

BASES DE DATOS I

Andrea Sofía Alfonsín
2020-2021
2º curso → 1º cuatrimestre

Tema-1

Modelo entidad-relación

El modelo E-R es una técnica de diseño de arriba a abajo.

El proceso de diseño

El proceso de diseño se divide en 4 fases

Fases

- **Fase inicial:** es la caracterización completa de las necesidades de los posibles usuarios de la BD. El resultado de esta fase es una especificación de requisitos del usuario.
 - **Fase de diseño conceptual:** se elige el modelo de datos y se traducen los requisitos en un esquema conceptual de la BD, que especifica las entidades, sus atributos, las relaciones y las restricciones. Esta fase da lugar a la creación de un diagrama e-r para tener una representación gráfica del esquema.
 - ↳ **diseño conceptual:** moderado de alto nivel independiente de cualquier modelo implementable.
 - **Fase de especificación de requisitos funcionales:** se describen los tipos de operaciones o transacciones, que se llevarán a cabo sobre los datos.
 - **Fase final:** se divide en dos partes
 - **Diseño lógico:** transformación del modelo conceptual abstracto en un modelo implementable concreto.
 - **Diseño físico:** especificación de las características físicas de la BD como la forma de organización de los archivos y las estructuras de almacenamiento interno.
 - El esquema físico es relativamente fácil de modificar, mientras que el lógico es más complicado.
- ***Transacción:** es la descripción de una operación a llevar a cabo sobre los datos almacenados en la BD. Puede suponer la creación, modificación, consulta o eliminación de los datos.
- Obtener un modelo para una BD no tiene una solución única, puede haber varios diseños completos y sin redundancia. Podemos decir que tiene mayor calidad aquel que restrinja al máximo las opciones que no se deberían poder realizar.

Al diseñar el esquema de una BD hay que evitar los **peligros**:

- **Redundancia:** es importante no repetir información para tener un buen diseño. El mayor problema de la información repetida es que al actualizar la información puede quedar alguna copia sin actualizar y se podría confundir todo.
- **Incompletitud:** puede hacer que sea muy difícil modelar algunos aspectos.

El modelo entidad-relación

Emplea 3 conceptos básicos, los conjuntos de entidades, los conjuntos de relaciones y los atributos. También tiene asociada una representación en forma de diagramas.

Conjunto de entidades: es un grupo de objetos con existencia física o conceptual (pueden ser objetos o ideas) suelen corresponder a sustantivos del lenguaje (lugares, personas, títulos...).

- La **extensión** de un conjunto de entidades se refiere a la colección real de entidades que pertenece al conjunto de entidades en cuestión.

Conjunto de relaciones: las relaciones moderan dependencias entre conjuntos de entidades, suelen corresponder a verbos del lenguaje (tener, comprar, asistir...).

- La asociación entre conjuntos de entidades se conoce como **participación**, es decir, los conjuntos de entidades E₁, E₂..., En participan en el conjunto de relaciones R.
- La función que desempeña una entidad en una relación se denomina **rol** de esa entidad. Los roles no se suelen especificar, pero resultan útiles cuando el significado de una relación necesita aclaración.

Atributos: son propiedades de las entidades y las relaciones de interés para la misión de la base de datos, sólo se incorporan a la BD aquellos atributos que sean necesarios.

- Para cada atributo hay un conjunto de valores permitidos denominados **dominio** o **conjunto de valores**.

Hay diferentes **tipos de atributos**:

→ **Simples y compuestos:** Los atributos compuestos se pueden dividir en subpartes, mientras que los simples no.

→ **Monovalorados y multivalorados;** Los atributos que tienen un único valor para cada entidad concreta se denominan monovalorados. También puede haber ocasiones en las que un atributo tenga un conjunto de valores para una entidad concreta, a estos atributos se les llama multivalorados.

↳ **Derivados:** son aquellos cuyo valor se puede obtener a partir del valor de otros atributos o entidades relacionados.

Los atributos tienen valores **nulos** cuando las entidades no tienen ningún valor para ese atributo (también se puede indicar "no aplicable").

Restricciones

Las restricciones determinan la semántica de la BD, es decir, su comportamiento. Una base de datos con las mismas conjuntos de entidades y relaciones será diferente si sus restricciones son diferentes.

La **correspondencia de cardinalidades** expresa el número de entidades a las que otra entidad se puede asociar mediante un conjunto de relaciones. Hay 4 tipos:

- **Uno a uno:** cada entidad de A se asocia a lo sumo con una entidad de B y viceversa.
- **Uno a varios:** cada entidad de A se asocia con cualquier número de entidades de B, pero cada entidad de B se puede asociar a lo sumo con una de A.
- **Varios a uno:** cada entidad de A se asocia a lo sumo con una de B, pero cada entidad de B se puede asociar con cualquier número de entidades de A.
- **Varios a varios:** cada entidad de A se asocia con cualquier número de entidades de B y viceversa.

Se dice que la participación de un conjunto de entidades E en un conjunto de relaciones R es **total** si cada entidad de E participa, al menos, en una relación de R.

Si solo algunas entidades de E participan en relaciones de R, se dice que la participación del conjunto E en R es **parcial**.

Clave primaria: todo conjunto de entidades debe tener una clave primaria. Una clave primaria es un conjunto mínimo de atributos que tienen valores diferentes para cada entidad en cada conjunto de entidades. La clave primaria debe mantener su condición de valores diferentes en el pasado (datos que estuvieron almacenados en la BD), en el presente (datos que están en la BD) y en el futuro (cualquier dato que se pueda almacenar en la BD). La estructura de las claves primarias de un conjunto de relaciones depende de la correspondencia de cardinalidad del conjunto.

Super clave: conjunto de atributos que diferencian a una entidad del resto.

Clave candidata: posible clave primaria.



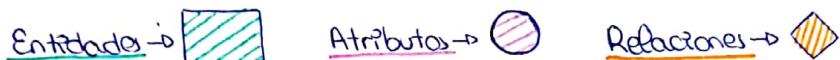
Atributos redundantes

Cada atributo que describe una propiedad debe estar solo una vez y en el conjunto de entidades o relaciones al que pertenece, si se necesita el atributo en otro lugar del modelo se obtiene a través de una relación, nunca repitiendo. Un diseño en modelo entidad-relación debe, desde el principio, realizarse sin redundancias.

Diagramas entidad-relación

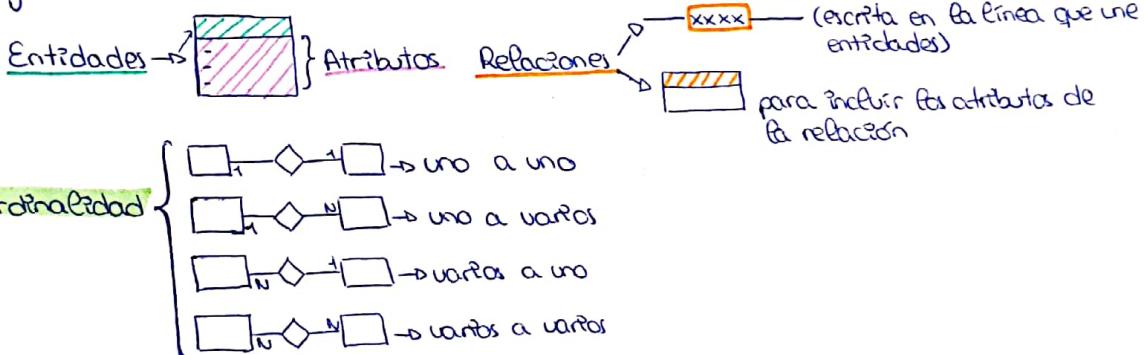
Hay diversas formas de representar un diagrama entidad-relación

1^ª forma



se unen mediante líneas en las que se representa la cardinalidad

2^ª forma



→ Ejemplo: una BD de una entidad educativa almacena materias y alumnos que están matriculados.



Modelo E-R extendido

Los conjuntos de entidades pueden incluir subgrupos de entidad que se diferencian de alguna forma de las demás entidades del conjunto. Las subclases tienen las mismas entidades que las superclases pero agrupadas en base al cumplimiento de alguna característica. Este proceso de establecimiento de subgrupos dentro del conjunto de entidades se denominará **especialización**.

Se llama **especialización simple** si permite varios conjuntos, y **especialización disjunta** si se permite uno como mucho.

El proceso de diseño puede ser ascendente o descendente.

- **Descendente:** es el refinamiento a partir del conjunto de entidades inicial en sucesivos niveles de subgrupos de entidades.
- **Ascendente:** varios conjuntos de entidades se sintetizan en un conjunto de entidades de nivel superior basado en características comunes.

La **generalización** es una relación de contención que existe entre el conjunto de entidades de nivel superior y uno o varios de nivel inferior.

Herencia de los atributos: se dice que los atributos de los conjuntos de entidades de nivel superior son heredados por los conjuntos de entidades de nivel inferior. Las subclases también heredan la participación en las conjuntos de relaciones en los que participa su superclase.

- **Jerarquía:** un conjunto de entidades dado sólo puede estar implicado como conjunto de entidades de nivel inferior en una relación, es decir, tienen **herencia única**.
- **Reticulo:** si un conjunto de entidades es de nivel inferior en más de una relación, tiene **herencia múltiple** y su estructura se denominará reticulo.

Restricciones a las generalizaciones

- deter-
minación
 - **Definida por el atributo (o por la condición):** la pertenencia a una subclase se evalúa en función del cumplimiento de una condición, es decir, se evalúan en función del mismo atributo.
 - **Definida por el usuario:** las subclases no están restringidas por una condición de pertenencia, sino que el usuario asigna las entidades a un conjunto dado.
 - **Restricción de completitud:** especifica si una entidad de la superclase debe pertenecer, al menos, a uno de los conjuntos (generalización o especialización total) o no (generalización o especialización parcial).

Agregación: las agregaciones generan conjuntos de entidades de nivel conceptual superior, agrupando conjuntos de entidades relacionadas mediante conjuntos de relaciones en un concepto único. Su objetivo fundamental es poder establecer conjuntos de relaciones entre conjuntos de relaciones.

Tema-2

Modelo relacional

Idea principal: todos los datos que maneja la base de datos se almacenan en una tabla de atributos. Una tabla se puede ver desde el punto de vista matemático como una relación, es decir, como el producto cartesiano de los atributos que la forman.

El modelo relacional es una técnica de diseño de abajo a arriba: se comienza identificando todos los atributos o características interesantes para la resolución del problema y luego se agrupan esos atributos en conjuntos de mayor nivel semántico.

El modelo relacional es hoy en día el principal modelo de datos para las aplicaciones comerciales de procesamiento de datos.

Estructura de las BD relacionales

Una relación entre atributos identificados en el mundo a modelar genera tuplas a través de su producto cartesiano. Estas tuplas se almacenan físicamente en las filas de las tablas del modelo.

Las columnas permiten almacenar los diferentes atributos interesantes de un elemento y las filas almacenan los valores (tupla) que tiene cada elemento para cada uno de los atributos considerados de interés.

El término **relación** se usa para referirse a una tabla, el término **tupla** a una fila y **atributo** a una columna.

El número de atributos de una relación se denomina **grado** de la relación, y el número de tuplas que tiene una relación se llama **cardinalidad**.

Llamamos **dominio** de un atributo al conjunto de valores permitidos para cada atributo de una relación. Un **dominio atómico** es aquel cuyos elementos se consideran unidades indivisibles. Es necesario para toda relación r que los dominios de todos los atributos de r sean atómicos.

En este modelo no hay
atributos compuestos ni
multivalorados.

Esquema de la base de datos

La definición de todos los tablas que forman una BD se denomina **esquema de la BD** (es el diseño lógico de la BD). El esquema es la definición general, los datos particulares almacenados en un momento dado se denominan **ejemplar de la BD**. En lenguaje de programación, el esquema se corresponde con el concepto de definición de tipos y el concepto de relación se corresponde con el de variable. En general, los esquemas de las relaciones consisten en una lista de los atributos y de sus dominios correspondientes.

Claves

Cada tupla dentro de una tabla debe ser distingible de todas las restantes tuplas de esa tabla, por tanto toda tabla deberá tener un conjunto de atributos que esté garantizado que tengan valores diferentes en cada una de las filas de la tabla. Esta necesidad debe cumplirse para las filas que estuvieron almacenadas en la tabla en el pasado, para las filas que están almacenadas en el presente y para cualquier fila que se pueda almacenar en el futuro.

- **Superclaves**: son los conjuntos de atributos que identifican cada fila de la tabla.
- **Clave candidatas**: son aquellas superclaves mínimas (no se puede eliminar ningún atributo sin perder la condición de superclave).
- **Clave primaria**: es una de las claves candidatas, normalmente se escoge la más estable en el tiempo.

El modelo relacional incorpora redundancia controlada para visualizar las relaciones conceptuales entre diferentes tablas. Esta redundancia controlada se plasma incorporando a una tabla atributos denominados **clave externa** que hacen referencia a claves candidatas de otras tablas. Los valores permitidos en estos atributos de clave externa están restringidos a valores que existan previamente en la clave candidata referenciada, generando lo que se conoce como **restricción de integridad referencial**.

Representación del esquema

No hay un estándar de representación diagramática del modelo relacional de una BD. Cualquier forma de representar el modelo relacional es válido siempre que deje muy claro cuáles son las claves externas y a qué claves candidatas hacen referencia.

IMPORTANTES: el modelo E-R no es una representación gráfica del modelo relacional, la gran diferencia entre ambos modelos es la existencia o no de redundancia de atributos

Lenguajes de consulta relacional

Para realizar una implementación operativa del modelo relacional es necesario que exista un lenguaje que permita crear, modificar y eliminar las tablas del esquema relacional de la BD y, una vez creadas las tablas, poder añadir tuplas con la información adecuada y obtener información a partir de los datos almacenados.

Los lenguajes de consulta pueden clasificarse en procedimentales o no procedimentales, y tienen 5 operaciones básicas (selección, proyección, producto cartesiano, unión de conjuntos) y diferencia de conjuntos) a partir de las cuales se puede hacer cualquier otra operación a realizar.

Álgebra relacional

Es un lenguaje de consulta procedimental (base del l. comercial SQL) que se emplea en la mayoría de gestores de bases de datos. Define un conjunto de operaciones que, operando sobre tablas, devuelven tablas.

Operaciones → pueden ser unarias (operan sobre una sola relación) o binarias (operan sobre pares de relaciones).

→ **Selección**: filtra tuplas que satisfacen una condición (unaria).

Selección → σ
Predicado → subíndice
Relación → entre paréntesis } **σ predicado (relación)**

El predicado puede ser una comparación a varías, con y (\wedge), o (\vee) o no (\neg)

→ **Proyección**: determina qué atributos se visualizan y cuáles se eliminan de los resultados (unaria).

Proyección → π
Atributos a mostrar → subíndice
Relación → entre paréntesis } **π_{atr1, atr2, ..., atrn} (relación)**

→ **Unión y diferencia**: corresponden a las operaciones lógicas de unión y diferencia sobre tablas. Las relaciones que participan en estas operaciones deben ser compatibles (mismo nº de atributos y mismo tipo de datos en cada atributo 1 a 1), el diseñador de la BD es responsable de que la semántica de los atributos emparejados sea coherente. (Relaciones binarias)

{ Unión → \cup
Diferencia → $-$

→ **Producto cartesiano**: comparte información procedente de dos relaciones y crea una relación cuyo nº de atributos es la suma de atributos de las 2 relaciones combinadas y su nº de tuplas es el producto de las tuplas de ambas relaciones (binaria).

Producto cartesiano → \times : **relación1 × relación2**

El resto de operaciones del álgebra relacional son derivadas, se pueden obtener a partir de las 5 básicas.

Existe una versión extendida del álgebra relacional que incorpora operaciones de agregación que permiten definir conjuntos de tuplas sobre los que aplicar operadores de resumen (contar, sumar, hallar máximo, hallar mínimo y calcular la media).

Cálculo relacional de tuplas / Cálculo relacional de dominios (CRT/CRD)

Son lenguajes no procedimentales para manipular el modelo relacional. Tienen todas las operaciones necesarias en un lenguaje de consulta relacional, es la base de los lenguajes comerciales QBE y Datalog. Es equivalente, con algunas restricciones, al álgebra relacional. Basados en lógica matemática

Un lenguaje que puede usarse para producir cualquier relación se denomina **relacionalmente completo**. Los tres explicados en este tema son **relacionalmente completos**.

Tema - 3

SQL : DDL y DML

SQL es el lenguaje de consulta de bases de datos más ampliamente utilizado. Aunque se le suele llamar lenguaje de consulta también tiene instrucciones para crear, modificar y eliminar tablas, insertar, modificar y eliminar tuplas y definir restricciones de integridad. Tiene varias versiones, pero en este tema nos centraremos en el estándar de 1992.

Lenguaje de definición de datos (DDL)

Es uno de los componentes del lenguaje SQL, proporciona comandos para la definición y modificación de esquemas de relación y el borrado de relaciones.

El DDL permite la especificación de un conjunto de relaciones y de su información relativa, incluyendo:

- { - El esquema de cada relación
- El dominio de valores asociado a cada atributo
- Las restricciones de integridad
- El conjunto de índices que se deben mantener para cada relación
- La información de seguridad y de autorización de cada relación
- La estructura de almacenamiento físico de cada relación en el disco.

Tipos básicos de datos.

- **char(n)** → Cadena de caracteres de longitud fija (n) también vale **character**
- **varchar(n)** → Cadena de caracteres de longitud variable (máximo n) también vale **character varying**.
- **int** → entero, también vale **integer**.
- **smallint** → entero pequeño
- **numeric(p,d)** → número de coma fija con p dígitos de los cuales d son decimales.
- **real, double precision** → números de coma flotante de simple y doble precisión.
- **float(n)** → número de coma flotante con precisión mínima n.

BLOB → binary (large object) es un objeto binario grande, se usa para almacenar datos binarios grandes

Definición básica de esquemas

- Las relaciones se definen mediante el comando **create table**

```
create table nombre_relación  
| (atributo1 tipodato,  
| atributo2 tipodato,  
| atributo3 tipodato,  
| primary key (atributo1));
```

- Se utiliza el comando **insert** para introducir datos en la relación

```
insert into nombre_relación  
values (atr1, atr2, ..., atrN);
```

- Para eliminar tuplas de una relación se utiliza el comando **delete**, que elimina todas las tuplas de una relación.

```
delete from nombre_relación
```

- Para eliminar una relación completa se utiliza el comando **drop table**.

```
drop table nombre_relación
```

- Para añadir o eliminar atributos a una relación se utiliza **alter table**

añadir → **alter table nombre_relación add nombre_atributo tipodato;**

eliminar → **alter table nombre_relación drop nombre_atributo**

- Para eliminar tuplas específicas de una relación se usa **delete where**, con un predicado que hará que se borrarán todas las tuplas que lo cumplan.

```
delete from nombre_relación
where predicado
```

- Para cambiar un valor dentro de una tupla sin cambiar todos los valores de la misma se utiliza la instrucción **update**.

```
update nombre_relación
set nombre_atributo = x
where nombre_atributo... (condición)
```

Si no se pone **where**, el cambio se produce en todas las tuplas de la relación.

Vistas

Son "relaciones virtuales" que contienen el resultado de las consultas para hacer visible al usuario los datos que éste puede ver. En SQL las vistas se definen con **create view**.

```
create view nombre_vista as <expresión de consulta>
```

Transacciones

Una transacción consiste en una secuencia de instrucciones de consulta o de actualización. La transacción debe finalizar con las expresiones **Commit work** o **Rollback work**.

- **Commit work** → hace que las actualizaciones realizadas por la transacción pasen a ser permanentes en la base de datos.
- **Rollback work** → deshace todas las actualizaciones realizadas por las sentencias de la transacción.

Restricciones de integridad

Las restricciones de integridad garantizan la consistencia de la base de datos, especialmente durante las actualizaciones de información, evitando que las modificaciones realizadas por usuarios den lugar a una pérdida de consistencia de los datos.

Restricciones sobre una sola relación

- not null**: prohíbe la inserción de valores nulos para ese atributo, las claves primarias reciben esta restricción automáticamente.
- unique**: indica que esos atributos forman una clave candidata, ningún par de tuplas puede tener el mismo valor para esos atributos.
- check(P)**: especifica un predicado que deben satisfacer todas las tuplas de una relación.

Restricción de integridad referencial: fuerza que los valores válidos en las claves externas sean valores que existan en las claves candidatas referenciadas (fundamental para la consistencia interna de la base de datos).

Tipos de datos fecha y hora

{

- date** → fecha del calendario que contiene el año, el mes y el día del mes.
- time** → hora del día en horas, minutos y segundos, se puede almacenar también el huso horario con time with timezone
- timestamp** → combinación de date y time
 - * se puede añadir la variante (p) para especificar el nº de cifras para los segundos.

Lenguaje de modificación de datos

Es el otro componente del lenguaje SQL, incluye un lenguaje de consultas como comandos para insertar, borrar y modificar tuplas en la base de datos.

Estructura básica de consultas

La estructura básica de consultas en SQL consta de tres cláusulas:

[
 select atributo₁, ..., atributo_N
 from relación
 where predicado]
]

Álgebra relacional
where = operación selección
select = operación proyección

A la hora de hacer la consulta, se debe plantear en el siguiente orden:

- 1º **from** → pensar qué incorporar en la cláusula from, lo que no se obtenga aquí como resultado nunca podrá aparecer en el resultado final.
- 2º **where** → razonar cómo filtrar los resultados obtenidos mediante la cláusula where para eliminar resultados no válidos generados por el producto cartesiano o para enfocar los resultados a unas necesidades de información concretas.
- 3º **select** → decidir con la cláusula select qué información contendrá la tabla resultante de la consulta.

Operación de renombrado

Cláusula **as**: es un mecanismo de renombrado de atributos de la relación resultado de una consulta, se puede usar tanto en la cláusula select como en la cláusula from.

→ Para cambiar el nombre del atributo consultado

```
select nombre_atr as nombre_nuevo  
from nombre_refaccion  
where predicado
```

→ Para cambiar un nombre de refacción (si es muy largo por uno más corto)

```
select R1.atr1, R2.atr2  
from refaccion1 as R1, refaccion2 as R2  
where predicado
```

Operaciones con cadenas de caracteres

SQL permite varias funciones que operan sobre cadenas de caracteres como `upper(s)` para convertir a mayúsculas y `lower(s)` para convertir a minúsculas, o `trim(s)` para eliminar espacios al final de las cadenas. Se pueden comparar patrones usando `like`, los patrones se escriben con caracteres especiales (% o _).

- {
 - % → coincide con cualquier subcadena de caracteres
 - ↳ 'Algo %': cualquier cadena de caracteres que empiece por "Algo"
 - ↳ '% algo %': cualquier cadena que contenga la subcadena "algo"
 - _ → coincide con cualquier carácter
 - ↳ '___': cualquier cadena que tenga 3 caracteres.
 - ↳ '___%': cualquier cadena que tenga mínimo 3 caracteres.

Los patrones se expresan con el comando `like`:

```
select attrib.  
from refaccion  
where atributo like patron
```

Para usar caracteres especiales como caracteres "normales" incluidos en el patrón se usa '\\' antes del carácter especial y con el comando escape

(%) like '5\%' escape '\'

* Like distingue entre mayúsculas y minúsculas, aunque hay BD con variantes que no distinguen. También se pueden buscar discordancias con `not like`.

Especificación de atributos

En la cláusula `select` se puede utilizar el símbolo '*' para indicar todos los atributos de la forma:

```
select nombre_refaccion*
```

Orden en la presentación de las tuplas

La cláusula `order by` coloca los elementos en orden en función de un atributo (o varios). Por defecto se ordenan de forma ascendente pero se puede especificar con `desc` o `asc`.

```

    | select *
    | from relación
    | order by atr1 asc, atr2 desc
  
```

→ se ordenan en función de atr1 y, si coincide, en función de atr2.

→ En la cláusula where se pueden utilizar los operadores de comparación **between** y **not between** para simplificar.

Operaciones sobre conjuntos

Las operaciones **union**, **intersect** y **except** operan sobre relaciones y se corresponden con las operaciones unión, intersección y diferencia del álgebra relacional.

Unión

A diferencia de la cláusula **select**, la operación **union** elimina los valores duplicados automáticamente, si se desea mantener los duplicados se puede escribir **union all** en lugar de **union**.

```

    | (select atrb.
    |   from relación
    |   where predicado)
    | union
    | (select atrb2
    |   from relación2
    |   where predicado2)
  
```

* lo mismo con **union all**

Intersección

La operación **intersect** también elimina los valores duplicados automáticamente si se quieren mantener se sustituye por **intersect all**

```

    | (select atrb.
    |   from relación
    |   where predicado)
    | intersect
    | (select atrb2
    |   from relación2
    |   where predicado2)
  
```

* lo mismo con **intersect all**

Excepto

La operación **except** da como salida todas las tuplas de su primera entrada que no están en la segunda. Al igual que **unión** e **intersección** elimina los duplicados, que se pueden mantener con **except all**. La semántica es la misma que en los casos anteriores.

Valores nulos

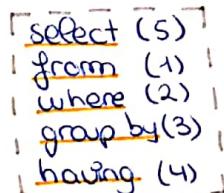
Los valores null son un problema, solo se debe recurrir a ellos cuando es absolutamente necesario. El resultado de las expresiones aritméticas es null si cualquiera de los valores de entrada es null. SQL trata como unknown el resultado de cualquier comparación en la que intervenga el valor null (excepto para `IS NULL` e `IS NOT NULL`). Si el resultado de la cláusula `where` se evalúa a unknown para una tupla, esta tupla no se añade al resultado.

Funciones de agregación

Las funciones de agregación permiten obtener un valor resumen para un conjunto de valores. SQL dispone de 5 operaciones de agregación, que se pueden aplicar al total de tuplas obtenidas en una consulta o a grupos de tuplas organizados mediante la cláusula `group by`. Cuando existen valores null, el proceso de las operaciones de agregación se complica.



En la estructura básica de consulta (pág 12) se incluyen las cláusulas `group by` y `having` al final pero entre `where` y `select` en el planteamiento:



Subconsultas anidadas

Las consultas válidas en SQL siempre generan una tabla, que puede ser usada dentro de otra consulta SQL, fundamentalmente en las cláusulas `from` y `where` para comparaciones y compruebas entre otras tablas.

Esta posibilidad de anidar varios niveles de consultas permite la estructura de consultas muy sofisticadas y complejas en SQL, es la parte más rica del lenguaje pero también la que más práctica necesita para llegar a ser dominada.

Unión natural

La operación de unión natural opera con dos relaciones y genera como resultado una relación con los pares de tuplas que tienen los mismos valores en los atributos que aparecen en los esquemas de ambas relaciones. Se utiliza la cláusula **natural join**.

```
select atr1, ..., atrn  
from r1 natural join r2  
where predicado
```

La cláusula **join...using** permite indicar el nombre de los atributos a comparar:

```
select atr1, ..., atrn  
from (r1 natural join r2) join relación  
using (atr)
```

La condición **on** permite la reunión de un predicado general sobre relaciones. A diferencia de natural join el resultado contiene los atributos comparados dos veces en la reunión (una por cada tabla).

```
select *  
from r1 join r2 on r1.an=r2.an
```

Reunión externa

La operación de reunión externa funciona de forma similar a la operación de reunión, creando tuplas en el resultado con valores nulos en los atributos de la relación que sólo aparecen en el esquema de una de las relaciones.

- (ljoj) **left outer join** → conserva las tuplas de la relación a la izquierda de la operación.
- (rjoj) **right outer join** → conserva las tuplas de la relación a la derecha de la op.
- (fjoj) **full outer join** → conserva las tuplas de ambas relaciones.

```
select *  
from r1 natural fjoj r2
```

* para cualquiera de los 3 tipos.

El tipo de reunión por defecto es reunión interna, aunque se puede especificar con la expresión **inner**.

Tipos de reunión

- inner join
- left outer join
- right outer join
- full outer join

Condiciones de reunión

- natural
- on <predicado>
- using(A1,A2,...,An)

La inclusión directa de las tablas en la cláusula **from** genera su producto cartesiano, la reunión natural es una operación que agrupa un prod.c. y un filtrado de valores sobre atributos con el mismo nombre.

Tema - 4

Transformación MER a MR

Atributo Compuesto	Se descompone en atributos atómicos
Atributo Multivalorado	Se quita el atributo de la entidad en la que estaba y se crea una tabla con una copia de la clave primaria y el atributo multivalorado PK: todos los atributos FK: copia de clave primaria
Entidad Fuerte	Se crea una tabla que incluya todos los atributos atómicos PK: clave primaria original
Entidad Débil	Se crea una tabla que incluya todos los atributos atómicos y una copia de la clave primaria de la entidad identificadora PK: copia de clave primaria y clave primaria parcial FK: copia de clave primaria
Relación Binaria 1:N	Se incluye una copia de la clave primaria del lado 1 en la tabla del lado N; Se incluyen los atributos de la relación en la tabla del lado N FK (lado N): copia de clave primaria
Relación Binaria N:N	Se crea una tabla con copias de las claves primarias de las entidades; Se añaden a la tabla los atributos de la relación PK: copias de claves primarias y clave primaria parcial FK: copias de claves primarias
Relaciones Ternarias y Superiores	Se crea una tabla con copias de las claves primarias de las entidades; Se añaden a la tabla los atributos de la relación PK: copias de claves primarias de lados N y clave primaria parcial FK: copias de claves primarias
Relación Binaria 1:1 Participación obligatoria en ambos lados	Combinar entidades en una tabla PK: una de las claves primarias
Relación Binaria 1:1 Participación obligatoria en un lado	Como relación binaria 1:N; lado opcional como lado 1; lado obligatorio como lado N
Relación Binaria 1:1 Participación opcional en ambos lados	Como relación binaria 1:N escogiendo el lado de forma arbitraria

Jerarquía Total, solapada	Una tabla con todos los atributos de la jerarquía y con discriminantes binarios (tantos como subclases) para saber a que subclases pertenece el elemento PK: clave primaria superclase
Jerarquía Parcial, solapada	Una tabla para la superclase con todos sus atributos PK: clave primaria superclase Otra tabla para todas las subclases con una copia de la clave primaria de la superclase, con todos los atributos de las subclases y con discriminantes binarios (tantos como subclases) para saber a que subclases pertenece el elemento PK: copia clave primaria superclase FK: copia clave primaria superclase
Jerarquía Total, disjunta	Una tabla para cada combinación superclase/subclase con todos los atributos de la superclase y de la subclase PK: clave primaria superclase
Jerarquía Parcial, disjunta	Una tabla para la superclase con todos sus atributos PK: clave primaria superclase Una tabla para cada subclase con todos los atributos de la subclase y una copia de la clave primaria de la superclase PK: copia clave primaria superclase FK: copia clave primaria superclase

Tema - 5

Normalización del MRL

La normalización del modelo relacional de una base de datos tiene una fuerte base matemática que se apoya en el concepto de dependencia funcional de atributos. El descubrimiento de dependencias funcionales entre atributos supone la representación de la semántica que se incorpora a la base de datos, se podría decir que es equivalente a descubrir conjuntos de entidades y relaciones en el MRL.

Una vez representada la semántica mediante las dependencias funcionales el proceso de normalización consiste en el cumplimiento sucesivo de determinadas condiciones denominadas formas normales.

A veces es necesario descomponer los esquemas en tablas más pequeñas siendo válidas únicamente las descomposiciones reversibles, sin pérdida, de forma que si se unen las tablas se vuelve a tener la original.

Primeras formas normales (1FN)

Cuando una tabla contiene únicamente atributos atómicos, se dice que está en la forma normal. En una base de datos del MRL esta 1^a es la única obligatoria, el resto de formas normales mejoran el diseño evitando los problemas que podrían aparecer.

- En general, se utilizan letras griegas para los conjuntos de atributos, latinas minúsculas seguidas de una mayúscula entre paréntesis para un esquema de relación, cuando se usa una griega minúscula, se refiere a un conjunto de atributos que puede ser o no un esquema, cuando se usa la latina indica que es un esquema. Una superclave se indica con K, se usa un nombre en minúsculas para las relaciones y el término ejemplar se refiere a un valor concreto en un momento dado.

La dependencia funcional es la base de la normalización. Un conjunto de atributos determina funcionalmente a otro si, para todos los tuplas de una relación, cuando los valores del primero se repiten, entonces los valores del segundo se repiten.

- Las diversas formas normales van imponiendo sucesivas condiciones a las relaciones que forman la base de datos.

Segunda forma normal (2FN) impide la existencia de dependencias funcionales parciales (en las dependencias funcionales parciales el lado izquierdo es una parte de una superclave pero no la superclave completa).

Tercera forma normal (3FN): impide la existencia de dependencias funcionales transitivas (en las dependencias funcionales transitivas el lado izquierdo no forma parte de una superficie).

Forma normal de Boyce-Codd (FNBC): obliga a que en todas las dependencias funcionales el lado izquierdo sea una superficie de la relación.

Cuando se encuentra alguna relación con alguna dependencia funcional que no cumple las condiciones se procede a descomponerla (habitualmente en 2). Hay que escoger descomposiciones de reunión sin pérdidas que preserven las dependencias.

Por encima de la FNBC existen otras formas normales que van imponiendo condiciones cada vez más restrictivas, pero normalmente no se usan en la práctica. El objetivo habitual de la normalización es conseguir que todas las relaciones estén en FNBC.

Teoría de las dependencias funcionales

Axiomas de Armstrong

Los axiomas de Armstrong y sus reglas adicionales permiten descubrir nuevas dependencias funcionales a partir de un conjunto de dependencias dadas.

→ Regla de la reflexividad: Si α es un conjunto de atributos y $B \subseteq \alpha$, entonces se cumple que $\alpha \rightarrow B$

→ Regla de la aumentatividad: si $\alpha \rightarrow \beta$ y r es un conjunto de atributos, entonces se cumple que $r\alpha \rightarrow r\beta$.

→ Regla de la transitividad: si $\alpha \rightarrow \beta$ y $\beta \rightarrow \gamma$ entonces $\alpha \rightarrow \gamma$

- reglas adicionales**
- { Regla de la unión: Si $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$, entonces $\alpha \rightarrow \beta\gamma$
 - Regla de la descomposición: Si $\alpha \rightarrow \beta\gamma$ entonces $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$
 - Regla de la pseudotransitividad: Si $\alpha \rightarrow \beta$ y $\gamma\beta \rightarrow \delta$ entonces $\alpha\gamma \rightarrow \delta$

Calcular el cierre de un conjunto de atributos porque si se cierra contiene todos los atributos de una relación, significa que el conjunto estudiado era una superficie. Conocer cuáles son las superficies de una relación es fundamental para saber si la relación está en la FNBC. Saber que una base de datos está en la FNBC implica comprobar todas las dependencias funcionales.

El **recubrimiento canónico** es el conjunto mínimo equivalente de dependencias funcionales. Dos conjuntos son equivalentes si sus cierres son iguales.

Pasos para la normalización:

1º Relación universal y dependencias funcionales

→ **Relación universal**: recoger los atributos descriptivos.
 $U = \{A_1, A_2, \dots, A_N\}$

→ **Dependencias funcionales**: localizar las dependencias entre atributos.
 $F = \{D_1; D_2; \dots; D_N\}$

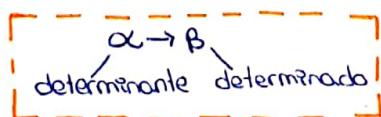
2º Primera forma normal → convertir todos los atributos en simples

3º Obtener el **recubrimiento canónico**

- 3.1 → Aplicar la regla de la **descomposición de Armstrong**
- 3.2 → Eliminar dependencias implícitas lógicamente por el resto
- 3.3 → Aplicar la regla de la **unión de Armstrong**

4º Descomponer hasta que todas las relaciones estén en **FNBC**

- 4.1 → Descomposición por **localización de atributos equivalentes**: se elimina de la relación original uno de los atributos equivalentes
- 4.2 → Descomposición por **eliminación de dependencias**: se crea una nueva relación por cada dependencia que impida la FNBC, y se eliminan de la original los atributos determinados (flecha derecha).



Cálculo del cierre: es un conjunto de atributos a partir de los cuales se obtiene U . Los atributos que no estén determinados por ningún otro atributo tienen que formar parte del cierre para que sea una clave candidata.

Tema - 6

El mundo de las bases de datos

Un **sistema gestor de bases de datos** (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos.

El **objetivo** de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera práctica y eficiente.

La **gestión** de los datos simplifica la definición de estructuras para almacenar la información y la provisión de mecanismos para su manipulación.

Inconvenientes a evitar en un SGBD

- Redundancia e inconsistencia de los datos
- Dificultad en el acceso a los datos
- Aislamiento de datos
- Problemas de integridad
- Problemas de atomicidad
- Anomalías en el acceso concurrente
- Problemas de seguridad

El propósito de los SGBD es paliar los inconvenientes que tienen los sistemas de procesamiento de archivos



Visión de los datos

Una de las principales finalidades de los sistemas de bases de datos es ofrecer a los usuarios una **visión abstracta** de los datos.

Niveles de abstracción

Nivel físico: describe cómo se almacenan los datos

Nivel lógico: describe qué datos se almacenan en la BD y qué relaciones existen entre ellos. Este nivel es el que usan los administradores de bases de datos.

Nivel de vistas: este nivel solo muestra parte de la BD, sirve para simplificar la interacción de los usuarios con el sistema.

La colección de información almacenada en una base de datos en un momento dado se denomina **ejemplar** de la BD. El diseño general de la BD se denomina **esquema** de la BD.

El **esquema físico** describe el diseño a nivel físico de la BD, el **esquema lógico** describe a su diseño a nivel lógico. Una BD también puede tener varios esquemas a nivel de vistas denominados **subesquemas**, que describen varias vistas de la BD.

El **esquema lógico** es el más importante.

Modelo de datos

Ofrece un modo de describir el diseño de las BD en los tres niveles. Pueden clasificarse en cuatro categorías diferentes.

- Modelo entidad-relación
- Modelo relacional
- Modelo de datos semiestructurados (elementos del mismo tipo pueden tener atributos diferentes)
- Modelo de datos basado en objetos (extensión del MER con encapsulación,...)

Para conseguir el objetivo de la visión abstracta, en primer lugar se establecen los 3 niveles independientes separando los aspectos de diseño de los de implementación. En segundo lugar se establecen los esquemas de estos niveles y, por último, se establece un modelo de datos que ofrece una colección de herramientas conceptuales para definir los esquemas de los diferentes niveles.

Lenguajes de bases de datos

Es un lenguaje que permite a los usuarios tener acceso a los datos organizados mediante el modelo de datos elegido, o manipularlos.

El **Lenguaje de definición de datos** expresa los esquemas definidos en un determinado modelo de datos. En él se definen las restricciones de integridad.

- **Restricciones de dominio**: valores que puede tomar cada atributo
- **Integridad referencial**
- **Asertos**: condiciones que la BD debe satisfacer siempre
- **AutORIZACIÓN**: de lectura, de inserción, de actualización o de eliminación.

El **Lenguaje de manipulación de datos** permite consultar la BD para extraer información y explotar la información introducida en ella. Un LMD puede ser procedimental o declarativo.

Una **consulta** es una instrucción para recuperar información, la parte del LMD implicada en la recuperación de información se denomina **lenguaje de consultas**.

Bases de datos relacionales → Se basan en el modelo relacional y usan un conjunto de tablas que representan los datos y las relaciones

En la fase de **diseño conceptual** de una BD, el diseñador recoge el modelo de datos y traduce los requisitos en un esquema conceptual de la BD.

Modelo entidad-relación → se basa en un conjunto de objetos denominados entidades y las relaciones que existen entre ellas.

Normalización → es un proceso cuyo enfoque es diseñar esquemas que se tengan en la forma normal adecuada.

Gestor de almacenamiento → es un módulo de programa que proporciona la interfaz entre los datos de bajo nivel almacenados en la BD y los programas de aplicación y las consultas.

Componentes del gestor

- Gestor de autorizaciones e integridad
- Gestor de transacciones
- Gestor de archivos
- Gestor de la memoria intermedia

Estructuras de datos del gestor

- Archivos de datos
- Diccionario de datos
- Índices

Gestión de transacciones

Requisitos

- **Atomidad** → una transacción debe producirse por completo o no producirse en absoluto
- **Consistencia** → es esencial que las transacciones preserven la consistencia.
- **Durabilidad** → los valores nuevos tras la transacción deben persistir aunque haya un fallo en el sistema.

Una **transacción** es un conjunto de operaciones que lleva a cabo una única función lógica en una aplicación de bases de datos.

El **componente de gestión de transacciones** es el responsable de garantizar las propiedades de durabilidad y atomidad. Es responsabilidad del **gestor de control de concurrencia** controlar la interacción de las transacciones concurrentes para garantizar la consistencia.

El término **minería de datos** se refiere, en líneas generales, al proceso de análisis semiautomático de grandes bases de datos para descubrir patrones útiles, intenta descubrir reglas y patrones en los datos.