



TECNICAS DE ALMACENAMIENTO DE DATOS

DDL (Data Definition Language)

Lenguaje de definición de datos

CREATE
ALTER
DROP

TABLE
INDEX
VIEW
SYNONYM
FUNCTION
PROCEDURE
PACKAGE
TRIGGER

DDL – TABLAS

```
CREATE TABLE NOMBRE_TABLA (  
    CAMPO_1          TIPO_DE_DATO,  
    CAMPO_2          TIPO_DE_DATO,  
    ...  
    CAMPO_N          TIPO_DE_DATO  
);
```

```
ALTER TABLE NOMBRE_TABLA  
ADD NOMBRE_COLUMNNA TIPO_DE_DATO;
```

```
DROP TABLE NOMBRE_TABLA;
```

DATATYPES

TIPO	EJEMPLO
NUMBER [(PRECISION, ESCALA)]	IMPORTE NUMBER (16, 2)
CHAR [(LONGITUD_MAXIMA)]	NOMBRE CHAR (30)
VARCHAR2 (LONGITUD_MAXIMA)	NOMBRE VARCHAR2 (30)
DATE	FECHA_DE_INGRESO DATE
BOOLEAN	HABILITADO BOOLEAN
BLOB	FOTO_CV BLOB
CLOB	PAPER CLOB

DDL – Ejemplos CREATE TABLE

```
CREATE TABLE PROVINCIA (  
    ID_PROVINCIA          INTEGER,  
    NOMBRE_PROVINCIA     VARCHAR2 (50)  
);
```

```
CREATE TABLE LOCALIDAD (  
    ID_LOCALIDAD          NUMBER NOT NULL,  
    NOMBRE_LOCALIDAD     VARCHAR2 (100),  
    ID_PROVINCIA          INTEGER NOT NULL  
);
```

DDL – Ejercicio 1

Crear una tabla llamada **ALUMNO** que incluya los siguientes campos:

- LEGAJO - INTEGER
- FECHA_DE_INGRESO - DATE
- NOMBRE - VARCHAR2(30)
- APELLIDO - VARCHAR2(30)

```
CREATE TABLE NOMBRE_TABLA (  
  CAMPO_1          TIPO_DE_DATO,  
  CAMPO_2          TIPO_DE_DATO,  
  ...  
  CAMPO_N          TIPO_DE_DATO  
);
```

DDL – Ejercicio 1

```
CREATE TABLE ALUMNO (  
    LEGAJO                INTEGER,  
    FECHA_DE_INGRESO     DATE,  
    NOMBRE                VARCHAR2 (30) ,  
    APELLIDO              VARCHAR2 (30)  
);
```

DDL – Ejercicio 2

Agregar las siguientes columnas a la tabla **ALUMNO** creada recientemente:

- DNI - NUMBER
- E_MAIL - VARCHAR2(100)

```
ALTER TABLE NOMBRE_TABLA  
ADD NOMBRE_COLUMNNA TIPO_DE_DATO;
```


DDL – Ejercicio 2

```
ALTER TABLE ALUMNO ADD (  
    DNI          NUMBER,  
    E_MAIL      VARCHAR2 (100)  
);
```

Ejercicio 3 (DML)

Insertar un registro en la tabla ALUMNO con sus datos personales.

```
INSERT INTO TABLA  
(CAMPO1, CAMPO2, CAMPO3)  
VALUES  
(VALOR, VALOR, VALOR);
```

DDL – VISTAS

- es una consulta guardada con un alias
- se pueden restringir los campos y registros de la tabla base a mostrar (seguridad)
- no existe una copia física de los datos (son consultas a los datos que hay en las tablas)
- en las columnas que incluyen resultados de funciones se debe indicar un alias

DDL – VISTAS

-- CREACION

```
CREATE VIEW NOMBRE_VISTA
AS
SELECT *
FROM NOMBRE_TABLA;
```

-- BORRADO

```
DROP VIEW NOMBRE_VISTA;
```

-- MODIFICACION

```
CREATE OR REPLACE VIEW NOMBRE_VISTA
AS
SELECT CAMPO1, CAMPO2, CAMPO3
FROM NOMBRE_TABLA;
```

DDL – Ejemplos CREATE VIEW

-- CREACION

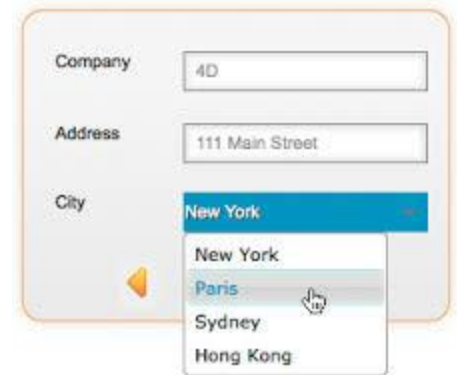
```
CREATE VIEW CIUDADES  
AS  
SELECT DISTINCT CITY  
FROM LOCATIONS;
```

-- BORRADO

```
DROP VIEW CIUDADES;
```

-- MODIFICACION

```
CREATE OR REPLACE VIEW CIUDADES  
AS  
SELECT DISTINCT CITY  
FROM LOCATIONS;
```



Company	4D
Address	111 Main Street
City	New York Paris Sydney Hong Kong

DDL – Consultas a VISTAS

- Aplica lo visto para tablas:
 - ✓ Selección de campos
 - ✓ Filtrar registros

```
SELECT *  
FROM NOMBRE_VISTA  
WHERE CAMPO = VALOR;
```

DDL – Ejercicio 4

- Crear una vista llamada **PAISES_DE_EUROPA** que muestre los nombres de los países (tabla COUNTRIES) que pertenecen a la región **Europe** (tabla REGIONS). Los nombres de los países deben mostrarse en mayúsculas (función UPPER)

```
CREATE OR REPLACE VIEW NOMBRE_VISTA  
AS  
SELECT CAMPO1, CAMPO2  
FROM NOMBRE_TABLA  
WHERE CAMPO = VALOR;
```

DDL – VIEW (with check option)

```
CREATE OR REPLACE VIEW VISTA_EMPLEADOS_WC
AS
    SELECT *
      FROM EMPLOYEES
     WHERE DEPARTMENT_ID = 10
    WITH CHECK OPTION;
```

Con la clausula **WITH CHECK OPTION** no se permiten modificaciones en aquellos campos que afecten a los registros que retorna la vista

DDL – SINONIMOS

- es un nombre alternativo que identifica una tabla en la base de datos
- en ocasiones se usa para simplificar el nombre original de la tabla
- también se suelen utilizar para evitar tener que escribir el nombre del propietario de la tabla
- existen sinónimos públicos y privados

```
CREATE PUBLIC SYNONYM CIUDADES  
FOR HR.CIUDADES;
```

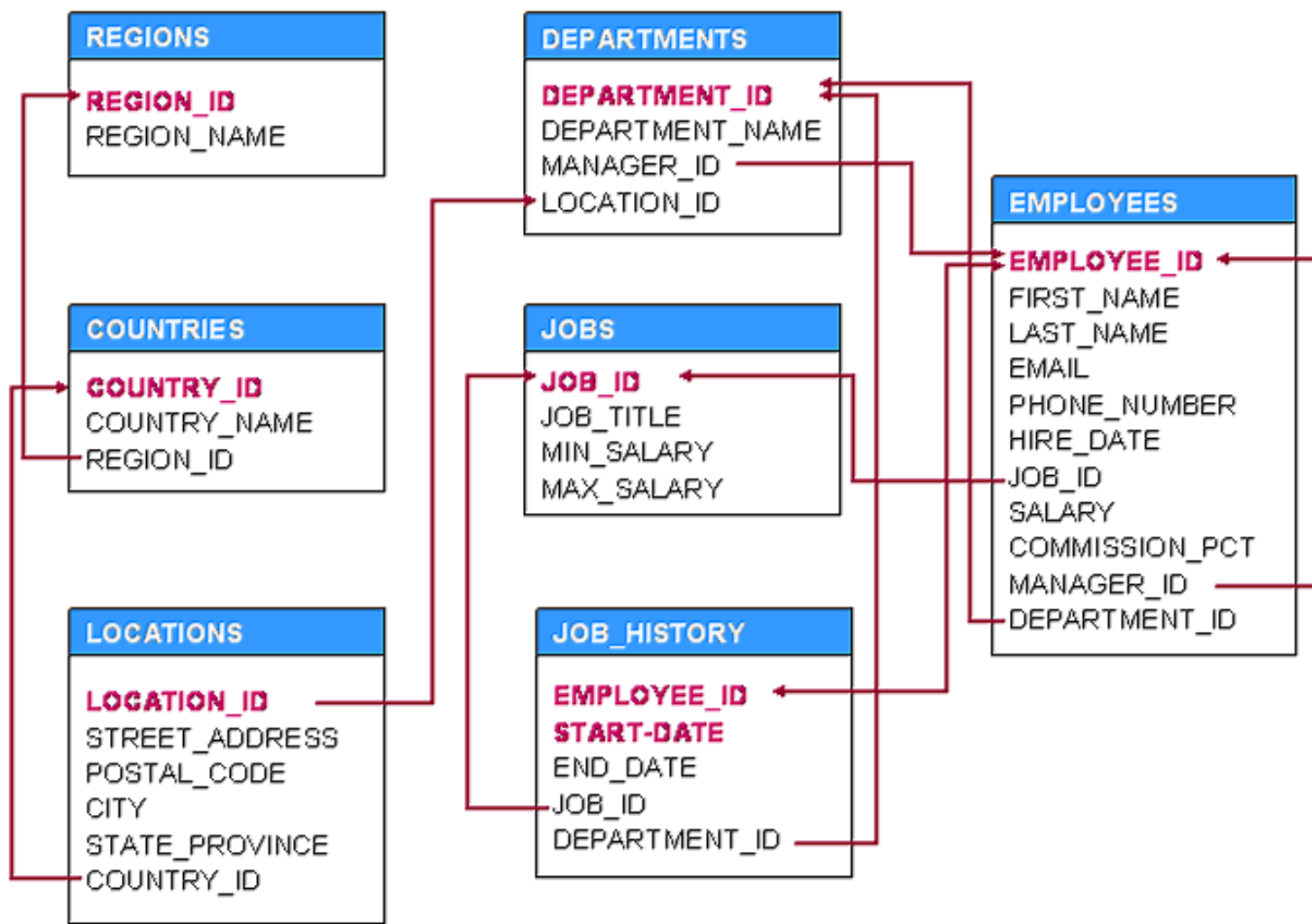
```
DROP PUBLIC SYNONYM CIUDADES;
```

DDL – Ejercicio 5

- Crear un sinónimo público para tabla **ALUMNO** creada anteriormente

```
CREATE PUBLIC SYNONYM NOMBRE_SINONIMO  
FOR OWNER.NOMBRE_TABLA;
```

DDL – INTEGRIDAD REFERENCIAL



DDL – INTEGRIDAD REFERENCIAL

-- PRIMARY KEY EN CREATE TABLE

```
CREATE TABLE TABLE_NAME
(
    COLUMN1 DATATYPE,
    COLUMN2 DATATYPE,
    ...
    CONSTRAINT CONSTRAINT_NAME PRIMARY KEY (COLUMN1, COLUMN_N)
);
```

-- PRIMARY KEY EN ALTER TABLE

```
ALTER TABLE TABLE_NAME
ADD CONSTRAINT CONSTRAINT_NAME PRIMARY KEY (COLUMN1, COLUMN_N);
```

-- ELIMINAR PRIMARY KEY

```
ALTER TABLE TABLE_NAME
DROP CONSTRAINT CONSTRAINT_NAME;
```

DDL – INTEGRIDAD REFERENCIAL

-- EJEMPLOS PRIMARY KEY

CREATE TABLE REGIONS

(

REGION_ID NUMBER,

REGION_NAME VARCHAR2(25),

CONSTRAINT REG_ID_PK PRIMARY KEY (REGION_ID)

);

ALTER TABLE REGIONS

ADD CONSTRAINT REG_ID_PK PRIMARY KEY (REGION_ID);

DDL – INTEGRIDAD REFERENCIAL

-- FOREIGN KEY EN CREATE TABLE

```
CREATE TABLE TABLE_NAME
(
    COLUMN1 DATATYPE,
    COLUMN2 DATATYPE,
    ...
    CONSTRAINT FK_COLUMN FOREIGN KEY
    (COLUMN1, COLUMN2, ... COLUMN_N)
    REFERENCES PARENT_TABLE
    (COLUMN1, COLUMN2, ... COLUMN_N)
);
```

-- FOREIGN KEY EN ALTER TABLE

```
ALTER TABLE TABLE_NAME
ADD CONSTRAINT CONSTRAINT_NAME
FOREIGN KEY (COLUMN1, ... , COLUMN_N)
REFERENCES PARENT_TABLE (COLUMN1, ... , COLUMN_N);
```

-- ELIMINAR FOREIGN KEY

```
ALTER TABLE TABLE_NAME
DROP CONSTRAINT CONSTRAINT_NAME;
```

DDL – INTEGRIDAD REFERENCIAL

-- EJEMPLOS FOREIGN KEY

```
CREATE TABLE COUNTRIES
```

```
(
```

```
    COUNTRY_ID    CHAR(2) ,
```

```
    COUNTRY_NAME  VARCHAR2(40) ,
```

```
    REGION_ID     NUMBER,
```

```
    CONSTRAINT COUNTRY_C_ID_PK PRIMARY KEY(COUNTRY_ID) ,
```

```
    CONSTRAINT COUNTR_REG_FK FOREIGN KEY(REGION_ID)
```

```
    REFERENCES REGIONS(REGION_ID)
```

```
);
```

```
ALTER TABLE COUNTRIES
```

```
ADD CONSTRAINT COUNTR_REG_FK FOREIGN KEY(REGION_ID)
```

```
REFERENCES REGIONS(REGION_ID) ENABLE;
```

DDL – INTEGRIDAD REFERENCIAL

-- DELETE CON FOREIGN KEY ACTIVAS

- **FOREIGN KEYS**

- **FOREIGN KEYS WITH CASCADE DELETE**

```
CONSTRAINT FK_COLUMN  
FOREIGN KEY (COLUMN1, COLUMN2, ... COLUMN_N)  
REFERENCES PARENT_TABLE (COLUMN1, COLUMN2, ... COLUMN_N)  
ON DELETE CASCADE
```

- **FOREIGN KEYS WITH "SET NULL ON DELETE"**

```
CONSTRAINT FK_COLUMN  
FOREIGN KEY (COLUMN1, COLUMN2, ... COLUMN_N)  
REFERENCES PARENT_TABLE (COLUMN1, COLUMN2, ... COLUMN_N)  
ON DELETE SET NULL
```


DDL – OTRAS CONSTRAINTS

UNIQUE CONSTRAINT

Es una restricción que permite identificar univocamente los registros de una tabla mediante uno o varios campos.

Ejemplo: Campo DNI en tabla ALUMNO

DIFERENCIA ENTRE PRIMARY KEY Y UNIQUE

PK → Ningún campo que forma parte de la clave puede ser nulo

UQ → Puede contener algún campo nulo en los campos definidos como unique mientras no exista un registro para la misma combinación

CHECK CONSTRAINT

Es una restricción que permite definir una condition que deben cumplir todos los registros de una tabla.

Ejemplo: Campo ES_ALUMNO_REGULAR en tabla ALUMNO ('S','N')

DDL – UNIQUE CONSTRAINT

-- UNIQUE CONSTRAINT

```
CREATE TABLE TABLE_NAME
(
    COLUMN1 DATATYPE [ NULL | NOT NULL ],
    COLUMN2 DATATYPE [ NULL | NOT NULL ],
    ...
    CONSTRAINT CONSTRAINT_NAME UNIQUE (COLUMN1, COLUMN2)
);
```

```
ALTER TABLE TABLE_NAME ADD
CONSTRAINT CONSTRAINT_NAME UNIQUE (COLUMN1, COLUMN2);
```

```
ALTER TABLE TABLE_NAME
DROP CONSTRAINT CONSTRAINT_NAME;
```

DDL – CHECK CONSTRAINT

-- CHECK CONSTRAINT

CREATE TABLE **TABLE_NAME**

(

 COLUMN1 DATATYPE [NULL | NOT NULL],

 COLUMN2 DATATYPE [NULL | NOT NULL],

 ...

 CONSTRAINT **CONSTRAINT_NAME** CHECK (COLUMN_NAME CONDITION) [DISABLE]

);

ALTER TABLE **TABLE_NAME**

ADD CONSTRAINT **CONSTRAINT_NAME** CHECK (COLUMN_NAME CONDITION) [DISABLE];

ALTER TABLE **TABLE_NAME**

DROP CONSTRAINT **CONSTRAINT_NAME**;

DDL – Ejercicio 6

- Crear las siguientes constraints en la tabla **ALUMNO**.

```
ALTER TABLE ALUMNO ADD  
CONSTRAINT CK_ALUMNO_NOMBRE  
CHECK (NOMBRE = UPPER(NOMBRE)) ;
```

```
ALTER TABLE ALUMNO ADD  
CONSTRAINT UQ_ALUMNO_DNI UNIQUE (DNI) ;
```

- Realizar un insert o update para verificar su funcionamiento.

DDL – INDICES

- similar al índice de un libro
- el objetivo de un índice es acelerar la recuperación de información
- es útil cuando la tabla contiene miles de registros
- permiten optimizar las operaciones de ordenamiento y agrupamiento
- al crear una restricción "primary key" o "unique" sobre una tabla, Oracle automáticamente crea un índice sobre el campo (o los campos) de la restricción y le da el mismo nombre que la restricción

DDL – INDICES

```
SELECT * FROM emp WHERE empno = 7900;
```

INDEX

EMPNO	ROWID
7369	1
7499	2
7521	3
7566	4
7654	5
7698	6
7782	7
7788	8
7839	9
7844	10
7876	11
7900	12
7902	13
7934	14

DATA

ROWID	ENAME	JOB	MGR	HIREDATE	SAL	...
2	ALLEN	SALESMAN	7698	81/02/20	1600	...
6	BLAKE	MANAGER	7839	81/05/01	2850	...
12	JAMES	CLERK	7698	81/12/03	950	...
4	JONES	MANAGER	7839	81/04/02	2975	...
1	SMITH	CLERK	7902	80/12/17	800	...
11	ADAMS	CLERK	7788	83/01/12	1100	...
5	MARTIN	SALESMAN	7698	81/09/28	1250	...
14	MILLER	CLERK	7782	82/01/23	1300	...
9	KING	PRESIDENT		81/11/17	5000	...
7	CLARK	MANAGER	7839	81/06/09	2450	...
10	TURNER	SALESMAN	7698	81/09/08	1500	...
13	FORD	ANALYST	7566	81/12/03	3000	...
3	WARD	SALESMAN	7698	81/02/22	1250	...
8	SCOTT	ANALYST	7566	82/12/09	3000	...

DDL – INDICES

```
CREATE [UNIQUE] INDEX INDEX_NAME  
ON TABLE_NAME (COLUMN1, COLUMN2, COLUMN_N);
```

```
ALTER INDEX INDEX_NAME  
RENAME TO NEW_INDEX_NAME;
```

```
DROP INDEX INDEX_NAME;
```

DDL – Ejercicio 7

- Crear el índice **IDX_ALUMNO_APELLIDO** para el campo **APELLIDO** de la tabla **ALUMNO**

```
CREATE [UNIQUE] INDEX INDEX_NAME  
ON TABLE_NAME (COLUMN1, COLUMN2);
```


DDL – Ejercicio 7

- Crear el índice **IDX_ALUMNO_APELLIDO** para el campo **APELLIDO** de la tabla **ALUMNO**

```
CREATE INDEX IDX_ALUMNO_APELLIDO  
ON ALUMNO (APELLIDO) ;
```

DDL – VISTAS UTILES

Se pueden consultar las siguientes vistas para ver los objetos creados:

- USER_OBJECTS
- USER_TABLES
- USER_VIEWS
- USER_CONSTRAINTS
- USER_INDEXES
- USER_IND_COLUMNS