DISEÑO DE BASES DE DATOS

Resumen teoría

Facultad de Informática
UNLP
2023

BERTONE, RODOLFO; THOMAS, PABLO Introducción a las Bases de Datos. Fundamentos y Diseño Primera Edición, Buenos Aires Prentice Hall - Pearson Education, 2011.

ÍNDICE

*** Falta contenido sobre seguridad e integridad en entornos concurrentes**

BASES DE DATOS

Una de las definiciones más divulgadas de una “Base de Datos” (BD) se refiere a una “colección o conjunto de datos interrelacionados” para la resolución de algún problema del mundo real (Universo de Discurso).

La colección de datos debe ser coherente. Un conjunto aleatorio de datos no puede considerarse una BD.

Una BD se diseña, se construye y completa de datos para un propósito específico. Está destinado a un grupo de usuarios concreto y tiene algunas aplicaciones preconcebidas.

Una BD está sustentada físicamente en archivos en dispositivos de almacenamiento persistente.

CONCEPTOS BÁSICOS

La definición de una BD consiste en especificar los tipos de datos, las estructuras y restricciones de los mismos.

La construcción de la BD es el proceso de almacenar datos concretos en algún dispositivo de almacenamiento bajo la gestión del DBMS.

La manipulación de BD incluye funciones tales como consultar la BD para recuperar datos específicos, actualizar los datos existentes, reflejar cambios producidos, etc

GESTORES DE BASES DE DATOS

Un Sistema de Gestión de Bases de Datos (DBMS en inglés) consiste en un conjunto de programas necesarios para crear y mantener (NO diseñar) una BD.

Es un sistema de software de propósito general que **facilita** (NO permite, facilita) los procesos de definición, construcción y manipulación de BD.

OBJETIVOS DE UN DBMS

- Evitar redundancia e inconsistencia de datos
- Permitir acceso a los datos en todo momento
- Evitar anomalías en el acceso concurrente
- Evitar anomalías en el acceso concurrente
- Restricción a accesos no autorizados -> seguridad
- Suministro de almacenamiento persistente de datos (aún ante fallos)
- Integridad de datos
- Backups

COMPONENTES DE UN DBMS

DDL (Data definition language): Especifica el esquema de BD.

DML (Data manipulation language): Recuperar, agregar, quitar y modificar información.

MODELADO

NIVELES DE ABSTRACCIÓN

- **Nivel de Vista:** corresponde al nivel más alto de abstracción. En este nivel se describe parcialmente la BD, solo la parte que se desea ver. Es posible generar diferentes vistas de la BD, cada una correspondiente a la parte a consultar.

- **Nivel Lógico:** en este nivel se describe la BD completa, indicando qué datos se almacenarán y las relaciones existentes entre esos datos. El resultado es una estructura simple que puede conducir a estructuras más complejas en el nivel físico.

- **Nivel Físico:** este es el nivel más bajo de abstracción, en el cual se describe cómo se almacenan realmente los datos. Se detallan las estructuras de datos más complejas de bajo nivel.

MODELO DE DATOS

El modelo de datos es un conjunto de herramientas conceptuales que permite describir los datos y su semántica, sus relaciones y las restricciones de integridad y consistencia.

Un modelo de datos sirve para hacer más fácil la compresión de los datos de una organización.

Se modela para:

- Obtener la perspectiva de cada actor asociado al problema.
- Obtener la naturaleza y necesidad de cada dato.
- Observar cómo cada actor utiliza cada dato.

Existen tres grupos de modelos de datos:

- Modelos lógicos basados en objetos.
- Modelos lógicos basados en registros.
- Modelos físicos.

DISEÑO DE DATOS

MODELO CONCEPTUAL

Los modelos de datos conceptuales son medios para representar la información de un problema en un alto nivel de abstracción.

A partir de la definición de modelos de datos conceptuales es posible captar las necesidades del cliente/usuario respecto del Sistema de Información que necesita. Esta descripción de la realidad permite mejorar la interacción usuario-desarrollador, dado que disminuye la brecha existente entre la realidad del problema a resolver y el sistema a desarrollar.

Esta solución es genérica, y no se relaciona con el DBMS o cualquier producto específico elegido.

Un modelo conceptual debe poseer cuatro características o propiedades básicas.

Estas son:

- **Expresividad**, es decir, capturar y presentar de la mejor forma posible la semántica de los datos del problema a resolver.
- **Formalidad**, que requiere que cada elemento representado en el modelo sea preciso y bien definido, con una sola interpretación posible. Esta formalidad es comparable a la formalidad matemática.
- **Minimalidad**, característica que establece que cada elemento del modelo conceptual tiene una única forma de representación posible y no puede expresarse mediante otros conceptos.
- **Simplicidad**, que establece que el modelo debe ser fácil de entender por el cliente/usuario y el desarrollador.

Entre las propiedades definidas es posible analizar que algunas de ellas son opuestas. Por ejemplo, un modelo expresivo (rico en conceptos) puede no ser simple; es decir, en pos de expresar mejor los datos de un problema se puede perder simpleza.

MODELO ENTIDAD RELACIÓN

El modelo ER (entidad-relación) consta de tres componentes básicos:

- Entidades
- Relaciones
- Atributos

ENTIDADES

Una entidad representa un elemento u objeto del mundo real con identidad, es decir, se diferencia unívocamente de cualquier otro objeto o cosa, incluso siendo del mismo tipo.

Un conjunto de entidades es una representación que, a partir de las características propias de cada entidad con propiedades comunes, se resume en un **núcleo**.

RELACIONES

Las relaciones representan agregaciones entre dos (binaria) o más entidades. Describen las dependencias o asociaciones entre dichas entidades. Por ejemplo, la entidad Dolores García cursa (relación) la materia Introducción a las Bases de Datos (otra entidad).

Un conjunto de relaciones es una representación que, a partir de las características propias de cada relación existente entre dos entidades, las resume en un **núcleo**.

Cada relación debe tener definida la cardinalidad máxima y mínima. La cardinalidad la define el problema (o el cliente).

ATRIBUTO

Un atributo representa una propiedad básica de una entidad o relación. Es el equivalente a un campo de un registro.

Los atributos también tienen asociado el concepto de cardinalidad. Esto es, cuando se define un atributo se debe indicar si es o no obligatorio y si puede tomar más de un valor (polivalente).

Las expresiones posibles son:

- **1..1 monovalente obligatorio**; en caso de que un atributo presente esta cardinalidad, no debe ser incluida explícitamente en el modelo. Está implícita.
- **0..1 monovalente no obligatorio.**
- **1..n polivalente obligatorio** (podrían existir múltiples valores para este atributo).
- **0..n polivalente no obligatorio.**

Atributos compuestos

Los atributos compuestos representan a un atributo generado a partir de la combinación de varios atributos simples. Un ejemplo para ilustrar esta situación puede ser la dirección de una persona. Se podría optar por modelar la dirección como un solo atributo simple donde se indican la calle, el número y, eventualmente, piso y departamento.

Un atributo compuesto podría ser polivalente o no obligatorio. Lo mismo podría ocurrir con los atributos simples que lo componen.

Atributo derivado

Un atributo derivado es un atributo que aparece en el modelo, pero cuya información, si no existiera almacenada en él, podría igualmente ser obtenida (calculado).

Tener un atributo derivado repite información, pero se gana en performance.

Los atributos derivados, atentan contra la minimalidad. Pero no es necesariamente malo.

Se debe analizar en cada caso si conviene dejarlo o sacarlo.

IDENTIFICADOR

Un identificador es un atributo o un conjunto de atributos que permite reconocer o distinguir a una entidad de manera unívoca dentro del conjunto de entidades.

El concepto de identificador está ligado a los conceptos de claves primarias y candidatas.

Se marcan como identificador todas las claves candidatas a ser clave primaria. No es objetivo del modelo conceptual definirlo, eso se elige en el modelo físico.

Los identificadores pueden ser de dos tipos:

- **Simples o compuestos:** de acuerdo con la cantidad de atributos que lo conforman, el identificador es simple si está conformado por solo un atributo y es compuesto en el resto de los casos.
- **Internos o externos:** si todos los atributos que conforman un identificador pertenecen a la entidad que identifica, es interno; en caso contrario, es externo.

Según la opinión del profesor Pablo Thomas, todas las entidades deben tener si o si identificador; en caso de no encontrarse naturalmente, hay que forzarlo. La opinión del profesor Rodolfo Bertone, es que es mejor no inventar.

JERARQUÍAS DE GENERALIZACIÓN

La generalización permite extraer propiedades comunes de varias entidades o relaciones, y generar con ellas una superentidad que las aglutine. Así, las características compartidas son expresadas una única vez en el modelo, y los rasgos específicos de cada entidad quedan definidos en su subentidad.

Toda jerarquía tiene cobertura. La cobertura es la manera en que se relacionan el padre con los hijos.

Posibles coberturas: (T, E), (T, S), (P, E), (P, S)

> **Relaciones de jerarquía total:** Ocurre cuando no existen registros del supertipo que no estén relacionados con el subtipo.
>
> **Relaciones de jerarquía parcial:** Esto pasa cuando registros del supertipo no están relacionados con ninguno de los subtipos.
>
> **Relaciones de jerarquía exclusiva:** Ocurre cuando un registro de la superentidad está relacionado de forma exclusiva con las subentidades.
>
> **Relaciones de jerarquía superpuesta:** Esto pasa cuando un registro del supertipo puede estar relacionado con más de un registro de los subtipos.

Subconjuntos

Los subconjuntos representan un caso especial de las jerarquías de generalización. Hay problemas donde se tiene una generalización de la que se desprende **solamente un hijo**.

Solamente hay que considerar que no es necesario indicar la cobertura para los subconjuntos, SIEMPRE es **Parcial** y **Exclusiva**. Esto se debe a que no puede tratarse de una cobertura total; si no, la especialización y la generalización serían lo mismo, representarían los mismos atributos. Además, no puede ser superpuesta, dado que no hay una segunda especialización con la cual superponerse.

REVISIONES DEL MODELO

- **Compleción:** Un modelo es completo si representa todas las características del dominio de aplicación (análisis de requerimientos)
- **Corrección:** Un modelo es correcto si se usan con propiedad los conceptos del diagrama E-R
- **Minimalidad:** Cada aspecto aparece una sola vez en el esquema. Los atributos derivados atentan contra la minimalidad y los ciclos de relaciones pueden o no atentar contra la minimalidad.
- **Expresividad:** Representa los requerimientos de manera natural y se puede entender con facilidad
- **Extensibilidad:** Un modelo se adapta fácilmente a requerimientos cambiantes cuando puede descomponerse en partes, donde se hacen los cambios.
- **Legibilidad:** Debe ser prolijo, evitar que se crucen líneas, asimetrías, etc…

MODELO LÓGICO

El propósito de la generación de un modelo ER lógico es convertir el esquema conceptual en un modelo más cercano a la representación entendible por el SGBD. El principal objetivo del diseño conceptual consiste en captar y representar, de la forma más clara posible, las necesidades del usuario (modelo conceptual) definidas en el documento de especificación de requerimientos de un problema. Una vez cumplido este paso, el diseño lógico busca representar un esquema equivalente, que sea más eficiente para su utilización sobre computadora.

El diseño lógico del modelo de datos de un problema produce como resultado el esquema lógico de dicho problema, en función de cuatro entradas:

1. Esquema conceptual: es el resultado tangible de la etapa inmediata anterior. El esquema conceptual representa la solución, a juicio del analista, respecto del problema original. El esquema lógico a obtener debe representar la misma información disponible en el esquema conceptual.

2. Descripción del modelo lógico a obtener: aquí se deben definir las reglas que se aplicarán en el proceso de conversión. Esas reglas están ligadas al tipo de SGBD seleccionado.

3. Criterios de rendimiento de la BD: durante la fase de diseño conceptual, se consideraron los requerimientos del usuario. No obstante, hay otro tipo de necesidades que no se pueden definir sobre el modelo conceptual. Estas necesidades tienen que ver con requerimientos, en general, no funcionales del problema, como por ejemplo performance de la BD. Así, una regla puede dar alternativas de solución y el analista deberá optar por aquella que permita alcanzar los estándares de rendimiento definidos para el problema.

4. Información de carga de la BD: este concepto aparece, en cierta forma, ligado al concepto anterior. Cuando se genera el esquema lógico, el analista debe observar cada entidad e interrelación definida, y ver la probable evolución de la cantidad de información contenida en esas estructuras. De este modo, la decisión final sobre el esquema de una relación o entidad dependerá del número probable de elementos que la compondrán, con el propósito de mantener la performance bajo control. Cuanto más grande es un archivo de datos, más lento es el proceso de búsqueda, inserción o gestión de la información.

DECISIONES SOBRE EL DISEÑO LÓGICO

Las decisiones sobre el diseño lógico están vinculadas, básicamente, con cuestiones generales de rendimiento y con un conjunto de reglas que actúan sobre características del esquema conceptual que no están presentes en los SGBD relacionales.

Atributos derivados:

Un atributo es derivado si contiene información que puede obtenerse de otra forma desde el modelo. Es importante detectar dichos atributos, y en el diseño lógico se debe tomar la decisión respecto de dejarlos o no.

La ventaja de un atributo derivado es, básicamente, la disponibilidad de la información.

La desventaja de un atributo derivado radica en que necesita ser recalculado cada vez que se modifica la información que contiene.

Por lo tanto, la pauta sobre los atributos derivados es dejar en el modelo a todos aquellos que son muy utilizados, y quitar los que necesitan ser recalculados con frecuencia.

Ciclos de relaciones

Nuevamente, los ciclos de relaciones atentan contra la minimalidad, y nuevamente la decisión de dejarlo o no recae sobre el analista, que pasa por tener el modelo mínimo, o por que posteriormente el modelo implique menos tiempo de procesamiento

Atributos compuestos

Los atributos compuestos son atributos que están conformados por varios atributos simples. Los SGBD, en general, no soportan esta variante.

Hay tres opciones:

- Generar un único atributo que se convierta en la concatenación de todos los atributos simples.
- Definir todos los atributos simples separados, sin un atributo compuesto que los resuma. Manteniendo el acceso a cada uno de los datos de forma independiente.
- Generar una nueva entidad y relacionarla.

Atributos polivalentes

Ningún SGBD relacional permite que un atributo contenga valores múltiples determinados dinámicamente. Es decir, se puede tener un atributo con múltiples valores, pero la cantidad máxima debe ser previamente determinada.

Entonces, se genera una entidad nueva, relacionada.

Jerarquías

El modelo relacional no soporta el concepto de herencia; por consiguiente, las jerarquías no pueden ser representadas.

Básicamente, hay tres opciones para tratar una jerarquía. Estas opciones son las siguientes:

- Eliminar las especializaciones (subentidades o entidades hijas), dejando solo la generalización (entidad padre), la cual incorpora todos los atributos de sus hijos. Cada uno de estos atributos deberá ser opcional (no obligatorio).

- Eliminar la entidad generalización (padre), dejando solo las especializaciones. Con esta solución, los atributos del padre deberán incluirse en cada uno de los hijos. (Solo posible en cobertura (T, E))

- Dejar todas las entidades de la jerarquía, convirtiéndola en relaciones uno a uno entre el padre y cada uno de los hijos. Esta solución permite que las entidades que conforman la jerarquía mantengan sus atributos originales, generando la relación explícita ES_UN entre padre e hijos.

MODELO FÍSICO

El modelo físico es la representación final sobre computadora.

El modelo físico o relacional utiliza un conjunto de tablas para representar las entidades y las relaciones existentes, definidas en el modelo ER

La estructura básica del modelo relacional es la tabla. Cada tabla tiene una estructura conformada por columnas o atributos que representan, básicamente, los mismos atributos definidos para el modelo ER. Cada fila (registro) de la tabla se denomina tupla o relación, mientras que cada columna representa un atributo del registro.

El tipo de datos que describe los tipos de valores de un atributo se denomina dominio (entero, string, etc, son dominio).

PASAJE DE LÓGICO A FÍSICO

Se deben realizar los siguientes pasos:

- **Eliminar identificadores externos**
    - El externo, se coloca como atributo interno, repetido.

- **Seleccionar clave primaria** (de entre los identificadores del ML).
    - La clave seleccionada se hashea. Por tanto, se busca muy rápidamente por esa clave, pero se vuelve muy ineficiente acceder secuencialmente o ordenar por esa clave. Es recomendable (no obligatorio, recomendable), lo más conveniente y lo más eficiente utilizar el autoincremental del DBMS. El ID autoincremental no se puede modificar.

- **Convertir entidades a tablas**
    - Todas las entidades pasan a tablas

- **Convertir relaciones a tablas o a columna, según corresponda.**
    - **Si es Muchos a muchos:** Se convierte en tabla con los 2 IDs (y los atributos de la relación, si los tuvieren)

    - **Uno a muchos:** No se convierte en tabla. Se coloca el ID (fk) de la otra entidad del lado del que conoce sólo a uno
        - **Concepto de clave foránea (fk):** atributo de una tabla que en otra tabla es Clave Primaria y que sirven para establecer un nexo entre ambas estructuras.
        - **Concepto de integridad referencial:** Propiedad deseable de las BD. Asegura que un valor que aparece para un atributo en una tabla, aparezca además en otra tabla. Se puede elegir entre modificar/eliminar en cascada, o bloquear la operación, establecerla en null, o no hacer nada.

    - **Uno a muchos cobertura parcial del lado de 1:** Dos opciones. No se convierte en tabla, genera un posible valor nulo. Lo que puede generar inconvenientes si uno no es prolijo al manejar los valores nulos.
      O si se convierte en tabla, agregando ambos IDs (y los atributos de la relación, si los tuvieren)

    - **Relaciones ternarias:** Se convierten en tabla, con las 3 IDs.

    - **Relaciones recursivas:** Se trata de igual forma que una relación normal, pues una relación recursiva no deja de ser una relación binaria.

RESTRICCIONES EN EL MODELADO

Estas restricciones pueden ser violadas por las operaciones ABM (Alta, baja, modificación). Si se viola una restricción, la operación se rechaza.

Restricciones de dominio

Especifican que el valor de cada atributo debe ser un valor del dominio de ese tipo. (Si un atributo en Entero, no puede contener el valor “Hola mundo” ni “2.5”)

Restricciones de clave

Evita que el valor del atributo una clave unívoca genere valores repetidos

Restricciones sobre nulos

Evita que un atributo tome valor nulo en caso de no ingresarle valor

Restricciones de integridad

Ningún valor de la clave primaria debe ser nulo

Restricciones de integridad referencial

Se especifica entre dos relaciones (tablas) y sirve para mantener la consistencia entre tuplas (datos) de las dos relaciones (tablas).

Establece que una tupla en una tabla que haga referencia a otra tabla deberá referirse a una tupla existente en esa tabla.

CONCEPTO DE DEPENDENCIA FUNCIONAL

Es una **restricción** (que se podría sumar a las restricciones anteriormente mencionadas). Es una restricción entre dos conjuntos de atributos de una BD.

Las dependencias funcionales no son ni buenas ni malas.

Hay dependencia funcional si un atributo X de una tabla, depende funcionalmente de otro atributo Y de la misma tabla. Es decir, conociendo X, puedo saber el valor de Y.

SIEMPRE hay dependencias funcionales. Porque **todos** los atributos de una tabla dependen funcionalmente de las clave unívocas (la clave primaria, y las no primarias).

Dependencia funcional completa

En caso de una clave primaria compuesta, una dependencia funcional se denomina **completa** si B depende de A pero de ningún subconjunto de A.

Dependencia funcional parcial (de 2FN)

Una dependencia funcional es parcial si existe algún atributo que pueda eliminarse de A y la dependencia continua verificándose

Dependencia funcional transitiva (de 3FN)

Una condición en la que hay 3 atributos.

Si A determina B, y B determina C, entonces A determina C.

Dependencia multivaluada (de 4FN)

La posible existencia de una DM se debe a 1FN que impide que una tupla tenga un conjunto de valores diferentes.

Entonces una dependencia multivaluada, es cuando dado un atributo A, puedo obtener múltiples valores para un atributo B ($A \rightarrow\rightarrow B$)

Por ejemplo, fecha de nacimiento multidetermina nombre, porque en una fecha de nacimiento pueden nacer muchas personas.

Dependencia de combinación (5FN)

Una dependencia de combinación es una propiedad de la descomposición que garantiza que no se generen tuplas espurias al volver a combinar relaciones mediante una operación del álgebra relacional. ($|X|$)

NORMALIZACIÓN

El proceso de normalización de una BD consiste en aplicar una serie de reglas a las tablas obtenidas a partir del diseño físico. Una BD debe normalizarse para evitar la redundancia de los datos, dado que cuando se repite innecesariamente la información, aumentan los costos de mantenerla actualizada. Además, la normalización ayuda a proteger la integridad de la información de la BD.

La ventaja de utilizar una BD normalizada consiste en disponer de tablas, cuyos datos serán para el usuario de fácil acceso y sencillo mantenimiento.

Definición: Técnica de diseño de BD que comienza examinando los nexos que existen entre los atributos (dependencias funcionales).

Propósito: Producir un conjunto de relacione (tablas) con una serie de propiedades partiendo de los requisitos de datos de una organización.

Entonces: Una base de datos que contiene sólo dependencias funcionales deseadas, está normalizada.

Ni para el usuario ni para el cliente es importante saber si la BD está normalizada o no.

El proceso de normalización es incremental y cada vez más restringido (por tanto menos vulnerable a repetición de información y a que la información se corrompa).

La primera forma normal se aplica siempre, es obligatorio. Las otras, son opcionales y se podrían obviar en función del problema específico.

PRIMERA FORMA NORMAL

Una tabla que tenga atributos polivalentes no está en primera forma normal.

Un modelo está en 1FN si todos los atributos son solo monovalentes.

SEGUNDA FORMA NORMAL

Una tabla que tenga atributos que dependan parcialmente de otras tablas no está en 2FN.

Un modelo está en 2FN si y sólo si está en 1FN y ninguna tabla tiene dependencias parciales.

TERCERA FORMA NORMAL

Una tabla que taringa atributos que dependen transitivamente de otros, no está en 3FN.

Un modelo está en 3FN si y sólo si está en 2FN y ninguna tabla tiene dependencias transitivas.

FORMA NORMAL DE BOYCE CODD

Una tabla que tenga atributos de acuerdo a la definición de Boyce Codd de otro no está en BCNF

Un modelo está en BCNF si y sólo si está en 3FN y para todas las dependencias los determinantes (Determinante → Y) son todas claves unívocas (primaria y candidatas).

Llevar una BD a BCNF evita redundancia de información pero pierdo control de los datos (porque pierdo una clave candidata). Si se deja en 3FN mantengo el control pero duplicó información.

CUARTA FORMA NORMAL

Un modelo está en 4FN si y sólo si está en BCNF y todas las tablas tienen sólo dependencias multivaluadas **triviales**.

QUINTA FORMA NORMAL

Un modelo está en 5FN si y sólo si está en 4FN y ninguna tabla tiene dependencia de combinación.

LENGUAJES DE CONSULTA

Son lenguajes utilizados para operar con la BD. Realizan operaciones de Alta, Baja, Modificación y Consulta.

ALGEBRA RELACIONAL

Es un lenguaje de consultas teórico. Es procedimental.

Las operaciones se realizan sobre una (unitarias) o dos (binarias) tablas de entrada que generan una nueva tabla como resultado.

Operaciones unitarias:

- Selección
- Proyección
- Renombre

Operaciones binarias:

- Producto cartesiano
- Unión
- Diferencia

CONSULTAS

Las consultas representan el 80% de las operaciones sobre una BD.

Selección

Dada una tabla, devuelve todas las tuplas que cumplen una condición booleana.

$$\sigma_{\text{Estadocivil = “casado” or Estadocivil = “divorciado”}} \text{(asociado)}$$

Proyección

Dada una tabla, la devuelve con columnas emitidas.

$$\pi_{\text{nombre, DNI}} (\text{asociado})$$

Producto cartesiano

Conecta dos entidades de acuerdo a la definición matemática de la operación. Junta todas las filas de una tabla, con todas las filas de la otra. Si no se hace correctamente, genera tuplas espurias (incorrectas).

$$\pi_{\text{asociado.nombre, ciudad.nombre}} (\sigma_{\text{asociado.cpCiudad = ciudad.cpCiudad}}(\text{asociado x ciudad}))$$

Renombre

El renombre es la operación que permite utilizar la misma tabla dos veces en la misma consulta.

$$\rho_{\text{city}} (\text{ciudad})$$

Unión

El operando unión es equivalente a la unión matemática de conjuntos. Se aplica a dos tablas o relaciones de la BD, generando una nueva tabla cuyo contenido es el contenido de cada una de las tuplas de las tablas originales involucradas.

$$\text{tabla1} \cup \text{tabla2}$$

La unión debe tener sentido. Es decir cuando se utiliza el operando de unión en AR, debe tenerse en cuenta que los dos conjuntos a unir deben ser unión-compatibles. Esto es, unir dos conjuntos que tengan la misma cantidad de atributos, y además el i-ésimo atributo de la primera tabla y el i-ésimo atributo de la segunda tabla deben tener el mismo dominio (i: 1..n). Caso contrario, el resultado obtenido no representa ninguna información útil.

Diferencia

El operando diferencia es equivalente a la diferencia matemática de conjuntos. Se aplica a dos tablas o relaciones de la BD, generando una nueva tabla cuyo contenido es el contenido de cada una de las tuplas de la primera tabla que no están en la segunda.

$$tabla1 - tabla2$$

Cuando se utiliza el operando de diferencia en AR, debe tenerse en cuenta que las dos tablas deben ser unión-compatibles.

OPERADORES ADICIONALES

Además de los seis operadores básicos definidos, existen otros operadores que otorgan mayor expresividad al AR, facilitando en muchos casos la construcción de las operaciones de consulta.

Producto natural

El producto natural directamente reúne las tuplas de la primera tabla que se relacionan con la segunda tabla, descartando las tuplas no relacionadas.

$$Alumnos \mid\times\mid localidades$$

Se debe notar que entre las tablas sobre las que se realiza el producto natural debe existir un atributo común, o sea, que tienen un atributo que se llama igual y que representa la misma información. En su defecto, de no existir un atributo común, el producto natural y el producto cartesiano actúan de la misma forma. Si hay más de un atributo que se llama igual, el producto natural no puede usarse porque el álgebra no sabría con cuál de los dos aplicarlo.

Intersección

El operando de intersección es equivalente a la intersección matemática de conjuntos. Se aplica a dos tablas o relaciones de la BD, generando una nueva tabla con las tuplas comunes a ambas tablas.

$$tabla1 \cap tabla2$$

Asignación

El operador de asignación no aporta una funcionalidad extra al AR, sino que solamente tiene como objetivo otorgar mayor legibilidad a las consultas. Con el operador de asignación se puede generar una subconsulta y volcar su resultado en una variable temporal de tabla, la cual puede ser utilizada posteriormente.

El operador de asignación es ←

División

Dadas dos tablas R1 y R2, R1 ÷ R2 tiene como resultado los valores de atributos de R1, que se relacionan con todas las tuplas de R2.

R1 % R2 si, y sólo si, el esquema de R2 está incluido en el esquema de R1, es decir que todos los atributos de R2 son atributos de R1.

ACTUALIZACIONES

El lenguaje original de AR permite realizar, además, las operaciones básicas de actualización: alta, baja y modificación de datos contenidos en la BD.

Altas

La operación de alta permite agregar una nueva tupla a una tabla existente. Para ello, se debe realizar una operación de unión entre la tabla y una tupla escrita de manera literal.

$$\text{Alumnos} \leftarrow \text{Alumnos} \cup \{\text{“Delia”, “30777987”, 2, 1}\}$$

Bajas

La operación de baja permite quitar una tupla de una tabla. Para ello, se debe realizar una operación de diferencia entre la tabla y una tupla escrita de manera literal. Ejemplo 14: Se debe quitar de la tabla alumnos al alumno López.

$$\text{Alumnos} \leftarrow \text{Alumnos} - \sigma_{\text{nombre = ‘Lopez’}}(\text{alumnos})$$

Modificación

A diferencia de las dos operaciones de actualización anteriores, que se resolvían a partir de la utilización de operadores básicos, la operación de modificación necesita representar el cambio con un operador, δ. El formato genérico de la operación es: δatributo = valor_nuevo (tabla)

$$\delta_{\text{ dni = “34567231”}}(\sigma_{\text{nombre = “Caporal”}}(\text{alumnos}))$$

Nótese que, previo a realizar la modificación, se debe seleccionar la tupla sobre la que se desea hacer el cambio. Si no se hiciera de esa forma, todos los dni de todos los alumnos de la tabla se modificarían.

LENGUAJE DE CONSULTAS ESTRUCTURADO (SQL)

DDL (Data definition language)

- CREATE DATABASE
- DROP DATABASE
- CREATE TABLE
- ALTER TABLE
- DROP TABLE

DML (Data modification language)

Estructura Básica

La estructura básica de una consulta SQL tiene el siguiente formato:

```
SELECT lista_de_atributos

FROM lista_de_tablas

WHERE condición
```

CONSULTA

Sentencias básicas

SELECT equivale a la proyección de AR

- Con **distinct** se eliminan tuplas duplicadas
- Con * se incluyen los atributos de las tablas que aparecen en el from
- Con **concat** se puede formatear la presentación

FROM equivale al producto cartesiano

INNER JOIN equivale al producto natural de AR

- Existen **NATURAL JOIN**, **LEFT JOIN**, **RIGHT JOIN** y **FULL OUTER JOIN**

WHERE equivale a la selección

- La sentencia Where **NO** es obligatoria
  *!!!!!!!!!!! PERO NO TE OLVIDES → https://www.youtube.com/watch?v=i_cVJgIz_Cs
- Puede llevar **AND** o **OR**
- Con **between** se filtra los que estén entre 2 valores
- Con **like** filtramos por subcadenas
- Con **day**, **month** o **year** manejamos **dates**

ORDER BY permite ordenar las tuplas

- Con **Desc** se elige orden descendente. (ascendente es el predeterminado)

UNION agrupa las tuplas resultantes de dos subconsultas

- **No** genera valores repetidos
- **Union all** si genera valores repetidos

INTERSECCIÓN es la intersección

EXCEPT es la diferencia

- No todos los DBMS implementan la diferencia

GROUP BY permite agrupar un conjunto de tuplas por algún criterio. Suele usarse en conjunto con las funciones de agregación. Se debe agrupar por claves unívocas

CREATE VIEW crea una subtabla temporal

IN anida una subconsulta, y pregunta pertenencia a un conjunto

EXIST devuelve verdadero si la subconsulta argumento no es vacía

Funciones de agregación

Son funciones que no se aplican a un tupla específica, sino que se aplican a un conjunto de tuplas.

- **NO** pueden aparecer en el where. (porque el where filtra una tupla, y la FA actúa sobre un conjunto de tuplas)

AVG: calcula el promedio

- Aplicable a atributos numéricos

MIN: retorna el elemento más chico, según el atributo especificado

MAX: retorna el elemento más grande, según el atributo especificado

SUM: realiza la suma matemática

- Aplicable a atributos numéricos

COUNT: cuenta la cantidad de tuplas resultantes

MODIFICACIÓN

Inserción

INSERT INTO

Borrado (https://www.youtube.com/watch?v=i_cVJglz_Cs !!!!!)

DELETE FROM

WHERE

Actualización

UPDATE

SET

WHERE

OPTIMIZACIÓN DE CONSULTAS

Cada resolución a un problema puntual tiene ventajas y desventajas respecto de otras alternativas. En algunos casos, resulta más sencillo para el programador resolver la consulta utilizando una determinada heurística; en otras situaciones, el estado actual de la BD hace más conveniente otro tipo de resolución. En general, el diseñador de la consulta trata de centrar sus objetivos en obtener el resultado esperado, en vez de analizar con detalle cuál sería la mejor secuencia de pasos a aplicar, para lograr una solución en el menor tiempo de respuesta posible.

COSTO DE LA EJECUCIÓN DE UNA CONSULTA

- Costo de acceso a almacenamiento secundario
- Costo de computo (operaciones sobre RAM)
- Costo de comunicación (desde que se envía la consulta, hasta que llega el resultado)

Cuando el DBMS recibe una consulta, la transforma a un formato interno (Parser), sobre eso realiza un proceso de optimización. Es por eso, que hasta el momento no se hizo hincapié en la eficiencia, pues el DBMS se encarga. Luego de eso selecciona los índices que considere para hacer el uso más óptimo posible de las tuplas y para minimizar los accesos a disco.

Repito, priorizamos **legibilidad** antes que eficiencia, porque de la optimización se encarga el DBMS.

Costo de selección

(CantidadTuplasTabla / CantidadValores (at, tabla)) * CantidadBytesTabla

Costo proyección

(CantidadBytes at1 + CantidadBytes at2 + … + CB atN) * CantidadTuplasTabla

Costo producto cartesiano

(CantidadTuplas t1 * CantidadTuplas t2) * (CantidadBytes t1 + CantidadBytes t2)

OPTIMIZACIÓN LÓGICA

- **Conviene realizar la selección lo antes posible.** Ya que todas las operaciones siguientes, se aplican sobre una menor cantidad de tuplas.
- **Conviene realizar más proyecciones** para disminuir la cantidad de información que se almacena en buffers de memoria. También aplica para la unión, intersección y diferencia si minimizan el número de resultados.

SEGURIDAD E INTEGRIDAD DE DATOS

Preservar la integridad de los datos contenidos en una BD es una de las principales tareas que debe llevar a cabo el diseñador de la BD. Para poder asegurar la información en todo momento, es necesario tomar algunos recaudos.

Uno de los primeros aspectos a considerar tiene que ver con asegurar la información contra destrucción malintencionada.

Sin embargo, existen muchas otras causas que pueden atentar contra el contenido de la BD sin que puedan atribuirse a la mala intención. Suponga que se daña el disco rígido de la computadora, que se cae un enlace de comunicaciones con el servidor de la BD o que sencillamente se corta la provisión de energía eléctrica. Si algún usuario se encuentra en ese momento operando contra la BD, puede ocurrir que se pierda parte del trabajo realizado. Esto acarrea un inconveniente importante; si el trabajo se realizó parcialmente, puede significar que parte de la BD tenga valores actualizados y otra parte no. Esta situación claramente viola el concepto de integridad de los datos. Una operación de usuario realizada de modo parcial atenta fuertemente contra la integridad de la información.

TRANSACCIONES

Una **transacción** es un conjunto de instrucciones que actúa como unidad lógica de trabajo. Esto es, una transacción se completa cuando se ejecutaron todas las instrucciones que la componen. En caso de no poder ejecutar todas las instrucciones, ninguna de ellas debe llevarse a cabo.

PROPIEDADES

Para garantizar que una transacción mantenga la consistencia de la BD, es necesario que cumpla con cuatro propiedades básicas (ACID):

- **Atomicidad:** garantiza que todas las instrucciones de una transacción se ejecutan sobre la BD, o ninguna de ellas se lleva a cabo. No es posible ejecutar una transacción a medias, o todo o nada.

- **Consistencia:** la ejecución completa de una transacción lleva a la BD de un estado consistente a otro estado de consistencia. Para esto, la transacción debe estar escrita correctamente. En el ejemplo del pago de un servicio, si la transacción resta $100 de la cuenta del usuario y suma 120 en la cuenta del prestatario, no resulta en una transacción consistente. Es responsabilidad del programador.

- **Aislamiento** *(isolation en inglés)***:** cada transacción actúa como única en el sistema, ignora el resto de las transacciones que se ejecutan concurrentemente en el sistema. Esto significa que la ejecución de una transacción, T1, no debe afectar la ejecución simultánea de otra transacción, T2.

- **Durabilidad:** una vez finalizada una transacción, los efectos producidos en la BD son permanentes. Realiza los cambios incluso si hay fallos en el sistema.

ESTADOS DE UNA TRANSACCIÓN

Una transacción puede estar en alguno de los siguientes cinco estados:

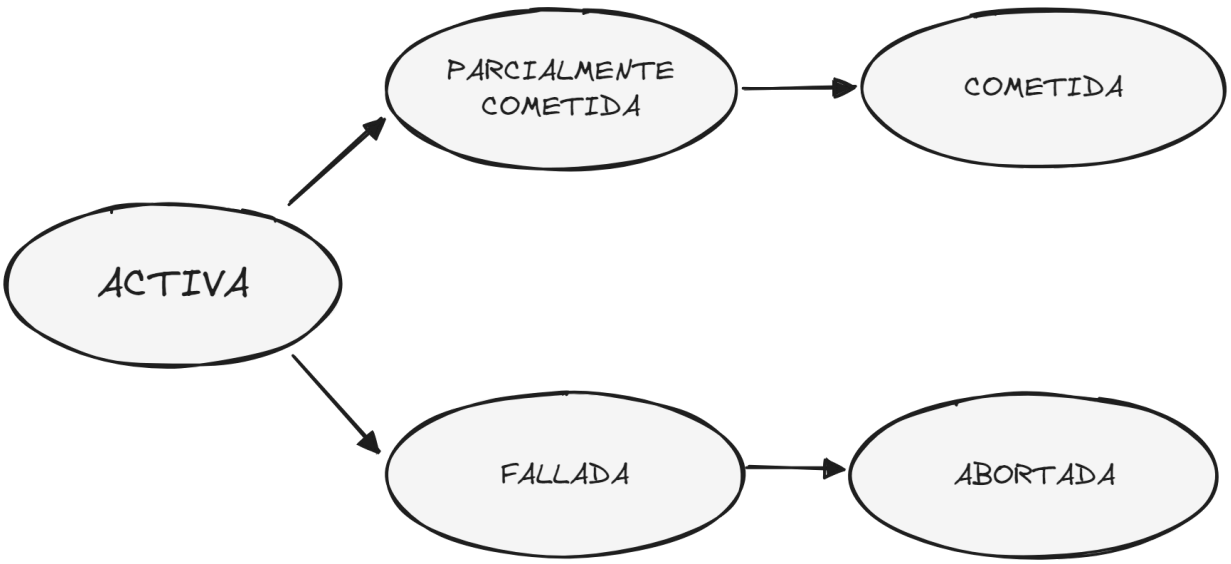
1. Activo: este estado se tiene desde que comienza su ejecución y se mantiene hasta completar la última instrucción, o hasta que se produzca un fallo.

2. Parcialmente cometido: este estado, también conocido como parcialmente finalizado, se alcanza en el momento posterior a que la transacción ejecuta su última instrucción. Esta última instrucción, por ejemplo, un UPDATE, se ejecuta sobre un buffer de memoria, que es volátil. Lo que asegura este estado es información correcta en buffer de RAM. Si se corta la luz, la información se podría perder generando inconsistencia con la BD, por ende se pasa a Fallado y se aborta. Por eso existe este estado, previo al “Cometido”.

3. Fallado: a este estado se arriba cuando la transacción no puede continuar con su ejecución normal.

4. Abortado: este estado garantiza que una transacción fallada no ha producido ningún cambio en la BD, manteniendo la integridad de la información allí contenida. Se está en este estado después de haber retrocedido la transacción y restablecido la BD al estado anterior al comienzo de la transacción.

5. Cometido: este estado, también conocido como finalizado, se obtiene cuando la transacción finalizó su ejecución, y sus acciones fueron almacenadas correctamente en memoria secundaria. (Se pasó del buffer a la memoria persistente de la BD).



ENTORNO DE LAS TRANSACCIONES

- Monousuario: No es posible que se generen problemas de Consistencia, Aislamiento, ni Durabilidad. El único inconveniente posible es la Atomicidad.

¿Qué hacer luego de un fallo?

Re-ejecutar la transacción fallada → No sirve

Dejar el estado de la BD como está → No sirve

El problema fue modificar la BD sin seguridad de que la transacción va a cometer.

Hay dos soluciones:

- Bitácora
- Doble paginación.

BITÁCORA (log)

El método de bitácora (log) o registro histórico plantea que todas las acciones llevadas a cabo sobre la BD deben quedar registradas en un archivo histórico de movimientos.

El método de bitácora consiste en registrar, en un archivo auxiliar y externo a la BD, todos los movimientos producidos por las transacciones **antes** de producir los cambios sobre la BD misma. De esta forma, en caso de producirse un error, se tiene constancia de todos los movimientos efectuados. Con este registro de información, es posible garantizar que el estado de consistencia de la BD es alcanzable en todo momento, aún ante un fallo.

Contenido de la bitácora:

T0 comienza

T0, dato1, valorViejo, valorNuevo

T0, dato2, valorViejo, valorNuevo

…

T0, datoN, valorViejo, valorNuevo

T0 finaliza

En caso de que la transacción falle:

Se agrega: T0 aborta

Si se cortó a la mitad:

No habrá ni un “finaliza” ni un “aborta”.

TÉCNICAS DE BITÁCORA

Modificación diferida de una base de datos

Las operaciones **write** se aplazan hasta que la transacción esté parcialmente cometida, en ese momento se actualiza la bitácora y la BD.

No importa en qué orden aparezca la operación write en la transacción. Primero se ejecutan el resto, y al final los write.

Si en la bitácora hay **Start y Commit**, debo re-ejecutar la transacción, así me aseguro que la BD queda bien.

Si hay **Start** y **NO hay Commit**, no hay que hacer nada.

Modificación inmediata de una base de datos

La actualización de la BD se realiza mientras la transacción está activa y se va ejecutando.

Se necesita el valor viejo, pues los cambios se fueron efectuando.

Ante un fallo:

- **Rehacer** *(REDO)*, si hay **Start** y **Commit**
- **Deshacer** *(UNDO)*, si hay **Start** y **NO hay Commit**

Este protocolo es más complejo que el de modificación diferida, pues necesita 2 operaciones extra, pero como ventaja tiene más distribuida la carga de trabajo.

CONDICIÓN DE IDEMPOTENCIA DE UNA BITÁCORA

Si hago o rehago 1 y 1000 veces una transacción de bitácora, el resultado siempre es el mismo.

PUNTOS DE VERIFICACIÓN

Para no revisar la bitácora entera luego de un fallo, se colocan **checkpoints,** asegurando que del checkpoint para atrás, todas las transacciones son correctas. Cuando los resultados de una transacción se guardan en memoria secundaria, se almacena un checkpoint en bitácora.

DOBLE PAGINACIÓN

El método alternativo al de bitácora recibe el nombre de doble paginación o paginación en la sombra. Este método plantea dividir al BD en nodos virtuales (páginas) que contienen determinados datos.

Se generan dos tablas en disco y cada una de las tablas direcciona a los nodos (páginas) generados.

El método trabaja de la siguiente forma. Ante una transacción que realiza una operación de escritura sobre la BD, la secuencia de pasos es la siguiente:

1. Se obtiene desde la BD el nodo (página) sobre el cual debe realizarse la escritura (si el nodo está en memoria principal, este paso se obvia).

2. Se modifica el dato en memoria principal y se graba en disco, en un nuevo nodo, la página actual que referencia al nuevo nodo.

3. Si la operación finaliza correctamente, se modifica la referencia de la página a la sombra para que apunte al nuevo nodo.

4. Se libera el nodo viejo.

En caso de producirse un fallo, luego de la recuperación, se desecha la página actual y se toma como válida la página a la sombra.

Este método presenta dos **ventajas** evidentes sobre el método de bitácora: se elimina la sobrecarga producida por las escrituras al registro histórico, y la recuperación ante un error es mucho más rápida (se recupera la página a la sombra como página actual, descartando la página actual insegura).

Sin embargo, el método también tiene algunas **desventajas** importantes. Primeramente, exige dividir la BD en nodos o páginas, y esta tarea puede ser compleja. En segundo lugar, la sobrecarga del método aparece en entornos concurrentes

* **Falta contenido sobre seguridad e integridad en entornos concurrentes**