

Base de datos 1

Resumen y notas de clases

Ultima modificación: 10 de mayo de 2005

Base de datos: conjunto de datos y sus relaciones

Modelos de base de datos

Modelo Jerárquico: acceso por un registro padre, el mas alto de esa jerarquía, por la cabecera. (similar a arboles)

Modelo de Redes: existen varios registros dueños que poseen registros. Ahora no hay 1 solo acceso a la estructura, puede haber mas de 1. Puede haber un registro con mas de un "padre". Cada registro tiene su propia cadena de apuntadores.

Cualquier tipo de entidad-relación puede ser representada en este tipo de base de datos.
Desventaja: es muy lento para actualizar/modificar.

La navegación del sistema jerárquico, o el de redes se hace registro a registro.

Modelo relacional: definido por el matemático James Codd

Se define como debe percibir el usuario los datos: con una percepción tabular (percibir los datos como tablas).

Tiene independencia de la estructura física. Es transparente para el usuario. Para consultar y modificar los datos, se hace en "lenguaje natural" (aun no implementado del todo, se usa SQL, Standard Query Language).

Hay 4 revisiones actualmente de SQL, los productos actuales cumplen con la 2, y migrando hacia la 3.

Los productos relacionales actuales no pueden manejar dominios.

Este modelo relacional es lo que se usa en la actualidad.

Modelo OOBD (orientado a objetos): representan el paradigma OOP, la misma base de datos es un objeto. Se implementaron las bases de datos relacionales orientadas a objetos.

Existen OOBD funcionando, algunas comerciales (ej : Jasmine)

Modelos textuales: documentales (tipo los de google) ; guardan documentos (no solo texto, vídeos, imágenes, etc).

Datos no estructurados y estructurados.

Para guardar textos se implementa con un archivo documentario y uno de referencias.

También se agrega un índice de descriptores (palabras clave del documento).

La búsqueda es rápida. Hay un inverso de descriptor también, y un archivo de sinonimia, uno de sinónimos categorizados (tesauro), otro para búsquedas con distancia, etc

Actualmente, los motores de indexación y búsqueda hacen algo muy similar a esto, con éxito.

Ventajas de usar bases de datos

- Reducción de la **redundancia** (no se elimina del todo, a veces es necesaria y buena cierta redundancia, se llama “redundancia controlada”)
- Reducción de **inconsistencias**
- Administrar restricciones de **seguridad**
- Administrar restricciones de **integridad**
- Administrar ambientes **multiusuario** (concepto de bloqueo de información [permitir que un solo usuario acceda a la vez a la información], concepto de transacción [operaciones que modifican la base de datos])
- Facilitar el **cambio de estructuras de datos** (se enmascara la estructura real de la base de datos con una visión abstracta para el usuario, independencia de las estructuras físicas).

Roles o tipos de usuario en una base de datos

- **Usuario:** programa o persona } interactúa con los datos
- **Programador** } interactúa con los datos y la estructuras
- **DBA** (administrador base de datos) } mantiene, administra, crea, interactúa con otros usuarios, le interesa la estructura y el funcionamiento de la base de datos

Los 3 son usuarios de la base de datos.

Administrador de datos (DA) – **NO** es un usuario de la base de datos ; su función es definir que cosas guardar, que cosas no, visibilidad/accesibilidad de los datos, como se guardan los datos, etc.

Hacen auditoría, seguridad informática, etc. Define políticas de seguridad, integridad, backup, y su implementación.

Funciones de los DBA

1. definir esquema conceptual (visión lógica de la base de datos) } DDL
2. definir esquema interno (el diseño físico) } DDL
3. interacción con los usuarios
4. definir integridad y seguridad. implementar las políticas definidas por el DA. } DIL y DSL
5. definir backup y restore (resguardo y recuperación: recovery) , backup en frío y en caliente
6. controlar el rendimiento de la base de datos

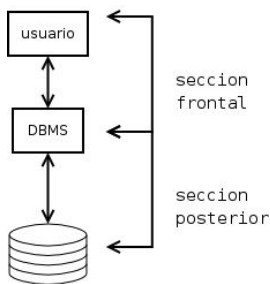
Funciones de un DBMS (Data Base Management System) –

Características básicas del software para ser una base de datos.

- tener un DDL (lenguaje de **definición** de datos)
- tener un DIL (lenguaje de **integridad** de datos)
- tener un DSL (lenguaje de **seguridad** de datos)
- tener un DML (lenguaje de **manipular** los datos)
- un modulo por el cual puedo ingresar las operaciones y comprender su resultado
- un modulo capaz de ejecutar las instrucciones que reciba
- algún mecanismo para almacenar la estructura y las reglas de seguridad e integridad (DDL, DSL, DIL)
- toda base de datos tiene algo llamado catalogo o diccionario de datos, donde guarda las 3 cosas antes mencionadas
- modulo de administración de seguridad (login y eso)
- manejo de la integridad
- modulo de recuperación

SQL = DDL + DIL + DSL + DML

Otra manera de ver la base de datos:



Frontal: todo lo que interactúa con el DBMS cumpliendo con las restricciones definidas.

Posterior: programas que interactúan con el DBMS, pero que no necesariamente cumplen con las restricciones de integridad.

Esta división se hace normalmente para los utilitarios (software)

Modelo relacional CODD (RM/V2)

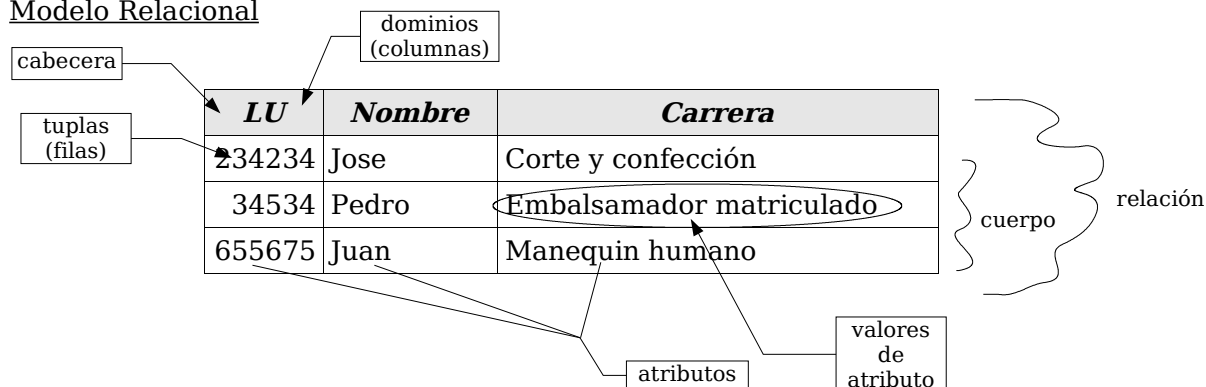
- El usuario percibe la DB como tablas
- No va a necesitar saber nada del hardware y la estructura física.
- Se encarga de
 - Modelo conceptual
 - Todo lo relacionado con la integridad de los datos
 - Alguna herramienta para manejar la DB

Datos:

- Estructura lógica
- Integridad de los datos
- Herramienta de manipulación de los datos

SQL **no** es un lenguaje de programación, usualmente se encuentra dentro de un lenguaje anfitrión o encapsulado en un servidor.

Modelo Relacional



Grado o aridad: cantidad de atributos (columnas)

Cardinalidad : Cantidad de tuplas (filas)

TODO el conjunto es la **RELACION**

Atributo: es la menor unidad de información con sentido propio.

Cumplen con la propiedad de **atomicidad**.

Estos atributos atómicos pueden tener una composición **simple** o **compuesta** (ej: fecha)

En el modelo relacional se habla de atributos simples/atómicos y atributos compuestos (NO permitidos en el modelo, se convierten en un conjunto de atributos simples).

Dominio: es el conjunto de valores legales posibles de los cuales los atributos extraen sus valores reales.

Cada atributo tiene un solo dominio asociado. Los atributos tienen un nombre que no cambia, no importa donde lo ponga.

Comportamiento: comparación entre valores, operaciones entre valores, y dominio del resultado.

(los productos relacionales actuales no implementan los dominios, implementan clases)

Tupla: (cada fila de la tabla :P) cada instancia o ítem de una relación, es el conjunto de valores de atributo que representan a esa instancia de la relación.

Cantidad de tuplas es la **cardinalidad**; todas esas tuplas forman el **cuerpo**, aun con cardinalidad cero (0) el cuerpo **existe!**

La **cabecera** esta formada por pares **atributo-dominio**

La cantidad de pares atr-dom de la cabecera se llama **grado/aridad** y va de 1..infinito (binaria, ternaria, primaria, etc)

Propiedades de las relaciones

1. **NO** existen tuplas duplicadas (unicidad, ya que el cuerpo es un conjunto)

Nota: No hay atributos duplicados en la cabecera

2. En las relaciones **NO** existe concepto de **orden** (p'q' es un conjunto y no existe el concepto de orden, no hay concepto de adyacencia o precedencia)

3. La cabecera de una relación **NO** esta ordenada.

4. Todos los atributos son **atómicos**.

NOTA: NO es lo mismo el modelo relacional que un producto relacional!

Tipos de relaciones

- Relación base: una relación autónoma con nombre propio, y tiene datos propios asociados
- Vistas: una relación derivada con nombre, existe mediante su definición en función de otras relaciones. No posee datos propios.

[me fui de la clase, faltan... :P]

Siempre debe existir conjunto de atributos de la relación que permite identificar a c/u de las tuplas de manera **unívoca**.

Variable de relación: (el contenido es dinámico) , definición de una cabecera+conjunto de tuplas variables, la cabecera NO cambia, cambia el cuerpo (varia a traves del tiempo).

Relación: es estática.

Como una relación no puede tener tuplas repetidas, **siempre** hay un identificador capaz de identificar c/u de las tuplas (así sea la cabecera entera).

Clave candidata: un subconjunto (puede ser toda la cabecera) de atributos de la cabecera de una relación, que debe cumplir estas 2 propiedades independientemente del tiempo:

- unicidad
- minimalidad

(o sea, lo mas corto posible, siendo único)

La clave candidata NO es parte del modelo relacional, el modelo relacional plantea una sola forma de acceder a una tupla determinada, mediante una clave primaria (PK: primary key)

La PK debe ser alguna de las claves candidatas, solo hay una PK por relación y NO existe relación sin PK.

La PK tiene unicidad y minimalidad también.

La PK es la **única** vía de acceso a una tupla determinada en el modelo relacional.

Para poder vincular relaciones, introducimos **redundancia controlada**, con restricciones.

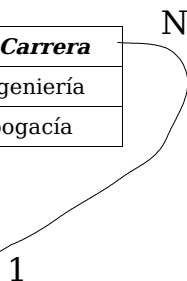
Ejemplo (relación muchos a 1):

Alumnos(relación referencial)

<i>NL</i>	<i>Nombre</i>	<i>Carrera</i>
123971293	Pedro Zaraza	ingeniería
749879879	Juan Gomez	abogacía

Carreras(relación referida)

<i>Facultad</i>	<i>Carrera</i>
Ciencias exactas	Ingeniería
Derecho	Abogacía



Referencial: clave foránea (ajena); un atributo toma un subconjunto de los valores del dominio. Las claves foráneas se colocan del lado del “muchos” en la relación.

Esa clave foránea (FK) siempre esta vinculada a una clave primaria (PK)

La relación que tiene la clave foránea se llama **relación referencial**.

La relación a la que “apunta” la clave foránea se llama **relación referida**.

Clave foránea (FK): es un subconjunto de atributos de la relación R1 y cumple:

- cualquier valor no nulo de la FK en R1 existe como valor de PK en R2
- si FK en R1 es compuesta, sus componentes o son nulos del todo, o son no nulos del todo.

R1 y R2 **no** necesariamente son distintas, puede haber **relaciones autoreferenciadas**.

El **ciclo referencial** es R1 referenciando a R2, R2->R3 y R3->R1, en la practica casi no se usa.

NOTA: el nulo se usa solamente en “propiedad no aplicable” o “valor no esta definido”

Si un atributo significa semánticamente lo mismo, debe tener el mismo nombre en todos los lugares que aparezca.


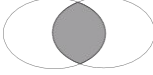

Álgebra relacional

Un lenguaje relacional es completo cuando cumple con todas las funciones del álgebra relacional. Actualmente, SQL lo cumple en su revisión 3.

Propiedades:

- Propiedad de cierre o clausura: el resultado de cualquier operación puede ser usado como argumento de una nueva operación.

Operaciones

Operación	Descripción	Notas
UNION	Hace la unión de relaciones. Requisitos: La cabeceras de la relaciones deben ser idénticas Resultado: relación con todas las tuplas de las relaciones unidas. La cardinalidad del resultado es \leq que la cadinalidades originales. Las tuplas repetidas se ELIMINAN .	conmutativa, asociativa. Gráfico: 
INTERSECT	Hace la intersección entre relaciones. Requisitos: las cabeceras deben ser idénticas. La cardinalidad del resultado es \leq que la cardinalidad de la menor relación.	conmutativa, asociativa. Gráfico: 
MINUS	Hace la diferencia . Requisitos: las cabeceras deben ser idénticas. Cardinalidad resultante \leq cardinalidad conjunto del cual estoy substrayendo.	NO conmutativa Gráfico: A minus B 
TIMES	Producto cartesiano extendido. Crea todas las combinaciones posibles de las tuplas que intervienen. Coalescencia: se obtiene una tupla formada por todos los valores de atributos de las 2 tuplas originales. Grado: suma grados de las relaciones . Cardinalidad: producto de cardinalidades de las relaciones. Nota: NO genera nueva información. Si hay atributos repetidos, se pone R1.Blah, R2.Blah, etc.	conmutativa, asociativa.
{ }	Proyección. Se selecciona entre { } los atributos que van a estar en el resultado. Ej: { Nombre, DNI } Ej: { * } (duplica la relación)	siempre elimina las tuplas duplicadas (por ser una relación, blah)
WHERE	Restricción: nueva relación formada por todas las tuplas que la condición se evalúa como verdadera. Resultado con grado igual, cardinalidad \leq original. Ej: ((R1 TIMES R2) WHERE Campo > 100) { Campo, Nombre }	El NULO no se incluye, pase lo que pase, en el resultado. Los operadores de conjunto (COUNT, SUM, MAX, MIN, AVG, NO se pueden usar con WHERE, por ser funciones estadísticas)
JOIN	Reunión: junta las tuplas en las cuales un atributo común esta repetido. Requisitos: debe haber al menos un atributo en común, y debe estar definido sobre el mismo dominio, y con igual significado semántico.	Es como un producto cartesiano extendido, donde solo se dejan las tuplas con atributos comunes iguales.

Operación	Descripción	Notas
DIVIDE BY	División: Dadas dos relaciones r1 y R2 donde la cabecera de R2 es un subconjunto de la cabecera de R1 definimos la operación de división de R1 por R2 como aquella operación que da como resultado una relación que tiene aquellos atributos de R1 que no forman parte de R2 y cuyos valores de atributo son los valores de las tuplas que poseen los mismos valores de atributos que el cuerpo de R3 y que poseen idéntico valor en el atributo independiente.	No asociativa ni conmutativa
EXTEND	Extender. Se agrega una operación a realizar y poner en un nuevo atributo: Sintaxis: EXTEND r1 ADD (operación) AS atributo_resultado Ejemplo: (EXTEND R1 ADD MAX(PU) AS Mayor, EXTEND R1 ADD MIN(PU) AS Menor, EXTEND R1 ADD AVG(PU) AS Promedio) { Mayor, Menor, Promedio }	Los operadores de conjunto se pueden usar con EXTEND
RENAME	Renombrar. Se cambia el nombre de un atributo o relación, útil cuando hay diferentes atributo con igual nombre involucrados en una operación. Ejemplo: RENAME atributo1 AS atributo2 Ejemplo: RENAME relacion1 as relacion2 } crea una relación con otro nombre Ejemplo: RENAME (r1 RENAME atrb1 as atrb2) AS r2	
SUMMARIZE	Resumir. Agrupa la relación según cierto criterio, donde hay un conjunto de tuplas para cada valor de ese criterio de forma de poder aplicar operadores de conjunto a cada grupo o conjunto de tuplas agrupadas. Sintaxis: SUMMARIZE r1 BY { atributo1, atributo2, ... } ADD operación AS nombre_atributo	A partir de ADD, es opcional, pero usualmente, necesario.

Operadores de conjunto (funciones estadísticas)

- **COUNT**(expresión)
- **SUM**(expresión)
- **MAX**(expresión)
- **MIN**(expresión)
- **AVG**(expresión)

Los operadores de conjunto se pueden usar con EXTEND, pero **no con WHERE**.

Ejemplo de SUMMARIZE

En este ejemplo, agrupa las tuplas, y genera un atributo imp, que tiene la suma de los totales para nlu.

NLU	Blah	Total
1	123123	5
2	34534	6
1	4565	8
2	6765	2
1	66	3
2	78789	4

(SUMMARIZE r1 **BY** { nlu } **ADD SUM**(total) **AS** imp) {nlu, imp}

NLU	Imp
1	16
2	12