Resumen Teórico de Promoción
“Diseño de Bases de Datos”

Contenido

Clase 1 "Bases de Datos"

Bases de Datos

¿Qué es una Base de Datos?

- Colección coherente de **archivos** de datos con significados inherentes relacionados diseñados para servir a múltiples aplicaciones.
- Un dato representa hechos conocidos que pueden registrarse y que tienen un resultado implícito.

Propiedades implícitas de una Base de Datos

- Representa algunos aspectos del mundo real, a veces denominado Universo de Discurso.
- Un conjunto aleatorio de datos no puede considerarse una BD. O sea los datos deben tener cierta lógica.
- Se diseña, construye y completa de datos para un propósito específico
- Sustentada físicamente en archivos en dispositivos de almacenamiento persistente de datos

DBMS (Data Base Management System) o SGBD (Sistema Gerenciador de Bases de Datos)

¿Qué es un DBMS o SGBD?

- Sistema de software de propósito general que facilita los procesos de definición, construcción y manipulación de Bases de Datos.

Objetivos de un DBMS

- Evitar **redundancia** e **inconsistencia** de **datos.**
- Permitir **acceso** a los datos en **todo** momento.
- Evitar anomalías en el acceso **concurrente**.

- Restricción a **accesos no autorizados (Seguridad).**
- Suministro de almacenamiento **persistente** de datos.
- **Integridad** en los **datos**
- **Backups**.

Componentes de un DBMS

- **DDL (Data Definition Languaje)**
  - ❖ Especifica el esquema de una Base de Datos generando como resultado un Diccionario de Datos.
- **DML (Data Manipulation Languaje)**
  - ❖ **Se encarga de**
    - ➢ **Consultar** información.
    - ➢ **Agregar** información.
    - ➢ **Borrar** información.
    - ➢ **Modificar** información.
  - ❖ **DML Procedimentales (SQL)**
    - ➢ Requieren que el usuario especifique **qué** datos se van a mostrar y **cómo** se van a obtener.
  - ❖ **DML No Procedimentales (QBE)**
    - ➢ Requieren que el usuario especifique **qué** datos se van a mostrar pero no el **cómo** se obtienen.

Actores involucrados con una Base de Datos

- **DBA o ADB**
  - ❖ Autoriza accesos, coordina y vigila la utilización de recursos de hardware y software, responsable ante problemas de violación de seguridad o respuesta lenta del sistema.
- **Diseñador de Base de Datos**
  - ❖ Definen la estructura de la BD de acuerdo al problema.
- **Analistas de Sistemas**
  - ❖ Determinan los requerimientos de los usuarios finales, generando la información necesaria para el diseñador.
- **Programadores**
  - ❖ Implementan las especificaciones de los analistas utilizando la Base de Datos.

- **Usuarios finales.**

Clase 2 "Abstracción, Modelo de Datos y Modelo Conceptual"

Abstracciones

- Proceso que permite seleccionar algunas características de un conjunto de objetos del mundo real, dejando de lado aquellos rasgos que no son de interés.

Niveles de Abstracción

1) **Visión**
   - ❖ Cada actor ve los datos que le interesan a él por lo tanto se pueden generar muchas **Vistas** para la misma Base de Datos.
2) **Conceptual**
   - ❖ Nivel donde se resumen todas las necesidades de todos los actores involucrados, generando un único **Modelo de Datos** que contiene toda la información (**Datos y Relaciones**).
3) **Físico**
   - ❖ Describe cómo se almacenan realmente los datos (**Archivos y Hardware**).

Abstracción de Clasificación

- Se usa para definir una clase.
- **Clase**:
  - ❖ Declaración o abstracción de objetos.
  - ❖ Se origina a partir de las características comunes de los objetos que la componen.
- Permite identificar los campos o atributos de los elementos individuales de datos.

Abstracción de Agregación

- Define una nueva clase a partir de un conjunto de otras clases que representan sus partes componentes.
- Permite agrupar los campos o atributos formando registros de datos.
- Genera correspondencia entre clases.

Abstracción de Generalización

- Define una relación de subconjunto entre los elementos de dos o más clases.
- Las especialidades o hijos heredan las características del padre.
- Genera correspondencia entre clases.

Propiedades de Correspondencia entre Clases

- **Agregación Binaria**
  - ❖ Correspondencia existente entre 2 clases diferentes.
  - ❖ Permite definir el concepto de **cardinalidad**.
- **Generalización**
  - ❖ Permite definir el concepto de **cobertura** de una **generalización**.

Modelo de Datos

- Conjunto de herramientas conceptuales que permiten describir la información que es necesario administrar para un **Sistema de Información**, las relaciones existentes, la semántica asociada y las restricciones de consistencia.
- Sirven para hacer más fácil la comprensión de 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.

Modelos Basados en Objetos

- Se utilizan para describir los datos de acuerdo con la visión que cada usuario tiene respecto a la Base de Datos. Permiten generar vistas de las necesidades que posee cada actor.
- Poseen Estructura flexible.
- Especifican restricciones explícitamente.
- Modelos de este tipo:
    - ❖ **Modelo Entidad-Relación.**
    - ❖ **Modelo Orientado a Objetos.**
    - ❖ Modelo de datos semántico.
    - ❖ Modelo de datos funcional.

Modelos Basados en Registros

- Permiten describir los datos desde la perspectiva de cada usuario.
- Permiten especificar la estructura lógica completa de la Base de Datos.
- Llevan su nombre debido a que la Base de Datos se estructura como registros de longitud fija, conformados por campos o atributos.
- Se dispone de lenguaje asociado para expresar consultas.
- Modelos de este tipo:
    - ❖ Modelo Orientado a Objetos.
    - ❖ **Modelo Relacional** (ejemplo más representativo de esta categoría).
    - ❖ Modelo Jerárquico.
    - ❖ Modelo de Red.

Modelo Físico de Datos

- Modelo actualmente inexistente, utilizado en los comienzos del diseño de Base de Datos.

Etapas del Modelado de Datos

1) **Conceptual**
    - ❖ Representación abstracta.
    - ❖ Integración de vistas.

- ❖ Es un modelo “maqueta” que se usa para **comunicarse con los clientes**, debe ser claro y entendible.
- ❖ Se genera una gran **interacción** y **participación** del **cliente**.
- ❖ Genérico, alejado del tipo de DBMS y alejado de un producto específico.

2) **Lógico**
   - ❖ Modelo donde se depura y convierte el **modelo conceptual** a términos informáticamente resolubles.
   - ❖ Empieza la especificación y el acercamiento al tipo de DBMS, todavía es alejado a un producto particular.
3) **Físico**
   - ❖ Determinar estructuras de almacenamiento físico.
   - ❖ Generar el modelo con uso del **DBMS** con una implementación directa en una computadora.
   - ❖ Es específico y orientado a un producto.

Modelo	Tipo de SGBD	SGBD específico
Conceptual	No debe decidirse	No debe decidirse
Lógico	Debe decidirse	No debe decidirse
Físico	Debe decidirse	Debe decidirse

Modelo Entidad-Relación

- Se basa en la concepción del mundo real como un conjunto de objetos llamados **Entidades** y las **Relaciones** existentes entre dichas **Entidades.**

Modelo Conceptual Entidad-Relación

Objetivos

- Representar la información de un problema en un alto nivel de abstracción
- Captar la necesidad de un cliente respecto del Sistema de Información que necesita.

- Mejora la interacción usuario/cliente - desarrollador disminuyendo la brecha entre la realidad del problema y el sistema a desarrollar

Características

- **Expresividad**
  - **Capturar** y **presentar** de la mejor forma posible la **semántica del problema.**
- **Formalidad**
  - Cada elemento representado en el modelo tiene que ser **preciso** y **bien definido**, con una sola **interpretación** posible.
- **Minimalidad**
  - Cada elemento tiene una **única forma de representación posible.**
- **Simplicidad**
  - El modelo debe ser fácil de **entender** por el **cliente/usuario** y por el **desarrollador**.
- **Característica adicional, aunque obligatoriamente deben estar presentes las 4 anteriores:**
  - El modelo debe ser fácil de leer. **Un modelo expresivo, formal, mínimo y simple debería ser fácil de leer.**

Componentes

- **Entidades**
  - Representa un elemento u objeto del mundo real con identidad
  - Se diferencia **unívocamente** de cualquier otro tipo de objeto.
- **Conjunto de Entidades**
  - Representación que, a partir de las características propias de cada entidad con propiedades comunes, se resume en un núcleo.
  - Se representan mediante un **rectángulo**.

- **Relaciones**
  - Representan **agregaciones** entre **dos o más entidades** (Binarias, Ternarias, N-arias).

- **Pueden ser recursivas**
  - Relación que une dos entidades del mismo conjunto.
- Describen las **dependencias** o **asociaciones** entre dichas entidades.
- **Cardinalidad**
  - Grado de relación existente en una **agregación**, presente cuando se definen las relaciones existentes entre **entidades**.
  - Cada **relación** debe tener una **cardinalidad mínima y una máxima.**
  - Es fundamental definir **precisamente** las **cardinalidades**, por lo tanto, **si se duda en un valor, la solución es consultar al cliente/usuario.**

- **Conjunto de Relaciones**
  - Representación que, a partir de las características propias de cada relación existente entre dos entidades, las resume en un núcleo.
  - Se representan mediante un **rombo**.
  - Puede existir más de un conjunto de relaciones entre 2 conjuntos de entidades.

- **Atributos**
  - Representa una propiedad básica de una entidad o relación.
  - Tienen asociado el concepto de **cardinalidad**.
    - Se debe indicar si es **obligatorio o no obligatorio**.
    - Se debe indicar si es **monovalente o polivalente**.

Componentes Adicionales

- **Jerarquía de Generalización**
  - Permiten extraer **propiedades comunes** de varias entidades o relaciones, y generar con ellas una **superentidad** que las **contenga**.
  - Las **características compartidas** son expresadas una única vez en el modelo.
  - Las **características específicas** de cada entidad quedan definidas en las **subentidades/especializaciones**.

- **Cobertura**
  - Depende del problema y si no se cuenta con la información necesaria para definirla, **consultar al cliente/usuario.**
  - Grado de **relación** entre **padre** e **hijos**.
  - **Primera cobertura:**
    - **Total**: Cada elemento del padre **está contenido** en alguno de los **hijos**.
    - **Parcial**: Pueden existir elementos del padre que **no estén contenidos** en los **hijos**.
  - **Segunda cobertura**:
    - **Exclusiva**: Un elemento del padre **solo puede estar contenido en un hijo.**
    - **Superpuesta**: Un elemento del padre **puede estar contenido en más de un hijo.**
- Hay concepto de **Herencia**
  - Las subentidades **heredan atributos e identificadores** de la **superentidad**.

- **Subconjuntos**
  - Caso especial de las **Jerarquías**.
  - Generalización de la que se desprende solo **una subentidad**.
  - No es necesario indicar la **cobertura**, siempre es **(parcial, exclusiva)**.
    - No es **total** porque si no la subentidad y la superentidad serían lo mismo.
    - No es **superpuesta** porque no hay una segunda subentidad para superponerse.

- **Atributos Compuestos**
  - Atributo generado a partir de la **combinación** de varios **atributos simples**.
  - Puede ser **polivalente o no obligatorio**. Lo mismo ocurre con los atributos simples que lo componen.

- **Identificadores**

- ❖ Atributo o conjunto de atributos que permite **distinguir** a una entidad de manera **unívoca** dentro de un **conjunto de entidades.**
- ❖ **Pueden ser de 2 tipos:**
  - ➢ **Simples o Compuestos:** De acuerdo con la cantidad de atributos que la conforman.
  - ➢ **Internos o Externos:** Si todos los atributos que conforman un identificador pertenecen a la entidad que identifica, es **interno**, sino es **externo**.

Clase 3 "Revisiones sobre el Modelo Conceptual y Modelo Lógico"

Revisiones sobre el Modelo Conceptual

Decisiones

- **Dependen de 3 factores:**
  - ❖ El problema.
  - ❖ Lo que el cliente espera obtener de la Base de Datos.
  - ❖ La experiencia del analista.
- **¿Conviene generar una entidad con un concepto nuevo? ¿O agregar un atributo a una entidad existente?**
  - ❖ La respuesta que ocurre generalmente a este problema es que es mejor a la larga generar una entidad con un concepto nuevo si se duda si un atributo es en verdad un atributo o una entidad.
- **¿Cuándo usar Generalización y cuándo usar Clasificación?**
  - ❖ Si existieran datos del modelo que tengan características propias, sería conveniente generar una Generalización/Jerarquía, sino estamos frente a una Clasificación.
  - ❖ La regla a aplicar debe controlar que no queden entidades definidas sin atributos o relaciones con otras entidades (Entidades Colgadas).

- **¿Convienen los atributos compuestos? ¿O es mejor generar muchos simples?**
    - ❖ Los compuestos se deberían utilizar si es posible.
    - ❖ En una etapa posterior se decidirá qué hacer con los compuestos generados.

Conceptos a tener en cuenta para mejorar el Modelo

- **Autoexplicación**
    - ❖ Se **expresa** a si mismo si puede representarse utilizando los elementos definidos, sin necesidad de utilizar aclaraciones en el lenguaje natural.
- **Completitud**
    - ❖ Está **completo** cuando todas las características del problema están contempladas en el modelo.
    - ❖ Revisar la especificación de requerimientos.
- **Corrección**
    - ❖ Es **correcto** si cada elemento en su construcción fue utilizado con propiedad.
    - ❖ Observar que estén expresadas todas las coberturas y cardinalidades, los identificadores, etc.
    - ❖ Revisar concepto de Herencia en las Jerarquías.
- **Expresividad**
    - ❖ Resulta **expresivo** si a partir de su observación es posible darse cuenta de todos los detalles que lo conforman.
- **Extensibilidad**
    - ❖ Resulta **extensible** si a es fácilmente modificable para incorporar nuevos conceptos en él.
- **Legibilidad**
    - ❖ Resulta **legible** si la representación gráfica es adecuada.
- **Minimalidad**
    - ❖ Resulta **mínimo** cuando cada concepto se representa una sola vez en el modelo.
    - ❖ **Hay dos factores que atentan a la Minimalidad**
        - ➢ Atributos derivados.
        - ➢ Ciclos de relaciones.

Modelo Lógico

Objetivos

- Convertir el esquema conceptual en modelo más cercano a la representación entendible por el DBMS.
- Representar un esquema equivalente, que sea más eficiente para su utilización.
- Necesario definir el tipo de DBMS (**Relacional**, Orientado a Objetos, Jerárquico y de Red)

Cambios a realizar en el Diseño Lógico

Atributos Derivados

- Un atributo es derivado si contiene la información que puede obtenerse de otra forma desde el modelo.
- **Ventaja de mantenerlos**
  - ❖ Disponibilidad rápida de la información.
- **Desventaja de mantenerlos**
  - ❖ Necesita ser recalculado cada vez que se modifica la información que contiene (Mantenimiento costoso).
- **Dejar en el modelo a todos aquellos que son muy utilizados, y quitar los que necesitan ser recalculados con frecuencia.**

Ciclos de Relaciones

- La decisión de mantener o no un ciclo de Relaciones radica por tener el modelo **mínimo** o por que posteriormente el modelo implique menos **tiempo de procesamiento**.

Atributos Polivalentes

- La solución para este caso consiste en quitar el atributo polivalente para representarlo generando una nueva **entidad**, estableciendo na relación con la entidad que lo contenía.

- **Respetamos la Primera Forma Normal**.

Atributos Compuestos

- **Tres soluciones posibles**:
  - ❖ Generar un único atributo que se convierta en la concatenación de todos los atributos simples que contiene el compuesto.
    - ➢ **Ventajas**
      - o Simple y sencillo de implementar.
    - ➢ **Desventaja**
      - o Se pierde la identidad de cada atributo simple.
  - ❖ Definir todos los atributos simples sin un compuesto que los resuma.
    - ➢ **Ventajas**
      - o Permite al usuario definir cada uno de los datos en forma independiente.
      - o Se adapta mejor a las situaciones de la vida real.
    - ➢ **Desventajas**
      - o La cantidad de atributos aumenta.
    - ➢ **Es la más usada.**
  - ❖ Generar una nueva entidad conformada por los atributos simples contenidos en el compuesto, debe estar relacionada con la entidad a la que pertenecía el compuesto.
    - ➢ **Ventajas**
      - o Capta mejor la esencia del atributo compuesto.
    - ➢ **Desventajas**
      - o Opción más compleja.

Jerarquías

- **El modelo relacional no soporta la Herencia**, por lo tanto hay que buscar otra forma de representar las Jerarquías.
- **Tres opciones posibles:**
  - ❖ Eliminar las subentidades, dejando a la superentidad la cual incorpora los atributos de las subentidades de manera no obligatoria y adquiere las relaciones de dichas subentidades.

        - Puede generar mayor complejidad de lectura si se cuenta con un gran número de subentidades, ya que aparecerán varios atributos no obligatorios y algunas de las cardinalidades de las relaciones se pueden ver afectadas.
    - Eliminar a la superentidad, dejando las subentidades incluyendo atributos y relaciones de la superentidad en todas las subentidades.
    - Dejar todas las entidades, convirtiendo a la jerarquía en relaciones uno a uno entre la superentidad y las subentidades "es un", manteniendo los atributos originales.
        - **Capta mejor la esencia de la Herencia.**
        - Genera mayor cantidad de entidades y relaciones, lo que puede generar problemas de performance.
- La cobertura de la jerarquía determina que solución es viables en cada caso:
    - En modelos donde no haya cobertura parcial, las tres soluciones son válidas.
    - Aquellos que presenten cobertura parcial no podrán utilizar la segunda opción (eliminar la superentidad).

Clase 4 "Modelo Físico"

Modelo Físico

- El modelo físico (relacional) representa la Base de Datos como una colección de relaciones.
- Una relación se asemeja a una tabla de valores, o a un archivo plano de registros.
- Cada relación está integrada por filas y columnas.
    - Cada fila representa un registro del archivo y se denominan tupla.
    - Cada columna representa un atributo del registro.

Pasos para pasar al Modelo Físico

Eliminación de Identificadores Externos

- Cada una de las entidades debe poseer sus **identificadores** definidos de forma **interna**. Para lograr esto, se deberá incorporar dentro de la entidad que contenga identificadores externos, aquellos atributos que permitan la definición del identificador de forma interna a la entidad.

Selección de Claves Primaria, Candidata y Secundaria

- **Para el usuario, el concepto de clave primaria y/o candidata no es importante**, solamente necesitan que estén los identificadores que permitan distinguir una entidad del conjunto de estas.
- **Si una entidad tiene un solo identificador**, este es la **Clave Primaria**.
- **Si una entidad tiene más de un identificador** la selección se hace del siguiente modo:
  - **Entre uno simple y uno compuesto**
    - Tomo el **simple**.
  - **Entre dos simples**
    - Tomo el de **menor tamaño físico**.
  - **Entre 2 compuestos**
    - Tomo el de **menor tamaño en bytes**.
- Los **identificadores** no elegidos se convierten en **Claves Candidatas**.
- Una entidad tiene solo **una Clave Primaria**, pero puede tener **más de una Candidata y/o Secundaria.**
- Cualquier Clave Primaria tratable directamente por el **usuario** puede ser **borrada o modificada**, para esto el **DBMS** presenta una solución:
  - Uso de un **atributo con tipo de dominio Autoincremental**
    - Es **tratada** por el **DBMS** de forma **exclusiva**.
    - El **usuario** solo tiene permitida la operación de **consulta**.
    - Actúa de la forma más **eficiente**, por lo tanto, mejora la **performance** de la Base de Datos.

Conversión de Entidades

- Generalmente, cada una de las entidades se convierte en una **tabla** del **modelo físico**.

- **Hay un caso especial donde lo anterior no se cumple**
  - ❖ Cuando la **cardinalidad de la relación** entre dos tablas es de **uno a uno con participación total de ambos lados**, se debe generar **una sola tabla** que **contenga** los **atributos de ambas entidades.**

Conversión de Relaciones

- **Concepto importante**
  - ➢ **Clave Foránea**: Atributo o grupo de atributos de una tabla que en otra tabla es/son Clave Primaria y que sirven para establecer un nexo entre ambas tablas.

- Se tienen en cuenta **3 situaciones diferentes**:
  - ❖ ***Cardinalidad muchos a muchos***
    - ➢ **Se generan 3 tablas, una para cada entidad y una para la relación** que puede definir su propia **Clave Primaria** mediante el uso de una **Autoincremental** o puede definir su **Clave Primaria** a partir de la **combinación de las Claves Primarias de las entidades que relaciona.**
    - ➢ Es mejor la opción del Autoincremental ya que define una Clave Primaria a partir de un **atributo simple**, la otra opción sería **compuesto**.
    - ➢ Se puede necesitar algún **atributo** de la **relación** para identificar **si no se usa el Autoincremental**.

  - ❖ ***Cardinalidad uno a muchos***
    - ➢ **Cardinalidad uno a muchos con participación total o parcial del lado de muchos**
      - o **Se generan 2 tablas, una para cada entidad.**
      - o La **Clave Primaria** del lado de **muchos** pasa como **Clave Foránea** a la tabla del lado de **uno**.
    - ➢ **Cardinalidad uno a muchos con participación parcial del lado de uno o participación parcial de ambos lados**
      - o Hay 2 soluciones, la segunda es mejor.
      - o **Generar 2 tablas, una por entidad** y definir una **Clave Foránea** que acepte valores **nulos** y haga

referencia a la entidad del lado de **muchos** en la entidad del lado de **uno**. Tomando valor solo cuando se da la **relación** entre las 2 entidades.

- **Generar 3 tablas, una por entidad y una para la relación** usando como **Clave Primaria** en la tabla de la **relación**, la **combinación** de las **Claves Primarias de las entidades que relaciona.**

❖ ***Cardinalidad uno a uno***

- **Cardinalidad uno a uno con participación total de ambos lados**
  - **Generar una sola tabla** que contenga los atributos de **ambas entidades**.
- **Cardinalidad uno a uno con participación parcial de un lado**
  - **Generar 2 tablas, una por entidad** y la tabla de la entidad del lado que posee **participación total** recibe como **Clave Primaria o como Clave Foránea** a la **Clave Primaria** de la entidad que posee **participación parcial**.
- **Cardinalidad uno a uno con participación parcial de ambos lados**
  - **Generar 3 tablas, una por entidad y una para la relación** que tendrá como **Clave Primaria** la **combinación** de las **Claves Primarias** de las **entidades** que relaciona.

- Las relaciones **Recursivas** y **Ternarias** siguen las reglas de la **cardinalidad** para resolverse.
- Tener en cuenta que las relaciones **Recursivas** llevan a la **replicación** de la **Clave Primaria** de la entidad.

Integridad Referencial

- Propiedad deseable de las Bases de Datos Relacionales.
- Asegura que un valor que aparece para un atributo en una tabla, aparezca además en otra tabla para el mismo atributo.

- Plantea restricciones entre tablas y sirve para mantener la consistencia de datos.
- Para que exista es necesario que haya un atributo en común entre dos tablas, normalmente ese atributo es Clave Foránea en una tabla, y en la otra Clave Primaria.
- No es condición necesaria que entre dos tablas que tienen un atributo en común, esté definida la Integridad Referencial.
- Formas que tiene el DBMS de restringir las operaciones de Modificación y Borrado:
  - **Restringir la operación**
    - Si se intenta borrar o modificar, la operación se restringe y no se puede llevar a cabo, uno tiene que ir borrando manualmente la cadena.
  - **Realizar la operación en Cascada**
    - Si se intenta borrar o modificar. La operación se realiza en cascada sobre todas las tuplas de la tabla que tiene definida la Integridad Referencial.
  - **Establecer la Clave Foránea en nulo**
    - Si se borra o modifica el valor del atributo que es Clave Primaria, sobre la Clave Foránea se establece valor nulo.
    - No es muy utilizado.
  - **No hacer Nada**
    - Se indica al DBMS que no es necesario controlar la Integridad Referencial.
    - Es equivalente a no definir la Integridad Referencial.

Dependencias Funcionales

- Restricción entre atributos/conjunto de atributos de una tabla de la Base de Datos.
- Un atributo Y depende funcionalmente de un atributo X (X → Y) cuando para un valor de X siempre se encuentra el mismo valor para el atributo Y. X e Y pueden ser un conjunto de atributos.
- Formalmente, si t1 y t2 son dos tuplas cualesquiera en una tabla, si t1 [X] = t2 [X] entonces debe ocurrir que t1 [Y] = t2 [Y].
- El atributo X determina al atributo Y.
- No son ni malas ni buenas para la Base de Datos.

Dependencia Funcional Completa

- Si X e Y son atributos de una relación, Y depende funcionalmente de manera completa de X, si Y depende de X pero de ningún subconjunto de X.
- No generan problemas

Dependencia Funcional Parcial

- Dado una Dependencia Funcional X → Y, esta es considerada parcial si, además, existe una dependencia Z → Y, siendo Z un subconjunto de X.
- Genera problemas
    - ❖ Trae aparejada la repetición de información.

Dependencia Funcional Parcial Transitiva

- Dado una Dependencia Funcional X → Y, esta se denomina transitiva si, además, existe un atributo Z tal que X → Z y Z → Y.

Dependencia Funcional de Boyce-Codd

- Dado una Dependencia Funcional X → Y, esta se denomina de Boyce-Codd, si, además, X no es una Clave Primaria o Clave Candidata, e Y es una Clave Primaria o una Clave Candidata o una parte de ella.

Dependencia Multivaluada

- Dados los atributos X e Y, una dependencia multivaluada X →→ Y ocurre cuando X multidetermina a Y, es decir, para un determinado valor de X se determina un conjunto de valores para Y.

Dependencia Multivaluada Trivial

- Dado una Dependencia Multivaluada X →→ Y, podemos decir que es trivial si Y está multideterminado por el atributo X y no por un subconjunto del atributo X.

Dependencia de Combinación

- Es una propiedad de la descomposición que nos garantiza que si a una relación R1 la descomponemos en dos relaciones R2 y R3 y luego las volvemos a combinar mediante operaciones de Álgebra Relacional, el resultado será nuevamente la relación R1 sin la aparición de tuplas espurias, es decir, combinaciones de tuplas que no tienen sentido.

Normalización

- Mecanismo de **diseño** que a partir de examinar las **dependencias funcionales** que existen entre atributos, busca el agrupamiento óptimo de estos con el fin de producir un conjunto de relaciones que cumplan con una serie de propiedades deseables partiendo de los requisitos de datos de una organización.
- La **ventaja** de usar una Base de Datos normalizada es que se disponen de tablas con **datos fácilmente accesibles** y de **sencillo mantenimiento.**
- **Se puede aplicar en cualquier etapa del diseño**, es mejor aplicar en el **Modelo Físico.**
- Es un proceso **deseado** pero **no estrictamente necesario**.
- Busca **reducir** la **redundancia de datos** y las **anomalías que pueden ocurrir a la hora de actualizar información**, y **evitar** la **pérdida de integridad de datos.**

Primera Forma Normal (1NF)

- Un modelo está en esta forma si para cada **tabla** no existen **atributos** que sean **polivalentes**.

Segunda Forma Normal (2NF)

- Un modelo está en esta forma si está en **Primera Forma** y además, para cada tabla no existen **Dependencias Parciales**.
- **Solución**
  - ❖ Quitar de la tabla el atributo o los atributos que generan la dependencia y situarlos en una nueva tabla. La dependencia sigue existiendo, pero en una ubicación donde ya no es una Dependencia Parcial.
  - ❖ Genera **segmentación** de información.

Tercera Forma Normal (3NF)

- Un modelo está en esta forma si está en **Segunda Forma** y además, para cada tabla no existen **Dependencias Transitivas**.
- **Solución**
  - ❖ Mismo proceso que en **Segunda Forma** pero tratando las **Dependencias Transitivas**.
  - ❖ Genera **segmentación** de información.
  - ❖ En este momento se tiene que tomar la decisión si seguir normalizando **(Menos redundancia, más segmentación, menos control)** o si frenar la normalización **(mantener la redundancia que se tenga, no segmentar más los datos).**

Forma Normal de Boyce-Codd

- Un modelo está en esta forma si está en **Tercera Forma** y además, para cada tabla no existen **Dependencias Boyce-Codd**.
- Es una Forma Normal bastante **restrictiva**, podemos perder **claves candidatas** al pasar a este modelo.
- También se puede estar en esta forma si y solo si todos los determinantes de las dependencias son Claves Candidatas o Claves Primarias.
- Se puede **evitar** y pasar a las siguientes Formas Normales.

Cuarta Forma Normal (4NF)

- Un modelo está en esta forma si está en **Tercera Forma o en Forma Boyce-Codd** y además, para cada tabla solo existen **Dependencias Multivaluadas Triviales.**

Quinta Forma Normal (5NF)

- Un modelo está en esta forma si está en **Cuarta Forma** y además, para cada tabla no existen **Dependencias de Combinación**.

Clase 5 "Lenguajes de Procesamientos de Consultas"

- **Existen 3 grupos de lenguajes teóricos:**
  - **Álgebra Relacional.**
  - Cálculo Relacional de Tuplas (no lo vemos).
  - Cálculo Relacional de Dominios (no lo vemos).

Álgebra Relacional

- **Lenguaje de consultas procedimental teórico**.
- Conjunto de **operadores** que toman las **relaciones/tablas** como operandos, y regresan otra **tabla** como **resultado**.
- Admite la **combinación** de **operandos** teniendo en cuenta que **el orden de las operaciones es importante.**
- **Operaciones Básicas**
  - **Unitarias (sobre una tabla)**
    - Selección.
    - Proyección.
    - Renombre.
  - **Binarias (sobre 2 tablas)**
    - Producto Cartesiano.
    - Unión.
    - Diferencia.
- **Operaciones Adicionales**
  - Producto Natural.
  - Intersección.

- ❖ Asignación.
- ❖ División.

Esquema Compatible

- Dos tablas son Esquema Compatible si tienen la misma cantidad de atributos y el i-ésimo atributo de la primera tabla y el i-ésimo atributo de la segunda son del mismo Dominio de tipo de dato.

Selección

- Permite filtrar tuplas de una tabla a partir del uso de un predicado o condición lógica que deben cumplir las tuplas para ser presentadas como resultado.

Proyección

- Permite presentar en el resultado algunos atributos de una tabla.

Producto Cartesiano

- Equivalente al producto cartesiano entre conjuntos.
- Vincula cada tupla de una relación con cada una de las tuplas de la otra relación.
- Genera tuplas espurias.
- Es necesario realizar una operación de selección luego de realizar un producto cartesiano para eliminar las tuplas espurias.
- Se utilizan prefijos para poder diferenciar atributos con mismos nombres.

Renombre

- Permite utilizar la misma tabla dos veces en la misma consulta mediante el cambio temporal del nombre de la misma.

Unión

- Equivalente a la unión matemática de conjuntos.
- Genera una nueva tabla cuyo contenido es el contenido de cada una de las tuplas de las tablas originales involucradas.
- Las dos tablas deben ser Esquema Compatible.

Diferencia

- Equivalente a la diferencia matemática de conjuntos.
- Genera una nueva tabla cuyo contenido es el contenido de cada una de las tuplas de la primera tabla que no están en la segunda tabla.
- Las tablas deben ser Esquema Compatible.

Producto Natural

- Reúne las tuplas de la primera tabla que se relacionan con la segunda tabla, descartando tuplas espurias.
- Tiene mejor performance que el Producto Cartesiano.
- Entre las tablas sobre las que se realiza esta operación debe existir un atributo en común.
    - ❖ Si no existe o hay más de uno actuará igual que el Producto Cartesiano.

Intersección

- Equivalente a la intersección matemática de conjuntos.
- Genera una nueva tabla con las tuplas que son comunes en las 2 tablas.
- Las tablas deben ser Esquema Compatible.
- Esta operación se puede realizar mediante la Diferencia o mediante la combinación de la Diferencia con la Unión.

Asignación

- Permite generar una subconsulta y volcar su resultado en una variable temporal la cual puede ser utilizada posteriormente.
- Otorga mayor legibilidad a las consultas.

División

- Dadas dos tablas R1 y R2, el resultado de realizar R1 % R2 son los valores de R1 que se relacionan con todos los valores de R2 si y solo si, todos los atributos de R2 son atributos de R1.
- El esquema de la tabla resultante es el equivalente a realizar R1 – R2.

Actualizaciones usando Álgebra Relacional

Altas

- Agregar una nueva tupla a una tabla existente conlleva una operación de Unión entre la tabla en cuestión y una tupla escrita de manera literal.

Bajas

- Eliminar una tupla de una tabla conlleva una operación de Diferencia entre la tabla en cuestión y una tupla escrita de manera literal.

Modificación

- Para modificar un atributo de una tabla debemos especificar la tabla que se modifica, el atributo y el valor nuevo a asignar. Se debe hacer una operación de Selección a la tabla para realizar la Modificación en una sola tupla, si no se hace se modificara ese atributo en todas las tuplas de la tabla.

Clase 6 “SQL (Structured Query Lenguage)”

- Lenguaje de consultas de Bases de Datos.
- Compuesto por 2 submódulos

- ❖ DDL (Data Definition Lenguage).
- ❖ DML (Data Manipulation Language).

DDL (Data Definition Lenguage)

- Módulo utilizado para la definición del modelo de datos.
- Podemos definir las tablas, atributos, la integridad referencial, índices y restricciones del modelo.
- Define tres cláusulas para la administración del modelo físico
  - ❖ CREATE.
  - ❖ DROP.
  - ❖ ALTER.

Operaciones de DDL

- **Crear una Base de Datos**
  - ❖ Se usa la sentencia CREATE DATABASE.
  - ❖ Se debe especificar el nombre de la Base de Datos.

- **Borrar una Base de Datos**
  - ❖ Se usa la sentencia DROP DATABASE.
  - ❖ Se debe especificar el nombre de la Base de Datos.
  - ❖ Incluye tablas y datos contenidos en ellas.

- **Crear una Tabla**
  - ❖ Se usa la sentencia CREATE TABLE.
  - ❖ Se debe indicar el nombre de la tabla y cada uno de los atributos que la componen (tipo de dato del dominio, si acepta valores nulos, si es Autoincremental, si es índice y de qué tipo, si es clave foránea, tipo de restricción de integridad referencial, etc.).

- **Borrar una Tabla**
  - ❖ Se usa la sentencia DROP TABLE.
  - ❖ Se debe especificar el nombre de la Tabla.

- **Modificar una Tabla**

- ❖ Se usa la sentencia ALTER TABLE.
- ❖ Se debe especificar el nombre de la Tabla y qué tipo de modificación se desea realizar sobre la tabla.

DML (Data Manipulation Language)

- Módulo para la operatoria normal de la Base de Datos.
- Se pueden realizar consultas, altas, bajas y modificaciones de datos sobre las tablas del modelo.
- Una consulta de SQL siempre retorna como resultado una tabla temporal.

Estructura Básica de una consulta

- SELCT lista_de_atributos
  FROM lista_de_tablas
  WHERE condición/predicado
- El SELECT y el FROM son obligatorios.

Cláusula SELECT

- Equivalente a la Proyección de Álgebra Relacional.
- Utilizada para listar contenido de una o varias tablas.
- Usamos el operador **"*"** para reemplazar la escritura literal de todos los atributos de una tabla.
- Se pueden realizar operaciones dentro de esta cláusula
  - ❖ Si el atributo a mostrar es integer o real acepta cualquier operación matemática.
  - ❖ Si el atributo a mostrar es string, admite cualquier operación del manejo de strings.
- **DISTINCT**
  - ❖ Se aplica a campos del SELECT.
  - ❖ Elimina tuplas repetidas.
- **ALL**
  - ❖ Se aplica a campos del SELECT.
  - ❖ No elimina tuplas repetidas, es la opción por defecto por lo tanto no se especifica.

- **AS**
    - ❖ Permite el renombre de atributos.

Cláusula FROM

- Equivalente al Producto Cartesiano de Álgebra Relacional.
- Sirve para especificar de qué tabla/tablas vamos a sacar la información.
- Se pueden renombrar las tablas utilizadas poniendo el nombre que queremos usar posteriormente al nombre de la tabla.
- **Producto Cartesiano**
    - ❖ Lista de tablas separadas por comas.
- **Producto Natural**
    - ❖ Se realiza mediante un INNER JOIN.
    - ❖ Reúne las tuplas de las relaciones que tienen sentido.
    - ❖ Más performante que el Producto Cartesiano.
    - ❖ Se utiliza la sentencia ON para especificar la condición que debe cumplirse.
    - ❖ No aplican las restricciones de un único atributo común con el mismo nombre en ambas tablas.
- **Producto Natural por equicombinación**
    - ❖ Se realiza mediante un NATURAL JOIN.
    - ❖ Análogo al producto natural de Álgebra Relacional.
    - ❖ Aplican las restricciones de Álgebra.
- **LEFT JOIN**
    - ❖ Variante del producto natural.
    - ❖ Contiene todos los registros de la tabla de la izquierda, si hay correspondencia con la tabla de la derecha, deja el valor correspondiente pero si no la hay, deja un valor nulo.
- **RIGHT JOIN**
    - ❖ Variante del producto natural.
    - ❖ Inversa del LEFT JOIN.
- **FULL JOIN**
    - ❖ Combinación del INNER, LEFT y RIGHT.

Cláusula WHERE

- Equivalente a la Selección de álgebra.
- Permite el uso de operadores lógicos.
- Necesaria la utilización de prefijos con atributos con el mismo nombre.
- **LIKE**
  - ❖ Brinda gran potencia a las consultas que requieren manejo de Strings.
  - ❖ **%**
    - Representa cualquier cadena de caracteres, inclusive la vacía.
  - ❖ **_**
    - Sustituye solo el carácter del lugar donde aparece.
  - ❖ Se usa **IS NULL e IS NOT NULL** para la verificación de valores nulos en los atributos.

Cláusula ORDER BY

- Permite ordenar las tuplas resultantes por el atributo que se le indique.
- Por defecto es en orden descendente pero se puede hacer ascendente usando **ASC**.
- Admite más de un criterio de orden, se irán aplicando en caso de empate.

Operaciones de Conjuntos

- **UNION**
  - ❖ Agrupa las tablas resultantes de dos subconsultas.
  - ❖ No conserva duplicados.
- **UNION ALL**
  - ❖ Lo mismo que la UNION pero conserva a los duplicados.
- **INTERSECT**
  - ❖ Intersección entre las tabla resultantes de dos subconsultas.
- **EXCEPT/MINUS**
  - ❖ Diferencia entre dos subconsultas.
  - ❖ Puede no estar implemento en todos los DBMS.

Funciones de Agregación

- Se usan en un **SELECT** o en un **HAVING**.
- Operan sobre un conjunto de tuplas de entrada produciendo un único valor de salida.
- **AVG**
  - ❖ Sacar promedio.
- **COUNT**
  - ❖ Cantidad de tuplas involucradas en el conjunto de entrada.
- **MAX**
  - ❖ Valor más grande del conjunto.
- **MIN**
  - ❖ Valor más chico del conjunto.
- **SUM**
  - ❖ Sacar la suma de los valores del atributo especificado.

Agrupamiento

- Se realiza mediante un **GROUP BY**.
- Agrupa las tuplas de una consulta por algún criterio con el objetivo de aplicar alguna función de agregación.
- **HAVING**
  - ❖ Se usa para restringir los grupos que aparecen en la tabla de resultados mediante alguna condición que tienen que cumplir los grupos.

Subconsultas

- Consiste en ubicar una consulta SQL dentro de otra. Se pueden usar los siguientes comparadores.
- **=**
  - ❖ Cuando una subconsulta retorna un solo valor simple, es posible compararlo contra otro valor simple
- **IN/NOT IN**
  - ❖ Comprueba si un elemento es parte o no de un conjunto
- **(>, =, >=, <=, <>)SOME**
  - ❖ (>, =, >=, <=, <>) que alguno.

- **(>, =, >=, <=, <>)ALL**
  - (>, =, >=, <=, <>) que todos.
- **EXIST/NOT EXIST**
  - Permite comprobar si una subconsulta generó o no alguna tupla como respuesta.

Vistas o Subconsultas Nominadas

- Se definen con la sentencia **CREATE VIEW ().**
- Manera de definir una subrutina dentro de SQL que puede ser referenciada posteriormente en otra consulta.
- Cada vez que se utiliza la vista, la subrutina almacenada es recalculada.
- Tienen la estructura de una tabla.
- **Permiten la modularización**.
- Generan **seguridad** en la Base de Datos ya que se puede especificar en el DBMS que usuario tiene permiso a ejecutar X Vista.

Altas, Bajas y Modificaciones

- **Altas**
- Se realizan mediante un **INSERT INTO**.
- Si una tabla maneja un atributo Autoincremental este no se debe especificar ya que es manejado por el DBMS.
- Genera solo una tupla cada vez que es ejecutada.
- Se debe tener cuidado con la integridad referencial as especificar atributos que sean **Claves Foráneas**.

- **Bajas**
  - Se realiza mediante un **DELETE FROM**.
  - Se puede borrar una tupla o un conjunto de tuplas.

- **Modificaciones**
  - Se realiza mediante un **UPDATE**.
  - Permite la modificación de uno o más atributos de una o un conjunto de tuplas.

Clase 7 "Optimización de Consultas"

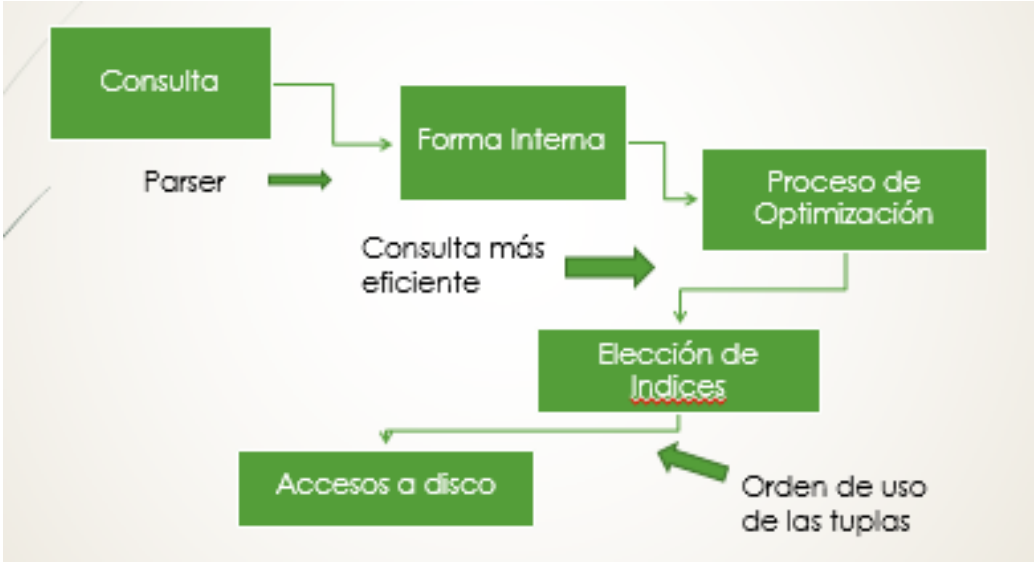
- Nosotros nos centramos en hacer consultas eficientes en términos de **legibilidad** en temas de **performance** se encarga el **DBMS**.

Optimizador de Consultas

- **Proceso del DBMS** encargado de encontrar una consulta equivalente a la generada por el usuario, que sea óptima en términos de **performance** para obtener el resultado deseado.

Pasos del Optimizador de Consultas

1) Un **analizador sintáctico (parser)** genera una expresión manipulable por el Optimizador de Consultas. El análisis sintáctico convierte la consulta del usuario en una estructura tipo árbol que resulta más fácil para el análisis.
2) A partir de la expresión interna generada, el **optimizador** obtiene una consulta equivalente más eficiente.
3) El **proceso de optimización** considera el estado actual de la Base de Datos y los índices definidos para resolver la consulta que se tiene hasta el momento, con el acceso a disco disponible.



Componentes del Costo de ejecución de una consulta

- **Costo de acceso a almacenamiento secundario**
  - Acceder **al bloque de datos que reside en disco**.
  - Se mide en milisegundos.
- **Costo de cómputo**
  - Costo de realizar operaciones sobre **memoria RAM**.
  - Se mide en nanosegundos.
- **Costo de comunicación**.
  - Costo de enviar la consulta y los resultados.
  - Hay que tenerlo en cuenta si es un **Sistema Distribuido**.

Clase 8 Parte 1 "Transacciones en Entornos Monousuario"

Seguridad de Datos

- Este concepto se relaciona con el uso indebido de la Base de Datos y eventualmente rotura de la misma a partir de la mala intención.

Integridad de Datos

- Este concepto se relaciona con la rotura de la información a partir del uso cotidiano de la Base de Datos o situaciones imprevistas para los usuarios, sin una mala intención de rotura.

Fallos

- **Tipos de fallos que pueden ocurrir:**
  - **Fallos que no producen pérdida de información**
    - Causados por transacciones que solo generen una consulta a la Base de Datos.
  - **Fallos que producen pérdida de información**
    - Causados por transacciones que conlleven altas, bajas o modificaciones en la Base de Datos.
    - Son las más difíciles de tratar.

- Los métodos de recuperación de información son los encargados de solucionar la pérdida de integridad de datos que estos fallos generan.

❖ **Fallos con poca probabilidad de ocurrencia**

- Corte de luz, rotura de disco, fallo de la CPU, sistema operativo que deja de funcionar, errores internos del DBMS, etc.

Transacción

- Conjunto de instrucciones que actúa como unidad lógica de trabajo.

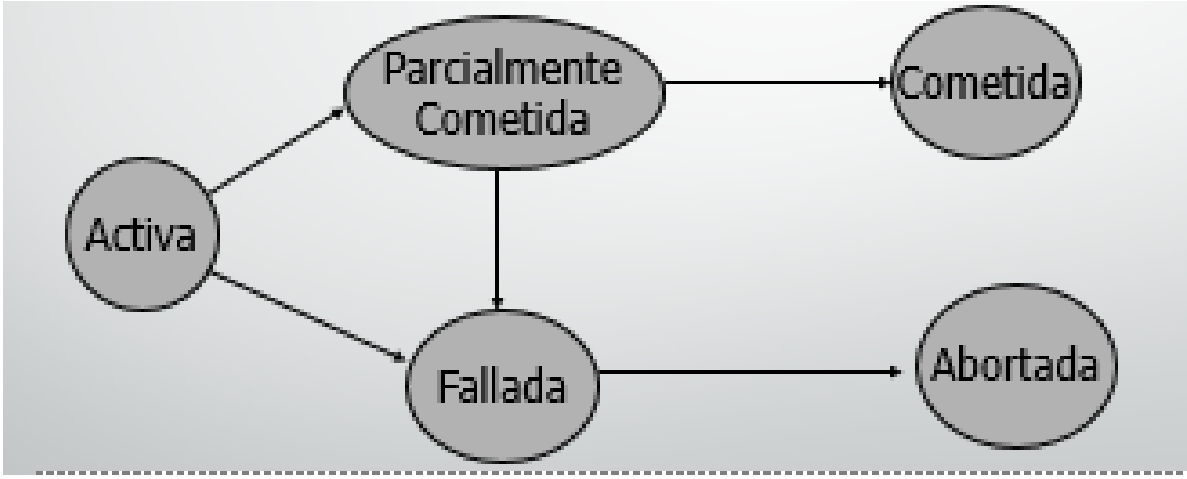
Propiedades ACID

- **Son 4 propiedades que una Transacción debe cumplir.**
- **Atomicidad**
  - ❖ Todas las instrucciones de una transacción se ejecutan o ninguna de ellas se lleva a cabo.
- **Consistencia**
  - ❖ La ejecución completa de una Transacción garantiza que el estado de la Base de Datos se mantenga consistente.
  - ❖ La Transacción debe estar bien escrita por el programador.
  - ❖ Se logra mediante la ejecución aislada de transacciones.
- **Isolation o Aislamiento**
  - ❖ Cada Transacción actúa como única en el sistema. La ejecución de una Transacción no debe afectar la ejecución simultánea de otra Transacción.
- **Durabilidad**
  - ❖ Una vez que finaliza una Transacción, los efectos producidos en la Base de Datos son permanentes.

Estados de una Transacción

1) **Activa**
   - ❖ Estado inicial que se tiene desde que comienza la Transacción y se mantiene hasta que se complete la última instrucción o hasta que se produzca un fallo.

2) **Parcialmente Cometida/Finalizada**
   - ❖ Se alcanza después de que se ejecuta la última instrucción de la Transacción.
   - ❖ Este estado existe ya que las operaciones de READ y WRITE se realizan sobre buffers, por lo tanto, pueden estar los datos en unos de estos buffers pero la Transacción no reflejo resultados en la Base de Datos.
3) **Cometida o Finalizada**
   - ❖ Se alcanza una vez que se finaliza la ejecución de la Transacción y sus acciones fueron almacenadas correctamente en memoria secundaria.
4) **Fallada**
   - ❖ Se obtiene cuando la Transacción no puede continuar con su ejecución normal.
5) **Abortada**
   - ❖ Garantiza que una Transacción fallada no ha producido ningún cambio en la Base de Datos, manteniendo la integridad de la misma.
   - ❖ Siempre que una Transacción falla debe abortarse para mantener la Base de Datos consistente.



Métodos de Recuperación de Integridad de una Base de Datos

- Se puede pensar en un primer instante que si una Transacción falla, podríamos **Re-ejecutar a la misma** o **podríamos Dejar el estado de la Base de Datos como está, pero esto no sirve.**

- Lo que si sirve es el **Registro Histórico/Bitácora** y la **Doble Paginación.**

Bitácora o Log

- Consiste en llevar un **registro histórico** en un **archivo auxiliar externo a la Base de Datos**, registrando todos los movimientos producidos por las transacciones **antes** de realizar los cambios sobre la Base de Datos.
- En caso de producirse un error se tiene constancia de todos los movimientos efectuados.
- **El estado de consistencia de la Base de Datos es alcanzable en todo momento, aún ante fallo.**
- **Solo se registran las operaciones de escritura.**
- Para este método toda transacción que tenga registro de comienzo pero no de cometida, debe abortarse.
- El DBMS identifica cada transacción mediante un número correlativo único en cada una de ellas.
- **Viendo la estructura del archivo:**
    - ❖ Se indica el comienzo de la transacción.
    - ❖ Se indica su cometida/finalización.
    - ❖ Se indica cada una de las acciones llevadas a cabo sobre la Base de Datos.
    - ❖ Se indica el Valor anterior del dato modificado y su Valor nuevo.
    - ❖ Se indica identificador de la transacción.
    - ❖ Se indica identificador en los elementos de datos.
    - ❖ Se indica su estado de abortada de ser necesario.
- **Puede ocurrir que una transacción no tenga ni registro de cometida ni registro de estar abortada.**
- Los algoritmos relacionados con el proceso de bitácora deben garantizar que **primero se genere el registro en la bitácora y luego en la Base de Datos**, para eso
    - ❖ El DBMS opera en contra del Sistema Operativo para poder manejar la descarga de los **buffers** de memoria, indicando que primero se debe escribir el buffer correspondiente a la bitácora

y luego el de la Base de Datos, y que se descargue primero el de la Bitácora.

- **Posee dos métodos de recuperación en caso de que se produzcan error**
  - ❖ Modificación diferida de la Base de Datos.
  - ❖ Modificación Inmediata de la Base de Datos.

Modificación Diferida de la Base de Datos

- **Demora** todas las **escrituras** en disco de las actualizaciones de la Base de Datos, hasta que la transacción alcance el estado de **cometida** en la Bitácora.
- Mientras la transacción esté activa se demora la escritura física de la Base de Datos.
- **Ventaja**
  - ❖ Si una transacción **activa** falla, se puede asegura que los cambios no fueron reflejados en la Base de Datos, manteniéndola **integra**.
- **Desventaja**
  - ❖ Se genera **sobrecarga de trabajo** cada vez que termina una **Transacción**.
- Este método **no necesita almacenar el Valor anterior de los datos** que se modifican ya que no es posible modificar la Base de Datos hasta que la transacción alcance su estado de cometida.
- **La Base de Datos puede perder su estado de integridad si:**
  - ❖ Ocurre un **fallo** mientras se están **escribiendo las modificaciones en la Base de Datos** una vez que la Transacción alcanzó su estado de cometida.
- **Ante una situación de fallo, luego de recuperarse**
  - ❖ Esta técnica lleva a cabo una función denominada **REDO** que consiste en rehacer todas las transacciones que tengan un registro de comienzo y un registro de cometida en la bitácora.
  - ❖ **Si una transacción no tiene registro de cometida es ignorada ya que no realizó cambios en la Base de Datos.**
  - ❖ Se puede notar que **esto puede llevar a rehacer transacciones que no tuvieron fallos.**

- Se puede realizar este **REDO** debido a la **propiedad de idempotencia de las transacciones**
    - ❖ Nos garantiza que aunque una transacción se reejecute n veces, se genera siempre el mismo cambio en la Base de Datos.

Modificación Inmediata de una Base de Datos

- Los cambios en la Base de Datos se efectúan a medida que la transacción avanza en su ejecución.
- **La sobrecarga de trabajo tiende a desaparecer ya que se distribuye en el tiempo.**
- En esta técnica **se necesita mantener el Valor anterior de los datos**.
- Puede suceder que el fallo de una transacción ocurra y que la Base de Datos esté parcialmente modificada.
- **Este método necesita de 2 funciones de recuperación**
    - ❖ **REDO**
        - ➢ Para toda transacción que tenga registro de comienzo y registro de cometida en la Bitácora.
    - ❖ **UNDO**
        - ➢ Consiste en retrotraer la Base de Datos al estado que tenía antes de iniciar la transacción mediante la devolución de los valores anteriores de los datos modificados que fueron registrados en la Bitácora, a la Base de Datos.
        - ➢ Para toda transacción que tenga registro de comienzo y no registro de cometida en la Bitácora.

Puntos de Verificación

- El **DBMS** agrega de forma **periódica** a la **Bitácora** una entrada denominada **punto de verificación/checkpoint** con el fin de **evitar la revisión de toda la Bitácora ante un fallo**. Haciendo que solamente se revise desde el punto de verificación en adelante.
- **Se agregan cuando se tiene la seguridad de que la Base de Datos está en estado consistente.**
- En **entornos monousuarios** siempre es posible colocar el punto de verificación en un **instante del tiempo sin transacciones activas**.

- **Periodicidad en la que se agregan los puntos**
  - **“Muy cerca uno del otro”**
    - Significa rehacer poco trabajo en caso de fallo.
    - Agrega sobrecarga a la escritura de la Bitácora.
  - **“Lejanos el uno del otro”**
    - Significa rehacer mayor cantidad de trabajo ante fallos.
    - Menos sobrecarga a la escritura de la Bitácora.

Doble Paginación

- Plantea **dividir la Base de Datos en nodos virtuales (páginas)** que contienen determinados datos.
- Se generan **2 tablas en disco** y cada una de ellas direcciona a las páginas generadas
  - **Tabla actual de páginas.**
  - **Tabla de páginas sombra.**
- **Pasos ante una transacción que realiza operaciones de escritura**
  1. Obtener desde la Base de Datos la página sobre la que se desea escribir. Si la página está en RAM este paso se saltea.
  2. Modificar el dato en RAM y grabarlo en disco en una página nueva, la página actual que apunte a la nueva página.
  3. Si la operación termina correctamente, se modifica la referencia de la página sombra para que apunte a la nueva página.
  4. Se libera la página vieja.
- En caso de fallo, luego de recuperarse, se desecha la página actual y **se toma como válida la página a la sombra**, **copiando la tabla de páginas sombra en memoria principal.**
- **Ventajas**
  - Elimina la sobrecarga de escrituras que posee la Bitácora.
  - La recuperación ante un error es más rápida.
- **Desventajas**
  - Exige dividir la Base de Datos en páginas, esta tarea puede ser compleja.
  - La sobrecarga de este método se ve en entornos concurrentes.
  - Genera fragmentación de datos.
  - Puede generar varias páginas ocupadas no referenciadas.

- ❖ Necesita del uso de algoritmos Garbage Collectors para liberar el espacio en memoria ocupado pero no utilizado.

Clase 8 Parte 2 "Transacciones en Entornos Concurrentes"

- Los DBMS permiten que varias transacciones se ejecuten simultáneamente compartiendo recursos, situación que puede generar problemas de inconsistencia.
- La inconsistencia se genera debido a que un entorno concurrente debe asegurar la propiedad de aislamiento de una transacción.
- Se puede pensar en secuencializar la ejecución de transacciones para eliminar este problema pero si bien es factible, se pierde todo aspecto de concurrencia y se vuelve monousuario, cosa que no queremos.
- **Transacciones correctas sin fallos pueden llevar a pérdida de integridad.**

Planificación

- En un **entorno concurrente**, el **orden de ejecución en serie de transacciones** se denomina **planificación**.
- **Involucra todas las instrucciones de las transacciones.**
- **Conservan el orden de ejecución de las mismas.**
- Si se **posee** un conjunto de **N transacciones**, estas tendrán **N! planificaciones en serie diferente.**
- Las planificaciones para un entorno concurrente no deben ser necesariamente en serie
  - ❖ Si dos transacciones utilizan recursos diferentes, pueden ser ejecutadas simultáneamente, garantizando el aislamiento.
- Dos planificaciones P1 y P2 son **equivalentes en cuanto a conflictos** cuando P2 se logra a partir del intercambio de instrucciones no conflictivas P1.
- Una planificación P2 es **serializable en cuanto a conflictos**, si existe otra planificación P1 equivalente en cuanto a conflictos con P2, y además P1 es una planificación en serie.

Conflicto en Planificaciones Serializables

- Dos instrucciones provenientes de dos transacciones diferentes son conflictivas si operan sobre el mismo dato, y al menos una de ellas es una operación de escritura.

Conclusiones

- **Se debe mantener la integridad de la Base de Datos**.
- La inconsistencia temporal puede ser causa de inconsistencia en planificaciones concurrentes.
    - ❖ La inconsistencia temporal se da cuando durante la ejecución de una transacción, llega otra y le mete "ruido" a la original.
- Para que una **planificación concurrente** sea **válida**, debe **equivaler** a una **planificación en serie.**
- Solo las operaciones de **READ y WRITE** deben tenerse en cuenta para realizar cualquier tipo de **análisis**.

Implementación del Aislamiento – Control de Concurrencia

- Existen dos variantes para lograr implementar el aislamiento
    - ❖ **Protocolo de Bloqueo.**
    - ❖ **Protocolo basado en hora de entrada.**

Protocolo de Bloqueo

- Se basa en que cada transacción debe solicitar y bloquear la información que necesita para su ejecución, utilizarla y luego liberarla para que otra transacción la use.
- **Existen 2 tipos de Bloqueos**
    - ❖ **Bloqueo Compartido**
        - ➢ Debe ser realizado por una transacción para leer un dato que no modificará.

        - Mientras dure este bloqueo se impide que otra transacción pueda escribir el dato.
        - Pueden coexistir bloqueos de este tipo sobre un mismo dato a la vez.
    - **Bloqueo Exclusivo**
        - Se genera cuando una transacción necesita escribir un dato.
        - Mientras dure este bloqueo se impide que otra transacción pueda leer o escribir el dato.
        - Es único, No puede coexistir con otros bloqueos.
- **Si una transacción pide un dato y este está bloqueado tiene 2 opciones**
    - Puede **esperar** por el dato si se posee certeza de que el dato va a ser liberado pronto. **En primer lugar siempre se espera.**
    - Puede **fallar y abortar**, para luego comenzar como una transacción diferente si el tiempo de espera es muy largo.
- **Formas en las que una transacción puede organizar sus pedidos de bloqueos (elegimos la primera ya que no se pierde consistencia en los datos)**
    - Las transacciones pueden trasladar los bloqueos de datos al inicio de las mismas, es decir, pedir todo lo que necesita al principio, usarlo y después liberarlo.
        - Permite reorganizar a las transacciones conflictivas para que se ejecuten en serie de manera aislada y a las no conflictivas concurrentemente con total aislamiento también.
        - Puede generar situaciones de **Deadlock**.
    - Las transacciones pueden realizar sus bloqueos cuando lo necesiten
        - Disminuye la probabilidad de **Deadlock**.
        - Aumenta el riesgo de pérdida de consistencia en los datos.
- **Deadlock**
    - Situación en que dos procesos obtienen un elemento y ambos solicitan el elemento que tiene el otro. Como ninguno de los dos puede obtener el segundo elemento, se genera un Deadlock y se impide continuar la ejecución de los procesos.

- ❖ Se soluciona mediante el mecanismo de elegir una “victima”, es decir, se elige uno de los procesos involucrado y se lo hace fallar y abortar para que libere los recursos.

- **Para asegurar la integridad de la Base de Datos, se lleva a cabo el protocolo de bloqueo en dos fases**
  - ❖ Requiere que las transacciones hagan sus bloqueos en dos fases
    - ➢ **Crecimiento**
      - Fase donde se **piden** los **datos** o **se pasa de bloqueo compartido a exclusivo.**
    - ➢ **Decrecimiento**
      - Fase donde se **liberan** los **datos** o **se pasa de bloqueo exclusivo a compartido**
  - ❖ Evita las situaciones de **Deadlock**.
  - ❖ **Garantiza seriabilidad en conflictos.**

Protocolo basado en hora de entrada

- El **orden de ejecución** se **determina** por **adelantado**, no depende de quien llega primero. Cuando uno hace un bloqueo de un dato, depende de que **transacción** llega primero para ver quien bloquea el dato.
- **No genera Deadlock.**
- Cada transacción recibe una **hora de entrada única** que puede ser la **hora del servidor** o un **contador**.
- A cada dato de la Base de Datos se le suma
  - ❖ **La hora en que se ejecutó el último WRITE.**
  - ❖ **La hora en que se ejecutó el último READ.**
- **En cuanto a cómo funciona el algoritmo para determinar si se puede ejecutar una operación de lectura/escritura**
  - ❖ La única forma de que una transacción pueda hacer **READ** a un dato es que su **hora de entrada** sea **mayor** que la **última hora** en que se hizo **WRITE** al dato.
    - ➢ La nueva **última hora de lectura** será el **máximo** entre la **hora de entrada de la transacción que logró leer** y **la última hora de lectura que tenía almacenada el dato.**

- ❖ La única forma de que una transacción pueda hacer **WRITE** a un dato es que su **hora de entrada** sea **mayor** que **la última hora** en que se hizo un **READ** al dato y a su vez sea mayor a **la última hora** en que se hizo un **WRITE** al dato.
  - ➢ La nueva **última hora de escritura** será la **hora de entrada** de la transacción que logró escribir.

- **Cualquier transacción que no pueda seguir con su ejecución falla y aborta.**

Granularidad

- El concepto de **granularidad** en el acceso a los datos está vinculado con el **tipo de bloqueo que se puede realizar sobre la Base de Datos.**
- La **granularidad gruesa** permite solamente **bloquear toda la Base de Datos** (reservado para el **diseñador**), y a medida que la **granularidad disminuye**, es posible **bloquear a nivel tabla, registro o hasta campos que la componen.**
- Los DBMS permiten **diversos niveles de bloqueo** sobre los datos.
- **Las transacciones en entornos concurrentes necesitan bloqueos a nivel de registros.**

Otras operaciones Concurrentes

- Operaciones como **insertar** o **borrar tuplas** de una **tabla** también necesitan garantizar su operación en forma aislada.
- Necesitan tener el control exclusivo de los datos o, en su defecto, con el protocolo basado en Hora de entrada deben cumplir las mismas condiciones que una operación de escritura.

Bitácora en Entornos Concurrentes

- **Se aplica de la misma forma que en entornos monousuarios**
  - ❖ Garantiza la **atomicidad** de la **transacción** ante posibles fallos originados en su ejecución.
- **Se agrega un nuevo tipo de fallo**

- ❖ Una transacción puede fallar por problemas de bloqueos o acceso a la Base de Datos.
- ❖ Como ante cualquier fallo, la transacción debe abortar, dejando a la Base de Datos en el estado de consistencia anterior al comienzo de su ejecución.

- Hace uso de funciones **REDO y UNDO** dependiendo del tipo de **modificación**.
- **Existe un único buffer de datos tanto para la Base de Datos como para la Bitácora.**
- Cada transacción tiene su propia área donde administra la información localmente.
- El fallo de una transacción significa deshacer el trabajo llevado a cabo por ella, y puede llevar al retroceso de otras transacciones.

Retroceso en cascada de transacciones

- **El retroceso de una transacción puede llevar a abortar otras.**
- **Puede llevar a deshacer gran cantidad de trabajo.**
- **Se agrega una nueva condición al problema**
  - ❖ Una transacción Tj no puede finalizar su ejecución, si una transacción Ti anterior utiliza datos que Tj necesita, y Ti no está en estado de finalizada. De esta forma, si Ti falla, Tj también deberá fallar.

Puntos de Verificación

- **No se puede garantizar la existencia de un momento temporal en que ninguna transacción se encuentre activa.**
- Se guarda el checkpoint junto a una **lista de transacciones** conformada por las que al momento de colocar el checkpoint, estaban **activas**.
- **Ante un fallo**
  - ❖ Se revisa **antes** del **Checkpoint** sólo aquellas **transacciones** que estén en la **lista**.
  - ❖ El **resto se ejecuta a partir de este**, como en un **entorno monousuario.**