BASE DE DATOS

1° AÑO

Clase N.º 5: Lenguaje SQL -Consultas-manipulación avanzadas de base de datos-Vistas en base de datos

Contenido: consultas avanzadas de básicas de datos, introducción a vistas de base de datos y manipulación de base de datos.

1. Presentación:

Bienvenidos a la clase N° 5 del espacio BASE DE DATOS

En esta clase sobre bases de datos debe ser, sin duda, Las habilidades y funciones de un administrador de bases de datos avanzados y Los pasos en el diseño de las primeras vistas.

Te invitamos a visitar el siguiente recurso para introducirnos en el tema

https://sfpdesarrolloweb.wordpress.com/2020/03/24/ud-34-consultas-sql-avanzadas/

2.Desarrollo y Actividades:

Diseño físico en SQL

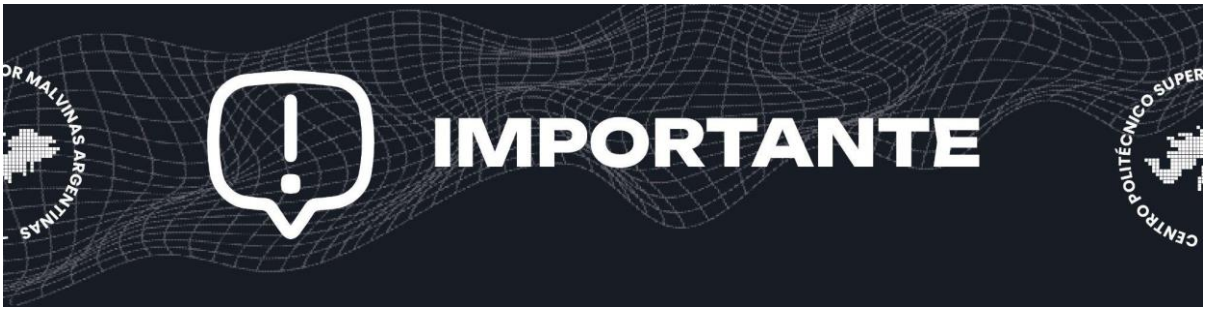
Introducción y objetivos

El diseño físico es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria, a partir del esquema lógico obtenido en la etapa anterior. Para especificar dicha implementación se debe determinar las estructuras de almacenamiento y

escoger los mecanismos necesarios para garantizar un acceso eficiente a los datos. Puesto que el esquema lógico utiliza el modelo relacional, la implementación del diseño físico se realizará en SQL.

Al finalizar este capítulo, el estudiante debe ser capaz de:

- Traducir el esquema lógico de una base de datos dada a un conjunto de sentencias SQL de creación de tablas que la implementen fielmente.
- Acudir a los manuales del SGBD escogido para la implementación y obtener en ellos toda la información necesaria para llevar a cabo la implementación sobre el mismo (sintaxis del lenguaje, los tipos de datos, etc.).
- Escoger las organizaciones de ficheros más apropiadas en función de las que tenga disponible el SGBD que se vaya a utilizar.
- Decidir qué índices deben crearse con el objetivo de aumentar las prestaciones en el acceso a los datos.
- Diseñar las vistas necesarias para proporcionar seguridad y facilitar el manejo de la base de datos.

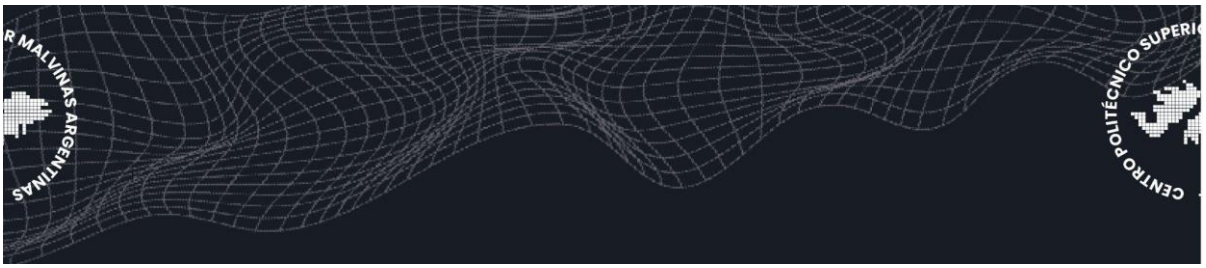


Metodología de diseño

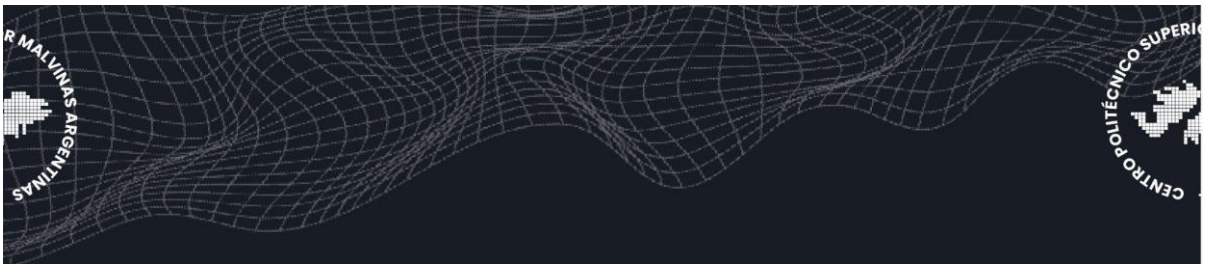
Mientras que en el diseño lógico se especifica qué se guarda, en el diseño físico se especifica cómo se guarda.

Para llevar a cabo esta etapa se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él. El diseñador debe conocer muy bien toda la funcionalidad del SGBD concreto y también el sistema informático sobre el que éste va a trabajar.

El diseño físico no es una etapa aislada, ya que algunas decisiones que se tomen durante su desarrollo, por ejemplo para mejorar las prestaciones, pueden provocar una reestructuración del esquema lógico. De este modo, entre el diseño físico y el diseño lógico hay una realimentación.



El propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior.
Concretamente, en el modelo relacional, esto consiste en:



Obtener un conjunto de sentencias para crear las tablas de la base de datos y para mantener las restricciones que se deben cumplir sobre ellas.

Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.

Diseñar el modelo de seguridad del sistema.

En los siguientes apartados se detallan cada una de las etapas que componen la fase del diseño físico.

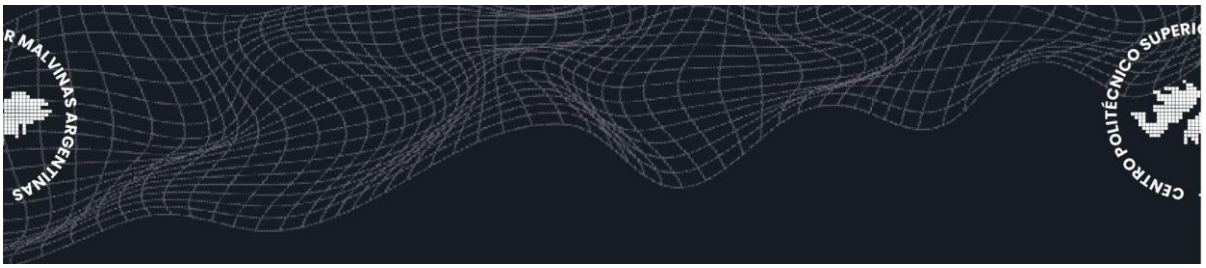
Traducir el esquema lógico

La primera fase del diseño físico consiste en traducir el esquema lógico a un esquema (físico) que se pueda implementar en el SGBD escogido. Para ello, es necesario conocer toda la funcionalidad que éste ofrece.

Sentencias de creación de las tablas

Las tablas se definen mediante el lenguaje de definición de datos del SGBD.

- El *nombre*. Es conveniente adoptar unas reglas para nombrar las tablas, de manera que aporten información sobre el tipo de contenido. Por ejemplo, a las tablas de referencia se les puede añadir el prefijo o el sufijo REF, a las tablas que almacenan información de auditoría ponerles el prefi- jo/sufijo AUDIT, a las tablas que sean de uso para un solo departamento, ponerles como prefijo/sufijo las siglas del mismo, etc.
- La *lista de columnas* con sus nombres. De nuevo resulta conveniente adoptar una serie de reglas para nombrarlas. Algunas reglas habituales son: poner el sufijo PK a las claves primarias (PRIMARY KEY), poner el sufijo FK a las claves ajenas (FOREIGN KEY), usar el nombre de la clave primaria a la que se apunta en el nombre de una clave ajena o el nombre de su tabla, usar el mismo nombre para las columnas que almacenan el mismo tipo de información (por ejemplo, si en varias tablas se guarda una columna con la fecha en que se ha insertado cada fila, usar en todas ellas el mismo nombre para dicha columna), etc. Además, para cada columna se debe especificar:



- Su *dominio*: tipo de datos, longitud y restricciones de dominio (se especifican con la cláusula CHECK).
- El *valor por defecto*, que es opcional (DEFAULT).
- Si admite *nulos* o no (NULL/NOT NULL).

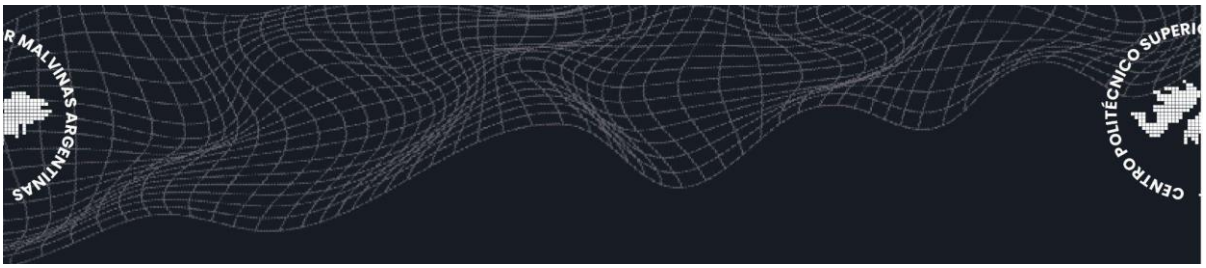
La *clave primaria* (PRIMARY KEY), las *claves alternativas* (UNIQUE) y las

claves ajenas (FOREIGN KEY), si las tiene.

Las *reglas de comportamiento* de las claves ajenas (ON UPDATE, ON DELETE).

A continuación, se muestra un ejemplo de la creación de las tablas FACTURAS y LINEAS_FAC (con las que se trabaja en el capítulo anteriores utilizando la especifi- cación de SQL del SGBD libre PostgreSQL.

```
CREATE TABLE facturas (
    codfac NUMERIC(6,0) NOT NULL, fecha DATE
                                                NO
    T NULL,
    codcli NUMERIC(5,0),
    codven NUMERIC(5,0),
    iva NUMERIC(2,0),
    dto     NUMERIC(2,0),
    CONSTRAINT cp_facturas PRIMARY KEY (codfac), CONSTRAINT
    ca_fac_cli FOREIGN KEY (codcli)
            REFERENCES clientes(codcli)
            ON UPDATE CASCADE ON DELETE SET  NULL,
    CONSTRAINT ca_fac_ven FOREIGN
            KEY (codven) REFERENCES
            vendedores(codven)
            ON UPDATE CASCADE ON DELETE SET  NULL,
    CONSTRAINT ri_dto_fac CHECK (dto BETWEEN 0 AND 50)
);
```




```
CREATE TABLE lineas_fac (
        codfac NUMERIC(6,0) NOT NULL, linea
                NUMERIC(2,0) NOT NULL,
        cant    NUMERIC(5,0)
        NOT NULL, codart VARCHAR(8) NOT
        NULL, precio NUMERIC(6,2) NOT
        NULL, dto
                NUMERIC(2,0),
        CONSTRAINT cp_lineas_fac PRIMARY KEY (codfac, linea),
        CONSTRAINT ca_lin_fac FOREIGN KEY (codfac)
                REFERENCES facturas(codfac)
                ON UPDATE CASCADE ON DELETE CASCADE,
        CONSTRAINT ca_lin_art FOREIGN KEY
                (codart) REFERENCES articulos(codart)
                ON UPDATE CASCADE ON DELETE RESTRICT,
        CONSTRAINT ri_dto_lin CHECK (dto BETWEEN 0 AND 50)
    );
```

Mantenimiento de restricciones y reglas de negocio

Las actualizaciones que se realizan sobre las tablas de la base de datos deben observar ciertas restricciones o producir determinadas consecuencias que imponen las reglas de funcionamiento de la empresa. Algunos SGBD proporcionan mecanismos que permiten definir restricciones y reglas, y vigilan su cumplimiento.
Un mecanismo para definir restricciones es la cláusula CONSTRAINT ...

CHECK. Un ejemplo de ella se puede observar en las sentencias de creación de las tablas FACTURAS y LINEAS_FAC, sobre la columna dto. Otro mecanismo son los *disparadores* (TRIGGER), que también se utilizan para establecer reglas de negocio en las que se requiere la realización de alguna acción como consecuencia de algún evento.

Diseñar la representación física

Uno de los objetivos principales del diseño físico es almacenar los datos de modo eficiente. Para medir la eficiencia hay varios factores que se debe tener en cuenta:

- *Rendimiento de transacciones.* Es el número de transacciones que se quiere procesar en un intervalo de tiempo.

- *Tiempo de respuesta.* Es el tiempo que tarda en ejecutarse una transacción. Desde el punto de vista del usuario, este tiempo debería ser el mínimo posible.

- *Espacio en disco.* Es la cantidad de espacio en disco que hace falta para los ficheros de la base de datos. Normalmente, el diseñador querrá minimizar este espacio.

Lo que suele suceder es que todos estos factores no se pueden satisfacer a la vez. Por ejemplo, para conseguir un tiempo de respuesta mínimo puede ser necesario aumentar la cantidad de datos almacenados, ocupando más espacio en disco. Por lo tanto, el diseñador deberá ir ajustando estos factores para conseguir un equilibrio razonable.

Para mejorar las prestaciones, el diseñador del esquema físico debe saber cómo interactúan los dispositivos involucrados y cómo esto afecta a las prestaciones:

- *Memoria principal.* Los accesos a memoria principal son mucho más rápidos que los accesos a memoria secundaria (decenas o centenas de miles de veces más rápidos). Generalmente, cuanta más memoria principal se tenga, más rápidas serán las aplicaciones. Si no hay bastante memoria disponible para todos los procesos, el sistema operativo debe transferir páginas a disco para liberar memoria (memoria virtual). Cuando estas páginas se vuelven a necesitar, hay que volver a traerlas desde el disco(fallos de página). A veces, es necesario llevar procesos enteros a disco (*swapping*) para liberar memoria. El hacer estas transferencias con demasiada frecuencia empeora las prestaciones.

- *CPU.* La CPU controla los recursos del sistema y ejecuta los procesos de usuario. El principal objetivo con este dispositivo es lograr que no haya bloqueos de procesos para conseguirla. Si el sistema operativo, o los procesos de los usuarios, hacen muchas demandas de CPU, ésta se convierte en un cuello de botella. Esto suele ocurrir cuando hay muchas

faltas de página o se realiza mucho *swapping*.

Entrada/salida a disco. Los discos tienen una velocidad de entrada/salida. Cuando se requieren datos a una velocidad mayor que ésta, el disco se convierte en un cuello de botella. Dependiendo de cómo se organicen los datos en el disco, se conseguirá reducir la probabilidad de empeorar las prestaciones. Los principios básicos que se deberían seguir para repartir los datos en los discos son los siguientes:

- Los ficheros del sistema operativo deben estar separados de los ficheros de la base de datos.
- Los ficheros de datos deben estar separados de los ficheros de índices.
- Los ficheros con los diarios de operaciones deben estar separados del resto de los ficheros de la base de datos.

Red. La red se convierte en un cuello de botella cuando tiene mucho tráfico y cuando hay muchas colisiones.

Cada uno de estos recursos afecta a los demás, de modo que una mejora en alguno de ellos puede influir en otros.

Escoger los índices a crear y sus tipos

Los índices son estructuras adicionales que se utilizan para acelerar el acceso a las tablas en respuesta a ciertas condiciones de búsqueda.
Algunos tipos de índices, los denominados caminos de acceso secundario, no afectan al emplazamiento físico de los datos en el disco y lo que hacen es proporcionar caminos de acceso alternativos para encontrar los datos de modo eficiente basándose en los campos de indexación. Hay que tener en cuenta que los índices conllevan un coste de mantenimiento que es necesario sopesar frente a la ganancia en prestaciones.
Cada SGBD proporcionará uno o varios tipos de índices entre los que escoger. Los más habituales son los índices basados en árboles B+ (o árboles B*) y los basados en la dispersión (*hash*).
Un índice con estructura de árbol B+ es un árbol de búsqueda que siempre está equilibrado (todas las hojas se encuentran al mismo nivel) y en el que el espacio desperdiciado por la eliminación, si lo hay, nunca será excesivo. Los algoritmos para insertar y eliminar son complejos para

poder mantener estas restricciones. No obstante, la mayor parte de las inserciones y eliminaciones son procesos simples que se complican sólo en circunstancias especiales: cuando se intenta insertar en un nodo que está lleno o cuando se intenta borrar en un nodo que está ocupado hasta la mitad. Las simulaciones muestran que un índice con estructura de árbol B+ de cuatro niveles contiene unos cien millones de nodos hoja, lo que indica que en cuatro accesos se puede llegar a los datos, incluso si la tabla es muy grande.

Un índice basado en la dispersión es un fichero disperso en el que las entradas se insertan en el índice aplicando una función sobre el campo de indexación.

Aunque el acceso a los datos es muy rápido (es casi un acceso directo), este tipo de índices sólo pueden usarse cuando la condición de búsqueda es la igualdad sobre el campo de indexación.

A la hora de seleccionar los índices a crear, se pueden seguir las siguientes indicaciones:

- Crear un índice sobre la clave primaria de cada tabla.

  La mayor parte de los SGBD relacionales crean un índice único de manera automática sobre la clave primaria de cada tabla porque es el mecanismo que utilizan para mantener la unicidad.

- No crear índices sobre tablas pequeñas. Si el SGBD ha creado índices automáticamente sobre este tipo de tablas, se pueden eliminar (DROP INDEX).

  Aquí conviene tener en cuenta que, en la mayor parte de los SGBD, no se permite eliminar un índice creado sobre una clave primaria a la que apunta una clave ajena, ya que este índice se utiliza para mantener la integridad referencial.

- Crear un índice sobre las claves ajenas que se utilicen con frecuencia en operaciones de JOIN.

- Crear un índice sobre los atributos que se utilizan con frecuencia para hacer restricciones WHERE (son condiciones de búsqueda).

- Crear un índice único sobre las claves alternativas que se

- utilizan para hacer búsquedas.

- Al igual que ocurre con las claves primarias, los SGBD suelen mantener la unicidad de las claves alternativas mediante un índice único que crean automáticamente.

- Evitar los índices sobre atributos que se modifican a menudo.

- Evitar los índices sobre atributos poco selectivos: aquellos en los que la consulta selecciona una porción significativa de la tabla (más del 15 % de las filas).

- Evitar los índices sobre atributos formados por tiras de caracteres largas.

- Evitar los índices sobre tablas que se actualizan mucho y que se consultan muy esporádicamente (tablas de auditoría o diarios). Si se han creado índices sobre este tipo de tablas, podría ser aconsejable eliminarlos.

- Revisar si hay índices redundantes o que se solapan y eliminar los que no sean necesarios.

Diseñar las vistas de los usuarios

El objetivo de este paso es diseñar las vistas o esquemas externos de los usuarios, correspondientes a los esquemas lógicos de cada grupo de usuarios. Cada esquema externo estará formado por tablas y vistas (VIEW) de SQL. Las vistas, además de preservar la seguridad, mejoran la independencia de datos, reducen la complejidad y permiten que los usuarios vean los datos en el formato deseado.

Diseñar las reglas de acceso

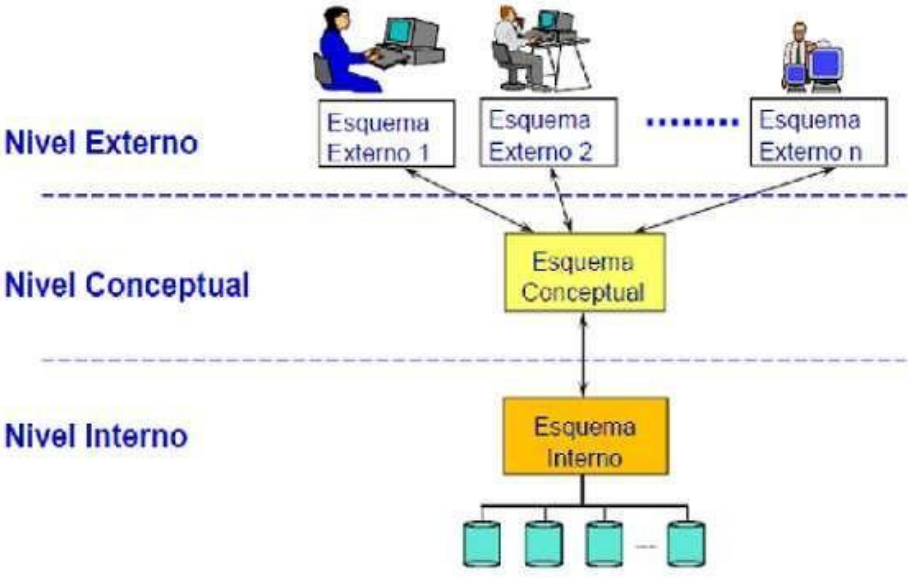
El administrador de la base de datos asigna a cada usuario un identificador que tendrá una contraseña asociada por motivos de seguridad. Para cada usuario o grupo de usuarios se otorgarán privilegios para realizar determinadas acciones sobre determinados objetos de la base de datos. Por ejemplo, los usuarios de un determinado grupo pueden tener permiso para consultar los datos de una tabla concreta, y no tener permiso para actualizarlos.

Monitorizar y afinar el sistema

Una vez implementado el esquema físico de la base de datos, ésta se debe poner en marcha para observar sus prestaciones. Si éstas no son las deseadas, el esquema deberá cambiar para intentar satisfacerlas. Una vez afinado el esquema, éste no permanecerá estático, ya que tendrá que ir cambiando conforme lo requieran los nuevos requisitos de los usuarios.
Los SGBD proporcionan herramientas para monitorizar el sistema mientras está en funcionamiento.

Vistas

Hay tres características importantes inherentes a los sistemas de bases de datos: la separación entre los programas de aplicación y los datos, el manejo de múltiples vistas por parte de los usuarios (esquemas externos) y el uso de un catálogo o diccionario para almacenar el esquema de la base de datos. En 1975, el comité ANSI-SPARC (*American National Standard Institute- Stan- dards Planning and Requirements Committee*) propuso una arquitectura de tres niveles para los sistemas de bases de datos, que resulta muy útil a la hora de conseguir estas tres características.



Arquitectura ANSI-SPARC para los Sistemas de Bases de Datos

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física. En esta arquitectura, el

esquema de una base de datos se define en tres niveles de abstracción distintos (ver figura):

1. En el *nivel interno* se describe la estructura física de la base de datos mediante un *esquema interno*. Este esquema se especifica mediante un modelo físico y describe todos los detalles para el almacenamiento de la base de datos, así como los métodos de acceso.

2. En el *nivel conceptual* se describe la estructura de toda la base de datos para una comunidad de usuarios (todos los de una empresa u organización), mediante un *esquema conceptual*. Este esquema oculta los detalles de las estructuras de almacenamiento y se concentra en describir entidades, atributos, relaciones, operaciones de los usuarios y restricciones. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar el esquema.

3. En el *nivel externo* se describen varios *esquemas externos* o *vistas de usuario*. Cada esquema externo describe la parte de la base de datos que interesa a un grupo de usuarios determinado y oculta a ese grupo el resto de la base de datos. En este nivel se puede utilizar un modelo conceptual o un modelo lógico para especificar los esquemas.

La mayoría de los SGBD no distinguen del todo los tres niveles. Algunos incluyen detalles del nivel físico en el esquema conceptual. En casi todos los SGBD que se manejan vistas de usuario, los esquemas externos se especifican con el mismo modelo de datos que describe la información a nivel conceptual, aunque en algunos se pueden utilizar diferentes modelos de datos en los niveles conceptual y externo.

Hay que destacar que los tres esquemas no son más que descripciones de los mismos datos pero con distintos niveles de abstracción. Los únicos datos que existen realmente están a nivel físico, almacenados en un dispositivo como puede ser un disco. En un SGBD basado en la arquitectura de tres niveles, cada grupo de usuarios hace referencia exclusivamente a su propio esquema externo.

La arquitectura de tres niveles es útil para explicar el concepto de *independencia de datos*, que se puede definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior. Se pueden definir dos tipos de independencia de datos:

La *independencia lógica* es la capacidad de modificar el esquema conceptual sin tener que alterar los esquemas externos ni los programas de aplicación. Se puede modificar el esquema conceptual para ampliar la base de datos o para reducirla. Si, por ejemplo, se reduce la base de datos eliminando una entidad, los esquemas externos que no se refieran a ella no deberán verse afectados.

La *independencia física* es la capacidad de modificar el esquema interno sin tener que alterar el esquema conceptual (o los externos). Por ejemplo, puede ser necesario reorganizar ciertos ficheros físicos con el fin de mejorar el rendimiento de las operaciones de consulta o de actualización de datos. Dado que la independencia física se refiere sólo a la separación entre las aplicaciones y las estructuras físicas de almacenamiento, es más fácil de conseguir que la independencia lógica.

Cada esquema externo estará formado por un conjunto de tablas (TABLE) y un conjunto de vistas (VIEW). En la arquitectura de tres niveles estudiada, se describe una vista externa como la estructura de la base de datos tal y como la ve un usuario en particular. En el modelo relacional, el término vista tiene un significado un tanto diferente. En lugar de ser todo el esquema externo de un usuario, una vista es una *tabla virtual*, una tabla que en realidad no existe como tal.

Una vista es el resultado dinámico de una o varias operaciones relacionales realizadas sobre las tablas. La vista es una tabla virtual que se produce cuando un usuario la consulta. Al usuario le parece que la vista es una tabla que existe y la puede manipular como si se tratara de una tabla, pero la vista no está almacenada físicamente. El contenido de una vista está definido como una consulta sobre una o varias tablas.

En SQL, la sentencia que permite definir una vista es la siguiente:

```
CREATE VIEW nombre_vista [  ( nombre_col, ... ) ]
        AS sentencia_SELECT
        [ WITH CHECK OPTION ];
```

Las columnas de la vista se pueden nombrar especificando la lista entre paréntesis. Si no se especifican nuevos nombres, los nombres son los mismos que los de las columnas de las tablas especificadas en la sentencia SELECT.

La opción WITH CHECK OPTION impide que se realicen inserciones y actúa

lizaciones sobre la vista que no cumplan las restricciones especificadas en la misma. Por ejemplo, si se crea una vista que selecciona los clientes con códigos postales de la provincia de Castellón (aquellos que empiezan por 12) y se especifica esta cláusula, el sistema no permitirá actualizaciones de códigos postales de clientes de esta provicia si los nuevos códigos postales son de una provincia diferente. Del mismo modo, a través de la vista sólo será posible insertar clientes con códigos postales de Castellón. Es como si se hubiera establecido una restricción de tipo CHECK con el predicado del WHERE de la definición de la vista.

Cualquier operación que se realice sobre la vista se traduce automáticamente a operaciones sobre las tablas de las que se deriva. Las vistas son dinámicas porque los cambios que se realizan sobre las tablas que afectan a una vista se reflejan inmediatamente sobre ella. Cuando un usuario realiza un cambio sobre la vista (no todo tipo de cambios están permitidos), este cambio se realiza sobre las tablas de las que se deriva. Las vistas son útiles por varias razones:

- Proporcionan un poderoso mecanismo de seguridad, ocultando partes de la base de datos a ciertos usuarios. El usuario no sabrá que existen aquellos atributos que se han omitido al definir una vista.

- Permiten que los usuarios accedan a los datos en el formato que ellos desean o necesitan, de modo que los mismos datos pueden ser vistos con formatos distintos por distintos usuarios.

CREATE VIEW domicilios (codcli,nombre,direccion,poblacion) AS SELECT c.codcli,c.nombre,c.direccion,c.codpostal || ' - '
|| pu.nombre || ' (' || pr.nombre || ')' FROM clientes c JOIN pueblos pu USING(codpue)
JOIN provincias pr USING(codpro); SELECT * FROM domicilios;

codcli	nombre	direccion	poblacion
210	Luis	C/Pez, 3	12540 - Vila-real (Castellón)

Se pueden simplificar operaciones sobre las tablas que son

complejas. Por ejemplo, se puede definir una vista que muestre cada vendedor con el nombre de su jefe:

```
CREATE VIEW vj ( codven, nombreven, codjefe,
    nombrejefe ) SELECT v.codven, v.nombre,
    j.codven, j.nombre
    FROM vendedores v LEFT OUTER JOIN
            vendedores j ON
            (v.codjefe=j.codven);
```

El usuario puede hacer restricciones y proyecciones sobre la vista, que el SGBD traducirá en las operaciones equivalentes sobre el JOIN.

```
SELECT f.codfac, f.fecha, vj.vendedor,
vj.jefe FROM facturas f JOIN vj
USING(codven) WHERE... ;
```

-

Las vistas proporcionan independencia de datos a nivel lógico, que también se da cuando se reorganiza el nivel conceptual. Si se añade un atributo a una tabla, los usuarios no se percatan de su existencia si sus vistas no lo incluyen. Si una tabla existente se reorganiza o se divide en varias tablas, se pueden crear vistas para que los usuarios la sigan viendo como al principio.

Las vistas permiten que se disponga de información expresada en forma de reglas generales de conocimiento relativas al funcionamiento de la organización. Una de estas reglas puede ser «los artículos en oferta son los que tienen descuento» y se puede definir una vista que contenga sólo estos artículos, aunque ninguna columna de la base de datos indique cómo ha de considerarse cada artículo (es el conocimiento).

```
CREATE VIEW articulos_oferta AS

SELECT * FROM          articulos

WHERE dto > 0;
```

Cuando se actualiza una tabla, el cambio se refleja automáticamente en todas las vistas que la referencian. Del mismo modo, si se actualiza una vista, las tablas de las que se deriva deberían reflejar el cambio. Sin embargo, hay algunas restricciones respecto a los tipos de modificaciones que se pueden realizar sobre las vistas. En el estándar de SQL se definen

las condiciones bajo las que una vista es actualizable o es insertable. Básicamente, una vista es actualizable si se puede identificar de modo único la fila a la que afecta la actualización.

- Una vista definida sobre varias tablas es actualizable si contiene las claves primarias de todas ellas y los atributos que no aceptan nulos.
- Una columna de una vista definida sobre varias tablas se podrá actualizar si se obtiene directamente de una sola de las columnas de alguna de las tablas y si la clave primaria de dicha tabla está incluida en la vista.
- Las vistas definidas con operaciones de conjuntos pueden ser actualizables, pero no son insertables (no se puede determinar en qué tabla hacer la inserción).
- Ya que el estándar permite que sean actualizables un conjunto muy restringido de vistas, en ocasiones será necesario hacer que una vista sea actualizable mediante disparadores o reglas del tipo *en lugar de*.

Algunos conceptos

1.-¿ Qué es un Diseño físico en SQL?

2.-¿Qué es Metodología de diseño?

3.- ¿Qué son los índices , cómo se crean y sus tipos?

5.-¿Que es una vista en base de datos?

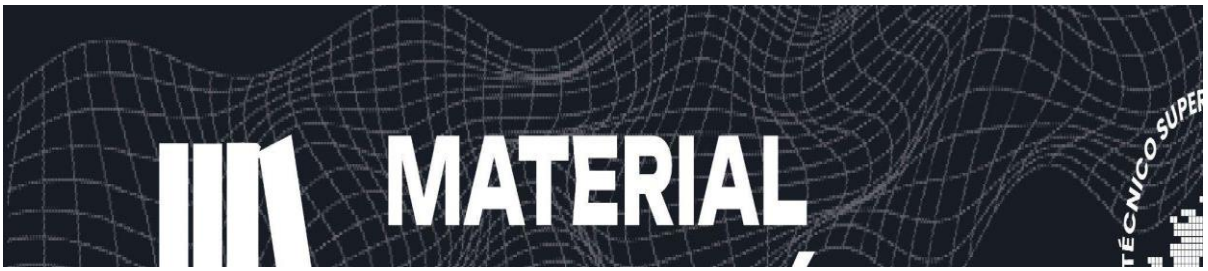
6.-¿Para qué sirve la vista de Base de datos?

ATENCIÓN: *Estas actividades deben contestarlas en el Foro de la plataforma Clase 5.-*

<u>2. Actividad Integradora de Cierre</u> (utilizar la base de datos de la clase Nº 4 anterior para sal siguientes consultas SQL).

Consultas

1. Contar el número de usuarios por marca de teléfono
2. Listar nombre y teléfono de los usuarios con teléfono que no sea de la marca LG
3. Listar las diferentes compañias en orden alfabético ascendentemente
4. Calcular la suma de los saldos de los usuarios de la compañia telefónica UNEFON
5. Mostrar el email de los usuarios que usan hotmail
6. Listar los nombres de los usuarios sin saldo o inactivos
7. Listar el login y teléfono de los usuarios con compañia telefónicaIUSACELL o TELCEL
8. Listar las diferentes marcas de celular en orden alfabético ascendentemente
9. Listar las diferentes marcas de celular en orden alfabético aleatorio
10. Listar el login y teléfono de los usuarios con compañia telefónica IUSACELL o UNEFON
11. Listar nombre y teléfono de los usuarios con teléfono que no sea de la marca MOTOROLA o NOKIA
12. Calcular la suma de los saldos de los usuarios de la compañia telefónica TELCEL



Cierre:

Tomamos un momento para repensar lo que hemos aprendido, En esta clase se aprendió sobre bases de datos como debe ser, sin duda, En general, al aprender sobre consultas-manipulación avanzadas de base de datos y vistas en base de datos, se pueden adquirir habilidades valiosas que permiten trabajar con grandes conjuntos de datos de manera eficiente y efectiva.

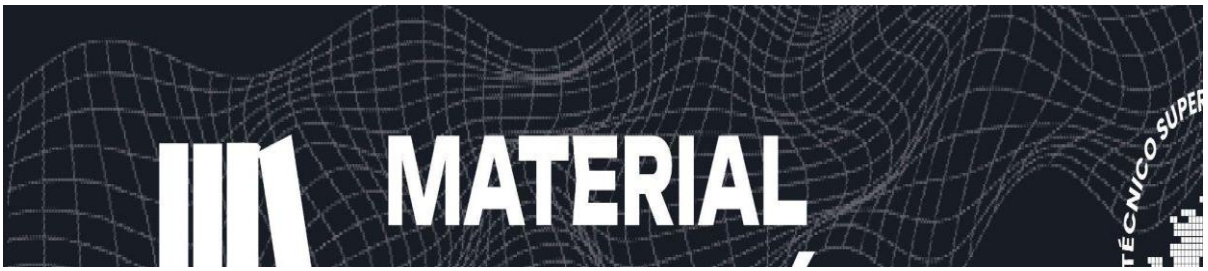
Sitio

(CONSULTA AVANZADAS MULTITABLAS)

- https://www.youtube.com/watch?v=IB8RxT0kCg8&ab_channel=UniversitatPolit%C3%A8cnicadeVal%C3%A8ncia-UPV

Ejercicios resueltos SQL

- https://www.studocu.com/es-ar/document/universidad-nacional-jose-faustino- sanchez-carrion/diseno-del-proyecto-de-investigacion/ejercicios-resueltos- sql/78209



- Marquez, M. (2011) : Base de datos. Editorial Universitat Jaume.
- Catheren M. R. (2009): Base de Datos. Edición McGraw Hill