DDL:

● CREATE, DROP, ALTER TABLE.
● CREATE,….., VIEW.
● CREATE,…., INDEX.
● CREATE, …., SYNONYM

Para los datos:

● VARCHAR2(tamaño), cadenas long variable, max 4000 bytes
● CHAR(tamaño) cadenas long fija, max 2000 bytes
● NUMBER(precision, escala)
● LONG, cadenas long variable, max 2 gigabytes
● DATE, fechas, siglo/año/mes/dia/hora/minutos/segundos
● RAW(tamaño), cadenas de bytes, máximo 2000bytes.
● LONG RAW, graficos, sonidos, 2 gigabytes.
● ROWID, cadena hexadecimal, direccion de fila en tabla.
● CLOB, NCLOB, BLOB: Objetos binarios de más de dos gigabytes.

Restricciones en DDL (Data Definition Language)
1. NOT NULL

Propósito: Obliga a que una columna no acepte valores nulos.
Ejemplo de uso:
sql
```
CREATE TABLE pedidos (
   cantidad NUMBER NOT NULL
);
```
2. UNIQUE

Propósito: Garantiza que los valores en una columna o conjunto de columnas no se repitan. Admite valores nulos.
Ejemplo de uso:
sql
```
CREATE TABLE usuarios (
   login VARCHAR2(50) UNIQUE
);
```
3. PRIMARY KEY

Propósito: Define una clave primaria única por tabla. Puede estar compuesta por múltiples columnas. Los campos de la clave primaria son automáticamente NOT NULL y UNIQUE.
Ejemplo de uso:
sql
```
CREATE TABLE personas (
   id NUMBER,
   lastName VARCHAR2(50),
   PRIMARY KEY (id, lastName)
);
```

4. FOREIGN KEY (REFERENCES)

Propósito: Define una clave ajena que referencia una clave primaria de otra tabla para garantizar la integridad referencial.

Opciones de integridad referencial:

-RESTRICT: Impide modificar/eliminar registros relacionados.
-CASCADE: Modifica/borra automáticamente los registros relacionados.
-SET NULL: Asigna NULL a los registros relacionados.
-NO ACTION: No realiza cambios en las claves foráneas.

Ejemplo de uso:
sql
```
CREATE TABLE telefonos (
   id_persona NUMBER,
   FOREIGN KEY (id_persona) REFERENCES personas(id) ON DELETE CASCADE
);
```

5. DEFAULT
Propósito: Establece un valor por defecto para una columna.

Ejemplo de uso:
sql
```
CREATE TABLE productos (
   stock NUMBER DEFAULT 0
);
```

6. CHECK

Propósito: Define una condición que los valores en una columna deben cumplir.

Ejemplo de uso:
sql
```
CREATE TABLE empleados (
   salario NUMBER CHECK (salario > 0)
);
```

Motores de almacenamiento en MySQL (aplicable a tablas)

InnoDB: Soporta transacciones seguras y claves foráneas.

MyISAM: Optimizado para consultas rápidas, pero sin soporte para claves foráneas.

Otros motores: BerkeleyDB, HEAP (MEMORY), ISAM, MERGE.

Ejemplo de uso:

sql
```
CREATE TABLE datos (
   id NUMBER
) ENGINE = InnoDB;
```

Ejemplos de constraint:

ALTER TABLE table_name ADD CONSTRAINT constraint_name [Constraint]

PRIMARY KEY -> ALTER TABLE table_name ADD CONSTRAINT constraint_name PRIMARY KEY (col1, col2);
FOREIGN KEY -> ALTER TABLE orders ADD CONSTRAINT users_fk FOREIGN KEY (user_id) REFERENCES users (user_id);
UNIQUE -> ALTER TABLE users ADD CONSTRAINT email_unique UNIQUE (email);
CHECK -> ALTER TABLE orders ADD CONSTRAINT check_revenue_positive CHECK (revenue >= 0);

ELIMINAR CONSTRAINT
ALTER TABLE table DROP CONSTRAINT constraint_name;

ELIMINAR TABLE
DROP TABLE table_name;

Creación de usuarios:

1.- Nos conectamos al servicio xepdb1 con el usuario system
2.- Ejecutamos las siguientes sentencias:
create user nombre_usuario identified by password_usuario;
grant connect, resource to nombre_usuario;
ALTER USER nombre_usuario quota unlimited on USERS;

Inserción de datos: Comando INSERT

Propósito:
Se utiliza para añadir datos a una tabla.

Sintaxis básica:

sql
```
INSERT INTO tabla [(listaDeCampos)]
VALUES (valor1 [, valor2 ...]);
```
Detalles importantes:
Lista de campos:

Si no se especifica, los valores deben seguir el orden de las columnas según su creación.
Orden obtenido con el comando DESCRIBE tabla.
Valores no indicados:

Si un campo tiene un valor por defecto (DEFAULT), se utiliza ese valor.
Si no tiene valor por defecto, se rellena con NULL.
Si el campo tiene la restricción NOT NULL, dará error si no se proporciona un valor.
Ejemplo de uso:

Con valores para todos los campos:

sql
INSERT INTO clientes
VALUES ('11111111', 'Pedro', 'Gutiérrez', 'Crespo', DEFAULT, NULL);

Indicando sólo algunos campos:

sql
INSERT INTO clientes(dni, nombre, apellido1, apellido2)
VALUES ('11111111', 'Pedro', 'Gutiérrez', 'Crespo');
Actualización de registros: Comando UPDATE

Propósito:
Se utiliza para modificar los datos existentes en una tabla.

Sintaxis básica:

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

Detalles importantes:
SET: Define las columnas y sus nuevos valores.

WHERE:
Es opcional, pero se recomienda para evitar modificar todos los registros de la tabla.

Sin condición WHERE:
La instrucción afecta a todos los registros de la tabla.

Ejemplo de uso:
Actualizar el nombre de un cliente con un DNI específico:

sql
UPDATE clientes
SET nombre = 'Juan'
WHERE dni = '11111111';