BLOQUE 3: Lenguajes de bases de datos relacionales
TEMA 7: SQL definicion de datos.
SQL: lenguaje de BD completo, cuyos comandos se organizan en

- **lengaje de definicion de datos (LDD)**
- **lenguaje de manipulacion de datos (LMD)**
- **otros**

ANSI SQL vs Modelo Relacional formal
ANSI SQL:
Utiliza los términos tabla, columna, fila, …
Una tabla puede tener filas idénticas. Permite definir tablas sin claves: las filas se identifican internamente, a nivel físico
☐ Columnas de tabla ordenadas en orden de creación
☐ Es posible indicar un orden de visualización de las filas
☐ Una clave ajena puede referenciar a una clave primaria o bien a una clave alternativa (UNIQUE)
Modelo Relacional formal:
Utiliza los términos formales relación, atributo, tupla, …
Una relación jamás contiene tuplas repetidas. Relación = conjunto (set) de tuplas
☐ Atributos de relación no ordenados
☐ Una clave ajena sólo puede referenciar a una clave primaria

El **Diseño Lógico Específico** consiste en escribir el Esquema Lógico con la sintaxis propia del modelo de datos particular del SGBD comercial elegido.

CREACION DE TABLAS:
Sentencia **CREATE TABLE** define (crea) una tabla: nombre, columnas y restricciones
⊙ Nombre único dentro del esquema
⊙ Para cada Columna hay que especificar:

- nombre
- tipo de datos
- restricciones de columna

⊙ Restricciones de tabla:

- de clave candidata,
- de integridad de entidad
- de integridad referencial
- restricciones de otro tipo

Ordenamiento de columnas y filas: una vez creada la tabla, sus columnas quedan ordenadas tal como aparecen en la instrucción CREATE TABLE. Las filas quedan en el orden de inserción
Las tablas creadas con CREATE TABLE son denominadas **tablas base** (el SGBD las almacena físicamente en algún fichero de la base de datos en disco)
Columnas de tabla, en la sentencia CREATE TABLE cada columna se define en una linea seguida de una coma. Para cada columna hay que especificar: el nombre de la columna, el tipo de datos y restricciones de columna.
Tipos de datos:

- Numericos
  - INTERGER, INT
  - REAL, DOUBLE PRECISION, FLOAT
  - **con formato:** NUMERIC(precision, escala)
- Cadenas de caracteres
  - longitud fija CHAR(n)
  - longitud variable VARCHAR(n)
- Cadenas de bits

    - longitud fija BIT(n)
    - longitud variable BIT VARYING(n)
- Temporales
    - DATE
    - TIME
    - TIMESTAMP fracciones de segundo y desplazamiento respecto al UTC(uso horario estandar)
    - INTERVAL periodo de tiempo para incrementar/decrementar el valor actual

Restricciones (de columna):

- Clausula **NULL o NOT NULL:** indica si una columna puede contener NULL o no.
- Clausula **DEFAULT valor:** (jsuto detras del tipo de datos), si una columna no tiene DEFAULT, su valor por defecto es NULL, si admite null y si no no admite valor por defecto.

Restricciones (de tabla): incluye la definición de **restricciones de integridad** que afectan al contenido de la tabla, es decir a todas las filas:

- Clausura **PRIMARY KEY(lista_columnas)** indica que columnas componen la clave primaria. Sus valores SIEMPRE seran unicos.
- Clausula **UNIQUE(lista_columnas)** indica que columnas forman una clave alternativa, hay que incluir una distinta para cada una de las claves alternativas que tenga la tabla. Se permite que una clave UNIQUE contenga el NULL. **Puede ser compuesta.**
- Clausula **CHEK(expresion)** para especificar resticciones adicionales, expresan una condicion sobre una o varias columnas que debe cumplir toda la fila de la tabla.
- Clausa **FOREIGN KEY(lista columnas) REFERENCES tabla(columnas_clave)** define que columnas forman una clave ajena.

Cada restricción se define en una línea y finaliza con una coma, salvo la última.

DEFINIR CLAVES AJENAS:

Cada clave ajena (FK) debe estar formada por tantas columnas como tenga la clave a la que referencia.

Cada columna de la FK debe tener el mismo tipo de datos que la columna correspondiente de la clave a la que referencia.

Al definir una clave ajena mediante FOREIGN KEY el **SGBD garantiza la Integridad Referencial**.

Al definir cada clave ajena hay que indicar las acciones de mantenimiento de la integridad referencial, o acciones referenciales, teniendo en cuenta su semántica. En la cláusula FOREIGN KEY hay que añadir las cláusulas **ON DELETE** acción y **ON UPDATE** acción.

Opciones de ON DELETE:

1.- Rechazar la operación, solo permite borrar si ninguna fila referencia a t. **NO ACTION.**

2.- Cascada, propagar la eliminacion, borrar todas las filas que referencian a t. **CASCADE.**

3.- Establecer nulo, toda fila de que referencia a t pasa a contener NULL en la FK. **SET NULL**

4.- Estableces valor por defecto, toda la fila que referencia a t toma su valor por defecto en la columna de la FK. **DEFAULT.**

Si tengo una relacion que proviene de una entidad debil se debe borrar en cascada siempre, en otros casos en los que usar el cascade tiene que estar muy claro, es muy raro usarlo. Lo habitual es el no acction.

Opciones de ON UPDATE:

1.- Rechazar la operación, solo se permite modificar si ninguna fila lo referencia.

2.- Cascada, propagar la modificacion, cambia su valor de la FK por el nuevo valor de la Ck de t.

3.- Estableces nulos toda fila que referencia a t pasa a contener NULL en la FK.

4.- Valores por defecto, toda fila de R2 que referencia a t toma su valor por defecto.

Hay que completar cada FOREIGN KEY con las clausulas ON DELETE y ON UPDATE.

NOMBRES DE RESTRICCION:
Anteponer a una cláusula PRIMARY KEY, FOREIGN KEY, UNIQUE o CHECK esto: CONSTRAINT nombre_RI, el nombre debe ser unico dentro de un mismo esquema.
Identifica una restricción, por si después debe ser eliminada o sustituida por otra y el SGBD lo usa en los mensajes de error generados ante intentos de incumplir la restricción.

Una **clave ajena es compuesta** cuando lo es la clave (primaria o alternativa) a la que referencia.

ALTERAR LA ESTRUCTURA DE UNA TABLA (ALTER TABLE):
Permite modificar la estructura de una tabla, existente, independientemente que contenga datos (filas).

- Añadir nueva columna **ADD**
- Modificar una columna ya existente **MODIFY, RENAME**
- Eliminar una columna. **DROP**
  - Solo se elimina una columna si no esta referenciada
  - No esta permitido una columna referenciada.
  - Se puede forzar su eliminacion con la clausula **CASCADE CONSTRAINTS**
- Añadir restriccion **ADD CONSTRAINT**
  - añadir una CHEK, una clave primaria, una clave alternativa.
  - Añadir una clave ajena
- Modificar una restriccion
  - Inhabilitar una restriccion de tabla ALTER TABLE tabla **DISABLE CONSTRAINT nombreRI;**
  - Habilitar una restriccion de tabla ALTER TABLE tabla **ENABLE CONSTRAINT nombreRI;**
- Eliminar una restriccion
  - eliminar la restriccion de clave primaria
  - eliminar una restriccion de clave alternativa
  - eliminar cualquier restriccion
  - eliminar una PRIMARY KEY o UNIQUE falla si existe alguna clave ajena que le haga referencia

TEMA 8 SQL: Manipulacion de Datos (DML).
CONSULTA:
SQL permite extraer información almacenada en las tablas de una base de datos relacional, hay que redactar cada consulta indicando qué datos queremos extraer: qué condiciones deben cumplir.
SQL. Sentencia SELECT:
Instrucción básica de obtención de información

```
SELECT lista_columnas
FROM lista_tablas
WHERE condición ;
```

La consulta selecciona las filas de lista_tablas que satisfacen condición y proyecta el resultado sobre las columnas de lista_columnas, el resultado sera una **tabla** con las columnas indicadas y las filas seleccionadas.
Puedes poner cualquier numero de condiciones en SELECT.
Una SELECT puede obtener filas repetidas, demasiado costoso. Cláusula **DISTINCT** (o UNIQUE): Para eliminar filas repetidas del resultado de una consulta.
Obtención de los valores de todas las columnas de las filas seleccionadas, uso del símbolo *****
Operador **LIKE,** comparación de cadenas de caracteres (textos)

- % significa “una cantidad cualquiera de caracteres cualesquiera (letras y dígitos)”
- _ significa un solo carácter (una letra o un dígito)

Operador || : concatenación de cadenas de caracteres
Operaciones **aritméticas**: Aplicación de operadores aritméticos (+, -, *, /) sobre valores numéricos. Se muestra el resultado de la operación, pero no se modifica ningún dato almacenado en la tabla.
Operaciones con **fechas**: funciones para convertir fechas (DATE) en cadenas de caracteres y al contrario:

- TO_CHAR (fecha, formato): convierte un valor de tipo DATE a una cadena de caracteres que expresa una fecha en el formato que se establece como segundo parámetro
- TO_DATE (cadena_texto, formato): convierte una cadena de caracteres, escrita en el formato indicado en el 2º parámetro, en un valor de tipo DATE

Función **EXTRACT** (campo FROM fecha): devuelve el valor de un campo concreto de una columna tipo DATE.
Operador **IN (NOT IN)** Un conjunto explícito de valores es una lista de valores, separados por comas, encerrada entre paréntesis, puede aparecer en la clausula where. Indica si el valor v pertenece al conjunto de valores V (devuelve true o false)
Operador **NULL** no es un valor, sino una marca que indica desconocimiento o ausencia de información. Comparar NULL usando =, <>, >, >=, <, <=, con cualquier cosa siempre da FALSE o UNKNOWN. NULL es distinto a cualquier otra cosa, incluso a otro NULL. Es necesario un comparador específico para comprobar si una columna contiene o no el NULL, Operador **IS NULL, IS NOT NULL**
Operador **ORDER BY,** presentar las filas resultado de una columna de ordenada.

- Ordenación según valores de una o varias columnas
- Ascendente ASC (por defecto) o Descendente DESC
- Suele ser una operación muy costosa
- Las filas no se ordenan en disco: se ven ordenadas, pero no cambian dentro de la tabla.

Operacion **JOIN**, sacar información de dos filas que están relacionadas entre sí, ese vínculo entre ellas está representado mediante la referencia que una fila tiene hacia la otra. Para especificar una reunión en la sentencia SELECT, hay que indicar los nombres de las tablas como operandos del JOIN en la cláusula FROM Y la condición de reunión se incluye en la cláusula **ON.**
La condición de reunión puede incluir más de una comparación por igualdad, ligadas con AND, si es necesario usar dos o más columnas para combinar correctamente las filas de una tabla con las filas de la otra tabla. **Esto ocurre cuando la clave ajena y la clave primaria son compuestas.**

En SQL los nombres de las columnas deben ser únicos dentro de cada tabla, pero distintas tablas pueden tener columnas denominadas igual. Si en una consulta aparecen varias columnas de igual nombre, pero de tablas distintas → AMBIGÜEDAD. Se puede utilizar pseudónimos (alias) para acortar los nombres de tabla que se usan para calificar.
Es posible dar nuevos nombres para las columnas del resultado de la consulta, es una nueva cabecera para la tabla resultado.
Es posible incluir varios operadores JOIN ON, para realizar reuniones entre más de dos tablas.

```
SELECT lista_columnas_de_R_y_S
FROM R, S
WHERE condición_de_reunión;
```

Selección incondicional: no incluir WHERE equivale a una condición TRUE para todas las filas. Si en la cláusula FROM aparece sólo una tabla, se obtienen todas las filas de dicha tabla. Si el FROM incluye más de una tabla, se obtiene el producto cartesiano entre dichas tablas (Se combina cada fila de una tabla con todas las filas de la otra)

El operador **JOIN ON**, ademas de permitir especificar una reunion “interna”, que ya hemos visto tambien permite especificar reuniones de otros tipos:

- Reunion Natural

☐ NO necesita condición de reunión. No incluye la cláusula ON condición
☐ El SGBD asume una condición de reunión para cada par de columnas con igual nombre en una y otra tabla
☐ Y sólo se incluye una de estas columnas en el resultado. Elimina una de las columnas idénticas
☐ La reunión natural sólo devuelve una de las dos columnas de reunió
☐ Si se usan alias para las tablas, no se puede calificar una columna de reunión fuera del natural join

- Reunion Externa: Necesaria si en una reunión se necesita obtener las filas con valor NULL en las columnas de reunión, o sin correspondencia en la otra tabla
  - LEFT [OUTER] JOIN Reunión externa izquierda
  - RIGHT [OUTER] JOIN Reunión externa derecha
  - FULL [OUTER] JOIN Reunión externa total o completa

Función **COALESCE()** : permite mostrar el valor que se desee en cierta columna del resultado de una consulta, en lugar de un NULL. Suele usarse con dos parámetros: COALESCE(columna, valor)

Operacion de conjuntos: **UNION, INTERSECT, MINUS.**
Las filas repetidas se eliminan, y se deja sólo una de ellas en la tabla resultado. Las tablas operando han de ser compatibles en tipo, igual número de columnas y columnas “correspondientes” (en la misma posición) deben tener el mismo tipo de datos.

- Union: La unión entre dos tablas obtiene una tabla con todas las filas de las dos tablas de origen (y sin duplicados). Como el resultado de una SELECT es una tabla, es posible hacer la UNION entre dos consultas (entre dos SELECT)
- Intersect: La intersección entre dos tablas obtiene una tabla con las filas que están a la vez en las dos tablas de origen (y sin duplicados). Como el resultado de una SELECT es una tabla, es posible hacer la INTERSECCIÓN entre dos consultas (entre dos SELECT).
- Minus:La resta entre dos tablas obtiene una tabla con las filas que están en la tabla de la izquierda y que no están en la tabla de la derecha (y sin duplicados). Como el resultado de una SELECT es una tabla, es posible hacer la RESTA entre dos consultas (entre dos SELECT).

CONSULTAS ANIDADAS
Una consulta anidada es una consulta SELECT completa, dentro de cláusula WHERE de otra consulta (consulta exterior)
Obtiene valores de la BD que se usan en la condición de otra consulta, para obtener otros datos.
Operador **IN** (también NOT IN) -- otro uso del operador t IN S – Indica si la fila t pertenece al conjunto de filas S (subconsulta).
Si la consulta anidada devuelve una sola columna y una única fila, el resultado es un valor escalar, y se permite usar cualquier comparador (=, <, >, <=, >=, <>) en lugar de IN.

- **Operador ANY o SOME** (otro uso del mismo operador)
  t op ANY S o t op SOME S,, op ∈ { >, ≥, <, ≤, < >, = }
  - Compara una fila **t** con las filas resultado de una consulta anidada **S**
  - Devuelve TRUE si **alguna** fila **e** de **S** cumple que **t op e**

- **Operador ALL** (otro uso del mismo operador)
  t op ALL S,, op ∈ { >, ≥, <, ≤, < >, = }
  - Compara una fila **t** con filas resultado de una consulta anidada **S**
  - Devuelve TRUE si **para toda** fila **e** de **S** se cumple que **t op e**

CORRELACION: A veces, una consulta anidada necesita usar columnas de una tabla declarada en el FROM de una consulta exterior. Una consulta exterior y otra anidada están **correlacionadas** si una condición de la anidada contiene columnas de una tabla declarada en la consulta exterior.
Si las columnas en las consultas exterior y anidada se llaman igual, entonces existe **ambigüedad**.
Solución: CALIFICAR la columna ambigua de la tabla exterior.
Operador EXISTS: comprobación de tablas vacías. Devuelve TRUE si S contiene al menos una fila, Devuelve FALSE si S está vacía (sin filas).

Funciones de agregados: (aparecer en clausula SELECT)

- **SUM()**
- **MAX()**
- **MIN()**
- **AVG()** (media aritmetica)

Admiten DISTINCT y ALL (por defecto).

Función COUNT( ): Cuenta número de filas (usando *) o valores no nulos en una columna poniendo entre parentesis el nombre de la columna (puede aparecer en la cláusula SELECT), para contar valores distintos usar dentro del count DISTINCT.

Clausula GROUP BY: para formar grupos de filas dentro de una tabla, los grupos se forman según el valor de ciertas columnas. Columnas de agrupación: las filas de cada grupo tendrán el mismo valor en las columnas de agrupación Aplicación de funciones agregadas a grupos de filas.

Clausula HAVING SIEMPRE JUNTO A GROUP BY. Condicion que deben cumplir los grupos de filas asociados a cada valor de las columnas de agrupacion. Un grupo que no cumple una condicion es descartado.

WHERE → filas individuales VS HAVING → grupos de filas

Oracle Online View: consulta dentro de la clausula FROM de otra SELECT, vista en linea.
NO es una consulta anidada ya que no esta dentro de la clausula where de otra consulta, si no en el FROM.

Division en SQL: Obtener de las filas de una tabla que estan relacionadas con todas y cada una de las filas de otra tabla. No existe un operación para realizar la division. REDACTAR CONSULTA DE FORMA QUE SEA POSIBLE OBTENER EL RESULTADO. **NEGAR EL ENUNCIADO.**
**Forma diapositiva 124 PENALIZA

MODIFICACION DE DATOS:
☐ Orden INSERT: añade una fila completa a una tabla, mismo orden con el que se creo la tabla. Para poder poner los valores de las columnas en cualquier orden, hay que especificar los nombres de las columnas antes de la cláusula VALUES. Se pueden omitir las columnas que tengan valores por defecto o que admitan null.
Comprobación automática (SGBD) de restricciones.
⊙ INSERT .. SELECT: Es posible rellenar una tabla extrayendo su contenido de la base de datos, mediante una SELECT
⊙ (LDD) CREATE AS SELECT: crear una tabla como “copia de estructura y contenido” de otra tabla CREATE TABLE AS SELECT … ;
☐ Orden DELETE: Elimina filas completas de una tabla (sólo una en el FROM). Cláusula WHERE para seleccionar las filas que se desea elimina. **Propagación de eliminaciones vía claves ajenas.**
☐ Orden UPDATE: Modifica valores de columnas en una o más filas de una tabla.

TEMA 9: Definicion de Reglas de Integridad y otros elementos (DDL).
9.1 Reglas de Integridad
Una BD es una representacion o modelo de una porcion del mundo real, por ello la informacion debe cumplir restricciones, reglas o normas.
Al crear un esquema de BD Relacional estas restricciones qeudan almacenadas junto a los datos de forma de **reglas de integridad.**
La integridad es la consistencia o correccion de los datos almacenados en la base de datos.
Las reglas de integridad (**RI**) afectan a los datos almacenados y se deben cumplir SIEMPRE, es importante en las etapas de **diseño** y el la **ejecucion**.

Componentes de una Regla de Integridad:
- Nombre: regla almacenada en los metadatos con ese nombre, aparecera en los diagnosticos producidos por el sistema como respuesta a intentos de incumplimiento de la regla.
- Restriccion: expresion boolean.
- Respuesta a un intento de incumplimiento de la regla, indica al SGBD que hacer si se intenta una operación que imcumple la RI. Por defecto RECHAZAR.

Categorias de las RI
- Reglas de integridad de **Dominio:** asociadas a un dominio de datos especificos.(No las vemos)
- Reglas de integridad de **Tabla:** RI de complejidad arbitraria incluidas en la definición de una tabla.
- Reglas de integridad **Generales**: RI de complejidad arbitraria, no incluidas en la definición de ninguna tabla. Son otro elemento más de la BD, al mismo nivel que una tabla o vista.

RESTRICCIONES DE TABLA
Restricciones asociada a una tabla especifica, no existe si no existe la tabla y se eliminan con la tabla. Se especifican dentro de CREATE TABLE.
Toda RI de tabla es comprobada inmediatamente: una operación de modificación sobre la tabla incluye el chequeo de todas sus RI (como paso final de la operación)+(una posible) acción.

RESTRICCIONES GENERALES (ASERTOS):
Restricciones especificas de cada base de datos particular, permiten especificar restricciones de integridad mas alla de las de dominio y de tabla.
Suelen implementar restricciones detectadas en el Diseño Conceptual y/o Logico.
Un aserto incluye un predicado que expresa una condición que la base de datos debe satisfacer siempre. Puede involucrar cualquier número de columnas de cualquier cantidad de tablas. Es un elemento de BD independiente de tablas y vistas existentes ☐ Tiene un nombre y consta de una condición.
Creacion: CREATE ASSERTION <nombre_aserto>⇨ nombre **obligatorio**
CHECK (<condicion>) ;
Eliminacion: DROP ASSERTION <nombre_aserto>;
Satisfaccion e incumplimiento de una RI general, se incumple el aserto si en alguna fila de la BD hace falsa la condicion.
Oracle SQL NO implementa directamente las sentencias CREATE ASSERTION ni DROP ASSERTION. Sí podremos ejecutar la SELECT que incluye y comprobar que devuelve una tabla vacía.
Condicion expresada en **negativo:** todo X satisface Y = ningun X satisface NO(Y).

Para crear una RI, el SGBD comprueba que los datos actuales cumplen todos esa RI, si no no deja crearla, para destruirla el sistema elimina su definicion del INFORMARTION_SCHEMA.
Una regla de integridad es **independiente de cualquier aplicación** especifica que acceda a la base de datos.

El SGBD rechaza todo intendo de INSERT o UPDATE que infrinja una especificacion de tipo de datos, **el incumplimiento de una RI de dominio o tipo de datos se detecta en tiempo de ejecucion**.
Comprobacion de restricciones: el SGBD comprueba una RI de inmediato, como último paso de la ejecución de una sentencia SQL. Si la RI es incumplida, la sentencia es cancelada y no tiene efecto en la BD. Pero a veces es necesario que ciertas restricciones no sean comprobadas hasta pasado un tiempo, pues si se hiciera de inmediato siempre fracasarían → hay que usar la sentencia ALTER TABLE para **desactivar (inhabilitar) temporalmente una restricción.**

9.2 Vista relacional
Es una "tabla" obtenida como resultado de la ejecución de una consulta, definida mediante una SELECT que extrae o deriva datos de ciertas tablas base. Presenta datos contenidos en una o más tablas (o vistas). Permite tratar el resultado de una consulta como una tabla.

Carasteristica fundamental → **actualizacion permanente.** Esto se debe a que la vista no se crea cuando se define, sino cada vez que se consulta. **Una vista no contiene informacion DEJA VER INFORMACION almacenada en sus tablas base.**

USOS: dar nombre y almacenar consultas complejas, presentar datos con una perspectiva diferente a las tablas base, ocultar la complejidad de los datos y por tanto simplificar las fentencias para el usuario, proporcionan seguridad (restringen el acceso a un conjunto de filas y/o de columnas predeterminadas a una tabla), aislar a las aplicaciones de los cambios en la definicion de las tablas base.

Nombres de columna: la vista 'hereda' los nombres de las columnas seleccionadas desde la tabla base, detras del nombre de la VIEW → (nuevos nombres).

Una vista puede aparecer en el FROM de la SELECT de definición de otra vista.

Creacion de vistas:

```
CREATE [ OR REPLACE ] [ [ NO ] FORCE ] VIEW nombre_vista
    [ (lista_nombres_columnas) ]
    AS consulta_definición
    [ WITH { READ ONLY
           | CHECK OPTION } [CONSTRAINT nombre_restricción] ]
```

OR REPLACE: permite reemplazar la definicion de la vista en el diccionario de datos, evitando eliminarla y volver a crearla.
Si la vista se definió con SELECT * y se añade columnas a las tablas base, la vista no incluirá esas nuevas columnas hasta ejecutar CREATE OR REPLACE
FORCE permite crear la vista siempre, aunque no existan las tablas base o el propietario de la vista no tenga permisos sobre ellas
WITH READ ONLY prohíbe hacer UPDATE, INSERT y DELETE sobre las tablas base a través de la vista.

Consulta de datos a traves de vistas:
Las vistas NO tienen limitación en operaciones de consulta. El usuario no distingue si el elemento al que accede es una tabla o una vista.

Manipulacion de datos a traves de vistas: El SGBD traduce (si puede) cualquier sentencia INSERT, UPDATE y DELETE sobre la vista a una expresión equivalente sobre sus tablas base, una vista no contiene datos asi que sobre una vista no puede modificar el contenido de la vista, **hay que modificar los datos almacenados en las tablas base.** Las actualizaciones de datos a traves de vistas si tiene **limitaciones.**
Algunas actualizaciones no tienen sentido.
Actualizar a través de una vista definida sobre varias tablas base suele dar problemas, pues puede haber ambigüedad, una modificacion puede traducirse a dos o mas actualizaciones distintas de las tablas base. El SGBD no puede saber qué actualización es correcta y cuál no, y si hubiera varias correctas, no sabría cuál elegir como la más adecuada. Así que no se garantiza que "toda vista sea actualizable". **Una vista sería actualizable si implicara una única actualización posible de las tablas base.**

- Una vista con una sola tabla base
  - **SÍ es actualizable si** sus columnas **contienen una clave** (primaria o alternativa) de la tabla base
    - Así se establece una correspondencia entre cada fila de la vista y una única fila de la tabla base
- Una vista definida sobre varias tablas mediante JOIN (reunión)
  - **NO** es actualizable
- Una vista definida con agrupación y funciones agregadas
  - **NO** es actualizable

9.3 Indices
El concepto de índice no pertenece al nivel conceptual ni lógico, sino al nivel físico (no incluido en SQL). En le modelo relacional se indica que las tablas no estan ordenadas. La ejecución de una consulta SQL implica la búsqueda de ciertas filas con base en ciertos datos y la recuperación de (parte de) su contenido. Para encontrar una fila → busqueda secuencial **No es eficiente.**
Un **índice** es una estructura de datos auxiliar que permite acelerar el acceso a las filas de una tabla con base en los valores de una o más columnas
Indice → entradas con dos campos:

- Valor existentes en una columna de indexacion
- Puntero al bloque dentro del fichero de datos que contiene el registro con dicho valor del campo de indexacion

Las entradas estan ordenadas según el valor del campo de indexacion → PERMITEN REALIZAR **BUSQUEDAS BINARIAS**.
Oracle lo hace àutomaticamente de toda clave primaria y alternativa.
Un campo **columna no clave,** que se usa muy frecuente en condiciones de selección o en operadores de reunion.
En una tabla **gande** sus consultas mas frecuentes recuperan menos del 20x100 de las filas
Creacion de indices:

```
CREATE INDEX nombre_índice
    ON nombre_tabla (columna [ASC | DESC]
            [, columna [ASC | DESC]...])
    ... ;
```

Son lógica y físicamente independientes de los datos de la tabla asociada

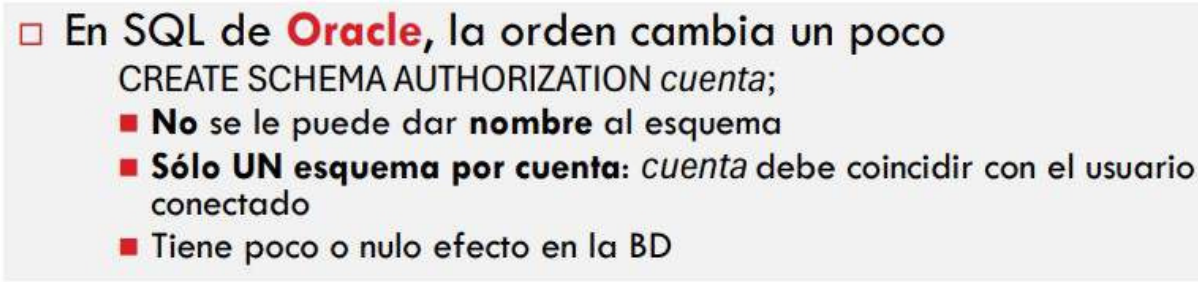
Eliminacion de indices:
Sentencia DROP INDEX nombre_índice;
Razones para eliminar un índice

Ya no se espera realizar consultas basadas en su campo de indexación
No se usa o no acelera las consultas

El espacio que ocupaba queda disponible y desaparece el coste de su mantenimiento (actualización permanente)
Para eliminar un índice asociado a UNIQUE o a PRIMARY KEY hay que desactivar (DISABLE) o eliminar (DROP) la restricción.

9.4 Esquema de Base de Datos Relacional.
Un esquema agrupa tablas y otros elementos relacionados de algún modo: los de un mismo usuario, los de cierta aplicación, etc.
Orden CREATE SCHEMA
CREATE SCHEMA nombre_de_esquema
AUTHORIZATION identificador_de_autorización
identificador_de_autorización: es el usuario/cuenta propietario del esquema
Una vez creado el esquema, a continuación se puede especificar las definiciones de los elementos contenidos en él.
Se puede indicar el esquema donde crear una tabla

- Esquema Explícito: Un usuario puede crear una tabla en el esquema de otro usuario (si tiene permiso para ello) CREATE TABLE empresa.empleado (...);
- Esquema Implícito en el contexto: Un usuario crea una tabla en el esquema activo en su cuenta CREATE TABLE empleado (...);



Orden DROP SCHEMA
Destruye un esquema de BD, junto con su definicion en el INIFORMATION_SCHEMA del catalogo.
DROP SCHEMA nombre_esquema opción; Con opcion:

- RESTRICT: Destruye el esquema sólo si no contiene elementos
- CASCADE: Elimina el esquema y además las tablas, dominios y demás elementos contenidos en dicho esquema

9.5 Catalogo de la base de datos relacional.
Conjunto nombrado de t**odos los esquemas** de BD existentes en un mismo entorno SQL. Contiene un esquema especial, INFORMATION_SCHEMA, que almacena metadatos: la definición de todos los elementos de todos los esquemas existentes en el catálogo (en oracle Data Dictionary).

Puede definirse restricciones de integridad referencial entre tablas que existan en esquemas dentro del mismo catálogo.