

## DDL. CREACIÓN DE ÍNDICES

---

**CREATE [UNIQUE] INDEX** *nombre\_índice*  
**ON** *tabla (columna {, columna })* ;

- Crea un índice sobre una tabla de la base de datos
- Se puede crear cualquier número de índices sobre una tabla. Cada uno de ellos puede incluir hasta un máximo de 32 columnas
- Los índices permiten mejorar los tiempos de ejecución de las consultas
- **UNIQUE** indica que no se aceptarán valores duplicados en el conjunto de las columnas que forman el índice

## DDL. CREACIÓN DE ÍNDICES

---

```
CREATE [UNIQUE] INDEX nombre_índice  
ON tabla (columna {, columna } ) ;
```

```
create index apellidos_nombre  
on lector ( ape_1, ape_2, nombre ) ;
```

## DDL. CREACIÓN DE VISTAS

---

```
CREATE VIEW nombre_vista [(columna {, columna })]  
AS sentencia_select  
[WITH CHECK OPTION]  
[WITH READ ONLY] ;
```

- Define una tabla virtual cuyo contenido será el resultado de la ejecución de la sentencia *select*
- La vista puede ser utilizada como tabla en posteriores consultas
- Una vista puede utilizarse para actualizar datos sólo si cumplen determinadas condiciones
- Si se incluye la cláusula *WITH CHECK OPTION* no se permitirá que mediante un *insert* o *update* se cree una fila que no cumpla las condiciones de la vista
- Si se incluye la clausula **WITH READ ONLY** no podrán ejecutarse operaciones *insert/delete/update* sobre la vista

## DDL. CREACIÓN DE VISTAS

---

```
CREATE VIEW nombre_vista [(columna {, columna })]  
AS sentencia_select  
[WITH CHECK OPTION];
```

### Condiciones para la actualización de datos a través de vistas

- la *select* sólo recupera filas de una única tabla (hay una única tabla base)
- todas las columnas seleccionadas se corresponden con columnas de la tabla base (no se utilizan funciones)
- la *select* no incluye *distinct*, *group by* ni *having*
- la inserción sólo será posible si todas las columnas con *not null not default* de la tabla original se recuperan en la *select*

## DDL. CREACIÓN DE VISTAS. EJEMPLOS

---

```
create view lectores_zam as select * from lector  
where provincia = 'ZAMORA' with check option;
```

```
update lectores_zam set provincia = 'SALAMANCA'  
where codigo= '7838294';
```

```
SQL> update lectores_zam set provincia = 'SALAMANCA' where codigo= '7838294';  
update lectores_zam set provincia = 'SALAMANCA' where codigo= '7838294'  
      *  
ERROR at line 1:  
ORA-01402: view WITH CHECK OPTION where-clause violation
```

## DDL. CREACIÓN DE VISTAS. EJEMPLOS

---

```
create view lectores_za  
as select * from lector where provincia = 'ZAMORA'  
with check option;
```

```
select codigo, ape_1, ape_2, nombre from lectores_za;
```

```
SQL> select codigo, ape_1, ape_2, nombre from lectores_za;
```

CODIGO	APE_1	APE_2	NOMBRE
7897445	HOYOS	MANCEBO	BRAULIO
11592038	BLANCO	MENDEZ	MARIA
2894123	MARTIN	RODRIGUEZ	ANGEL JOSE
11543192	ROLDAN	ESCUDERO	FRANCISCO
41743122	BLANCO	AGUDO	OSCAR
7838294	RODRIGUEZ	ESTEVEZ	ALONSO
6883822	ABARCA	MARTIN	PEDRO

7 rows selected.

## DDL. CREACIÓN DE SINÓNIMOS

---

**CREATE SYNONYM** *nombre\_sinónimo*  
**FOR** *tabla / vista / índice* ;

- Crea un sinónimo para una tabla, vista o índice
- Un sinónimo es un alias, un nombre alternativo para un objeto

```
create synonym usuario for lector;
```

## DDL. CREACIÓN DE SECUENCIAS

---

**CREATE SEQUENCE** *nombre\_secuencia*  
**[INCREMENT BY** *incremento***] [START WITH** *inicio***] ;**

- Una secuencia es un mecanismo para obtener listas de números secuenciales
- Se pueden utilizar para obtener valores que se pueden usar como claves primarias
- Las secuencias son un elemento propio de ORACLE
  - En otros SGBD se usan otros mecanismos: atributos AUTO\_INCREMENT
- CURRVAL y NEXTVAL son pseudocolumnas de las secuencias
  - *nombre\_secuencia.CURRVAL* : devuelve el valor actual de la secuencia
  - *nombre\_secuencia.NEXTVAL* : calcula y devuelve el siguiente valor de la secuencia



## DDL. CREACIÓN DE SECUENCIAS

```
create table mitabla (  
codigo integer not null primary key,  
otro integer);
```

```
create sequence clave_mitabla;
```

```
insert into mitabla  
values (clave_mitabla.nextval, 55);
```

```
insert into mitabla  
values (clave_mitabla.nextval, 23);
```

```
SQL> select clave_mitabla.currval from dual;
```

CURRVAL
2

```
SQL> select * from mitabla;
```

CODIGO	OTRO
1	55
2	23

```
SQL> 
```

## DDL. ELIMINACIÓN DE OBJETOS DE LA BASE DE DATOS

---

**DROP [ TABLE | VIEW | INDEX | SYNONYM | SEQUENCE ]**  
*nombre\_objeto;*

```
DROP INDEX apellidos_nombre;
```

```
DROP VIEW lector_zam;
```

```
DROP SYNONYM usuario;
```

```
DROP SEQUENCE clave_mitabla;
```

## CONCESIÓN DE PRIVILEGIOS DE ACCESO A TABLAS Y VISTAS

---

```
GRANT ALL [ PRIVILEGES ] | privilegio {, privilegio }  
ON nombre  
TO PUBLIC | usuario {, usuario }  
[ WITH GRANT OPTION ];
```

- El propietario de una tabla o vista es el usuario que la crea. Los demás usuarios no pueden acceder a ella mientras no les sean concedidos los privilegios de acceso.
- Los privilegios de acceso pueden ser *select*, *update*, *delete*, *insert*, *index*, *references*, *alter* o *all*

```
grant all on table lector to OPS$I000001;  
grant select on table autor to public;
```

## CONTROL DE ACCESO MEDIANTE VISTAS. ACCESOS POR USER

---

- Haciendo uso del valor de USER se pueden crear vistas que recuperen unos datos distintos según el usuario que las consulte
- Alguna de las tablas sobre las que se defina la vista deberán incluir una columna con la identificación del usuario que puede consultar la tupla

```
create table persona (  
  Usuario          varchar(15) not null primary key,  
  Datopersonal    varchar(10));  
  
create view misdatos as select * from persona  
where usuario=user;  
  
grant select on misdatos to public;
```

## CONTROL DE ACCESO MEDIANTE VISTAS. ACCESOS POR USER

---

```
SQL> select * from misdatos;
```

USUARIO	DATOPERSON
OPSS\$[REDACTED]	89357

```
SQL> select * from persona;
```

USUARIO	DATOPERSON
OPSS\$I[REDACTED]	3000
OPSS\$I[REDACTED]	3034
OPSS\$I[REDACTED]	3068
OPSS\$I[REDACTED]	3102
OPSS\$I[REDACTED]	3136
OPSS\$I[REDACTED]	3170
OPSS\$I[REDACTED]	3204

Consultando la vista  
cada usuario obtiene una  
tupla distinta

La tabla base contiene  
muchas tuplas

## REVOKE PRIVILEGIOS DE ACCESO A TABLAS Y VISTAS

---

**REVOKE { SELECT | DELETE | USAGE | ALL [PRIVILEGES] } [, ...] ON objeto FROM { PUBLIC | nombre\_usuario [, ...] };**

**REVOKE { INSERT | UPDATE | REFERENCES } [, ...] [ ( columna [, ...] ) ] ON objeto FROM { PUBLIC | nombre\_usuario [, ...] };**

- Revoca privilegio/s a un usuario/s sobre un objeto
- La sintaxis para el comando **REVOKE** tiene capacidades para rescindir privilegios en columnas individuales en tablas

```
revoke select on lector from OPS$I10101010;
```

## DDL. MODIFICACIÓN DE DEFINICIÓN DE TABLAS

---

**ALTER TABLE** *tabla*

**ADD** [**CONSTRAINT** *nombre*] *restricción\_tabla*

| **DROP CONSTRAINT** *nombre*

| **ADD** *especificación\_columna*

| **DROP COLUMN** *columna* [**CASCADE CONSTRAINT**] ;

- Permite añadir o eliminar columnas y condiciones de integridad a tablas ya creadas
- Al eliminar una columna se puede especificar si se eliminará el elemento indicado y todos los que dependan de él:
  - ***CASCADE CONSTRAINT***

## DDL. MODIFICACIÓN DE DEFINICIÓN DE TABLAS

---

**ALTER TABLE *tabla***  
**ADD [CONSTRAINT *nombre*] *restricción\_tabla***  
**| DROP CONSTRAINT *nombre***  
**| ADD *especificación\_columna***  
**| DROP COLUMN *columna* [CASCADE CONSTRAINT ] ;**

```
alter table autor  
  
drop column codigo cascade constraint;
```