**UF1845: Módulo Bases de Datos**

**15. Bases de datos**

Niveles de abstracción

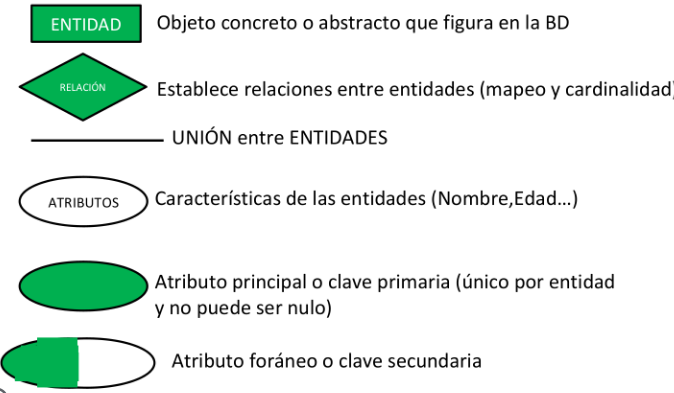
Simplifica usabilidad de usuarios con Bases de datos (BBDD) y Sistema.

* Nivel físico: Dispositivos de almacenamiento Local o remoto. Nivel más bajo de abstracción. Dónde tenemos nuestra BD.
* Nivel conceptual: Definición + Relación de DATOS: Interno, como está construido.
* Nivel visión: Usuario visualiza BD en el sistema. Nivel más alto de abstracción. Diferentes vistas

Diagrama/modelo Entidad – Relación

La clave primaria es un identificador único. Solo lo tiene ese valor esa clave primaria.

Elementos:



Clave foránea: Una que relaciona con otras tablas.

Cuando solo es uno se pone un 1:

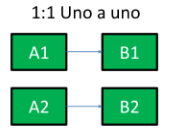
Cuando es muchos se pone una n:

Quien obtiene la clave foránea es quien tiene muchos.

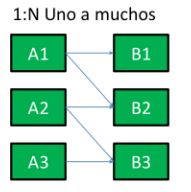
Diagrama Entidad – Relación: Grado de cardinalidad

Unión de dos cardinalidades de dos entidades.

1:1. Uno a uno. Una entidad, solo puede tener otra entidad.



1:N Uno a muchos: Una entidad puede tener muchas entidades.



N:M Mucho a muchos

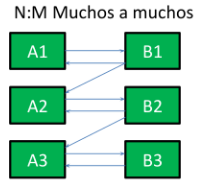


Diagrama Entidad – Relación



* **Ventajas**: Diseño de alto nivel, ya que refleja con bastante precisión el esquema conceptual. Los diagramas ER permiten mantener una visión global del diseño y favorecer la comunicación entre diseñadores.
* **Desventajas:** Falta de soporte formar, ya que los SGBD no lo implantan directamente. Casi siempre hay que transformarlo en un modelo de más bajo nivel, esto significa, más cercano a la comprensión del sistema – máquina:
  + **Un nivel intermedio:** El lenguaje SQL
  + **Un nivel alto:** el usuario lo puede entender más fácilmente pero la máquina que computa el código, no tanto.
  + **Un nivel bajo:** más cercano a la comprensión del sistema – máquina. Entiende bytes pero no los textos.

Unified Modeling Language

Es un modelo internacional, que ayudaba al diseño e implementación de sistemas de software complejos. Cuando quieres hacer un flujo de bases de datos (como se hace un préstamo por parte de un usuario con entidades, Estudiantes-preéstaos-libros). Se crea para los diagramas que definen la estructura y el comportamiento.

El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. UML tiene aplicaciones más allá del desarrollo de software, p. ej., en el flujo de procesos en la fabricación.

Los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene.

UML no es un lenguaje de programación, pero existen herramientas que se pueden usar para generar código en diversos lenguajes usando los diagramas UML.

Vamos a establecer información relacionada, se acerca a la base de datos, pero no va hacer que yo pueda insertar el préstamo de un libro en ningún sitio concreto. El UML no es un lenguaje de base de datos, pero es más avanzado que el Modelo Entidad-Relación.

\*Cuando hacemos el modelo Entidad Relación ponemos atributos, pero hemos establecido la relación que tenga la forma en la que un libro es leído por un alumno. Que sepas la parte del libro que ha sido leído.

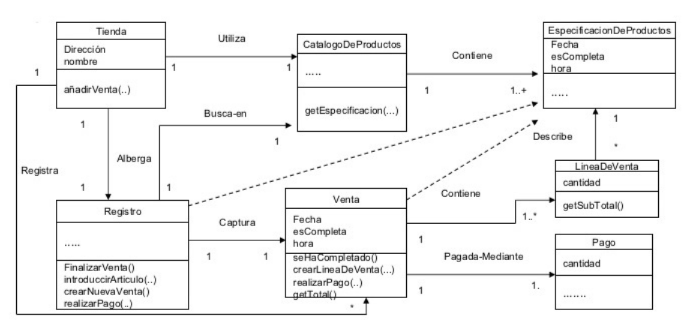
UML: Diagrama de casos de usos

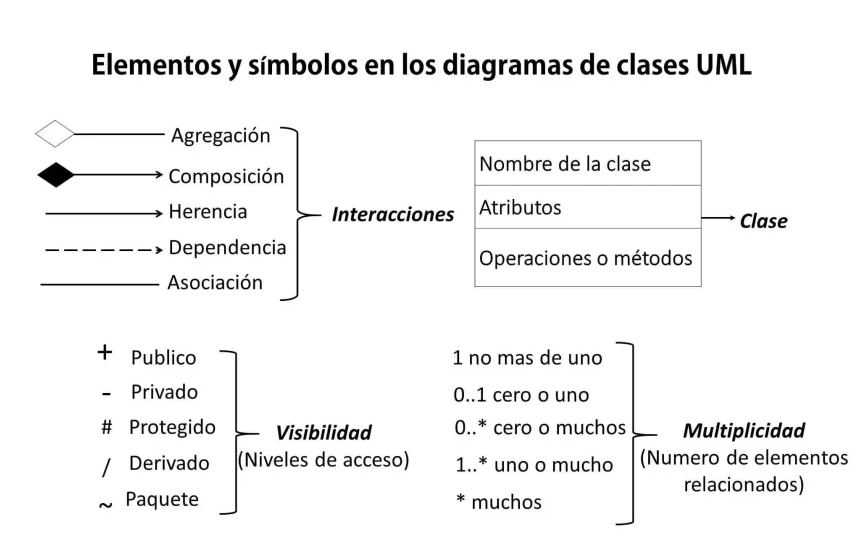
Representa gráficamente los casos de usos, interacción con el sistema.



UML: Diagrama de clases

Muestra un conjunto de clases interfaces y sus relaciones

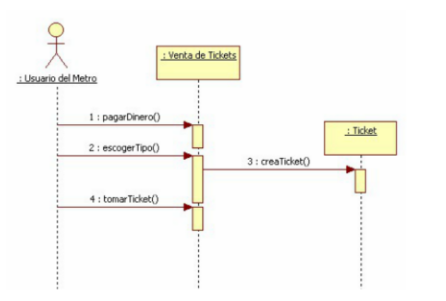


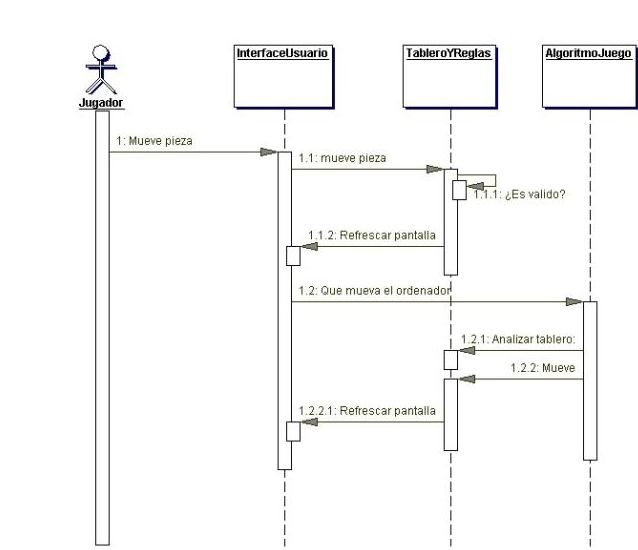


UML: Diagrama de secuencia

Muestra la interacción de los objetos que componen un sistema de forma **temporal**.

Representa gráficamente los casos de usos, interacción con el sistema. Es un diagrama de alto nivel.





EJERCICIO: DIagram ade secuencia de un juego “Buscaminas”.

**15.2 Concepto y origen de las Bases de Datos (DDBB) y de los Sistemas Gestores de Bases de Datos (DBMS). Evolución de la Base De Datos (BD).**

Si trataremos los datos solo con ficheros, no tendríamos información relacionada.

\* En el mundo actual es fundamental el procesamiento de datos para obtener información que permita tomar decisiones, cada vez se genera y se utilizan más datos, lo cual dificulta su procesamiento.

Etapas de evolución del tratamiento de información:

* 1ª etapa, principios de lo 1960, primeras aplicaciones integran datos en programas

en forma de CONSTANTES, se analizan partes del mundo científico

* 2ª etapa, mediados 60s, aparecen los FICHEROS que representan partes del mundo

real, solo se accedían de forma secuencial y poco después de forma aleatoria.

* 3ª etapa, finales 60s, para evitar problemas en ficheros aparecen las BASES DE
* DATOS, que pretenden modelar grandes partes del mundo real mediante la relación de varios archivos, evitando la redundancia e incoherencias.

Una base de datos es “una colección de datos que están relacionados entre sí, que tienen una definición y una descripción comunes y que están estructurados de forma particular”

Los elementos que forman las bases de datos.

1. **Los datos**: representación de hechos, conceptos o instrucciones, estructurados, para comunicar, interpretar y representar datos de informaciones almacenados y relacionados entre si:
   * Microdatos: datos simples, como por ejemplo nombres, direcciones, …
   * Macrodatos: aglomeración de datos, como las tablas.
2. **El diccionario de datos (DD):** conjunto de metadatos que contiene datos sobre los datos, que da lugar a una metabase, una BD que describe otra BD
   * Esquemas externos, conceptuales e internos
   * Descripciones de registros, campos y relaciones a cada nivel
   * Procedimientos de autorización de accesos y validación de datos
   * Correspondencias externas/conceptual y conceptual/interna
   * Referencias cruzadas
     + Un programa qué datos utiliza
     + Unos datos y qué programas lo utilizan o lo modifican

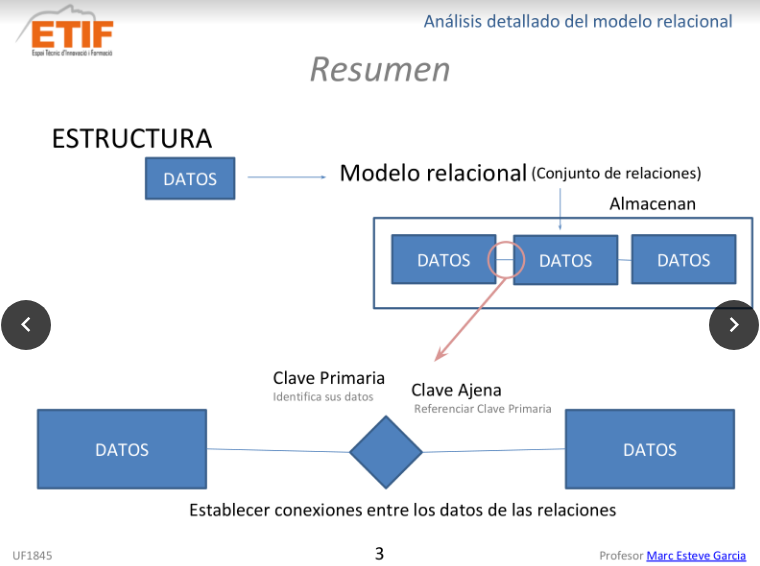
//Es lo que describe la base de datos. Sus esquemas internos. Todos los campos y relaciones describen la base de datos.

1. **El sistema de gestión de la base de datos** (SGBD): Software o conjunto de programas que gestionan las BD.Ell software encargado de administrar y elaborar bases de datos.
   * Funciones: Crear, actualizar, consultar, proteger las BBDD
2. **Los usuarios: todas las personas que utilizan las BD**
   * Informáticos (especializados y comprenden la parte interna)
     + Administradores (ABD): gestionan la base de datos y proporcionan herramientas para el resto de usuarios
     + Desarrolladores: complementan a los administradores, analizan y programan aplicaciones para usuarios finales.
   * No informáticos: serán los tipos de usuarios a los que va dirigida la Base de Datos, utilizarán datos para el trabajo cotidiano y accederán los datos según sus permisos para facilitar el acceso y manipulación

Una cosa es la biblioteca de datos y otra cosa es la definición de estas bases de datos.

**5.3 Análisis detallado del modelo relacional**

3.1 Estructura de los datos. Estructura.



**Lenguaje de consulta SQL:** es un lenguaje declarativo de acceso a las bases de datos relacionales, que permite definir y manipular las bases de datos.

**El modelo relacional** es un modelod e datos, y, como tal, hay que tener en cuenta los aspectos de los datos:

* La estructura
* La manipulación
* La integridad

Las relaciones están formadas de:

* **Atributo:** Se trata de cada una de las columnas de la tabla. Vienen definidas por un nombre y pueden contener un conjunto de valores.
* **Tupla:** También llamado registro. Esa fila de valores. Se trata de cada una de las filas de la tabla, es importante señalar que no se pueden tener tuplas duplicadas en una tabla.

**\*Dominio**: conjunto de valores atómicos o indivisibles, es decir, que por muy largo que sea un valor, no tiene una estructuración interna para un sistema de gestión de base de datos relacional. Serán de una tipología determinada: (establece los valores que va a tener). Pueden ser de dos tipos:

* Dominio predefinido: Corresponde a los tipos de datos que normalmente proporcionan los lenguajes de base de datos. Como por ejemplo los enteros, las cadenas de texto, los reales, los numéricos, etc.
* Los dominios definidos por el usuario: que pueden ser más específicos, esta definición de dominio debe constar, como mínimo del nombre del dominio y de la descripción de los valores que la forman.

La relación se componen de:

* Esquema:
* Extensión:

Nosotros haremos relaciones entre tablas. Se relacionaran por sus atributos

**Hay una serie de características que las relaciones deben cumplir:**

* La atomicidad de los valores de los atributos:
* Las tuplas no se pueden ordenar:
* Los atributos no se pueden ordenar

**Hay diferentes claves que se pueden encontrar en las relaciones:**

* Las claves: conjuntos de atributos que identifica a una relación.
  + Super clave: Atributo o conjunto de atributos que identifica de forma única a una identidad. Es la capacidad de ser clave primaria en las diferentes relaciones.
    - La clave prima pase a convertirse en superclave ya que define directamente los atributos.
  + Clave candidata: “Todos aquellos atributos pertenecientes a una relación que pueden identificar inequívocamente a cada tupla”.
  + Clave foránea:

**Operadores del modelo**

* La actualización de datos: consiste en hacer que los cambios que se producen en la vida real, queden reflejados en las relaciones de la base de datos. Hay **tres operaciones básicas de actualización**:
  + Inserción: sirve para añadir una o ´mas tuplas a una relación.
  + Borrado: que sirve para eliminar una o más tuplas de una relación
  + Modificación: que sirve para cambiar alos valores que tienen una o más tuplas de unarelación para uno o más de sus atributos.
* El valor NULO “Es un atributo que no contiene nigun valor
* El **grado** de una relación “ Es el número de atributos que pertenecen a su esquema”.
* La **cardinalidad de una relación** \* Es el numero de tuplas que pertenecen a su extensión.
* La consulta de datos: consiste en la obtención de datos a parti d elas relaciones que contiene la base de datos.
  + Lenguajes basados en el álgebra relacional
  + Los lenguajes basados ene l cálculo relacional

**La integridad**

Es la propiedad de los datos de corresponder a representaciones creíbles del mundo real. Es necesario que se cumplan varias condiciones, en general, estas condiciones garantizan la integridad de los datos y pueden ser de dos tipos.

* Las restricciones de integridad del usuario. Condiciones específicas que se deben cumplir en una base de datos concreta.
* Las reglas de integridad de modelo.

Existen reglas que todo sistema gestor de base de datos (SGBD) relacional debe garantizar que se cumpla:

* **Regla de integridad de unicidad de la clave primaria:** Toda clave primaria que se elija para una relación no debe tener valores repetidos, es decir, la clave primaria o el conjunto de atirbutos que forman la clave priamria de una relación no puede tener en nignun momento dos tuplas cn la isma cominaciónd e valores para los atributos de clave primaria. El SGBD relacional deberá garantizar el ciumplimiento de esta regla de integridad en todas las inserciones y modificaciones que afecten a atirbutos que pertenecen a la clave primaria de la relación.
* **Regla de integridad de entidad de la clave primaria**: Establece que los atributos de la clave primaria de una relación no pueden tener vlaores nulos. La clave prima o el conjunto de atirbutos que formen la clave primaria de una relación, no peude tener ninguna tupla cna aglun valor nulo ar algún
* **Regla de la integridad referencial:** Esta regla determina que todo slos valores que toma una clave ajena deben ser valores nulos o valores que existen en la clave primaria a la que referencia.

Los dominios pueden ser de dos tipos, predefinidos o definidos por el usuario. El último nos resulta muy útil, ya que nos permite determinar de forma más específica cuáles serán los valores administrados por el atributo.

Hay una serie de reglas para mantener la integridad del dominio. Relacionada con al idea de dominio y establece dos condiciones. Regla de integridad del dominio:

* La primera condición: Consiste en que un valor no nulo de un atributo debe pertenecer al dominio del atributo.
* La segunda condición: Es más difícil, sobretodo en el caso de dominios definidos por el usuario y en los sistemas gestores de bases de datos (SGBD) actuales que no lo soportan para este tipo de dominios. Esta condición sirve para establecer que los operadores que pueden aplicarse sobre los valores dependen de los dominios de estos valores, es decir, un operador determinado solo se pude aplicar sobrevalores que tengan dominios que le sean adecuados.

Algebra relacional

El álgebra relacional se basa en al teoríad e conjuntos para especifciar consultas en una base de datos relacional. Es necesario efinir varios pasos que sirven para ir construyendo, mediante operaciones de álgebra relacional, una nueva relación que contenga los datos que responden a

Es el paso previo a un lenguaje SQL. Se pueden dividir **las operaciones de algebra lineal:**

**Según se pueden expresar o no en términos de otras operaciones:**

* **Operaciones primitivas:** aquellas operaciones a partir de las cuales podemos definir el resto.
* **Operaciones no primitivas:** el resto de las operaciones del álgebra relacional que no son estrictamente necesarias, porque se pueden expresar en términos de las primitivas, sin embargo, las operaciones no primitivas permite formular algunas consulta de forma más cómoda. Existen distintas versiones del álgebra relacional, según las operaciones no primitivas que se incluyen.

**\***Las que se utilizan con mayor frecuencia son la intersección y la combinación.

**Según el número de relaciones que tienen como operando:**

* **Operaciones binarias:** Son aquellas que tienen dos relaciones como operando. Son binarias **todas las operaciones, excepto la selección y la proyección.**
* **Operaciones unarias:** Son aquellas que tienen una sola relación como operando. Son operaciones unarias la selección y la proyección.

**Según se parecen o no a las operaciones de la teoría de conjuntos**

* **Operaciones conjuntistas**: aquellas que se parece a las de la teoría de conjuntos. Se trata de la **unión**, **la intersección, la diferencia y el producto cartesiano.**
* **Operaciones específicamente relacionales:** Son el resto de las operaciones, como la selección, la proyección y la combinación.

**\***Las operaciones del álgebra relacional obtienen como resultado una nueva relación.

Supongamos, que tenemos EMPLEADOS\_ADM UU EMPLEADOS-PROD para obtener la unión de las relaciones EMPLEADOS\_ADM y EMPLEADOS\_PROD, el resultado de la operación es una nueva relación que tiene la unión de las tuplas de las relaciones de originales.

Operadores

* **Redenomina**r :=Es uno de los principales operando. **Asigna un nombre** a la relación que resulta de una operación de álgebra relacional y se hace de la siguiente forma, R := E, siendo R la relación y E la operación del álgebra relacional. Estamos hablando de asignar un valor.
* **La Unión (U).** Es un operador binario. Tabla T unida de la tabl S -> T U S
  + Deben ser **compatibles**. Decimos que dos relacionaes T y S son relaciones compatibles si:
    - Tienen el mismo grado
    - Tiene que tener el mismo dominio.
    - No se repiten. Si se repiten no se ponen.



EMPLEADOS\_ADM **U** EMPLEADOS\_PROD

El resultado será el siguiente:



Tienen el mismo grado. En la unión todo lo que no se repita lo coloca.

* **Intersección (∩ al revés):** En las relaciones las que son tuplas idénticas, las podremos encontrar en la intersección. Es binaria. Han de ser compatibles. Solo se encontrarán en la nueva relación aquellas tuplas cuyos atributos de los campos son comunes en ambas tuplas.

R : = EMPLEADOS\_ADM **∩** EMPLEADOS\_PROD

**\*Ejemplo, en el caso de las dos tablas anteriores. Solo se mostrará la siguiente tabla:**



* **La diferencia (-):** es una operación en la que, a partir de dos relaciones, se obtiene una nueva relación formada por todas las tuplas que están en la primera relación y no están en la segunda. Solamente los que coinciden son los que se restan. La diferencia es binaria también.

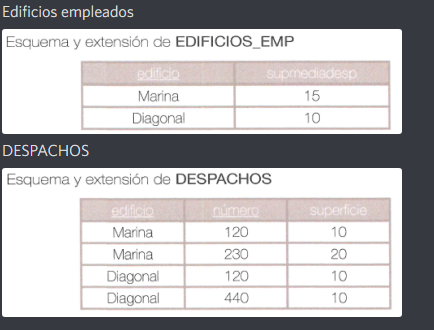
**Solo tiene sentido si se aplica a relaciones que tengan tuplas similares.** Es necesario que las relaciones sean compatibles.

\*Los atributos del esquema de la relación resultante de T – S coinciden con los atributos del esquema de la relación T y la extensión de la relación resultante de T – S es el conjunto de tuplas que pertenecen a la extensión de T, pero no a al de S.

R := EMPLEADOS\_ADM (T) **-** EMPLEADOS\_PROD

* **Producto cartesiano (X)**: A partir de dos relaciones obtiene una nueva relación formada por todas las tuplas que resultande concatenar tuplas de la priemra relación con tuplas de la segunda.
  + No necesita ser del mismo grado para llevar a cabo eso.

//acabar de entender



Escasamente se utiliza de forma explícita. El resultado que da no suele ser útil para resolver las consultas habituales. Es una operación primitiva del álgebra relacional a partir de la cual se define otra operación del álgebra 🡪 esta combinación se utiliza con mucha frecuencia.

Para realizar el producto cartesiano de las relaciones DESPACHOS y EDIFICIOS\_EMP es necesario redenominar atributos primeramente:

**EDIFICIOS(nombreedificio, supmediadesp) := EDICIOS\_EMP(edificio, supmediadesp)**

R := EDIFICIOS x DESPACHOS

El resultado da esto:



Operadores específicamente relacionales

* **Selección.** Es un tipo de operador unario. De una sola tabla especificamos solo una tupla o varias.
  + **Ø** dentro del dominio.

Edificio = Marina y superficie > 12; se indicaría DESPACHOS (edificio = Marina y superficie > 12)

//esto es el proceso previo a escribirlo con el lenguaje SQL.

* **Proyección.** Una operación que sirve para elegir algunos atributos de una relación y eliminar el resto, es decir, la proyección es una operación que, a partir de una relación, se obtiene una nueva relación formada por todas las tuplas de la relación de origen, que resultan de eliminar unos atributos especificados.
  + Sintaxis 🡪 RELACIÓN [nombre, apellido]
  + Atributos y tuplas.

EMPLEADOS\_ADM[nombre, apellido]

Elimina implícitamente todas las tuplas repetidas. El resultado de una proyección es una relación válida y no puede tener repeticiones de tuplas.

* + Es una operación unaria. Y se extraen columnas.

Usuario ‘COLUMNAS’

RELACIÓN[nombre, apellido] Atributos y tuplas.

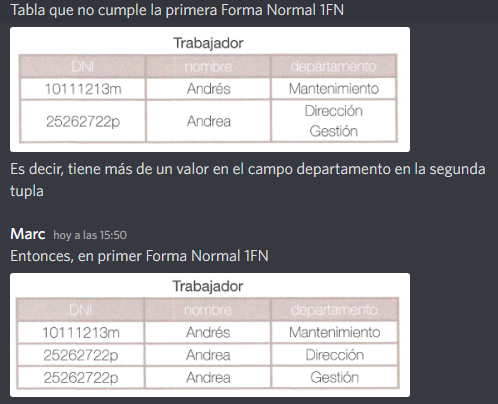
* **Combinación**. Como una proyección pero la forma todas las tuplas que resultan de concatenar tuplas de la primera relación con tuplas de la segunda que cumplen una condición de combinación especificada. Concatena aquellas que cumplan una condición determinada:
  + UNARIO ‘COLUMNAS’
  + RELACIÓN [nombre, apellido]
  + Atributos y tuplas.
* REL1[atrirel1=atrirel”, condición]RELL2

Limitaciones

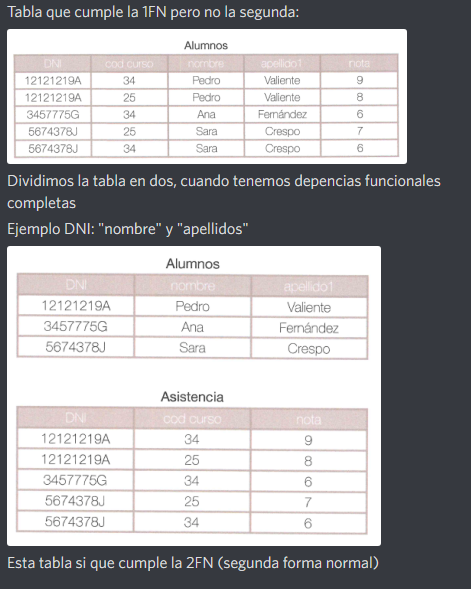
Reglas formales

\*Si tenemos la tercera forma normal significa que es más restrictiva que la primera. Incluye las otras dos.

* Primera forma normal

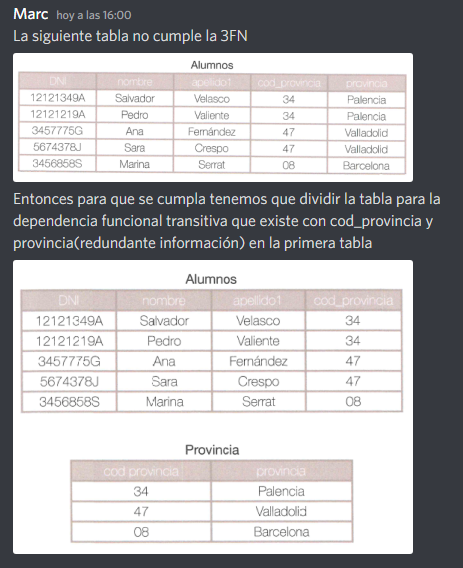


* Segunda forma normal



* Tercera forma normal
  + Hay una dependencia transitiva entre ambos valores. Provincia depende del código provincia. Lo que hace no esté en tercera forma normal. Cuesta más de entender que la segunda porque lo que se le está añadiendo aquí, además, es el componente de la dependencia transitiva entre ambos campos.

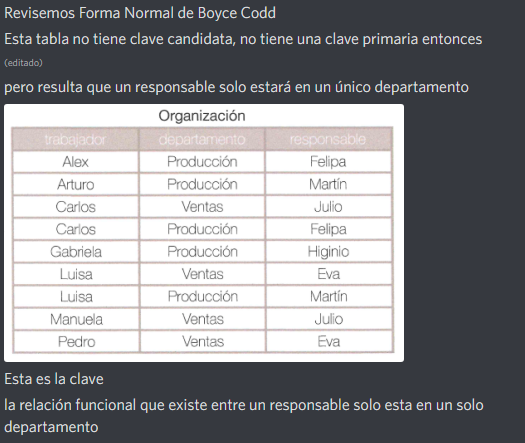
Ex:

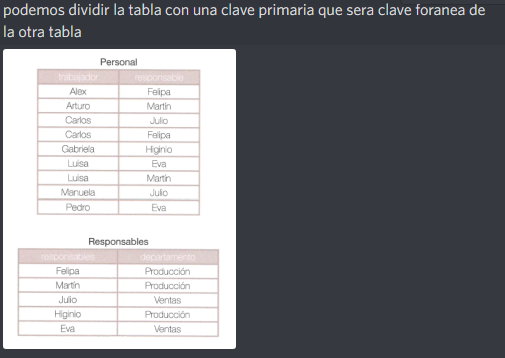


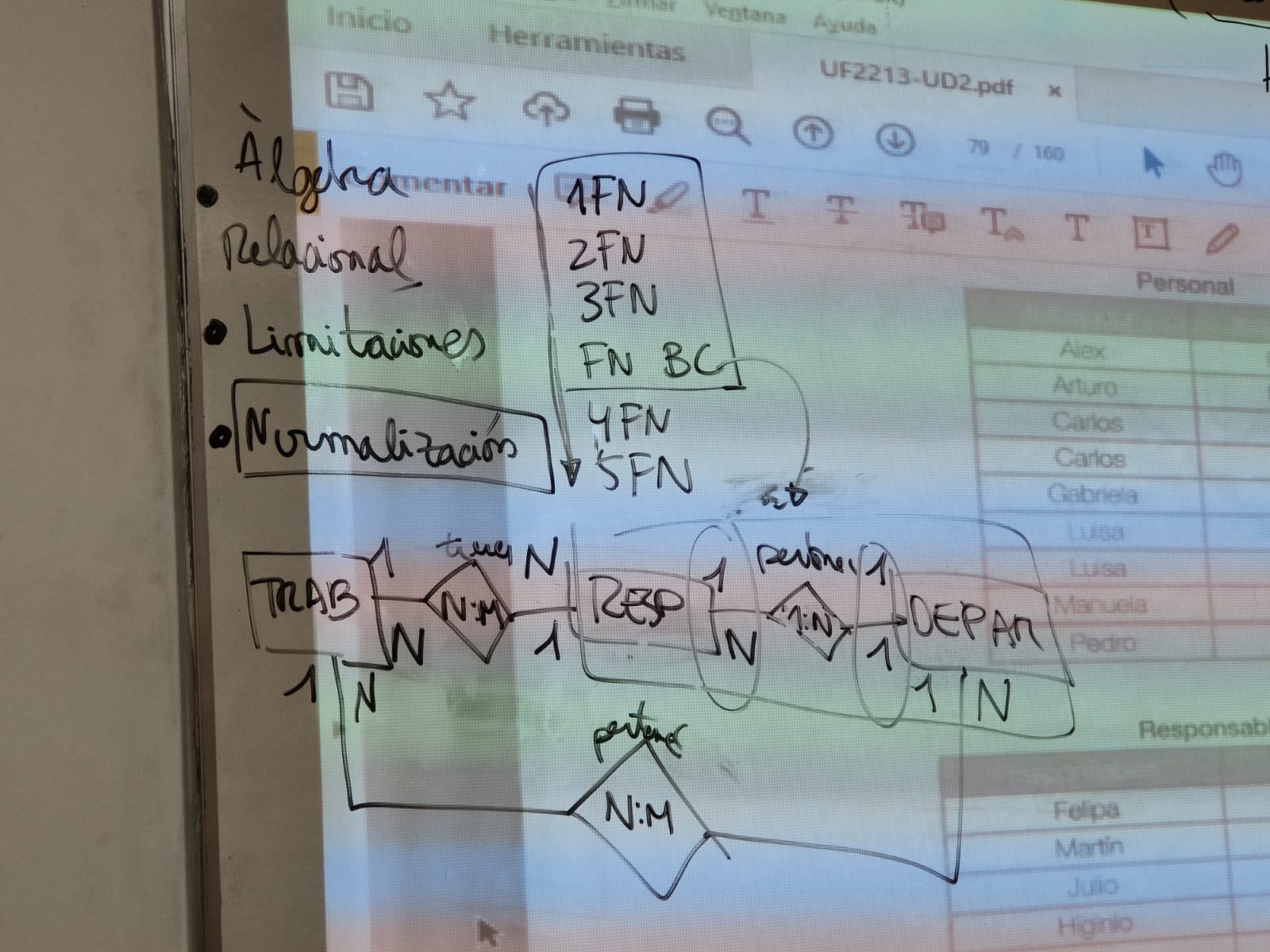
* Forma normal de Boyce-Codd:

No hay una clave primaria clave y todas son candidatas.

Lo que hay que hacer es estrechar al máximo las relaciones. En este caso vemos que una persona es como máximo responsable de un departamento, por lo tanto solo con que indiquemos sus trabajadores. Y posteriormente los responsables de cada departamento, ya entendremos el departamento para el que trabaja cada trabajador. No hay una relación de uno a muchos que no hayas separado.







Hasta aquí ya tendríamos una base de datos bastante buenas.

La dependencia multivaluada es la base de la cuarta forma normal. Si las funcionales eran la base de la segunda, la tercac y Boyce-Codd, stas son la base de la cuarta forma normal.

* Cuarta forma normal
  + Entre ellas dependen por separado. Todas son relaciones de muchos a muchos. En la Boyce-Codd alguno encontramos paor separado. Pero aquí no se puede dejar ninguna. Todas las relaciones son de muchos a muchos. Entre dos atributs no hay una dependencia clara. En Boyce Scodd si que tenemos la relación uno a muchos.
  + Lo puedes dividir entre diferentes elementos, aunque lo suyo es ver cual es más reducido, en este caso por cada profesor, cuales son de cursos diferentes.
* Quinta forma normal: A veces se ha dudado de ella por que no se sabe si realmente mejora.
  + Dependencia Join: la tabla resultante de tomar un subjconjunto de los atributos con una tabla. Tabla formada por unas cuantas columnas de la tabla original. Si queremos normalizarla para reducir información podemos hacer una división pero a triple. Puede ser que sea más complejo. Se reduce tanto que se divide la tabla era en tres.

1. B 🡪 La estructura, la manipulación y la integridad
2. D 🡪 Foráneas
3. B 🡪 La inserción, el borrado, y la modificación.
4. B 🡪 Las restricciones de integridad de usuario y las reglas de integridad del modelo.
5. B 🡪 Operaciones primitivas y no primi
6. A 🡪 Redenominar
7. D 🡪 Intercesión
8. Combinación y proyección. También puede ser combinación con corchetes.
9. C 🡪 También incluye la cuarta
10. D 🡪 Está en la página 84.

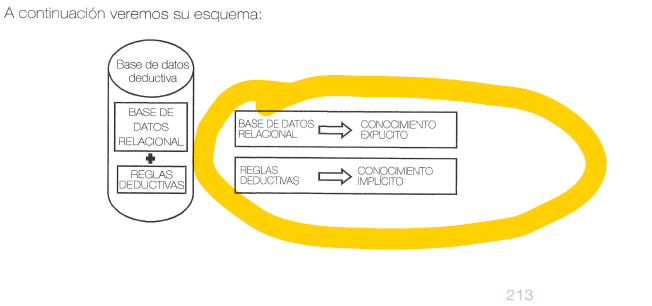
**15. 4 Modelos avanzados de BD**

4.1 BD Deductivas

Son un sistema de gestión de bases de datos que permiten mediante información **extensión al de reglas y hechos**, la **deducción de información extra que complemente la que nuestra base de datos tiene almacenada**.

Aparecen en la década de los 80 y su desarrollo ha contribuido a obtener mejoras en cuanto a los resultados y desarrollos fundamentales relativos al campo de la lógica y la inteligencia artificial.

* Este tipo de base de datos se usa con frecuencia en el campo de la deducción automática.



El lenguaje más utilizado es el datalog (no procedmental basado en un lenguaje de programación lógica de Prolog que simplifica la escritura de consultas simples y las hace más sencillas.

Esta base de datos presenta un alto grado de lógica matemática. A su vez, de computación. 🡪 Por la estrecha relación que existene tnre abmos conceptos: por esto se puede indentificar el término base de datos deductiva con el término base de datos lógicas.

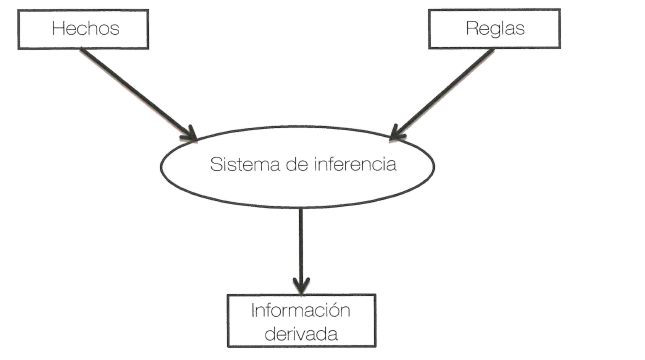
* Almacenar el mayor número de dato smediante un programa lógica e intentando selccionar los datos para guardar en la mejroia principal, de odo que se almacene una determinada información.
  + Cuando seleccionas los datos para guardar en la memoria principal, se almacena una determinada información.
* Se utilizan estos tipos de especificaciones:
  + Hechos: Conocimientos explícitos y se conocen como la base de datos extensional 🡪 podemos identificar con las siglas EDB.
    - Son las tplas en una base de datos relacional con la diferencia de que **se indica el nombre de tupla pero no se indican los nombres de los atributos**.
      * Una tupla intenta describir algún hecho de la realidad que dependen de de los atributos. Por ello se puede concluir que los atributos dependen de la posición en la tupla.

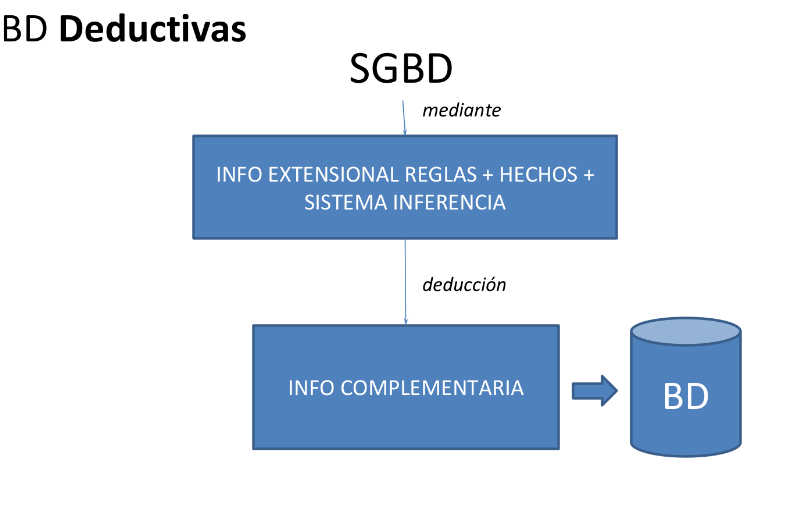
\*Recordar que una **tupla** ”es cada una de las filas de una tabla. Es importante señalar que no se puede tener filas duplicadas en una tabla”.

* + Reglas (pagina 95): Son los conocimientos implícitos y son conocidas como la base de datos intencional, la cual podemos identificar con las siglas **IDB**.
    - Puede tener parecido a las vistas relacionales. Indican como obtener mediante los hechos guardados nuevos hechos y las relaciones indirectas entre las entidades.
    - Normalmente se escriben utilizando un lenguaje dclarativo, se indica lo que se desea, pero no el algoritmo para llegar a esa meta. Las bases de datos deductivas combinan el modelo relacional para representar los datos con el modelo de la programación lógica para la representación y manejo de las reglas.

\*Hay que recordar que las **vistas** “son tablas cuyos datos derivan de la base de datos global, por estos datos no existen realmente, solo se visualizan como una tabla virtual, por eso no hay ningún archivo que la represente”.

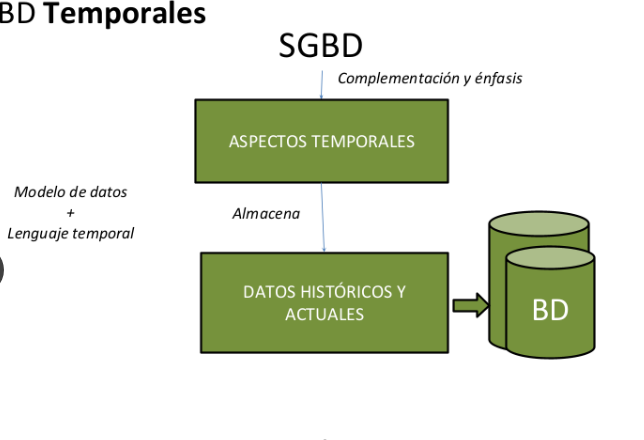
* + Sistemas de interferencia: también llamado mecanismo de deducción puede deducir hechos nuevos a partir de la base de datos interpretando las reglas. El modelo empleado por la base de datos deductivos está íntimamente relacionado con el modelo de datos relacional y sobre todo con el formalismo del cálculo relacional.
    - Relacionado también con el campo del a prgramación lógica y el lenguaje Prolog. Este es el punto de partida de los trabajos sobre bases de datos basados en lógica.
    - Datalog es un lenguaje similar a Prolog, está formado por reglas que junto a un cojunto de relacionex existentes componen el lenguaje, aunque la estructura gramatical se parece a la del Prolog, su semántica es diferente y nos permite trabajar con datos que no estén en la memoria principal.
  + Podemos establecer dos tipos de interferencia computacional basados en la interpretación de las reglas por la teoría de la demostración:
    - **Mecanismo de interferencia ascendente**: &//encadenamiento hacia delante o resolución ascendente. El sistema de interferencia parte de los hechos y aplica las reglas para generar hechos nuevos.
    - **Mecanismo de inferencia descendente**: //También llamado encadenamiento hacia atrás o resolución descendente.
      * Parte del predicado que es el objetivo de la consulta e intenta encontrar coincidencias con las variables que conduzcan a hechos válidos de la base de datos.
      * Retrocede desde el objetivo buscado para determinar hechos que lo satisfacen.
      * Si no existieran los hechos que buscamos, el sistema entonces buscará la primera regla cuya cabeza tenga el mismo nombre de predicado que la consulta.





Son datos que no existen, sino que se generan a partir de las reglas.

4.2 BD Temporales



A partir de la página 98. Un sistema de gestión identificado con las siglas DBMS, es decir, data base management system.

Hay tres tipologías de bases de datos temporales, dependiendo de como utilizan o manejan el aspecto del tiempo y estas son::

* Base de datos temporales de tiempo transaccional:
  + Tiempo de acuerdo al momento en el que se almacena un hecho.
* Bases de datos temporales de tiempo válido: Registran el tiempo en el cual un hecho ocurrió en la realidad, 🡪 realizar correcciones sobre los datos reistrado.
* Bases de datos bitemporales: Cada estado se puede modificar para ctualziar el conocimiento de la realidad pasada, presente o futura. Se pueden actualizar generando nuevas versiones de los mismos estados.

Cada hecho almacenado posee una marca de tiempo asociada, y son:

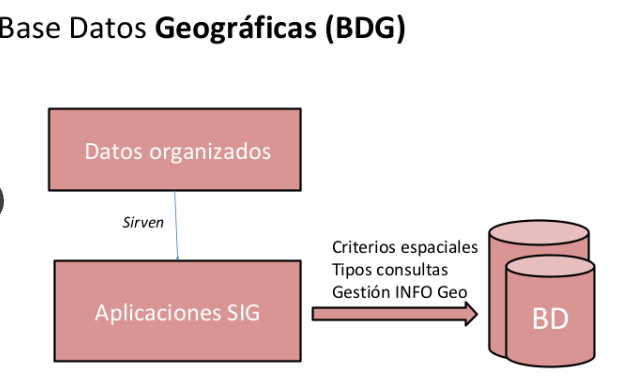
* Tiempo valido: Este concepto hace referencia al conjunto de intervalos de tiempo durante los que el hecho es verdadero.
* Tiempo de transacción: Engloba el intervlao de tiempo durante el cual ese hecho es cierto en el sistema ade base de datos.

Tres tipos de datos más comunes son:

* Dato temporal: Es el más importante, y se pueden definir como una unión ifnita de intervalos. El dato temporal se considera la base sobre la que se construye la base d edatos temporal.
* Dato estático: Se considera como una constante sobre todo el univeros del tiempo. Se suele identificar este dato como eterno, en contraste con el tipo de ddato temporal es válido apra un periodo o intervalo específico de tiempo.
* Dato instantáneo: dato solo vigente en un momento o instante actual

BD Geográfica

Aguardaremos unos datos mediante datos geográficos. Pueden ser de cualquier tipo. Son una colección de datos que sirven para aplicaciones de sistemas de información geográficas. Permiten el almacenamiento estructurada de los datos de acuerdo a **criterios espaciales,** tipos de consultas y **gestión de información geográfica**.



Quizá en una cinta de montaje no lo necesita toda la maquinaria pero quizá uno si que necesita datos geográficos. AGV 🡪 Robot que necesita una serie de datos geográficos. Puede ser un SIG (sistema de naegaicón geográfica) o bien puede ser global,e tc.

La lectura, la forma de acceder a estos datos de manera geográfica.



Interoperabilidad 🡪 Habilidad de dos o más sistemas o componentes para intercambiar información y utilizar información intercambiada.

Normalización 🡪 Se entiende como la utilización de forma adecuada de los procedimientos, datos y servicios.

El Instituto Geográfico Nacional, ha definido su Catálogo de Objetos Geográficos. Se puede definir como aspectos del mundo real relacionados con una ubicación concreta en el terreno. Se caracterizan por tener atributos que pueden ser representados mediante una geometría primitiva, tales como:

* Puntos: Se encuentran determinados por las coordenadas terrestres medidas por la latitud y longitud.
* Líneas: Objetos abiertos que cubren una distancia dada y comunican varios puntos o nodos.
* Polígonos: Figuras planas conectadas por distintas líneas.
* Celdas de un ráster: La rasterización, permite a través de una malla asociar daots a una imagen, es de cir, se pueden relacionar paquetes de información al os píxeles de una imagen digitalizada.

En teoría tendremos un usuario que accede a eso datos de manera geográfica.

Los SIG modelan la realidad territorial para convertirla en datos geográficos que son manipulados en un entorno informatizado. Utilizan los modelos de representación raster y vectorial.

* El modelo ráster
* El modelo de representación vectorial

Las bases de datos distribuidos, tenemos unos datos en el sistema que se ejecutan en el aplicativo global, pero se puede hacer una consulta de manera local. A su vez está repartido entre ordenadores de una misma red, ya sea a nivel local, o cada uno en una diferente localización geográfica. A un sistema que da información a dispositivos locales.

* En las criptos, cada uno tiene un aplicativo que revisa que existan. No hay un sitio global. El blockchain es todo lo contrario a base de datos distribuidas. +

Es un híbrido entre la centralizada y la descentralizada.

Base de Datos distribuida

**Centralized**: Una base de datos central a la que tienen acceso los diferents usuaris.

**Distributed**: Tendrá un aplicativo global. Tendrán particiones. Tendremos centralizaciones.

**Descentralized**: Sería algo parecido al blockchain. No hay un sistema central que tenga ese control.

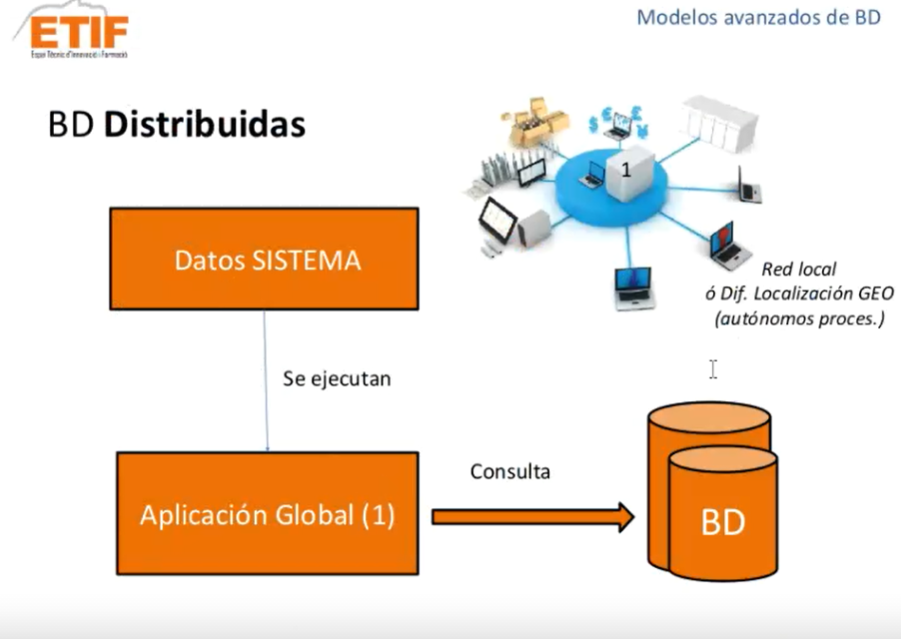
//Pendiente de repasar PDF.

Autonomía: Se pueden presentar en diferentes niveles, como son:

* Autonomía de diseño: Está relacionada con el mismo diseño
* Autonomía de comunicación: Cuando se comunica con otros sistemas manejadores de bases de datos (SMBD). Ellos pueden comunicarse autónomamente con esa base de datos.
* Autonomía de ejecución. Ejecuta operaciones locales como quiera.
* Autonomía de ejecución: Ejecuta operaciones locales como quiera.

Se debe definir un modelo de referencia para un esquema de estandarización en las bases de datos, cuyo propósito es dividir el trabajo en piezas y esas piezas se relacionen unas con otras:

* BD Distribuidas



Que servicios vamos a ofrecer?. Eso es importante tenerlo en cuenta. Que datos vamos a ver en esa base de datos. Podemos tener diferentes tipologías de base de deatos dentro de una arquitectura.

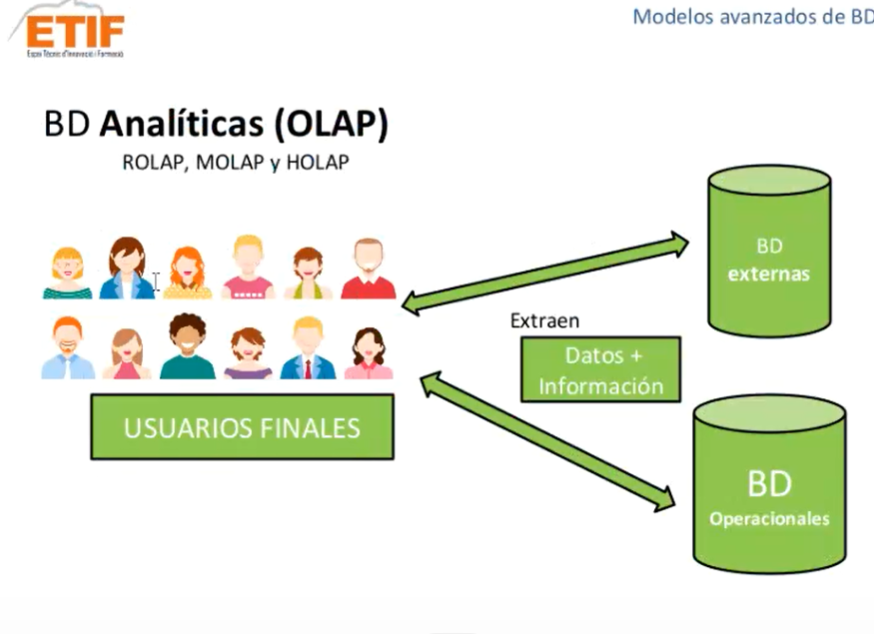
BD Analíticas (OLAP=.

* ROLAP
* OLAP

Son bases de datos que almacenan datos e información que se extraen de bases de datos operaciones y externas seleccionadas. Se componen de los datos y la información resumida que más necesitan los usuarios finales.

No tiene por qué ser locales. Las bases de datos analíticas también el nombre de bases de datos gerenciales o bases de datos de información. Es el acrónimo en inglés de procesamiento analítico en línea (On-Line Analytical Processing).

Se componen del a información resumida que más necesitan los usuaris finales.

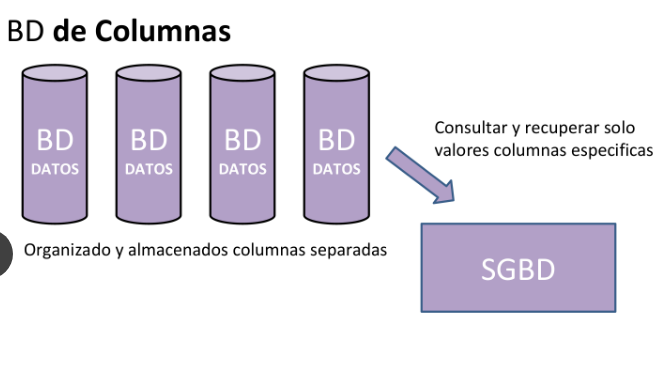


En la operacional podríamos tener una serie de datos. Y en las externas una serie de información que se extrae de esa base de datos operacional. La razón es la rapidez de su respuesta. Una base de datos relacional almacena entidades de tabla (falta penúltimo parágrafo de la página 109).

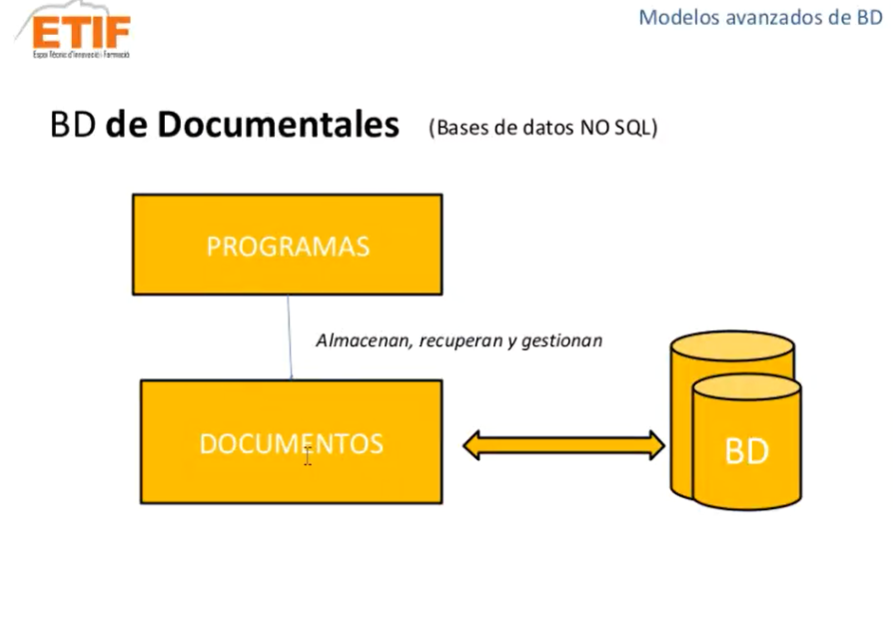
* ROLAP: Implementación OLAP que almacena los datos en un motor relacional. Los datos son detallados, evitando las agregaciones y las tablas se encuentran **desnormalizadas**. Sacamos un dato concreto.
  + Tendríamos diferentes estructuras de diseño. Serían basos de datos relacionales.
* MOLAP: Podmeos obtener datos en una base de datos multidimensional. Pueden tener más de una información agregada en esa información. Almacena los datos en una base de datos multidimensional.
  + No sigue la primera forma normal porque tenemos más de un valor. Pero tenemos más de un valor.
* HOLAP. Almacena algunos datos en un motor relacional y otros en una base de datos multimendional. Es Híbrid.

BD de Columnas: Son bases de datos orientadas a columnas, es decir, tienen sus datos organizados y almacenados por columnas separadas, debido a esto, el sistema puede evaluar las columnas que están siendo acceddias en una consulta y recuperar sólo los valores solicitados desde las columnas específicas.

* Estan definidos por columnas y no por tablos.
* Ventajas:
  + Acceso rápido a los datos, como el modelo DSM, el cual nos permite consultar rápidamente los datos columna a columna, al guardarse físicamente de manera contigua.
  + Lee solo los valores de las columnas necesarias para el procesamiento de una consulta determinada, por lo que tiene una amyor eficiencia en entornos de almacenes, donde las consultas, típicas incluyen los agregados realizas por un gran número de elementos de datos.
  + Comprime la información asignable de cada columna con el fin de mejorar el procesamiento desde el ancho de banda del acceso a disco.
  + Los cambios en el esquema tiene menor impacto y por lo tanto el coste de ralizarlos es menor.
* Desventajas //
* La función es sobretodo para el acceso a esos campos de manera más eficiente posible.
* Se considera un modelo avanzado de base de datos.



Bases de datos documentales



Filezilla almacena, recupera y gestiona una serie de documentos de una base de datos. La URL de Firebase (por ejemplo) tiene una URL, a una base de datos con una serie de archivo. Tenemos una serie de archivos en esa base de datos. Podemos tener una base de datos que guarde un archivo concreto. Se puede ir combinando.

A diferencia de las bases de datos relacionales, estas bases de datos están diseñadas alrededor de una noción abstracta de documento. Mientras cada implementación de base de datos orientada a documentos difieren en los detalles, en general todas ellas comparten el principio de que los documentos encapsulan y codifican datos

Más de la sencilla correspondencia clave-documento o clave-valor usada para recuperar un documento. Ofrece un API o un lenguaje de interrogación para recuperar documentos según su contenido. Así poder, por ejemplo, preguntar por todos los documentos que tienen un valor dado en un campo. El conjunto de características del API o del lenguaje de interrogación , así como lo que se obtiene, varía significamente entre distintas implemtaciones. Se utiliza para:

* Colecciones
* Etiquetas
* Metadatos ocultos.
* Jerarquía de los contenidos.

Se puede llegar a cambinar. Podríamos tener Netflix con una base de datos documental, dentro de la cual tuviéramos una serie de particiones distribuidas. Entre las que los usuarios pueden acceder.

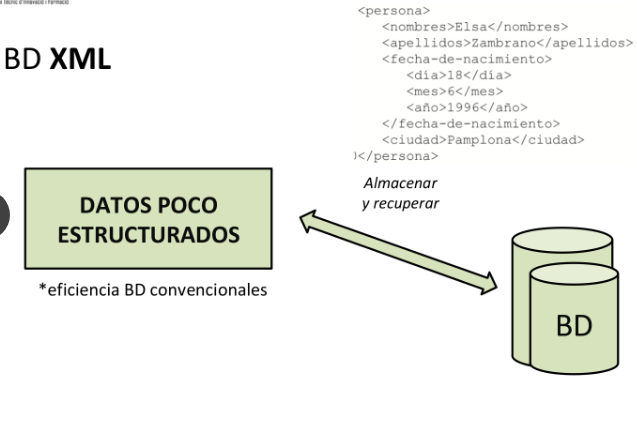
WEBANGULAR HTML Es un framework, para aplicaciones web. Es un sistema que te ayuda a crear aplicativos. Desarrollado por Typscrit.

BD XML: Si os suena, XML. Esto es similar a copiar la bases de datos. Sirve para almacenar y reucuperar datos poco estructurados. No tienen relación entre ellos a no ser que tengan estructuras anidadas en concreto.

* Title tiene algo que ver con link? En HTML, en verdad no. Porque están en el mismo canal, pero entre ellos no tienen relación, realmente.
* Centrados en datos
* Centrados en documentos.

Hacemos una exportación de una base de datos como una fotografía, como datos anidades.

El XML es el documento que guarda esa bases de datos. Las otras tienen otras extensiones, depende del sistema. No es un archivo en concreto.



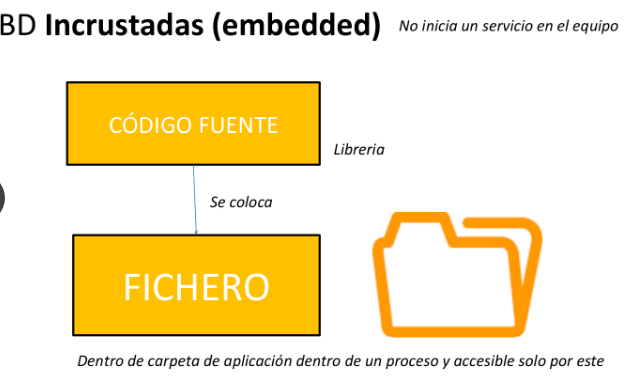
BD Incrustadas (embedded)

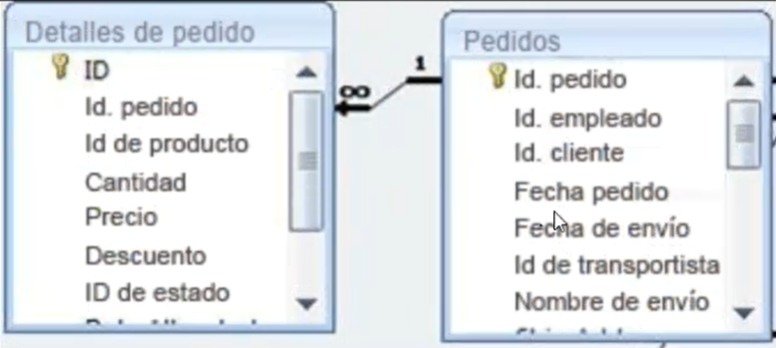
No inicia un servicio en el equipo

Código fuente – Librería

La base de datos está incrustada en un servicio. El próximo sistema gestor de la base de datos está creando esos archivos. Con su propio lenguaje está creando esa base de datos. Nosotros mediante un fichero generaremos la base de datos. Nosotros veremos esa relación mediante el propio sistema. Tendremos un Excel que en si, no será una base de datos, podemos importar esa base de datos que incrústalo dentro del sistema gestor de base de datos. En ese momento pasará a ser una base de datos.

El próximo sistema gestor de base de datos. De esta tabla y esta tabla voy a generar esa base de datos. Podremos modificar la base de datos, desde el propio sistema. Podemos acceder a esa relación y añadir más registros extras.





1. ¿Cuáles son los tipos de factores que utilizan las bases de datos deductivas?

b) Los hechos, las reglas y el sistema de inferencia.

\*Teniamos una serie de reglas que nos darán hechos. Nos darán una serie de datos. Son deductivos.

2.

d) El tiempo de validez y el tiempo de transición

3.

c) El dato temporal (el más importante), el dato estático y el dato instantáneo.

4.

b) Los puntos, las líneas, los polígonos y las celdas de un raster (esas representaciones primitivas a la hora de guardar una base de datos geográfica).

5.

c) En tres niveles

\*Vimos tres niveles:

El nivel de autonomía, etc. Lo vemos en tres niveles.

6.

b. ROLAP, MOLAP y HOLAP

7. ¿De qué esta compuesta las filas de BD de columnas?

c) Las filas de una columna son las tuplas. Serían los registros.

8.

c) Colecciones, Etiquetas, Metadatos ocultos y Jerarquías de directorios.

9.

d)

\*Películas quizá si que podría estar dentro de documentos.

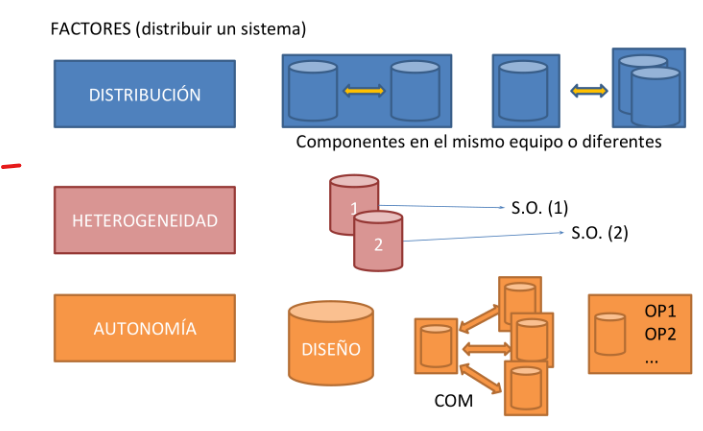
10.

a)

Página 116. La Scimore DB, no es la más importante hoy en día dentro. Si buscas los tipos de base de datos incrustadas, casi que no sale ScimoreDB.

**T.15.5 Análisis detallado de la distribución de la BD**

5.1 Formas de distribución



Distribución: Los componentes del sistema están localizados en un mismo equipo o en diferentes equipos.

Heterogeneidad: Cuando existen componentes que se ejecutan en diversos sistemas operativos.

Autonomía: Se puede presentar en diferentes niveles, como son:

* Autonomía de diseño: Está relacionadas a su propio diseño.
* Autonomía de comunicación: Cuando se comunica con otros sistemas manejadores de bases de datos (SMBD).
* Autonomía de ejecución: Ejecuta operaciones locales como quiera.

Modelo de referencia para un esquema de estandarización en las bases de datos: propósito: dividir el trabajo en piezas y esas piezas se relacionan unas con otras.

* Basado en componentes: Se definen las componentes del sistema junto con las relaciones entre ellas.
* Basado en funciones: Se identifican las diferentes clases de usuarios junto con la funcionalidad que el sistema ofrecerá para cada clase.
* Basado en datos: Se identifica los diferentes tipos de descripción de datos y se especifica un marco de trabajo arquitectural el cual define las unidades funcionales que realizarán y usarán los datos de acuerdo con las diferentes vistas.

\*ANSI/SPARC.

La transparencia

Es la división del nivel semántico y la implementación del sistema.

* Ocultar al usuario los detalles del diseño. No tiene que saber que se encuentra trabajando sobre un sistema distribuido, por ejemplo, la independencia de los datos es una forma de transparencia.
* El **propósito** de la transparencia en el ambiente distribuido es la independencia de datos:
  + Manejo de la red de comunicación
  + Manejo de copias repetidas
  + En la distribución o fragmentación de la información

La independencia de datos: inmunidad de las aplicaciones de usuarios a los cambios en la definición y organización de los datos y viceversa.

* **Independencia lógica** de datos: inmunidad de las aplicaciones de usuario a los cambios en la estructura lógica de la base de datos, permite un cambio en la definición de un esquema no debe afectar a las aplicaciones de usuario. Cuando cambie la estructura, no puede afectar a las aplicaciones.
  + Agregar un nuevo atributo a una relación, creación de una nueva relación, reordenameinto lógico de algunos atributos.
* **Independencia física de datos:** ocultamiento de los detalles sobre las estructuras de almacenamiento a las aplicaciones de usuario, es decir, **la descripción física de datos puede cambiar sin afectar a las aplicaciones de usuario**.
  + Los datos pueden ser movidos de un disco a otro, o la organización de los datos puede cambiar.

Hay niveles de transparencia: con el propósito de establecer una arquitectura de un sistema de bases de de datos distrubuidas es ofrecer u nivel de transparencia adecuado para el manejo de la información.

* **Primer nivel:** se soporta la transparencia de red
* **Segundo nivel:** se permite la transparencia de replicación de datos.
* **Tercer nivel:** se permite la transparencia de la fragmentación
* **Cuarto nivel:** se permite transparencia de acceso, por medio de un lenguaje de manipulación de datos.

Que son la transparencia de localización de framgentación, de replicación de datos y de localización.

* La **transparencia de localización**: permite aceder al usuario a los datos sin tener en cuenta la ubicación de estos. Debe ser transparente al usuario ya que no necesita saber dónde está el dato par autilizalo. Se consigue cuando los administradores de transacciones distribuidas pueden determinar la localización de los datos y emitir acciones a los administradores apropiados.
  + Se puede ejecutar cuando los administradores de transacciones tienen acceso a los directorios de localizaciones de datos. Los administradores de transacciones necesitan conocer si los datos cambian de lugar, ya que las transacciones ignoran la modificación en la localización.
* La **transparencia de fragmentación**: El acceso a una base de datos distribuida debe hacerse forma transparente, los usuarios deben comportarse como si los datos en realidad no estuvieran fragmentados, lo cual es necesario por razones de rendimiento.
  + El sistema maneja la conversión de consultas de usuario definidas sobre relaciones globales a consultas definidas sobre fragmentos. Las respuestas a las consultas fragmentadas para obtener una sola respuesta a una consulta global.
* La **transparencia de réplica**: si existen réplicas de objetos de la base de datos, está debe ser controlada por el sistema manejador de base de datos (SMBD).
  + La función principal de la transparencia de replicación es la de mantener la consistencia entre los copias, esta funciona en forma transparente a las aplicaciones.
  + La replicación puede mejorar el funcionamiento y proteger la disponibilidad de las aplicaciones, porque alternas opciones de acceso de los datos existentes.
  + Existe una **copia principal** y varias **copias secundarias**, las que se extienden a lo largo de las modificaciones en forma asíncrona.
  + Hay dos tipos de propagación de modificaciones:
    - Incrmeental: la información que se envía desde la copia principal a las secundarias son las variaciones en los datos.
    - Total: Se envía toda la copia principal completa

Utilidad: hacer que el sistema sea menos sensible a los fallos. Ya que si la copia principal no está disponible, se peude seguir usando alguna de las copias secundarias. La replicación es necesaria por las siguientes razones.

* + Mayor rendimiento: Debido a que se dispone de **copias locales.**
  + Mayor disponibilidad: Debido a que los datos son accesibles siempre al tenerse varias copias.

Principal desventaja: hay que mantener actualizadas todas las copias de ese objeto o dato replicado. Esto nos lleva al problema de la propagación de las actualizaciones.

5.2 Arquitectura ANSI/X3/SPARC (130-145)

5.3 Transacciones distribuidas

5.4 Mecanismos de distribución de datos

**Mecanismos de distribución de datos transacciones:**

1. Distribución de datos
2. Durabilidad
3. Características BD
4. Duración
5. Estructura

Autoevaluación

1.

b) La distribución, la heterogeneidad y la autonomía

2.

d) La independencia lógica de datos y la independencia física de datos.

3.

4.

5.

6.

7.

8.

9.

10.

**UF1846: Módulo de SQL**

**T.16 Introducción SQL**

**\***En sí, no es un lenguaje de programación. No tiene funciones, por ejemplo.

El SQL es el Structured Query Lenguage. Lenguaje estructura de consultas. Inventado por IBM (SEQUEL).

* No solo consultas, no confundirse con el nombre, si no conoces lenguaje puedes pensar que solo se realizan consultas pero SQL sirve para crear BD de cero, crear campos nuevos, eliminarlos, modificar propiedades campos, establecer relación entre tablas de una base de datos, etc.

\*Cual es la consulta. Lo que sea. Nosotros haremos una acción concreta, en esa base de datos. Que tablas de esa base de datos estamos modificando. Que campos vamos a mencionar.

* Que tablas estamos modificando.
* Que campos vamos a mencionar
* Cual va a ser la relación de la bases de datos.

\***Vamos a pedir:** “dime un dato concreto”. Puede ser un campo, puede ser una consulta. \*No solo es un lenguaje de consultas.

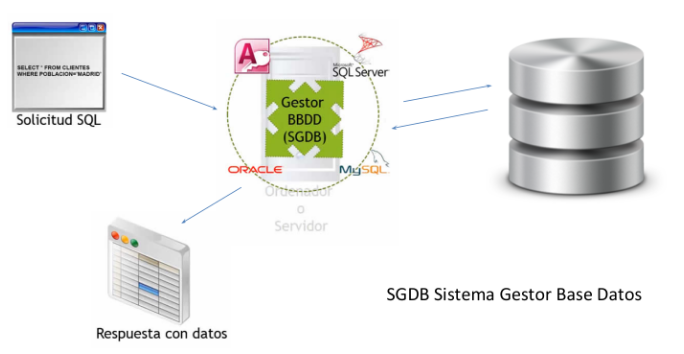
Más tablas, que se conectan con más tablas. Si yo tengo una tabla de clientes, con ventas. Puedo decirle las que ha tenido, y los datos de esa venta. SQL nos sirve para prácticamente todo en una base de datos.

\*La mayoría de **acciones** se llaman acciones **CRUD**

* Create
* Read
* Update (into) o insert
* Delete

\*Aunque se llame lenguaje estructurado de consulta, no solo hace consultas. Hace más casos.

Esquema básico de funcionamiento de este lenguaje de acceso y manipulación de datos en base de datos relacionales



Nosotros tendremos una solicitud concreto de SQL:

\*Select determinada tabla y mostrarme una serie de clientes.

En el propio SGDB haremos esa consulta. Puede ser como:

* Acces: En el mismo paquete office.
* AQL Server
* Oracle
* My SQL: Lo tenemos instalado con XAMPP.

\*Heide SQL, Terminal.

Tendremos una serie de datos, eso lo importamos al sistema gestor de base de datos.

Estándar SQL



Ese estándar se lo enviaremos a un Sistema Gestor de Base de Datos. Enviaremos un universal como es el caso del SQL.

Acces, normalmente entraran.

1. Select [E-ail Adress], Company,
2. From Contacts
3. Where City = Seattle

Los cambios de un programa a otro pueden ser muy pequeños como es el caso de los corchetes. Tenemos diferentes formas (dialectos) de escribir el lenguaje estandarizado.

ANSI (Instituto de Estandarización). Establece una nomenclatura ANSI y una nomenclatura ISO para este tipo de lenguaje.

Dialecto como tal.

Grupos de comandos SQL

* DDL: Lenguajes de definición de datos:
  + Crear y modificar la estructura de la base de datos BD.
  + Crear, eliminar, tablas…modificar estructura tablas agregando eliminados campos, etc.

\*Crearía la estructura sobre la que se articula la base de datos. Puede eliminar los campos o los registros.

* DML: Lenguaje de manipulación de datos:
  + Seleccionar registros de una BD, consultas, insertar nuevos registros, actualizar, borrar información.
  + Consultas de selección y acción.
* DCL: Lenguajes de control de datos:
  + Proporcionan seguridad a la información en la base de datos.
* TCL: Lenguajes de control de transacciones:
  + Gestión de los cambios en los datos.

DDL: Create, Truncate, Drop, Alter

DML: Select, Delete, Update, Insert

DCL: Grant, Revoke

TCL: Cimmit, Rollback, Savepoint

Cláusulas SQL:

* **From:** especifica la table de la que se quieren obtener los registros
* **Where:** especifica las condiciones o criterios de los registros seleccionados
* **Group By:** para agrupar los registros seleccionados en función de un campo
* **Having:** especifica las condiciones o criterios que deben cumplir los grupos
* **Order by:** ordena los registros seleccionados en función de un campo

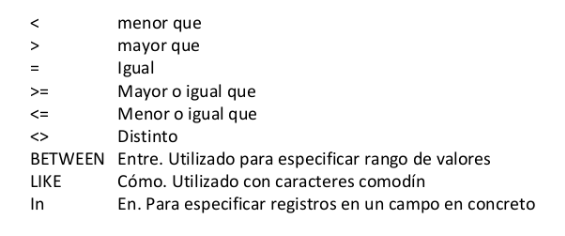
Instrucción SQL:

**COMANDO\* + CLÁUSULAS\* + OPERADORES + FUNCIONES**

Construimos mediante estos componentes una frase en inglés una instrucción SQL

Operadores SQL

* Comparación:



* Lógicos: (el and multiplica y el or solo está sumando)
  + And ----- Y lógico
  + Or ------ O lógico
  + Not ----- Negación lógica

Orden de escritura

1. Comando\*
2. Cláusulas\*

From, Where, Group By, Having, Order By.

Tipos de datos

* Si es un string: VARCHAR
* Si es un número:

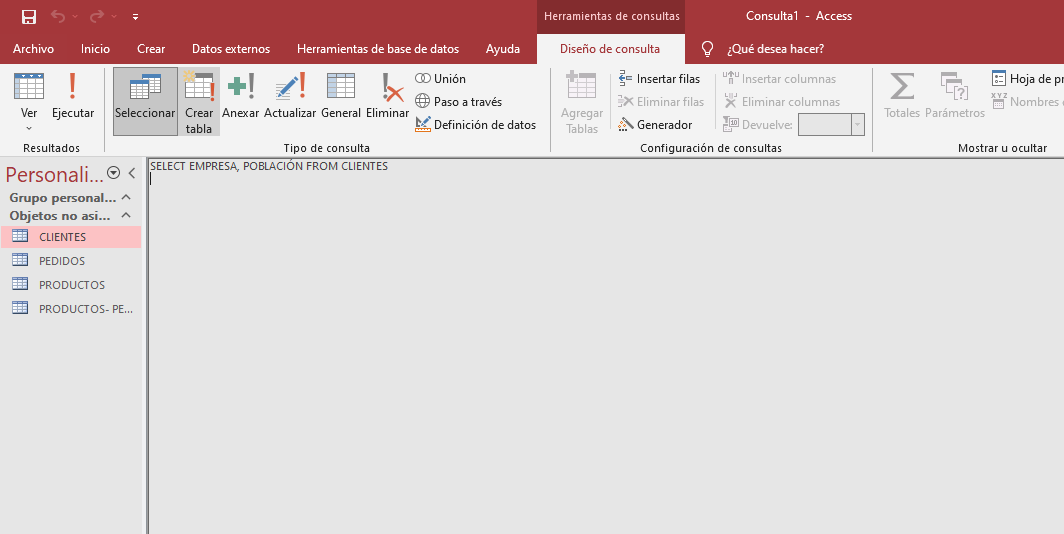
Ejercicios SQL

**Ejercicio 1: Realizar una consulta que muestre los campos “Empresa” y “Población” de la talba “Clientes”.**

SELECT EMPRESA, POBLACIÓN FROM CLIENTES

Procedimiento Acces:

En diseño de consulta 🡪 Se crea una nueva opción

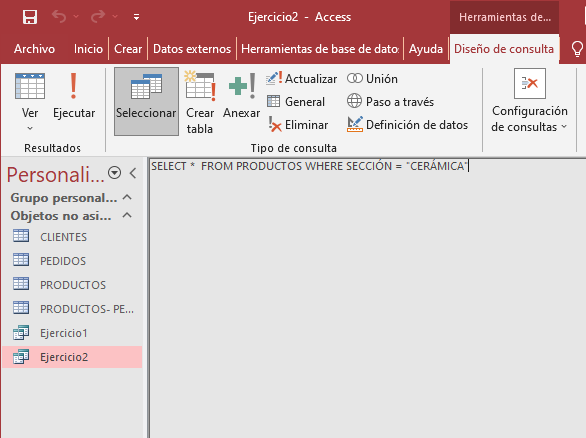


**Ejercicio 2: Realizar una consulta que muestre los artículos de la sección “Cerámica”.**

**SELECT \* FROM PRODUCTOS WHERE SECCIÓN = "CERÁMICA"**

\*No especifica que sea el nombre del artículo.

\*Para que muestre todo. Ponemos un \* y lo mostrará todo.



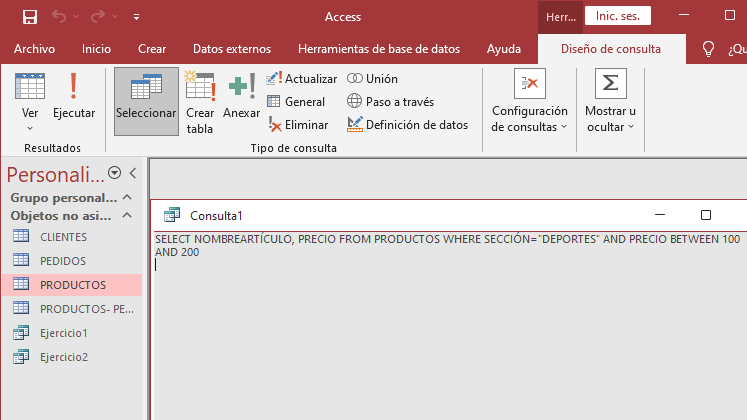
**Ejercicio 3. Realizar una consulta que muestre los productos que muestre los productos de la sección “Deportes” cuyo precio esté entre 100 y 200€. En la consulta solo se mostrarán los campos “Nombre de artículo” y “Precio”.**

SELECT NOMBREARTÍCULO, PRECIO FROM PRODUCTOS WHERE SECCIÓN="DEPORTES" AND PRECIO BETWEEN 100 AND 200

\*También podría ser PRCIO >= 100 AND PRECIO <=200

🡪 Con el BETWEEN, nos estamos ahorrando un PRECIO aquí.

**\*Se puede utilizar and con más de un operador.**



**Ejercicio 4. Realizar una consulta que muestre los productos cuyo país no sea España.**

Crear 🡪 Diseño de consulta 🡪

SELECT \* FROM PRODUCTOS WHERE NOT PAÍSDEORIGEN="ESPAÑA";



**Ejercicio 5. Realizar una consulta que muestre los artículos españoles de la sección “Deportes” o aquellos cuyo precio sea superior a 350€ independientemente de cual sea su sección o país de origen.**

Access

1. SELECT \* FROM PRODUCTS WHERE PAÍSDEORIGEN=”ESPAÑA” AND SECCIÓN=”DEPORTES” OR PRECIO>350 AND

\*Cuidado con la conjunción i/o.

El or está englobando desde where país de origen hasta deportes. Porque lo primero que se hace es el producto/multiplicación que es paísdeorigen y sección=deportes.

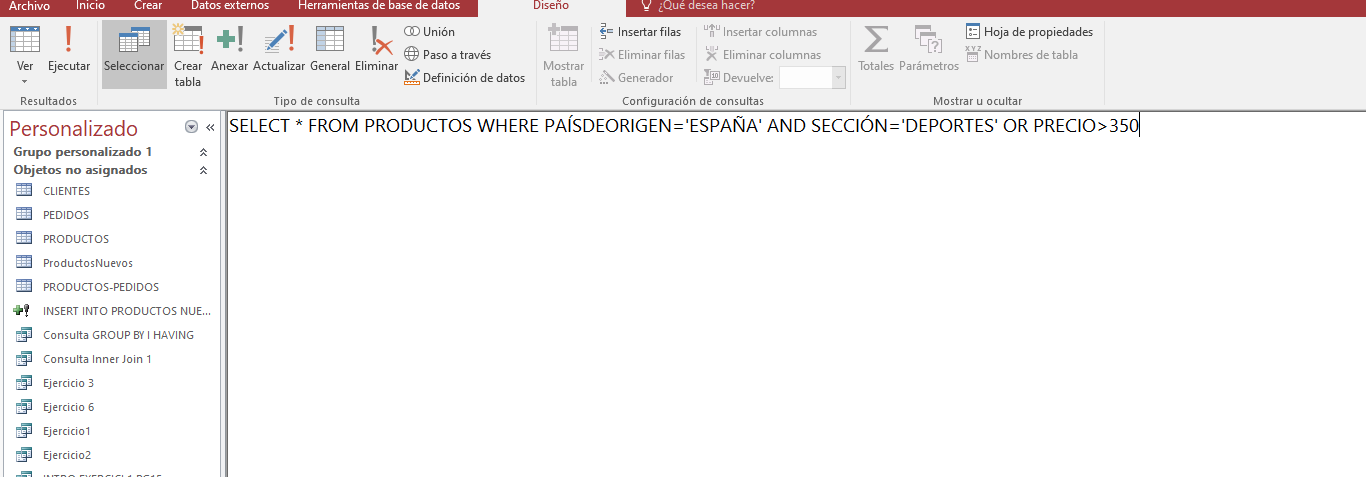
0+0🡪0

0+1🡪1

1+0🡪1

1+1\_1

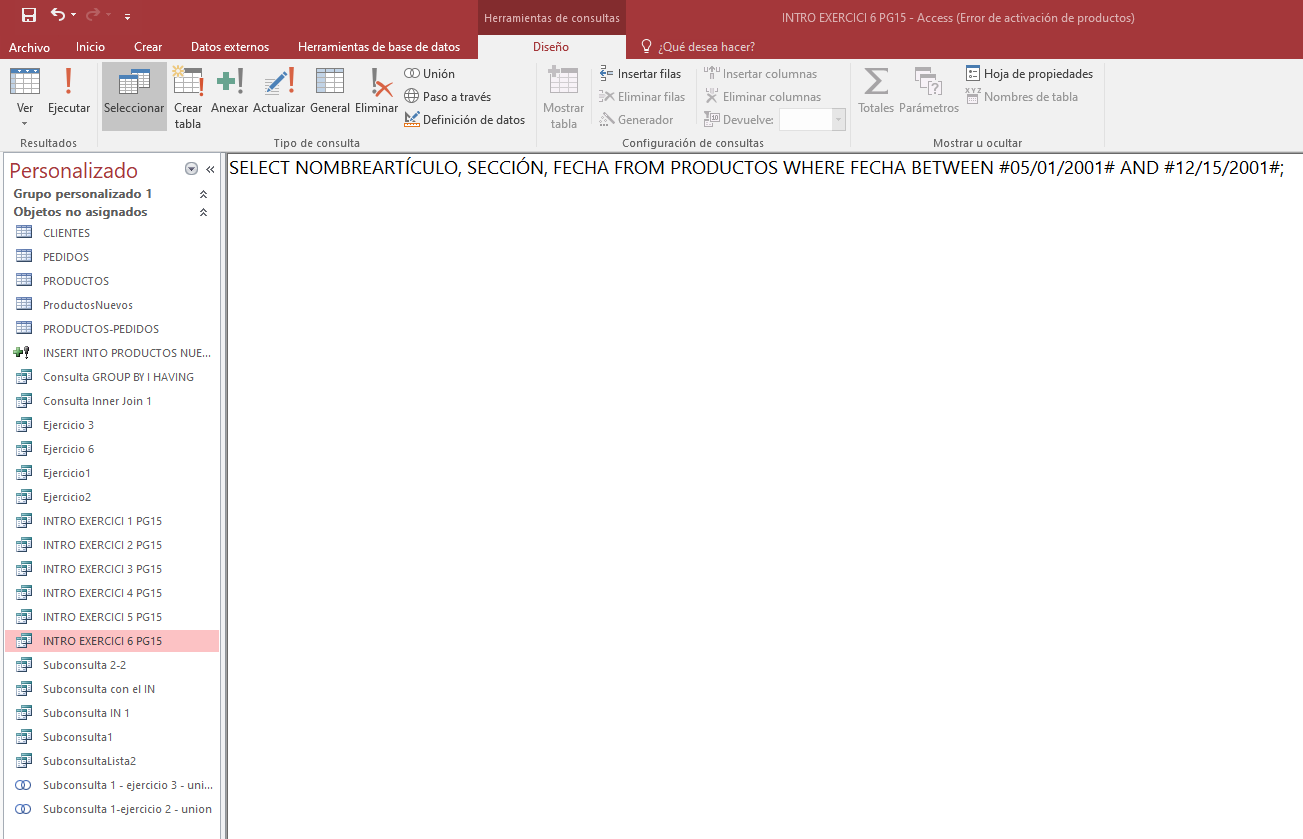
(1X1)+1



**Ejercicio 6. Realizar una consulta que muestre los productos cuya fecha esté entre 1/05/2001 y 15/12/2001. En la consulta solo se visualizarán los campos “Nombre de artículo”, “Sección” y “Fecha”.**

Access

SELECT NOMBREARTÍCULO, SECCIÓN, FECHA FROM PRODUCTOS WHERE FECHA BETWEEN #05/01/2001# AND #12/15/2001#;

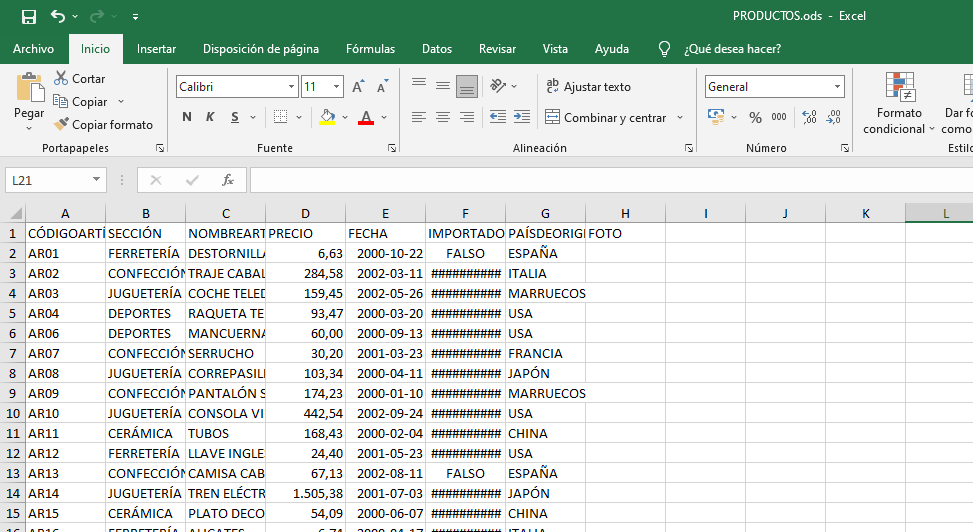


Importación de la base de datos en **PHPMyAdmin**

**En SQL el varchar es el tipo de dato.**

1. **Crea la base de datos**
2. **Inserta las diferentes tablas**
3. **Inserta las diferentes filas y tuplas aquí.** Las comas que utiliza para los valores son els accents oberts en català.
   1. **La primera no la coge porque le hemos dicho que era el nombre del campo.**

**\***Con el caso de los clientes hemos creado una base de datos y le hemos puesto nosotros el nombre. Luego hemos importado la base de datos y hemos vuelto a seleccionar la opción de que son los nombres del campo.



Donde pone general podemos cambiar y poner más formatos de número, vamos a fecha y allí podemos escoger el tipo de fecha que queremos.

Todos estos parámetros se guardan en el diccionario de datos. A su vez están en las diferentes relaciones entre los objetos.

\*Le interesa saber el tipo de archivo; o INT o DOBLE O VARCHAR.

Instrucciones

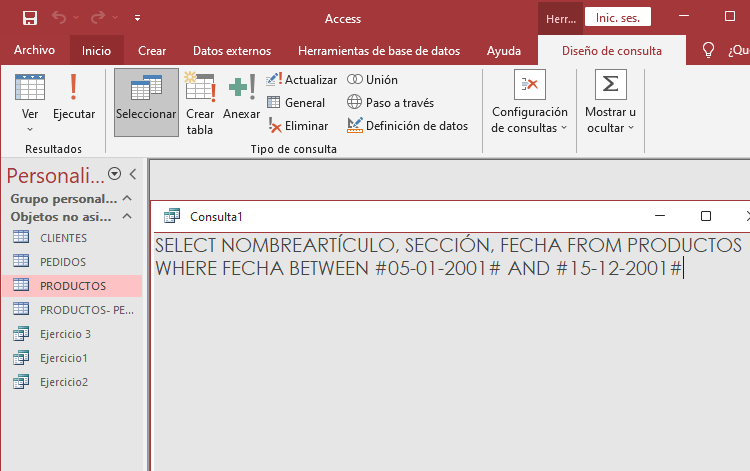
\*La **instrucción**

\*La **instrucción** que va a dar para destruir una base de datos: DROP DATABASE `ods\_db`;.

**Ejercicio 6. Realizar una consulta que muestre los productos cuya fecha esté entre 1/05/2001 y 15/12/2001. En la consulta solo se visualizarán os campos “Nombre de artículo”, “Sección” y “Fecha”.**

Para Acces es:

SELECT NOMBREARTÍCULO, SECCIÓN, FECHA FROM PRODUCTOS WHERE FECHA BETWEEN #05-01-2001# AND #15-12-2001#



Para PHP My SQL es:

SELECT NOMBREARTÍCULO, SECCIÓN, FECHA FROM PRODUCTOS WHERE FECHA BETWEEN ‘2001-05-01’ AND ‘2001-12-15’



En el caso de PHP my admin, se trata de un software diferente, y por lo tanto estamos hablando de un dialecto diferente. Es la manera como interpretan el lenguaje SQL.

En cifras las fechas es una comparativa. Podríamos tener >=100 PASADO <=150.\*Para un mismo dialecto nosotros podemos tener diferentes maneras o diferentes vías de encontrar el resultado que ns está proponiendo.

//Quedaría guardar en Acces, l’exercici 4-5-6.

Cláusula ORDER BY

Ordena los registros seleccionados en función de un campo

Si queremos realizar una ordenación en nuestros registros, ejemplo:

SELECT \* FROM PRODUCTOS WHERE SECCIÓN=’DEPORTES’ OR SECCIÓN=’CERÁMICA’ + **ORDER BY SECCIÓN**

El texto por defecto sale de A a la Z, números de menos a mayor

* DESC: *invierte el orden por defecto*

SELECT \* FROM PRODUCTOS WHERE SECCIÓN=’DEPORTES’ OR SECCIÓN=’CERÁMICA’ ORDER BY SECCIÓN, PRECIO

No se puede ordenar por **dos campos** a la vez pero si por varios criterios de ordenación, primero uno, después los siguientes.

\*El desc se debe poner en cada uno de los campos, solo afecta a uno. Por ejemplo si queremos atribuirlo a sección, ponemos sección desc, precio.

🡪 SELECT \* FROM PRODUCTOS WHERE SECCIÓN='DEPORTES' OR SECCIÓN='CERÁMICA' ORDER BY SECCIÓN, PRECIO DESC



**ORDER BY. EJERCICIOS PÁGINA 17**

**1.Realizar una consulta que muestre aquellos productos españoles cuya fecha**

**esté comprendida entre febrero y mayo del 2002. Ordenar los resultados**

**por el campo “Nombre artículo” descendentemente.**

SELECT

\* todos los campos

PRODUCTOS

Condiciones:

BETWEEN Febrero y Mayo del 2002

PAÍSDEORÍGEN="ESPAÑA"

Ordenar NOMBREARTÍCULO DESC

MySQL

SELECT \* FROM PRODUCTOS

WHERE PAÍSDEORIGEN='ESPAÑA' AND FECHA BETWEEN '2002-02-01' AND '2002-05-31' ORDER BY NOMBREARTÍCULO DESC

(Otra Condición posible: WHERE FECHA>='2002-02-01' AND FECHA<='2002-05-31')

ACCESS

SELECT \* FROM PRODUCTOS WHERE PAÍSDEORIGEN='ESPAÑA' AND FECHA BETWEEN #02-01-2002# AND #05-31-2002# ORDER BY NOMBREARTÍCULO DESC

**2. Realizar una consulta que muestre los campos “Empresa”, “Dirección” y “Población” de la tabla “Clientes”. Ordenar la consulta por el campo “Población” ascendentemente y por “Dirección” descendentemente.**

SELECT EMPRESA, DIRECCIÓN, POBLACIÓN FROM CLIENTES ORDER BY POBLACIÓN, DIRECCIÓN DESC

**3. Realizar una consulta que muestre todos los campos de la tabla “Clientes”.**

**Ordenar los resultados por el campo “Población” ascendentemente.**

SELECT

TODOS LOS CAMPOS \*

TABLA CLIENTES

POBLACIÓN ASC

SELECT \* FROM CLIENTES ORDER BY POBLACIÓN

Funciones de agregado

AVG 🡪 Calcula el promedio de un campo

COUNT 🡪 Cuenta los registros de un campo

SUM 🡪 Suma los valores de un campo

MAX 🡪 Devuelve el máximo de un campo

MIN 🡪 Devuelve el mínimo de un campo

Ejercicios página 19. **Funciones de agregado.**

1. **Realizar una consulta sobre la tabla “Clientes” que muestre los campos “Dirección”, Teléfono” y “Población”. Este útlimo debe aparecer en la consulta con el nombre de “Residencia”. Los registros aparecerán ordenados descendentemente por el campo “población”.** 
   1. Para que aparezca con un nombre diferente habrá que poner AS, de alias.

My SQL:

SELECT DIRECCIÓN, TELÉFONO, POBLACIÓN AS RESIDENCIA, FROM CLIENTES ORDER BY RESIDENCIA DESC

ACCES:

SELECT DIRECCIÓN, TELÉFONO, POBLACIÓN AS RESIDENCIA, FROM CLIENTES ORDER BY POBLACIÓN DESC;

Es el mismo código lo único que lo que cambie es que no coge el alias directamente, habrá que poner el original en el caso de acces.

1. **Realizar una consulta que muestre que poblaciones hay en la tabla “Clientes”.** 
   1. Con la **clausula GROUP BY,** podremos evitar el order by y solo mostrar aquellas poblaciones o aquellos atributos que no estén repetidos.
   2. Si lo quiero especificar primerio aplico las condiciones y luego hago el group by. Si no agrupas no vas a poder coger un montón de registros del campo precio.

SELECT POBLACIÓN FROM `clientes`GROUP BY POBLACIÓN;

1. Realizar una consulta de agrupación que muestre la suma del precio de los artículos de todas las secciones. Mostrar en al consulta los campos sección y suma por sección.

SELECT SECCIÓN, SUM(PRECIO) AS SUMS\_SECCIÓN FROM PRODUCTOS GROUP BY SECCIÓN

Si no le pusiera el Group By, no tendría sentido porque sería la suma total. Entonces no tendría sentido que pusiera “sección” porque realmente no hay que mostrar el resto de los campos.

1. Realizar una consulta de agrupación que muestre la media del precio de todas las secciones menos de juguetería. En la consulta deberán aparecer los campos “Sección” y “Media por sección”.

SELECT SECCIÓN, AVG(PRECIO) AS MEDIA\_POR\_SECCIÓN FROM PRODUCTOS WHERE SECCIÓN<>'JUGUETERÍA' GROUP BY SECCIÓN ORDER BY AVG(PRECIO);(Añadimos ordenación que no nos lo pide)

SELECT SECCIÓN, AVG(PRECIO) AS MEDIA\_POR\_SECCIÓN FROM PRODUCTOS WHERE SECCIÓN <> `JUGUETERÍA` GROUP BY SECCIÓN

1. Realizar una consulta que muestre cuantos artículos hay de la sección “Deportes”.

SELECT \*, Count(CÓDIGOARTÍCULO), FROM PRODUCTOS

SELECT COUNT(CÓDIGOARTÍCULO) AS NºARTÍCULOS\_DEPORTES FROM PRODUCTOS WHERE SECCIÓN = 'DEPORTES'

SELECT SECCIÓN, COUNT(SECCIÓN) AS TOTAL\_ARTICULOS FROM PRODUCTOS WHERE SECCIÓN="DEPORTES" GROUP BY SECCIÓN

\*Si no muestro ninguna sección no es necesario agrupar. Pero si lo muestro si que hay que agrupar.

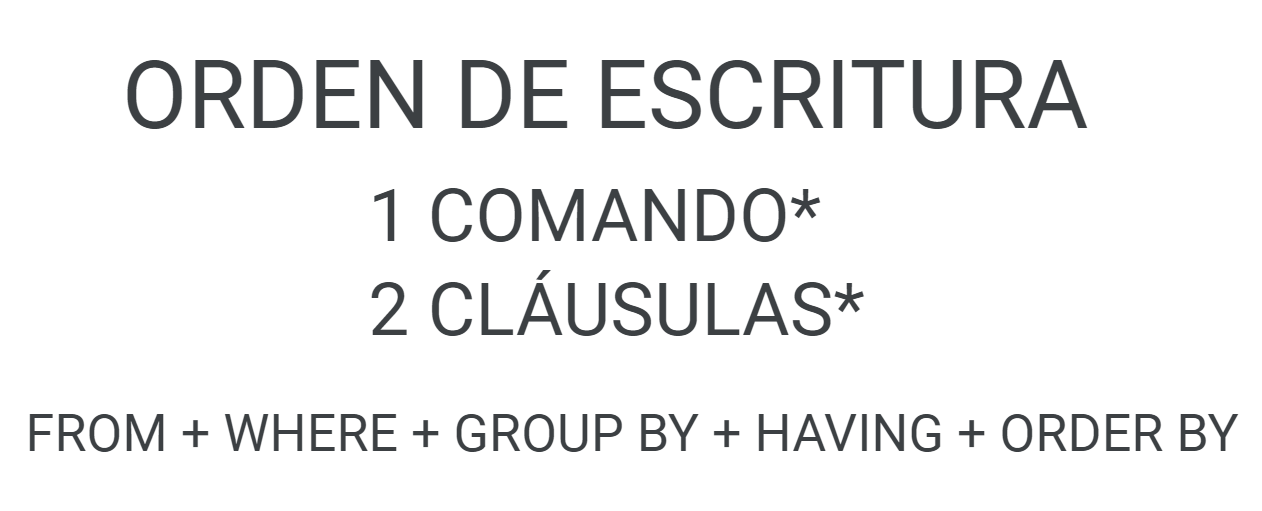
SELECT COUNT(CÓDIGOARTÍCULO) AS NºARTÍCULOS\_DEPORTES FROM PRODUCTOS WHERE SECCIÓN = 'DEPORTES'

Where – Having

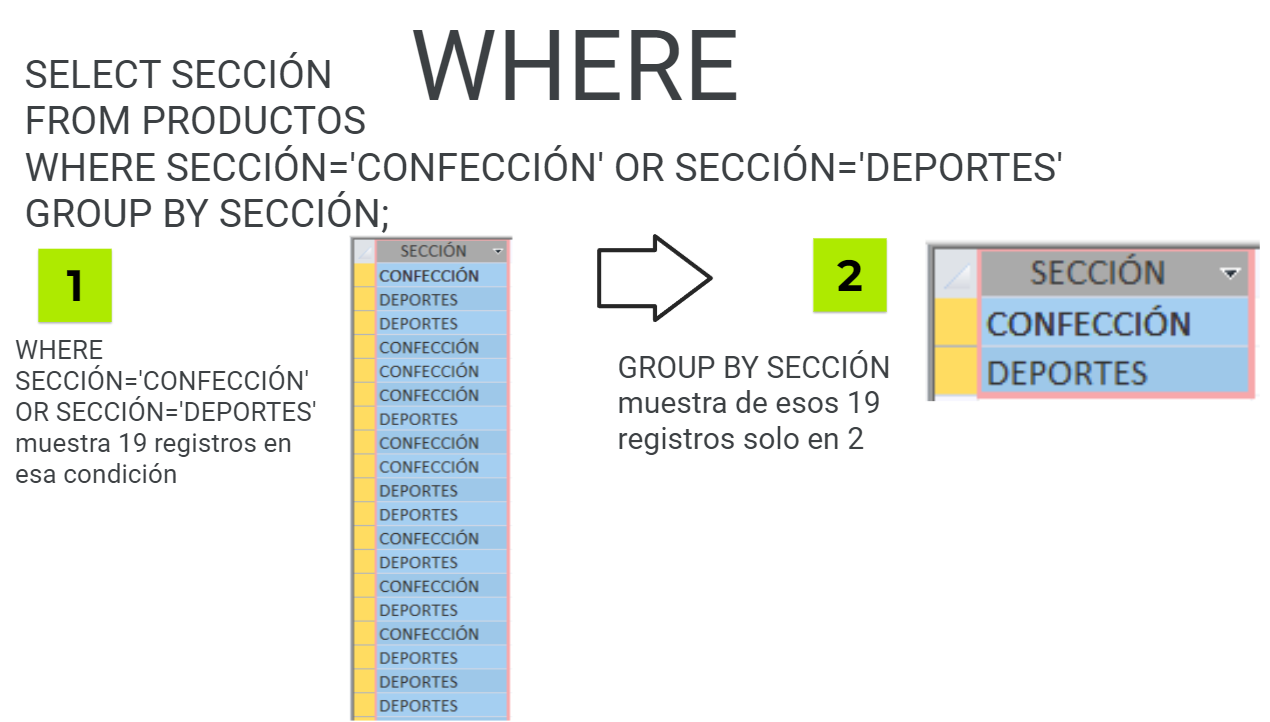
Tanto el having como el where: son condiciones. Es una cuestión de efectividad. Depende de hacerlo antes o después. Es por una cuestión de velocidad. Técnicamente hacen lo mismo. Uno va antes o después de

**Falta TEMA HAVING I OPERADORES ANTERIORES**

**Alguns exemples dels exercicis:**

****

**WHERE**

****

****

**Si solo tenemos una sección, y agrupamos por secciones solo nos mostrará un registro.**

**Having**

****

**En este caso se utiliza having en el SQL para explicitar que se mostrarán aquellos elements que en la sección tenga Confección o bien Deportes. En el caso del Group By pasará solo a mostrar dos registros en esa condición de los 6 registros anteriores.**

****

**Diferencia entre el uso de WHERE y HAVING**

**SELECT SECCIÓN FROM PRODUCTOS WHERE SECCIÓN='CONFECCIÓN' OR SECCIÓN='DEPORTES' (19 REGISTROS)**

**SELECT SECCIÓN FROM PRODUCTOS WHERE SECCIÓN='CONFECCIÓN' OR SECCIÓN='DEPORTES' GROUP BY SECCIÓN (2 REGISTROS)**

**SELECT SECCIÓN FROM PRODUCTOS GROUP BY SECCIÓN (6 REGISTROS)**

**SELECT SECCIÓN FROM PRODUCTOS GROUP BY SECCIÓN HAVING SECCIÓN='CONFECCIÓN' OR SECCIÓN='DEPORTES' (2 REGISTROS)**

**SELECT SECCIÓN, MAX(PRECIO) AS MAXIMO FROM PRODUCTOS GROUP BY SECCIÓN HAVING SECCIÓN='CONFECCIÓN' OR SECCIÓN='DEPORTES'**

Consultas de cálculo

**Ejercicio 1. Cálculo**

**1. Realizar una consulta que visualice los campos NOMBRE ARTÍCULO, SECCIÓN, PRECIO de la tabla PRODUCTOS y un campo nuevo que nombramos con el texto “DESCUENTO\_7”. Debe mostrar el resultado de aplicar sobre el campo PRECIO un descuento de un 7 %. El formato del nuevo campo para debe aparecer con 2 lugares decimales.**

SELECT PRODUCTOS.NOMBREARTÍCULO, PRODUCTOS.SECCIÓN, PRODUCTOS.PRECIO, PRECIO\*0.07 AS DESCUENTO\_7, Round(PRECIO-PRECIO\*0.07) AS PRECIO\_FINAL\_DESC

**FROM PRODUCTOS;**

**Ejercicio 2.**

**Realizar una consulta visualizando los campos FECHA, SECCIÓN, NOMBRE ARTÍCULO y PRECIO de la tabla PRODUCTOS y un campo nuevo que nombramos con el texto “DTO2 €\_EN\_CERÁMICA”. Debe mostrar el resultado de aplicar sobre el campo PRECIO la resta de 2 € sólo a los artículos de la sección CERÁMICA. El formato del nuevo campo debe aparecer con 2 lugares decimales. Ordenar el resultado de la consulta por el campo FECHA descendente.álculo**

SELECT FECHA, SECCIÓN, NOMBREARTÍCULO, ROUND(PRECIO-2, 2) AS DTO2€\_EN\_CERÁMICA FROM PRODUCTOS WHERE SECCIÓN="CERÁMICA" ORDER BY FECHA DESC

**Ejercicio 3. Cálculo.** **Realizar una consulta visualizando los campos NOMBRE ARTÍCULO, SECCIÓN, PRECIO de la tabla PRODUCTOS y un campo nuevo que nombramos con el texto “PRECIO\_AUMENTADO\_EN\_2”. Debe mostrar el PRECIO con un incremento de un 2% del PRECIO. Sólo debemos tener en cuenta los registros de la sección FERRETERÍA. El nuevo campo debe aparecer en Euros y con 2 lugares decimales.**

SELECT NOMBREARTÍCULO, SECCIÓN, PRECIO, ROUND(PRECIO+PRECIO \* 0.02,2)AS PRECIO\_AUMENTADO\_EN\_2 FROM PRODUCTOS WHERE SECCIÓN='FERRETERÍA';

**Ejercicio 4. Cálculo (ejercicio extra\_estava en discord)**

**Muestra NOMBREARTÍCULO, SECCIÓN, PRECIO de la tabla PRODUCTOS solo de las secciones deportes y cerámica y ordena primero por sección y luego por precio. Haz un incremento de +10 unidades que se llame SUMA10 y de +10% que se llame INCREMENTO10 en dos campos extra.**

SELECT NOMBREARTÍCULO, SECCIÓN, PRECIO, PRECIO+10 AS SUMA10, PRECIO\*1.10 AS INCREMENTO10 FROM PRODUCTOS WHERE SECCIÓN='DEPORTES' OR 'CERÁMICA' ORDER BY SECCIÓN, PRECIO

**Consola SQL**

Cuando existía el MS2, había que irse moviendo por lso archivos para modiificarlos, etc. Para movernos por nuestro sistema operativo. La interfaz que nuestro Windows pero mediante comandos. Da instrucciones. Nos conectaremos al terminal y podrá dar instrucciones. Creará un base de datos en PHP My Admin, pero no es el teminal quien la tiene.

Aún así, se puede abrir, crear, y gestionar una base de datos. Se pueden ver los datos introducidos. Que datos no.

Requisitos

* phpMyAdmin(instalación con XAMPP) Maria DB(es el nombre comercial).
* Terminal o consola de Windows
* Consultas:
  + Create Databases
  + Create Table
  + Select
  + Insert into
  + Update
* Otras instrucciones.

Si no tenemos phpMyAdmin no se puede hacer.

El terminal sería lo equivalente a la pestaña de MySQL que aparece en PHP My Admin.

PHP My Admin

* Instalación local con XAMPP
* User: root (sin password)
* Para crear Base de datos, tablas,
* Insertar registros, campos…
* Debemos tener XAMPP encendido con los Servicios:
  + Aparche
  + MySQL
* Que funcione correctamente el navegador con localhost/phomyadmin

Terminal o consolda de Windows

\*Pasos necesarios para acceder a la Base de datos desde CONSOLA (o también llamado “Terminald e Windows” o “Símbolo del Sistema”).

* Nos situamos en el directorio de XAMPP:

C:\xampp\mysql\bin>

* En el explorar de archivos escribimos “cmd” y se abrirá una ventana

similar al terminal, pero con la dirección correcta

\*Con el cd, podemos escoger el sitio del directorio al cual queramos acceder. Si le damos DIR aparecerán todos los directorios posibles.

Consultas (Creación DB y tabla)

C:\xampp\mysql\bin>-u

"-u" no se reconoce como un comando interno o externo,

programa o archivo por lotes ejecutable.

C:\xampp\mysql\bin>mysql -u root -p

Enter password:

Welcome to the MariaDB monitor. Commands end with ; or \g.

Your MariaDB connection id is 505

Server version: 10.4.24-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>

Consultas (Creación DB)

INSTRUCCIONES:

* Show databases:
* Create database prueba\_consola;
* Use prueba\_consola;

Creación tabla “usuarios”

Create table usuarios

(id INT AUTO\_INCREMENT,

Nombre VARCHAR (100) NOT NULL,

Email VARCHAR(100) NOT NULL,

PRIMARY KEY, (id));

\*Cuando hablamos de campos obligatorios hablamos de campos NO NULL\*

Consultas

SELECT \* FROM usuarios; //No surt res

INSERT INTO usuarios VALUES (null, 'Marc', 'marc@pr0j3ct.com’);

SELECT \* FROM usuarios;

INSERT INTO usuarios (id,nombre) VALUES (null, 'pepito’);

SELECT \* FROM usuarios;

En el último caso lo que hace es que pasa al siguiente ya que

CRUD en SQL:

Nuestro PHP ejecutá consultas CRUD, que son las siglas.

C 🡪 Create (CREATE) + INSERT INTO

R 🡪 Read (SELECT)

U 🡪 Update (UPDATE)

D 🡪 Delete (DELETE)

\*Habría que ir con cuidado con los acentos. Hay algunos UTF que no lo aceptan.

\*Null es que tiene un tipo de valor nulo, no tiene ni tipo ni valor, pero no está vacío.

INSERT INTO (insertar registros)

INSERT INTO usuarios VALUES (null,’valor2’,’valor3’);

INSERT INTO usuarios (id, nombre) VALUES (null, ‘valor2’);

INSERT INTO tabla (campo1, campo2) VALUES (‘valor1’, ‘valor2’).

Consulta de actualización UPDATE

**UPDATE usuarios SET nombre=’Manuel’ WHERE id=3;**

UPDATE usuarios SET nombre='Manuel' WHERE id=3;

//No puc fer perquè no existeix

INSERT INTO usuarios (id,nombre) VALUES (null, 'juanito’);

SELECT \* FROM usuarios;

UPDATE usuarios SET nombre='Manuel' WHERE id=3;

SELECT \* FROM usuarios;

UPDATE usuarios SET email='manuel@correo.com' WHERE id=3;

SELECT \* FROM usuarios;

Consultas ELIMINAR

SELECT \* FROM usuarios;

DELETE para eliminar registros, contrario al INSERT INTO:

DELETE FROM usuarios WHERE id = 2;

DROP sirve para eliminar tablas o bases de datos, es lo contrario al CREATE

DROP TABLE usuarios;

DROP DATABASE prueba\_consola;

CTRL + C //exit

// Lo que hacemos cuando le damos a UPDATE productos SET nombre\_articulo=’table’ WHERE código\_articulo=3;

Lo que hacemos es que **actualiza** la tabla productos dando valor al nombre\_articulo igual a tablet, pero la condición es que el codigo\_articulo es igual a 3, solo cambia este registro

**17. Subconsultas**

Subconsultas: Representa una consulta dentro de otra consulta

* Escalonadas: Tendremos un criterio respecto a la subconsulta.
* SELECT AVG(edad) FROM Persona

En este caso estaría calculando la edad media del campo edad de la tabla persona.

\*Posteriormente esto acaba derivando en la consulta:

Select Nombre, Apellidos

FROM Persona

WHERE Edad < (

SELECT AVG(edad)

FROM Persona

)

* De lista

El resultado de esa subconsulta será una lista de precios. El select hará referencia a un hijo que será más de un valor en concreto. No solo un registro.

* Correlacionada

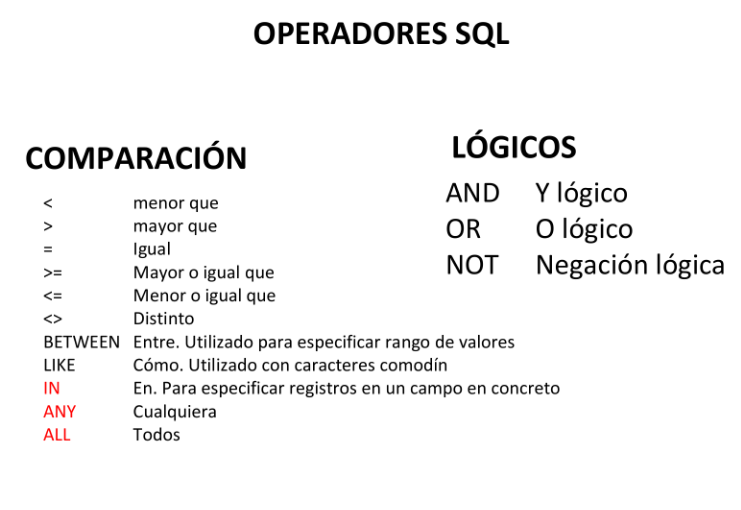
Vamos a establecer un SELECT hijo. Mediante una lista, podremos hacer el filtro de otra consulta.

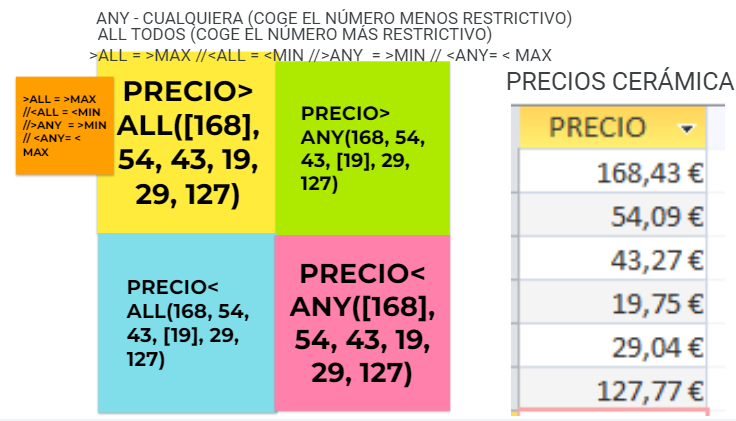
**En el caso de las subconsultas, al final estará entre paréntesis el “hijo” que es el criterio o filtro.**

Veremos tres operadores nuevos que son el IN, ANY, ALL: \*Hay un Jambord que explica lo de los 3. A veces es algo complicado de entender.

* Cuando hablamos de ANY: Hablamos de cualquier de ellos. Es decir, cualquier que esté por debajo del número que estamos buscando, porejemplo. Normalmente el ANY compara el menor.
* Cuando hablamos de ALL: Siempre serás más excluyente. Si es mayor que cualquiera de ellos si que lo cumpliría.
* Cuando nos referimos a IN: Estaos hablando de aquellos elementos o registros que se encuentra en un campo en concreto.

Estos tres elementos se suman a los operadores de comparación y lógicos de las consultas.







Esta sintaxis sí que es la correcta.

Subconsulta – Escalonadas. Página 5. PPT.

**Ejercicio 1 (primer ejercicio):**

**Buscar la media de precios de Ferretería y muestra los precios que sean menores que esa media.**

Ejercicio resulto.

SELECT \* FROM PRODUCTOS WHERE PRECIO < (SELECT AVG(PRECIO) FROM PRODUCTOS WHERE SECCIÓN='FERRETERÍA')

**Ejercicio 2:**

**Muestra los artículos de confección que están por encima de la media de precios de juguetería.**

\*1)Media de precios de jugueteria:

SELECT AVG(PRECIO) FROM PRODUCTOS WHERE SECCIÓN='JUGUETERÍA';

Muestra los artículos de confección mayores que (\*1):

SELECT NOMBREARTÍCULO, PRECIO FROM PRODUCTOS WHERE SECCIÓN='CONFECCIÓN' AND PRECIO > (mediajuegueteria \*1);

**Resultado final:**

SELECT NOMBREARTÍCULO, PRECIO FROM PRODUCTOS WHERE SECCIÓN='CONFECCIÓN' AND PRECIO>(SELECT AVG(PRECIO) FROM PRODUCTOS WHERE SECCIÓN='JUGUETERÍA');

Subconsulta de lista

**Ejercicio 1 (el primer ejercicio de lista está en la página 5 del PPT):**

**Busca los precios de Juguetería y muestra TODOS los precios de Marruecos que sean menores que esa lista de precios de Juguetería (luego muestra cualquier sea menor).**

SELECT PRECIO FROM PRODUCTOS WHERE SECCIÓN=’JUGUETERÍA’

Esto son todos los precios de juguetería.

Si no especifica cuales tiene que son los nombres, hay que poner asterisco.

SELECT \* FROM PRODUCTOS WHERE PAÍSDEORIGEN=’MARRUECOS’ AND PRECIO<ALL (SELECT PRECIO FROM PRODUCTOS WHERE SECCIÓN=’JUGUETERÍA’);

**Ejercicio 2:**

Muéstrame los artículos del 2000 que tengan un precio por debajo de todos los artículos de deportes (luego con cualquiera). (Si estoy comparando el precio en la subconsulta) estaré comparando el precio. Tienes que compararlo con el mismo tipo, porque si no no te dará un resultado correcto.

SELECT PRECIO FROM PRODUCTOS WHERE SECCIÓN=’DEPORTES’

SELECT \* FROM PRODUCTOS WHERE FECHA BETWEEN #01-01-2000# AND #12-31-2000# AND PRECIO<ALL (SELECT \* FROM PRODUCTOS WHERE SECCIÓN=’DEPORTES’).

---- Si tuviéramos que hacer la consulta en MySQL – PHP My Admin ----

SELECT \* FROM PRODUCTOS WHERE FECHA BETWEEN “2000-01-01” AND “2000-12-31” AND PRECIO<ALL (SELECT \* FROM PRODUCTOS WHERE SECCIÓN=’DEPORTES’).

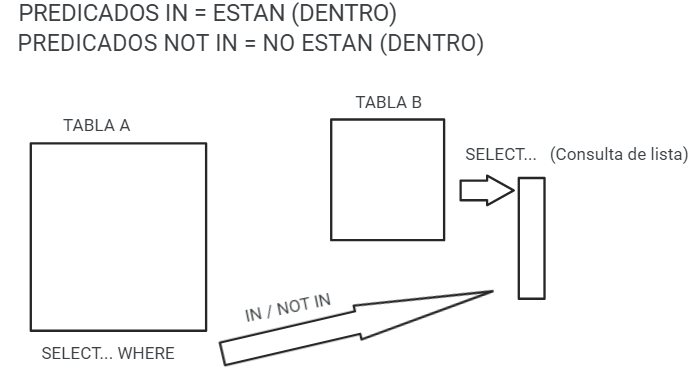
\*L’únic que ha canviat són les cometes.

Relaciones

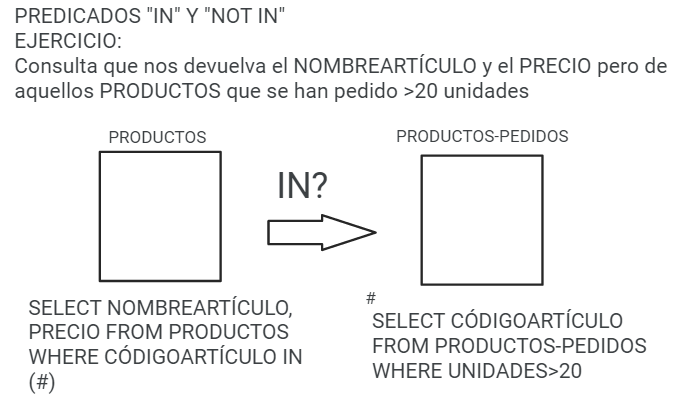
Una base de datos es un conjunto de datos relacionados. Esas relaciones no se pueden romper. Entonces para evitar que se produzcan hay que llevar a cabo las restricciones.

Predicados IN / NOT IN

En a tabla A hacemos un condicional, una búsqueda, una lectura 🡪DE si esta en una lista de la Tabla B. Es lo que hacíamos antes pero en otra tabla.



* 1 Consultas: Consulta que nos devuelva el NOMBREARTÍCULO y el PRECIO pero de aquellos PRODUCTOS que se han pedido >20 unidades



Hay que poner código artículo que es la que se compara:

**Ejercicio 1:**

**Consulta que nos devuelva el NOMBREARTÍCULO y el PRECIO pero de aquellos PRODUCTOS que se han pedido >20 unidades**

SELECT NOMBREARTÍCULO, PRECIO FROM PRODUCTOS WHERE CÓDIGOARTÍCULO IN (SELECT ‘CÓDIGO ARTÍCULO’ FROM ‘PRODUCTOS- PEDIDOS’ WHERE UNIDADES>20);

Sería lo mismo sino colocamos IN en vez de colocar OUT.

**Ejercicio 2:**

**Muestra empresas, población de la tabla clientes solo los pedidos que no se han pagado con tarjeta.**

SELECT EMPRESA, POBLACIÓN FROM CLIENTES WHERE CÓDIGOARTÍCULO IN (SELECT `NÚMERO DE PEDIDO` FROM PEDIDOS WHERE `FORMA DE PAGO` = TARJETA);

<>🡪 TAMBIÉN SE PODRÍA HABER PUESTO ese símbolo que indica diferencia entre uno y el otro.

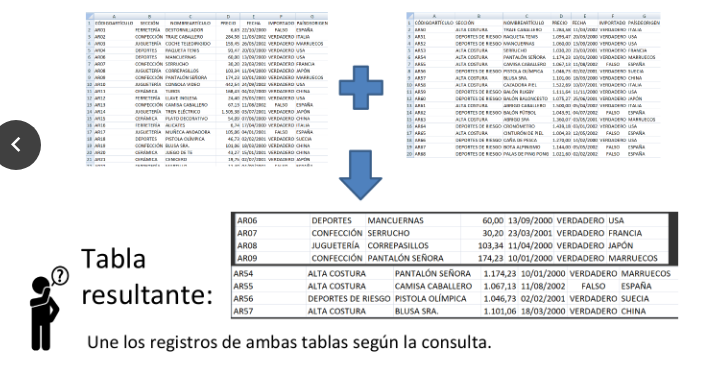
El examen llegará hasta subconsultas de listas.

**Union y Joins SQL**



UNION / UNION ALL (Unión externa)

* Unión de dos (o+) tablas = número de campos y datos compatibles.
* Cuando hablamos de unión: no archivos repetidos
* Union All: Archivos repetidos también.



\*Pueden tener campos con nombres diferentes, la consulta toma el nombre de los campos de la primera tabla. Se pueden unir más de dos tablas.

Consulta para unir por completo dos tablas:

SELECT \* FROM PRODUCTOS

UNION

SELECT \* FROM PRODUCTOSNUEVOS

1. Hemos añadido de manera manual en acces una fila.
2. Hemos añadido esa fila con el código en PHPMyAdmin – MySQL:

INSERT INTO ProductosNuevos VALUES ('AR01','FERRETERÍA','DESTORNILLADOR',6.6280,'#22/10/2000#',FALSE,'ESPAÑA');

Nos hemos equivocado y hemos tenido que actualizar una fila: SELECT \* FROM `productosnuevos` WHERE 1

**Ejercicios de práctica – Consultas Union**

**\*Para poder mostrar solo los campos que se piden se lo ponemos a la primera\***

**Ejercicio 1: Realizar una consulta que visualice los registros de la tabla PRODUCTOS con un precio entre 200 y 500 y además los registros de la tabla PRODUCTOSNUEVOS donde el precio sea superior a 1100.**

* Acces:

**Ejercicio 2**: Realizar una consulta visualizando los campos FECHA, SECCIÓN, NOMBRE ARTÍCULO y PRECIO de la tabla PRODUCTOS sólo a los artículos de la sección CERÁMICA. También que se vean los productos de ALTA COSTURA de la tabla PRODUCTOSNUEVOS.

* Acces:

SELECT FECHA, SECCIÓN, NOMBREARTÍCULO, PRECIO FROM PRODUCTOS WHERE SECCIÓN='CERÁMICA'

UNION

SELECT FECHA, SECCIÓN, PRECIO, NOMBREARTÍCULO FROM PRODUCTOSNUEVOS WHERE SECCIÓN='ALTA COSTURA';

* PHP My Admin:

**Ejercicio 3: Une las dos tablas mostrando solo los productos del año 2001. Da igual si hay campos repetidos, muéstralos.**

\*Para que la fecha esté en un formato correcto lo que hay que hacer un cambio en Excel. En el valor del campo.

\*Estándar FECHA aaaa-mm-dd.

\*Las tablas hay que importarlas en ODS.

* Acces:

SELECT \* FROM PRODUCTOS WHERE FECHA BETWEEN #01-01-2001# AND #12-31-2001#

UNION ALL

SELECT \* FROM PRODUCTOSNUEVOS WHERE FECHA BETWEEN #01-01-2001# AND #12-31-2001#

\*En este caso al decirnos que mostrarnos también los que puedan estar repetidos vamos a poner UNION ALL en vez de UNION.

INNER / LEFT /RIGHT JOIN (Unión Interna)

La consulta une los campos de dos tablas que contienen relación



Refleja la información común en ambas tablas.

* Inner Join
* Left Join
* Right Join





Ejercicio ProductosNuevos2

1. Copiamos el Excel de productos, el primero y borramos los valores y el campo foto.
2. Una vez eso ponemos bien el campo fecha y guardamos.
3. Lo podemos importar en Administración
4. En insertar ponemos en CÓDIGOARTÍCULO, un valor de AR69
   1. En SECCIÓN DEPORTES
   2. En NOMBREARTÍCULO BICICLETA
   3. En PRECIO: 1500
   4. En FECHA
   5. En IMPORTADO FALSO
   6. En PAÍSDEORIGEN HAWAII
5. La consulta tenía que ser un UNION de todos.
   1. Los UNIONS muestran solo los que se repiten.
   2. Se van a unir las siguientes tablas:
      1. PRODUCTOSNUEVOS2
      2. PRODUCTOSNUEVOS
      3. PRODUCTOS

\*Para modificar los registros, está en examinar de la tabla.

🡪 Asegurarse que en estructura uno sea la clave primaria. En examinar ya podremos editar.

* Es importante poner claves primarias.

INNER / LEFT /RIGHT JOIN (Unión Interna)

La consulta une los campos de dos tablas que contienen relación



Refleja la información común en ambas tablas.

* Inner Join
* Left Join
* Right Join



Cuando unes dos tables, es la unión de información entra las dos. Lo que tiene de información común esas dos tablas.

En esa relación tendremos también un Left Join o un Right Join en función de lo que queda en la parte izquierda o deerrecha del espacio del medio de las dos circunferencias.



Aquí tenemos la unión de elementos que son en común y no. LA unión no se va hacer por registros si no por campos

Ejercicios INNER JOIN

**EJERCICIO 1:** Queremos saber los clientes que han hecho pedidos de Madrid.

**Como se prepara un INNER JOIN**

1. Que campos van a estar involucrados?
   1. Como no nos han dicho nada: pues vamos a poner todos los campos 🡪

SELECT \*

1. Cuando hacemos el from, le vamos a poner las dos tablas separadas por un inner join (las dos tablas que quieres que estén) 🡪 FROM CLIENTES INNER JOIN PEDIDOS
2. Le añadiremos el on (cuando hayan de la tabla clientes más un punto seguido del campo de la tabla clientes (CÓDIGOCLIENTE).
3. Después esto cuando será =a la tabla [código cliente] de la tabla.
4. Al final pondremos la condición más sencilla que es WHERE POBLACIÓN=’MADRID’

SELECT \* FROM CLIENTES INNER JOIN PEDIDOS ON CLIENTES.CÓDIGOCLIENTE=PEDIDOS.[CÓDIGO CLIENTE] WHERE POBLACIÓN=’MADRID’

\*Se podrían escoger una serie de campos en vez de mostrarlos todos como se hará en el siguiente ejercicio:

**EJERCICIO 2:** Queremos mostrar los clientes con pedidos de Madrid solo los siguientes campos CÓDIGO CLIENTE (de la tabla Clientes), POBLACIÓN, DIRECCIÓN, NÚMERODEPEDIDO, CÓDIGO CLIENTE (de la tabla Pedidos) y FORMADEPAGO

SELECT CLIENTES.CÓDIGOCLIENTE, POBLACIÓN, DIRECCIÓN,`NÚMERO DE PEDIDO`, PEDIDOS.`CÓDIGO CLIENTE`, `FORMA DE PAGO` FROM CLIENTES INNER JOIN PEDIDOS ON CLIENTES.CÓDIGOCLIENTE = PEDIDOS.[CÓDIGO CLIENTE] WHERE POBLACIÓN='MADRID'

**EJERCICIO 3: Muestra los clientes que han hecho pedidos de “ACoruña. Muestra solo los campos Código Cliente de la tabla pedidos), el nombre de la empresa, el teléfono y la forma de pago.**

SELECT PEDIDOS. [CÓDIGO CLIENTE], EMPRESA, TELÉFONO, [FORMA DE PAGO] FROM CLIENTES INNER JOIN PEDIDOS ON CLIENTES.CÓDIGOCLIENTE = PEDIDOS. [CÓDIGO CLIENTE] WHERE POBLACIÓN='A CORUÑA';

**EJERCICIO 4:**

SELECT NOMBREARTÍCULO, PRECIO, UNIDADES FROM PRODUCTOS INNER JOIN [PRODUCTOS-PEDIDOS] ON [PRODUCTOS-PEDIDOS].[CÓDIGO ARTÍCULO]=PRODUCTOS.CÓDIGOARTÍCULO WHERE UNIDADES>20

Da igual el orden que se ponga después de ON, va a dar lo mismo.

\*Las instrucciones son completa: no se ejecutan por partes.

🡪 Al final nos vamos a una instrucción que se ejecuta de manera completa. Aunque para entender se hace por separado, la instrucción hay que mirarla desde la totalidad.

Cuando estamos uniendo los elementos que no tienen en común. En el caso de clientes y pedidos, nunca va a ver pedidos que no tengan clientes.

**Probar ejercicio con un LEFT JOIN:**

SELECT CLIENTES.CÓDIGOCLIENTE, POBLACIÓN, DIRECCIÓN, [NÚMERO DE PEDIDO], PEDIDOS.[CÓDIGO CLIENTE], [FORMA DE PAGO] FROM CLIENTES LEFT JOIN PEDIDOS ON CLIENTES.CÓDIGOCLIENTE=PEDIDOS.[CÓDIGO CLIENTE] WHERE PEDIDOS.[CÓDIGO CLIENTE] IS NULL

(Los clientes que no han hecho pedidos)

**Probar ejercicio con un RIGHTJOIN:**

**Ejercicio: Subconsulta:**

SELECT NOMBREARTÍCULO, PRECIO FROM PRODUCTOS WHERE CÓDIGOARTÍCULO IN (SELECT [CÓDIGO ARTÍCULO] FROM [ORIDYCTIS . OEDUDIS] WHERE UNIDADES > 20)

**Inner Join**

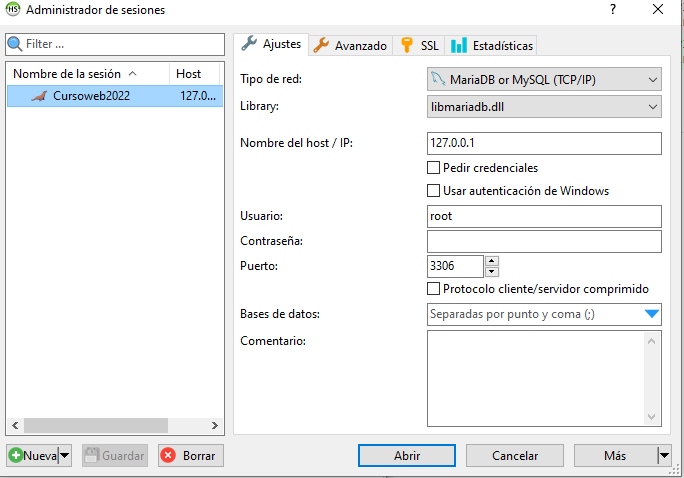
SELECT NOMBREARTÍCULO, PRECIO FROM PRODUCTOS INNER JOIN [PRODUCTOS- PEDIDOS] ON PRODUCTOS.CÓDIGOARTÍCULO? [PRODUCTOS- PEDIDOS].[CÓDIGO ARTÍCULO] WHERE UNIDADES > 20

**DISTINCT**

**Conexión PHP con MySQL**

**Se necesitan algunos elementos para hacer la conexión. Como el nombre de usuario, la dirección, etc. El index mejor guardarlo en una carpeta que no tenga el index. Si lo cogemos por carpetas o directorios los tenemos por separados.**

Descarga de los archivos de Google ClassRoom. Descarga de Heidi y conexión con la base de datos.



MYSQLi\_CREATE

Con este archivo PHP podemos crear una base de datos con el new mysqli. Cuando le damos desde el localhost podemos crear el archivo.

MySQLi\_escribir-datos

//21-07\_2022

Se ha hecho escribir datos y escribir datos 2. (mirar github)

Se hace bien la conexión poniendo el nombre de la base de datos correctamente.

* En el caso de que queramos escoger el nombre de la tabla se pone

MySQL\_leer-datos

\*$statement puede ser una de las variables válidas, pero también se puede poner como $sql, u otras similares. Lo importante aquí es la instrucción que se pone después.

Aquí lo que estamos haciendo

* Hacemos una conexión a la base de datos
* Nada más haber generado la conexión establece un condicional que es life
* Cuando tenemos un fetchassociation. Lo que nos dice es muéstrame el resultado de saltar a la siguiente fila, pero solamente el nombre.
  + Cada vez que haces ese fetch\_assoc estás haciendo un salto.

Ejercicio :

Que tenemos que cambiar para que se muestre la tabla:

Mysqli\_delete

Es otra de las instrucciones CRUD. Nos sirve para eliminar tablas.

* En la conexión de connm hay que poner el nombre del servidor que es localhost, el nombre de usuario que es root, contraseña no tenemos así que se pondrá como un string vacío y luego el nombre de la base de datos “prueba\_mysqli”.
* Te conectas a la base de datos realizando esa conexión y luego se hace un mysqli\_close($conn); para eliminar una de las tablas.

Mysqli\_prepared-statements.php

* Primero hacemos la conexión
* Prepared statements: Lo que hacemos es como si estuviéramos enviando un formulario, pero realmente lo que hacemos es un envío a través del método GET.

    if(isset($\_GET['nombre']) && isset($\_GET['edad'])){

        $nombre = $\_GET['nombre'];

        $edad = $\_GET['edad'];

Por eso lo estamos haciendo a través del navegador:

$statement->bind\_param('ssi', $ID, $nombre, $edad);

    $ID = NULL;

En el bind\_param estamos escogiendo los valores.

Si solo quisiéramos poner dos habría que modificar el SSI y solo poner un S i un I. En nuestro caso hemos creado una columna más que es un string y por lo tanto podemos dejar el mimos que no cambia nada.

* No podemos poner el ID en VSC como un Integer porque el nulo solo podría ser 0 y no se haría el autoincremento.

Si se envía por post no se visualiza el navegador. El get tiene doble sentido, Desde el formulario lo puedes enviar a la variable y desde la variable lo puedes enviar al navegador.

**Resum de la tornada al setembre**

A la diferencia con Xampp, sí que ejecuta un servidor y una base de datos. A través de Xampp generamos un servidor local. Ese servidor nos sirve para poder alojar archivos de PHP. Solamente a través de un servidor podemos visualizar esos archivos.

* Es el aplicativo que está instalado en este servidor de base de datos.

Alguien pone esa dirección y no irá a nuestro servidor va a ir al propio. Internet es precisamente eso la conexión entre varios ordenadores.

HeidiSQL se puede conectar a diferentes bases de datos.

Una consulta no es solo que vayas a buscar algo en un buscador. Si no que puedes modificarlo, puedes crear, creación lectura, actualización y modificación.

1. Mediante un programa PHP creamos el aplicativo.
   1. Le aplicamos filtros para hacerle pruebas de seguridad a los contenidos.
   2. En Correo y en Nombre. Comprobábamos y validábamos si era un número de teléfono que tiene 9 cifras.
2. Mediante el método POST / GET, hacíamos esa conexión.
3. A través de Heidi SQL podemos gestionar esa base de datos.
   1. A través de aquí es desde donde podemos crear la base de datos.

**Form\_db**

Análisis del archivo index view

<form action="<?php echo htmlspecialchars($\_SERVER['PHP\_SELF']); ?>" method="post">

            <input type="text" class="form-control" id="nombre" name="nombre" placeholder="Nombre:" value="<?php if(!$enviado && isset($nombre)) echo $nombre ?>">

* El método es post.
  + Hay método de envío.
* El action es donde lo envíamos.
* La vatrable global $\_SERVER guarda la dirección de donde va. Es la forma más segura de enviarlo. Es una variable array.
  + El input name es el nombre del campo que va a salir
  + Al colocar un PHP dentro de eso valor. Si no se ha enviado y tiene un nombre le vamos a colocar ese mismo nombre.
    - Si se ha enviado el nombre y está bien de formato se muestra en el campo.
* <?php if (!empty($errores)): ?>
* <div class="alert error">
* <?php echo $errores; ?>
* </div>
* <?php elseif($enviado): ?>
* <div class="alert success">
* <p>Enviado Correctamente</p>
* </div>
* <?php endif ?>

Si lo hemos enviado bien, saldrá otro recuadro que será, lo hemos enviado correctamente. Si todo está correcto y validado podemos enviarlo. Entonces enviaremos a la base de datos.

Análisis del archive index.php

if (isset($\_POST['submit'])) {

        $nombre = $\_POST['nombre'];

        $correo = $\_POST['correo'];

        $mensaje = $\_POST['mensaje'];

Si se ha enviado se van a cumplir con estas variables. Guarda estos array asociativos POST. Recopilamos los datos que nos dan a través del formulario.

//sanear (recibir textos y no nada más)

        if (!empty($nombre)) {

            $nombre = trim($nombre);

        } else {

            $errores .= 'Por favor escribe un nombre <br />';

Hay que asegurarnos de que, por ejemplo, en el caso de nombre se trata de un nombre y nada más. En el caso del nombre se le llama Trimar.

$errores .= 'Por favor escribe un nombre <br />'; // el .= representa la concatenación

En el caso de los errores se pueden ir concatenando. Además en el caso del <br /> nos puede servir para ahorrar espacio.

if(!$errores){

            $enviar\_a = 'tunombre@tuempresa.com';

            $asunto = 'Correo enviado desde miPagina.com';

            $mensaje\_preparado = "De: $nombre \n";

            $mensaje\_preparado .= "Correo: $correo \n";

            $mensaje\_preparado .= "Mensaje: " . $mensaje;

            //mail($enviar\_a, $asunto, $mensaje\_preparado);

            $enviado = 'true';

La función del mail, lo que hace es que guarda un correo, es a quien se lo envías. Un asunto, que es lo que va a encontrar la persona cuando recibe el correo y un mensaje preparado.

El final $enviado era igual a false o a vacío. El post recibía datos, pero no hacía un enviado. En el momento que no hay errores podemos hacer ese enviado.

Si no lo revisáramos de la base de datos eso se esfumaría. Por eso a través de la función mail podremos recibir los datos con lo que se ha completado el formulario.

Análisis de la parte “require” de “index.php”

Cuando nosotros queremos establecer una conexión con PDO lo utilizamos con este bucle condicional de try and catch.

try {

            $conexion = new PDO('mysql:host=localhost;dbname=form\_db', 'root', '');

La función de tipo PDO, genera un objeto de tipo conexión. Tenemos varios datos:

* Nombre de la base de datos
* Usuario de esa base de datos.
* Contraseña de esa base de datos.

//Insertar datos del formulario

            $statement = $conexion->prepare('INSERT INTO usuarios\_form VALUES (null, :nombre, :correo, :mensaje)');

En este caso ya tenemos la sentencia SQL.

$statement->execute(

                array(':nombre'=> $nombre, ':correo'=> $correo, ':mensaje'=> $mensaje)

            );

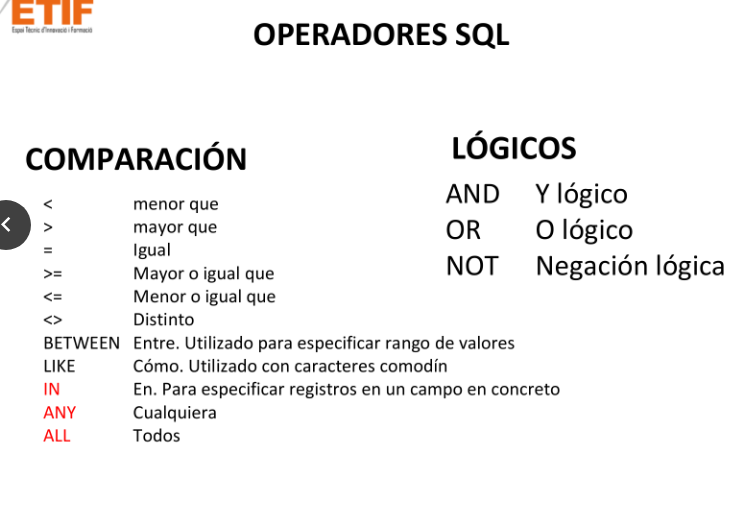
En este caso haríamos un insert dentro de la tabla “usuarios\_form”. El primero le ponemos NULL (porque el usuario no tiene que escribirla, somos nostros quienes hacemos la asignación”. Si no hacemos una consulta preparada, no podemos revisar que esos datos sean benignos o malignos. La función execute filtra que lo que recibes de esa variable se puedan enviar a la base de datos sin ningún tipo de problema.

Se hace por si a caso, para confirmar que hay errores de sintaxis. Divide la sentencia SQL de los datos que recibes.

Son **tres fases** en total. Lo que prima aquí es que la base de datos no deje de tener datos incorrectos.

* En la primera fase hay que establecer correctamente la conexión.
* En el segundo caso de revisa que la sentencia SQL es correcta para el envío de datos a la base de datos.
* Tercera Fase ¿???

SDASDASJDAISDD

* a
* asd
* a

F