio
* En un sistema ideal, las transacciones deberían garantizar todas las propiedades ACID (ACID es un acrónimo de Atomicity, Consistency, Isolation, Durability).

# Tipos de datos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Descripción** | **Nomenclatura** | **Tipo de almacenamiento** | **Formato** | **Función relacionada** |
| **Texto de anchura fija** | **CHAR(n)** | **Texto** |  |  |
| **Texto de anchura variable** | **VARCHAR2(n)** | **Texto** |  |  |
| **Texto de anchura fija para caracteres nacionales** | **NCHAR(n)** | **Texto** |  |  |
| **Texto de anchura variable para caracteres nacionales** | **NVARCHAR2(n)** | **Texto** |  |  |
| **Entero pequeño** | **SMALLINT** | **Número** |  |  |
| **Entero normal** | **INT** | **Número** |  |  |
| **Entero grande** | **BIGINT** | **Número** |  |  |
| **Entero de precisión decimal** | **NUMBER(n)** | **Número** |  |  |
| **Decimal de coma variable** | **NUMBER** | **Número** |  |  |
| **Decimal de coma fija** | **NUMBER(n,d)** | **Número** |  |  |
| **Fecha** | **DATE** | **Fecha** | **01-01-1970** | **TO\_DATE()** |
| **Fecha y hora** | **TIMESTAMP** | **Fecha** | **1489135745** |  |
| **Intervalo** | **INTERVAL** | **Fecha** | **INTERVAL '4 10:12:7,352' DAY TO SECOND(3)** | **INTERVAL [time] TO [time]** |
| **Texto de gran longitud** | **LONG** | **Texto** |  |  |
| **Binario de gran longitud** | **RAW** | **Números binarios** |  |  |

# DDL

Lenguaje de definición de datos. Se encarga de la creación, modificación y eliminación de los objetos de la base de datos.

Cada usuario de una base de datos posee un esquema. El esquema suele tener el mismo nombre que el usuario y sirve para almacenar los objetos de esquema, es decir los objetos que posee el usuario.

Esos objetos pueden ser: tablas, vistas, índices y otras objetos relacionados con la definición de la base de datos. Los objetos son manipulados y creados por los usuarios. En principio sólo los administradores y los usuarios propietarios pueden acceder a cada objeto, salvo que se modifiquen los privilegios del objeto para permitir el acceso a otros usuarios.

En este tipo no se pueden realizar RollBacks o Commits.

## Creación de bases de datos

**Método simplificado:**

CREATE DATABASE prueba;

**Método completo:**

CREATE DATABASE prueba

LOGFILE prueba.log

MAXLOGFILES 25

MAXINSTANCES 10

ARCHIVELOG

CHARACTER SET WIN1214

NATIONAL CHARACTER SET UTF8

DATAFILE prueba1.dbf AUTOEXTEND ON MAXSIZE 500MB;

## Tablas

### Creación de tablas

El usuario ha de poseer permisos para la creación de las tablas sobre el esquema actual. Además, deben cumplir con las siguientes restricciones impuestas por ORACLE:

* Deben comenzar con una letra
* No deben tener más de 30 caracteres
* Sólo se permiten utilizar letras del alfabeto (inglés), números o el signo de subrayado.
* No puede haber dos tablas con el mismo nombre para el mismo esquema, aunque sí para distinto esquema.
* No puede coincidir con el nombre de una palabra reservada.
* Para diferenciar mayúsculas y minúsculas en una cadena, se utilizan comillas dobles.

**Método sencillo:**

CREATE TABLE nombreDeTabla (nombreDeLaColumna1 tipoDeDato(longitud));

**Método con valor por defecto:**

CREATE TABLE nombreDeTabla (pueblo VARCHAR(5) DEFAULT ‘Palencia’);

**Método completo:**

CREATE TABLE esquema.nombreDeTabla (nombreDeLaColumna1 tipoDeDato [DEFAULT valor] [restricciones]);

### Borrado de tablas

El usuario ha de poseer permisos para la eliminación de las tablas sobre el esquema actual. Además, se ha de tener en cuenta que:

• Desaparecen todos los datos, así como su estructura.

• Cualquier vista y sinónimo referente a la tabla seguirá existiendo, pero ya no funcionará (conviene eliminarlos)

• Las transacciones pendientes son aceptadas (COMMIT.

**Método sencilllo:**

DROP TABLE nombreDeTabla;

**Método completo:**

DROP TABLE esquema.nombreDeTabla;

**Método con eliminación de claves foráneas:**

DROP TABLE nombreDeTabla CASCADE constraints;

### Vaciado de tablas

El usuario ha de poseer permisos para el vaciado de las tablas sobre el esquema actual. Además, se ha de tener en cuenta que:

• Desaparecen todos los datos, pero se conserva su estructura.

• Cualquier vista y sinónimo referente a la tabla seguirá existiendo, pero ya no funcionará (conviene eliminarlos)

• Las transacciones pendientes son aceptadas (COMMIT).

**Método completo:**

TRUNCATE TABLE nombreDeTabla;

## Columnas (Tablas)

### Añadir columna a tabla

El usuario ha de poseer permisos para el manejo de las tablas sobre el esquema actual.

**Método completo:**

ALTER TABLE nombreTabla ADD(nombreColumna TipoDatos [Propiedades] [,columnaSiguiente tipoDatos [propiedades]...)

### Eliminar campos de tablas

El usuario ha de poseer permisos para el manejo de las tablas sobre el esquema actual.

**Método completo:**

ALTER TABLE nombreTabla DROP(nombreColumna [, siguienteColumna…])

### Modificar campos de tablas

El usuario ha de poseer permisos para el manejo de las tablas sobre el esquema actual.

**Método completo:**

ALTER TABLE nombreTabla MODIFY(nombreColumna tipoDeNuevoDato(longitud) [, siguienteColumna…])

### Renombrar campos de tablas

El usuario ha de poseer permisos para el manejo de las tablas sobre el esquema actual.

**Método completo:**

ALTER TABLE nombreTabla RENAME COLUMN nombreAnt TO nombreNuevo

### Renombrar campos de tablas

El usuario ha de poseer permisos para el manejo de las tablas sobre el esquema actual.

**Método completo:**

{CREATE TABLE nombreTabla | ALTER TABLE nombreTabla {ADD | MODIFY}} (campo tipoDeDatos [propiedades] [[CONSTRAINT nombreRestricción ]] tipoRestricción (columnas) [,siguienteCampo…] [,CONSTRAINT nombreRestricción tipoRestricción (columnas) ...)

Para la nomenclatura de nombres de las restricciones se recomienda seguir los siguientes parámetros:

* Tres letras para el nombre de la tabla
* Carácter de subrayado
* Tres letras con la columna afectada por la restricción
* Carácter de subrayado
* Dos letras con la abreviatura del tipo de restricción. La abreviatura puede ser:
  + **NN**. NOT NULL.
  + **PK**. PRIMARY KEY
  + **UK**. UNIQUE FK.
  + **FK**. FOREIGN KEY
  + **CK**. CHECK

### Valor por defecto de un campo

El usuario ha de poseer permisos para el manejo de las tablas sobre el esquema actual.

**Método completo:**

CREATE TABLE nombreTabla (campoUno tipoDato(longitud) DEFAULT valorPorDefecto);

### Mostrar columnas de una tabla

La opción cascade hace que se desactiven también las dependencias (regla de integridad referencial).

DESCRIBE tabla;

## Restricciones

### Restricción: Prohibir nulos

Está implícito en toda clave primaria. **Es la única restricción que no requiere de un constraint.**

CREATE TABLE nombreTabla(campo1 tipoDato(longitud) NOT NULL);

### Restricción: Valor único

Está implícito en toda clave primaria.

CREATE TABLE nombreTabla(campo1 tipoDato(longitud));

CONSTRAINT nombreRestriccion **UNIQUE** (campo1,campo2));

### Restricción: Clave primaria

Posee implícitamente las restricciones **NOT NULL** y **UNIQUE**.

CREATE TABLE tabla(campo1 tipodato(longitud),

CONSTRAINT nombreRestriccion **PRIMARY KEY**(campo1,campo2));

### Restricción: Clave foránea

Posee implícitamente las restricciones **NOT NULL** y **UNIQUE**.

* **ON DELETE SET NULL**. Coloca nulos todas las claves secundarias relacionadas con la borrada.
* **ON DELETE CASCADE**. Borra todos los registros cuya clave secundaria es igual que la clave del registro borrado.
* **ON DELETE SET DEFAULT**. Coloca en el registro relacionado el valor por defecto en la columna relacionada
* **ON DELETE NOTHING**. No hace nada.

**Método simple:**

CREATE TABLE tabla(campo1 VARCHAR2(9),

CONSTRAINT campo1 FOREIGN KEY(campo1) REFERENCES clientes(dni));

**Método con validación de integridad referencial:**

CREATE TABLE tabla(campo1 VARCHAR2(9),

CONSTRAINT campo1 FOREIGN KEY(campo1) REFERENCES clientes(dni) [ON DELETE | ON UPDATE [SET NULL | CASCADE | DEFAULT]);

### Restricción: Check

CREATE TABLE ingresos(cod NUMBER(5) PRIMARY KEY,

concepto VARCHAR2(40) NOT NULL,

importe NUMBER(11,2),

CONSTRAINT importe\_min CHECK (importe>0));

### Añadir restricción a una tabla existente

ALTER TABLE tabla ADD [CONSTRAINT nombre] tipoDeRestricción(columnas);

### Eliminar restricción de una tabla existente

Si esta restricción se viera afectara por la regla de integridad referencial deberíamos eliminarla en modo cascade.

ALTER TABLE tabla DROP {PRIMARY KEY | UNIQUE(campos) | CONSTRAINT nombreRestricción [CASCADE CONSTRAINTS]}

### Desactivar restricciones

La opción cascade hace que se desactiven también las dependencias (regla de integridad referencial).

ALTER TABLE tabla DISABLE CONSTRAINT nombre [CASCADE];

### Reactivar restricciones

Muestra la información relacionada al nombre de la tabla, tipo de dato y valor nulo de la estructura de una tabla.

ALTER TABLE tabla ENABLE CONSTRAINT nombre [CASCADE];

### Renombrar restricciones

Muestra la información relacionada al nombre de la tabla, tipo de dato y valor nulo de la estructura de una tabla.

ALTER TABLE table RENAME CONSTRAINT nombreViejo TO nombreNuevo;

### Mostrar restricciones

**Mostrar restricciones del usuario:**

SELECT \* FROM USER\_CONS\_COLUMNS;

**Mostrar tipo de restricciones del usuario:**

SELECT \* FROM USER\_COL\_CONSTRAINTS;

**Mostrar TODAS las restricciones (Sólo SYSDBA):**

SELECT \* FROM ALL\_CONSTRAINTS;

## Índices

Los índices son objetos que forman parte del esquema que hacen que las bases de datos aceleren las operaciones de consulta y ordenación sobre los campos a los que el índice hace referencia.

**Se aconseja crear índices en campos que:**

* Contengan una gran cantidad de valores
* NO Contengan una gran cantidad de nulos
* Sean parte habitual de cláusulas WHERE, GROUP BY u ORDER BY
* Seann parte de listados de consultas de grandes tablas sobre las que casi siempre se muestran como mucho un 4% de su contenido.

**No se aconseja en campos que:**

* Pertenezcan a tablas pequeñas
* No se usen a menudo en las consultas
* Pertenecen a tablas cuyas consultas muestran menos de un 4% del total de registros
* Pertenecen a tablas que se actualizan frecuentemente
* Se utilizan en expresiones

### Creación de índices

CREATE INDEX nombre ON tabla (columna1 [,columna2...]);

## Vistas

Una vista no es más que una consulta almacenada a fin de utilizarla tantas veces como se desee, de una forma más rápida y sencilla. Existen dos tipos de vistas:

**Simples.** Las forman una sola tabla y no contienen funciones de agrupación. Su ventaja es que permiten siempre realizar operaciones DML sobre ellas.

* **Complejas.** Obtienen datos de varias tablas, pueden utilizar funciones de agrupación. No siempre permiten operaciones DML.

### Creación de vistas

* **OR REPLACE**. Si la vista ya existía, la cambia por la actual
* **FORCE**. Crea la vista aunque los datos de la consulta SELECT no existan
* **WITH CHECK OPTION**. Hace que sólo las filas que se muestran en la vista puedan ser añadidas (INSERT) o modificadas (UPDATE). La restricción que sigue a esta sección es el nombre que se le da a esta restricción de tipo CHECK OPTION.
* **WITH READ ONLY**. Hace que la vista sea de sólo lectura. Permite grabar un nombre para esta restricción.

CREATE [OR REPLACE] [FORCE|NOFORCE] VIEW vista

[(alias[, alias2...])]

AS consulta SELECT

[WITH CHECK OPTION [CONSTRAINT restricción]]

[WITH READ ONLY [CONSTRAINT restricción]];

### Mostrar vistas

SELECT \* FROM USER\_VIEWS;

### Mostrar vista

DROP VIEW nombreDeVista;

# DML

Lenguaje de manipulación de datos. Es requerido un COMMIT para aplicar los datos. Estos datos se han de confirmar antes del cese de la sesión en la cual se añaden. Para ello, debemos ejecutar el comando **COMMIT**. Si deseamos descartar los cambios realizados en este COMMIT, debemos ejecutar la instrucción **ROLLBACK**.

**Durante una transacción DML:**

* Se puede volver a la instrucción anterior a la transacción cuando se desee
* Las instrucciones de consulta SELECT realizadas por el usuario que inició la transacción muestran los datos ya modificados por las instrucciones DML
* El resto de usuarios ven los datos tal cual estaban antes de la transacción, de hecho los registros afectados por la transacción aparecen bloqueados hasta que la transacción finalice. Esos usuarios no podrán modificar los valores de dichos registros.
* Tras la transacción todos los usuarios ven los datos tal cual quedan tras el fin de transacción. Los bloqueos son liberados y los puntos de ruptura borrados.

## Insertar registro

El usuario ha de poseer permisos para el manejo de las tablas sobre el esquema actual.

**Método completo:**

INSERT INTO tabla [(listaDeCampos)] VALUES (valor1 [,valor2 ...])

## Modificar registro

El usuario ha de poseer permisos para el manejo de las tablas sobre el esquema actual.

**Método completo:**

UPDATE tabla SET columna1=valor1 [,columna2=valor2...] [WHERE condición]

## Eliminar registro

El usuario ha de poseer permisos para el manejo de las tablas sobre el esquema actual. Para mantener la regla de integridad referencial, si el registro borrado posee la propiedad **ON DELETE CASCADE** se borrarán más registros aparte del eliminado.

**Método completo:**

DELETE [FROM] tabla [WHERE condición]

## Especificar inicio de transacción

Especifica que va a empezar una transacción.

**Método completo:**

BEGIN TRAN;

## Especificar fin de transacción exitosa

Le indica al motor que puede considerar la transacción completada con éxito.

**Método completo:**

COMMIT TRAN;

## Especificar fin de transacción errónea

Indica que se ha alcanzado un fallo y que debe restablecer la base al punto de integridad.

**Método completo:**

ROLLBACK TRAN;