BD

Conceptos básicos de BD

Definiciones de BD

- Es una colección de datos relacionados.
- Colección de **archivos** diseñados para servir a múltiples aplicaciones.
- Colección o conjunto de datos interrelacionados con un propósito específico vinculado a la resolución de un problema del mundo real.
- Un dato representa hechos conocidos que pueden registrarse y que tienen un resultado implícito.

Propiedades implícitas de una BD

- Una BD representa algunos aspectos del mundo real, a veces denominado Universo de Discurso.
- **Una BD es una colección coherente de datos con significados inherentes**. Un conjunto aleatorio de datos no puede considerarse una BD. O sea **los datos deben tener cierta lógica**.
- Una BD se diseña, construye y completa de datos para un propósito específico. Está destinada a un grupo de usuarios concretos y tiene algunas aplicaciones preconcebidas en las cuales están interesados los usuarios.
- Una BD está sustentada físicamente en archivos de dispositivos de almacenamiento persistente de datos.

Resumen

- 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 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.

DBMS o SGBD

- Las siglas -> Data Base Management System o Sistema Gerenciador de Base de Datos.
- Es una colección de programas que permiten a los usuarios crear y mantener la BD.
- Es un sistema de software de propósito general que 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.
- Restricción accesos no autorizados -> seguridad.
- Suministro de almacenamiento persistente de datos (aún ante fallos).
- Integridad en los datos.
- Backups.

Componentes de un DBMS

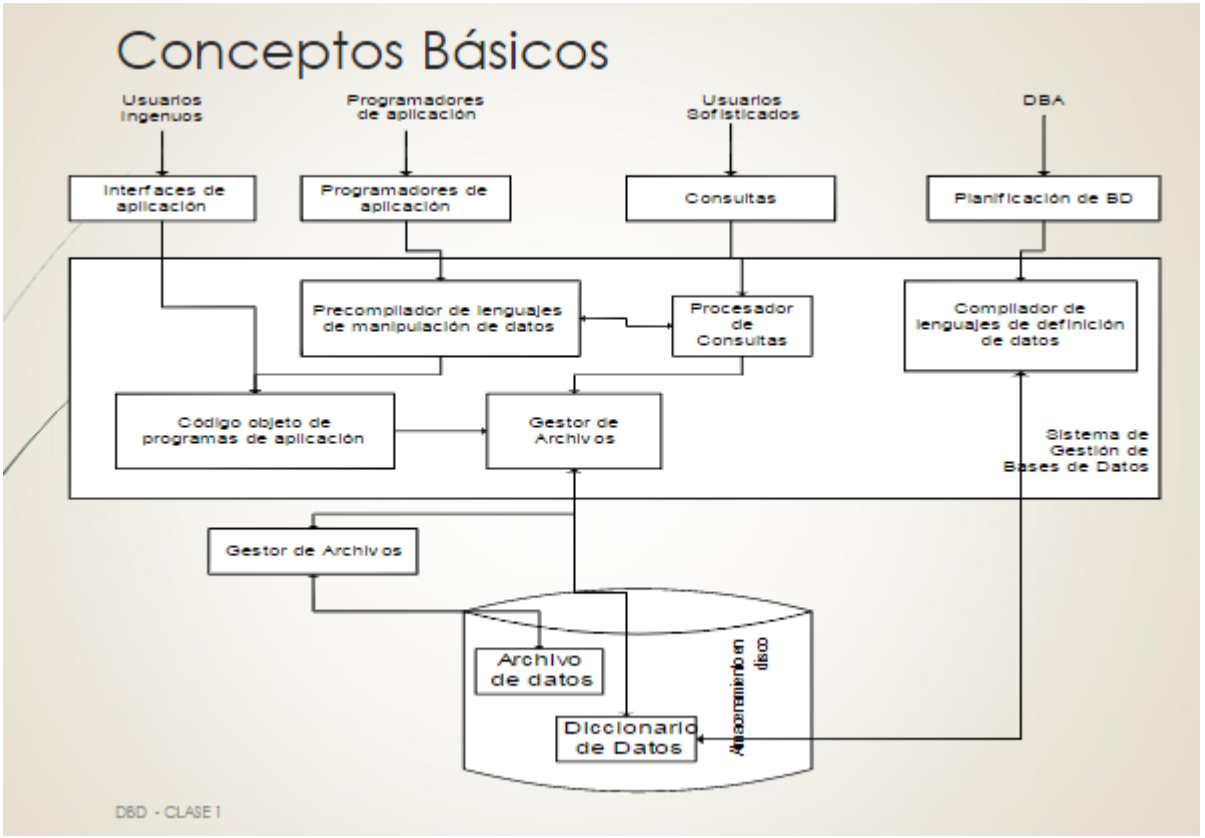
- **DDL** (data definition language):
    - Especifica el esquema de BD.
    - Resultado: Diccionario de datos.
- **DML** (data manipulation language):
    - Recuperación de información.
    - Agregar información.
    - Quitar información
    - Modificar información.

Características de DML

- **Procedimentales (SQL)**: Requieren que el usuario especifique **qué** datos se muestran y **cómo** obtener esos datos.
- **No Procedimentales (QUE)**: Requieren que el usuario especifique **qué** datos se muestran y **sin especificar cómo** obtener esos datos.
    - Más fáciles de utilizar
    - Aprendizaje más sencillo

Actores involucrados con una BD

- **DBA o ADB**
    - Administra el recurso, que es la BD. 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 BD**
    - Definen la estructura de la BD de acuerdo al problema del mundo real que esté representando.
- **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 BD generada por el diseñador.
- **Usuarios (distintos tipos)**



Propósitos más relevantes

- Aprender a definir una BD.
    - Construcción del modelo de datos -> Diseño
    - Normalización.
- Aprender a manipular una BD.
    - Lenguaje de trabajo clásico con BD.
- Estudio de seguridad e integridad de la información.

Modelado

- Colección de herramientas conceptuales para describir datos, relaciones entre ellos, semántica asociada a los datos y restricciones de consistencia.
- Abstracciones:
    - **Visión**: Vé sólo los datos de interés (muchas vistas para la misma BD) a través de programa de aplicación.
    - **Conceptual**: **qué** datos se almacenan en la BD y **qué** relaciones entre ellos.
    - **Físico**: Describe **cómo** se almacenan realmente los datos (archivos y hardware).

Modelos

- **Basado en objetos** (visión, conceptual).
    - Estructura flexible, especifican restricciones explícitamente.
    - Representan la información de un problema del mundo real con un esquema de **alto nivel de abstracción**
    - Por ej:
        - Modelo de Entidad-Relación
            - Integrado por entidades y sus relaciones
            - **Entidad**: Objeto de la realidad que se describe por un conjunto de características denominadas **atributos**
            - Ofrece simplicidad y potencia expresiva
        - Modelo Orientado a Objetos.
            - Representa la realidad con objetos de similares características del mundo real
- **Basado en registros** (conceptual, físico).
    - La BD se estructura en registros de formato fijo.
    - Se dispone de lenguaje asociado para expresar consultas.
    - Por ej:
        - OO.
        - Relacional.
            - Utiliza un **conjunto de tablas para representar la información**
            - Cada fila (tupla) representa un **registro** y cada columna un **campo** o atributo
            - Estas tablas **se vinculan entre sí y establecen relaciones de integridad** entre los datos que la componen
            - El más utilizado desde hace tres décadas
        - Jerárquico
            - Registros vinculados entre sí formando una estructura de árbol
        - Red.
            - Registros vinculados que generan una estructura de grafo
- **Físico de datos**.

Independencia de datos

- Capacidad de modificar esquemas sin alterar otro nivel.
    - **Físico** (modificar el esquema físico sin provocar que los programadores tengan que reescribir los programas de aplicación. Generalmente para mejorar el funcionamiento).
    - **Lógico** (modificar el esquema conceptual).

Categorías de software de procesamiento de datos

- Sin independencia de datos (SO, transferencia a un sector en particular).
- Independencia física (leer un registro de un archivo, SO).
- Independencia lógica parcial (leer siguiente registro de un archivo).
- Independencia lógica y física (leer siguiente registro de un tipo particular, DBMS).
- Independencia geográfica (BD distribuidas).

Diseño de datos

Tres etapas

- **Conceptual** (representación abstracta).
    - Integración de vistas.
- **Lógico** (representación en una computadora).
- **Físico** (determinar estructuras de almacenamiento físico).

Modelo de datos

- Un modelo de datos sirve para hacer más fácil la comprensión 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.
    - Obtener como cada actor utiliza cada dato.

Abstracciones

- Proceso que permite seleccionar algunas características de un conjunto de objetos del mundo real, dejando de lado rasgos que no son de interés.
- Tres abstracciones:
    - Clasificación.
    - Agregación.
    - Generalización.

Abstracción de clasificación

- Define una clase.
- Clase:
    - Declaración o abstracción de objetos
    - Se origina a partir de características comunes que tienen los objetos que la componen
- Un objeto puede pertenecer a varias clases
- Por ej:
    - Clase: dia_de_semana
        - Características: laborable o no, hora de comienzo y fin de actividades, etc
        - Lunes, martes, etc, pertenecen a la clase
    - Clases: auto, camión, vehículo naftero, vehículo gasolero
        - Un auto naftero pertenecería a la clase auto y a vehículo naftero
- La relación entre objeto y clase suele representarse con una flecha de puntos desde el objeto hacia la clase: ⇢
- Permite identificar los campos o atributos de los elementos individuales de datos

Abstracción de agregación

- Define una nueva clase a partir de otras clases que representan sus partes componentes.
- Por ej:
    - Un auto está compuesto por motor, chasis, carrocería y ruedas, entre otros
- Estos componentes serían “clases hoja”, las cuales representan una relación “es parte de” de la clase raíz
- La relación entre estas clases suele representarse con ⇒ desde la clase hoja hacia la clase raíz
- Permite agrupar campos o atributos formando registros de datos

Abstracción de Generalización

- Define una relación de subconjunto entre los elementos de dos o más clases.
- Las especialidades (hijos) heredan las características del padre.
- Representan una relación de “es un” entre los hijos y el padre
- Por ej:
    - Clases: Persona, Alumno y Docente
    - La clase padre, Persona, tendrá atributos comunes como nombre, dni, etc
    - Las clases hijas, Alumno y Docente, tendrán atributos representativos de ellas, como número de alumno o jerarquía docente, entre otros
    - Un objeto Alumno, tendría los atributos de la clase Alumno, y los heredados de la clase Persona

Propiedades de las abstracciones

- **Agregación binaria**
    - Correspondencia existente entre dos clases diferentes
    - Por ej:

        - Nacido_en establece una correspondencia entre persona y ciudad
    - **Cardinalidad**
        - Nivel de correspondencia existente entre los objetos de la clase
        - Así, es posible definir el nivel mínimo de correspondencia (Cardinalidad mínima) y el máximo (Cardinalidad máxima) entre los objetos de la clase
        - Card mínima
            - Si es 0, indica que en la agregación hay una participación opcional
            - Si es 1, la participación es obligatoria
        - Card máxima
            - Si la card max(a,b) = 1 y card max(c,b) = 1, se dice que la cardinalidad es uno a uno
            - Si la card max(a,b) = 1 y card max(c,b) = n, se dice que la cardinalidad es uno a muchos
            - Si la card max(a,b) = n y card max(c,b) = 1, se dice que la cardinalidad es muchos a uno
            - Si la card max(a,b) = n y card max(c,b) = n, se dice que la cardinalidad es muchos a muchos
    - También existen agregaciones n-arias
- **Cobertura en generalización**
    - Grado de relación entre padres e hijos
    - Dos coberturas
        - Total o parcial
        - Exclusiva o superpuesta
    - Cobertura Total
        - Cada elemento del padre está contenido en alguno de los hijos
    - Cobertura Parcial
        - Pueden existir elementos del padre que no se instancien sobre los hijos
    - Cobertura Exclusiva
        - Un elemento del padre puede estar sólo en un hijo
    - Cobertura Superpuesta
        - Un elemento del padre puede estar en varios hijos
    - Por ejemplo
        - Clases Persona, Docente y Alumno
        - Si puede existir un elemento exclusivamente de la clase Persona, la cobertura es Parcial, si no, es Total
        - Si un elemento puede ser Docente y Alumno a la vez, la cobertura es superpuesta, si no, exclusiva

Modelo Conceptual ER

- Técnica de modelado de datos más ampliamente utilizada
- Presenta las entidades de datos, sus atributos asociados y las relaciones entre estas entidades

Objetivos

- Representar la información de un problema en un alto nivel de abstracción.
- Captar la necesidad de un cliente respecto del problema que enfrenta.
- Mejora la interacción cliente / desarrollador disminuyendo la brecha entre la realidad del problema y el sistema a desarrollar.

Características

- **Expresividad**: Disponer de todos los medios necesarios para describir un problema.
- **Formalidad**: Cada elemento representado sea preciso y bien definido, con una sola interpretación posible.
- **Minimalidad**: Cada elemento tiene una única representación posible.
- **Simplicidad**: El modelo debe ser fácil de entender por el cliente y por el desarrollador.

Componentes

- Entidades
- Relaciones
- Atributos

Entidades

- Representa un elemento u objeto del mundo real con identidad.
- Se diferencia de cualquier otro objeto o cosa.
- Conjunto de entidades
    - Representación que, a partir de las características propias de cada entidad con propiedades comunes, se resume en un núcleo.

Relaciones

- Representan agregaciones entre dos (binaria) o más entidades.
- 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.
- Tipos de relación
    - Binaria
    - Ternaria
    - N-aria
    - Recursiva
- Cardinalidad de la relación
    - Define el grado de relación existente en una agregación
    - Cardinalidad Máxima
    - Cardinalidad Mínima

Atributos

- Representa una propiedad básica de una entidad o relación.

- Equivale a un campo de un registro.
- Cardinalidad.
    - Monovalente/polivalente
    - Obligatorio/opcional (nulo)

Componentes adicionales

- Identificadores
- Atributos compuestos
- Jerarquías/subconjuntos

Identificadores

- Es un atributo o conjunto de atributos que permite reconocer una entidad de manera unívoca dentro del conjunto de entidades.
- Puede ser
    - Simples o compuestos.
    - Internos o externos.

Atributos compuestos

- Representan un atributo generado a partir de una combinación de atributos simples.
- Puede ser polivalente y no obligatorio.
- Sus atributos simples pueden ser polivalentes y no obligatorios también.

Jerarquías

- Permite extraer propiedades comunes de varias entidades (o relaciones) y generar una superentidad que las contenga.
- Cobertura:
    - Total o parcial.
    - Superpuesta o exclusiva.

Ventajas y desventajas

- Ventajas
    - **Expresividad**: Disponer de todos los medios necesarios para describir un problema.
    - **Formalidad**: Cada elemento representado sea preciso y bien definido, con una sola interpretación posible.
    - **Minimalidad**: Cada elemento tiene una única representación posible.
    - **Simplicidad**: El modelo debe ser fácil de entender por el cliente y por el desarrollador.
- Desventajas
    - Lograr que un modelo sea expresivo ayuda a comprender el problema, pero puede atentar contra la simplicidad
    - La forma de expresar y determinar la cardinalidad y los identificadores no es sencilla
        - La forma gráfica y semántica de expresar la cardinalidad puede variar según el libro o texto que se utilice

    - Las relaciones n-arias también presentan inconvenientes

Revisiones del Modelo conceptual

Decisiones

- **Compleción o completitud**: Representa todas las características del dominio de aplicación (análisis de requerimientos).
- **Corrección:** Usar con propiedad conceptos E-I.
    - **Sintáctica:** Conceptos E-I se usan correctamente.
    - **Semántica:** Conceptos se usan de acuerdo a su definición. Errores más frecuentes:
        - Usar atributos en lugar de entidades.
        - Olvidar una generalización.
        - Olvidar una propiedad de herencia.
        - Usar entidades en lugar de interrelaciones.
        - Olvidar un identificador de una entidad.
        - Omitir cardinalidad
- **Minimalidad:** Cada aspecto aparece una sola vez en el esquema.
    - Ciclo de relaciones
        - Cuando una entidad A está relacionada con una entidad B, la cual está relacionada con una entidad C, la que a su vez se relaciona con la entidad A
        - Si una de las relaciones puede ser quitada y aún así representar la misma información, el esquema no es mínimo
    - Atributos derivados
        - Atributo del modelo cuya información puede obtenerse aunque no se encontrara en el mismo
            - Por ejemplo, cantidad de materias aprobadas, que podría verse contando las relaciones con las materias y las notas
        - Si existen atributos derivados, el esquema no es mínimo
            - Pueden existir igualmente en ciertos casos
            - La decisión puede pasarse al esquema lógico
- **Expresividad:** Representa los requerimientos de manera natural y se puede entender con facilidad.
- **Autoexplicación:** El esquema se explica a si mismo cuando puede representarse un gran número de propiedades usando el modelo conceptual, sin otros formalismos.
  Transformaciones para expresividad y autoexplicación:
    - Eliminar sub-entidades colgantes de la generalización.
        - Cuando una entidad hija no tiene atributos
    - Eliminar entidades colgantes.
    - Crear generalización: Dos entidades similares, crea una jerarquía de generalización.
    - Dividir relaciones que puedan estar unidas innecesariamente

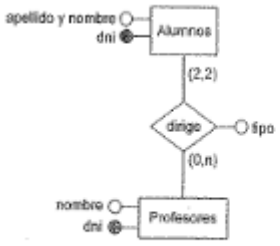


Figura 10.18 a)

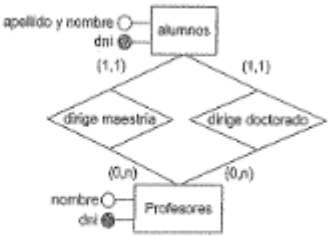


Figura 10.18 b)

-
    - Crear subconjuntos
        - Por ejemplo en casos en que ciertas relaciones y/o atributos serán usados cuando se defina este subconjunto

Figura 10.19 a)

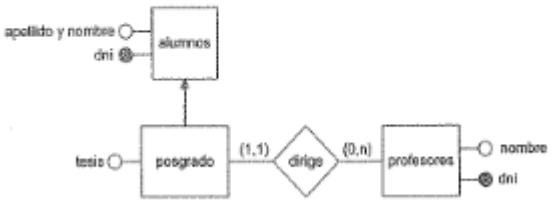


Figura 10.19 b)

- **Extensibilidad:** Un esquema se adapta fácilmente a los requerimientos cambiantes cuando puede descomponerse en partes, donde se hacen los cambios.
- **Legibilidad:**
    - Es legible si la representación gráfica es adecuada
    - Utilizar herramientas automatizadas.
    - Estructuras simétricas.
    - Se minimiza el número de cruces.
    - generalización sobre los hijos.

Modelo lógico

- Diseño lógico de alto nivel usando E-R
    - Convertir el esquema conceptual en un esquema lógico.
    - Enfoque global del diseño lógico.

Objetivo

- Convertir el esquema conceptual en un modelo más cercano a la representación entendible por el SGBD (el cual debe definirse el tipo en este paso, **relacional**, OO, jerárquico o de red)
- Representar un esquema equivalente al conceptual, que sea más eficiente para su utilización

Entradas

- **Esquema conceptual**
    - Representa la solución, a juicio del analista
    - El esquema lógico debe representar la misma información disponible en el conceptual
- **Descripción del modelo lógico a obtener**
    - Se deben definir las reglas que se aplicaran en el proceso de conversión
    - Están ligadas al tipo de SGBD seleccionado (**relacional**, en esta materia)
- **Criterios de rendimiento de la BD**
    - Durante la fase anterior, se consideran los requerimientos del usuario, pero hay otras necesidades que no pueden definirse en el modelo conceptual
    - Por ejemplo, performance de la BD
    - 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
- **Información de carga de la BD**
    - Ligado al concepto anterior
    - El analista debe observar cada entidad e interrelación definida, y ver la probable evolución de la información contenida en esas estructuras
    - La decisión 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

Decisiones

- Atributos derivados
- Atributos polivalentes
- Atributos compuestos
- Ciclo de relaciones
- Jerarquías
- Partición de entidades

Atributos derivados

- Ahora se debe tomar la decisión de dejarlos o no
- Ventaja: Disponibilidad de la información
    - Si una información es muy requerida, se ahorra el cálculo repetitivo y se obtiene más rápido
- Desventaja: Debe ser modificado cada vez que se modifica la información que lo contiene
- Pauta
    - Dejar aquellos que son muy utilizados
    - Quitar los que necesitan ser recalculados con frecuencia
- En caso de que un atributo sea muy utilizado y con mucho recálculo, queda a criterio del analista

Ciclo de relaciones

- La decisión del analista pasa por tener el modelo mínimo, o por que posteriormente el modelo implique menos tiempo de procesamiento

Atributos polivalentes

- Los SGBD, en general, permiten que sus atributos contengan múltiples valores, estilo vectores en memoria, pero de tamaño estático (máximo fijo)
- Pero ningún SGBD permite que un atributo contenga valores múltiples determinados dinámicamente
- Predefinir un máximo en un atributo polivalente puede causar espacio desperdiciado en algunos objetos e insuficiente en otros
- El criterio de corrección es la Primera Forma Normal
    - Los atributos de entidades o relaciones deben ser atributos simples
- Solución
    - Quitar el atributo, generar una entidad con el mismo y una relación entre la nueva entidad y la que originalmente poseía el atributo polivalente
    - La relación será uno a muchos o muchos a muchos (En general, la segunda), dependiendo de la naturaleza del problema

Atributos compuestos

- Los SGBD, en general, no soportan estos atributos
- Tres respuestas posibles:
    - Generar un atributo único, concatenando los atributos simples del atributo complejo
        - Se pierde la identidad de cada atributo
        - Dificulta operaciones con los mismos
    - Definir los atributos simples en la entidad, sin uno complejo que los resuma
        - Mantiene la identidad de los atributos
        - Aumentan los atributos en la entidad
        - En general, la más indicada
    - Generar una nueva entidad
        - Capta la esencia del atributo compuesto
        - Es la opción más compleja

Jerarquías

- Punto más importante de convertir un modelo conceptual en lógico
- El modelo relacional no soporta el concepto de herencia
- Se necesita encontrar un mecanismo que represente las jerarquías, captando el dominio del conocimiento de estas, bajo un esquema que no administre herencia
- Tres opciones
    - **Eliminar las especializaciones** (subentidades o entidades hijas)
        - Dejar sólo la generalización (entidad padre)

        - Incorporar todos los atributos de sus hijos, de manera opcional (no obligatoria)
    - **Eliminar la generalización** (Entidad padre)
        - Dejar sólo las especializaciones
        - Incorporar los atributos del padre en cada entidad hija
    - **Dejar todas la entidades de la jerarquía**
        - Crear relaciones “ES_UN” uno a uno entre padre y cada uno de los hijos
        - Se mantienen los atributos originales
        - Es la que capta mejor la esencia de la herencia
- La cobertura de la jerarquía es la que más determina la solución viable en cada caso
    - Si la cobertura de la jerarquía fuese parcial, la segunda solución no resultaría aplicable
        - Rompe la equivalencia entre el modelo conceptual y lógico, ya que se pierde información
    - Si la cobertura es superpuesta, la segunda opción es aplicable, pero poco práctica.
        - Podría repetirse información en las subentidades generadas si un elemento perteneciera a varias especialidades
- Conclusión
    - El responsable de la elección de la política para eliminar las jerarquías es el diseñador de la BD
    - La tercera solución es la que capta mejor la esencia de la herencia, pero también la que genera mayor número de entidades y relaciones en el modelo final
    - Deben tenerse en cuenta la cantidad de entidades y relaciones para evitar problemas de performance en la utilización de la BD

Partición de entidades

- Reorganizar la distribución de las entidades en un conjunto de entidades (partición horizontal) que la componen, o de los atributos (partición vertical) que conforman cada conjunto de entidades
- Su objetivo es mejorar la performance de las operaciones futuras sobre la BD
- Horizontal
    - Permite separar las entidades que conforman un conjunto
    - Por ejemplo, una entidad persona que agrupa clientes y proveedores, puede dividirse en dos conjuntos de entidades entidades, uno para cada grupo
- Vertical
    - Analiza un conjunto de entidades con múltiples atributos
    - Agrupa distintos atributos en nuevas entidades
    - Por ejemplo, un empleado podría tener sus atributos divididos en tres conjuntos diferentes: datos personales, datos laborales y datos salariales

Modelo físico

- El modelo físico (relacional) representa la BD como una colección de relaciones.

    - En otros términos -> Cada relación se asemeja a una tabla de valores, o a un archivo plano de registros.
    - Un registro o un elemento de una relación (tabla) se denomina tupla.
    - Un atributo mantiene su nombre.
    - Cada tabla de valores resultante se denomina *relación*.
        - Cada relación se obtiene a partir de una entidad o una relación ER.
    - El tipo de datos que describe los tipo de valores de un atributo se denomina dominio.

Pasos

- Eliminación de identificadores externos
- Selección de claves
    - Primaria
    - Candidata
- Conversión de entidades
- Relaciones

Eliminación de identificadores externos

- Cada una de las entidades deben poseer sus identificadores definidos en forma interna
- Deben incorporarse dentro de la entidad que contenga identificadores externos, aquellos atributos que permitan la definición del identificador de forma interna a la entidad
- En caso de existir más de un identificador, se deberá elegir el que luego será la clave primaria

Selección de claves: primaria, candidate y secundaria

- Para el usuario de la BD, estos conceptos no son importantes
    - Sólo es necesario que estén definidos identificadores que permitan distinguir una entidad del conjunto de entidades
- El administrador de la BD debe decidir cuáles de los identificadores se convierten en clave primaria o candidata
- Cuando se genera el esquema físico sobre el modelo relacional, se debe decidir el criterio de definición de clave primaria a partir de los identificadores reconocidos de cada entidad
- Si una entidad tiene definido un identificador es clave primaria de la tabla
- Si una entidad tiene varios identificadores definidos, la selección de la Clave Primaria (**CP**) debería realizarse del siguiente modo
    - Entre un identificador simple y uno compuesto, debería tomarse el simple, ya que es más fácil de tratar y/o usar
    - Entre dos simples, se debe optar pòr aquel de menor tamaño físico
    - Entre dos compuestos, se debe optar por aquel que tenga menor tamaño en bytes
- El resto de los identificadores serán definidos como Clave Candidata (**CC**)

- Los SGBD ofrecen el uso de índices secundarios a partir de claves candidatas y secundarias
- Mientras que el acceso físico al archivo de datos se logra a partir del uso de la CP, con los índices secundarios podemos acceder al índice primario
- Cualquier CP elegida que sea directamente tratable por el usuario puede sufrir borrados o modificaciones, que influyen negativamente en la performance final de la BD, dado que impactarán los índices primarios y/o secundarios
- Actualmente los SGBD permiten el uso de un atributo del dominio Autoincremental como CP, que es tratado exclusivamente por el SGBD
    - El usuario sólo puede consultarlo, no generarlo, borrarlo, ni modificarlo
    - Es la opción más eficiente
- Superclave
    - Conjunto de uno o más atributos que permiten identificar de forma única una entidad de un conjunto de entidades
    - Similar a una CP o CC, pero puede contener atributos innecesarios
    - Si un atributo es superclave, entonces también lo es cualquier superconjunto que incluya a dicho atributo

Conversión de entidades

- En general, cada una de las entidades definidas se convierte en una tabla del modelo
- Tabla:
    - Entidad = (Atributo_id, atributo1, atributo2, atributo3….)
    - Los atributos subrayados indican que son CP
- En el modelo físico, las CC no son indicadas
- La primer regla se rompe en caso de que existan dos entidades con una relación uno a uno con cobertura total entre ellas, siendo lo más indicado el uso de una única tabla
    - La existencia de una determina la existencia de la otra

Relaciones

- Cardinalidad Muchos a muchos.
- Cardinalidad Uno a Muchos.
    - **Clave foránea:** Atributo/s de una tabla que en otra tabla es/son CP y que sirven para establecer un nexo entre ambas estructuras.
    - Cobertura total.
    - Cobertura Parcial.
- Relaciones recursivas.
- Relaciones ternarias.

Cardinalidad muchos a muchos

- La solución es independiente de la cardinalidad mínima
- La relación N a N se convierte en tabla
- Los atributos que definen la CP de cada una de las entidades conforman la CP de la tabla

    - Si el par entre ambas CP puede repetirse entre distintas relaciones, se debe agregar otro atributo de la relación que, junto a las CP de las entidades, funcione de CP de la relación

Cardinalidad uno a muchos

- Con participación total
    - Se agrega la Clave Foránea (**CF**) a la tabla correspondiente a la entidad cuya cardinalidad máxima de la relación es 1
    - No necesita generar otra tabla
    - La CF es clave secundaria en la tabla donde aparece
- Con participación parcial del lado de muchos
    - Misma solución que con participación total
- Con participación parcial del lado de uno
    - Se genera una tabla como la resolución de muchos a muchos
    - Puede resolverse como con participación total, pero permite valores nulos, lo cual no es recomendado
- Con cobertura parcial de ambos lados
    - Igual que con participación del lado de uno

Cardinalidad uno a uno

- Con participación total
    - La CP de una entidad debe agregarse como CF a la otra
        - Puede elegirse cualquier entidad para esto
    - Se recomienda unir ambas entidades en una
- Con participación parcial de un lado
    - La CP del lado con participación parcial se utiliza como CP del lado con participación total
    - Es la opción utilizada al resolver jerarquías manteniendo todas las entidades
- Con participación parcial de ambos lados
    - No es muy común
    - Se recomienda generar una tabla como cardinalidad muchos a muchos

Relaciones recursivas

- Se resolverán como las otras relaciones, dependiendo de la cardinalidad

Relaciones ternarias

- Se resolverán como las otras relaciones, dependiendo de la cardinalidad

Integridad referencial

- Propiedad deseable de las BD.
- Asegura que un valor que aparece para un atributo en una tabla, aparezca además en otra tabla.
- Tipos de IR:

    - Restringir la operación.
        - Si se intenta borrar o modificar la tupla que tiene IR con otra, la operación se restringe
        - Por ejemplo
            - Si se tienen las clases Cliente y Factura, con idCliente en Factura
            - Para borrar el Cliente
                - No debería tener Facturas con su CP
                - Primero debemos borrar todas sus Facturas
            - Lo mismo si modificamos la CP sobre la tabla Cliente
    - Realizar la operación en cascada.
        - Si se intenta borrar o modificar una tupla sobre la tabla donde está definida la CP de la IR, la operación se realiza en cadena sobre todas las tuplas de la tabla que tiene definida la CF
        - Por ejemplo
            - Eliminar un Cliente, eliminaría todas las facturas del cliente
            - Al modificar la CP de un Cliente, se modificarían las CF en las tuplas de Factura
    - Establecer la clave Foránea en nulo.
        - Si se borra o modifica el valor del atributo que es CP, sobre la CF se establece valor nulo
        - Esta opción no es muy utilizada ni está presente en todos los SGBD
    - No hacer nada.
        - Se le indica al SGBD que no es necesario controlar la IR
        - Es equivalente a no definir restricciones de IR

Restricciones

- Restricciones de dominio.
    - Especifican que el valor de c/atributo A debe ser un valor atómico del dominio de A.
- Restricciones de clave.
    - Evita que el valor del atributo clave genere valores repetidos
- Restricciones sobre nulos
    - Evita que un atributo tome nulo en caso de no ingresarle valor.
- Restricciones de integridad
    - Ningún valor de la clave primaria puede ser nulo.
- Restricción de integridad referencial
    - Se especifica entre dos relaciones y sirve para mantener la consistencia entre tuplas de las dos relaciones
    - Establece que una tupla en una relación que haga referencia a otra relación deberá referirse a una tupla existente en esa relación.
    - Clave foránea está representada por un atributo de una relación que en otra es clave primaria.
- Las operaciones de Alta, Baja y Modificación (ABM) pueden generar violaciones a las restricciones anteriores.
    - Alta

        - Puede violar: Valor nulo para clave, repetición de la clave, integridad referencial, restricciones de dominio.
        - Si se viola la regla, la operación se rechaza
    - Baja
        - Puede violar: Integridad referencial (se procede como en el caso anterior).
    - Modificación
        - Puede violar: Cualquiera de las operaciones.

Dependencias Funcionales

Definición

- Una DF es una restricción entre dos conjuntos de atributos de la BD.
- Formalmente, una DF X -> Y entre dos conjuntos de atributos X e Y que son subconjuntos los atributos (R) de una relación (r), especifica una restricción sobre las posibles tuplas que podrían formar un estado de la relación r en R.
- La restricción indica que si t1 y t2 son dos tuplas cualesquiera en r y que si t1[X] = t2[X] entonces debe ocurrir que t1[Y] = t2[Y]
- Ésto significa que los valores del componente Y de una tupla de r depende de los valores del componente X.
- De un atributo que es CP se desprenden DF al resto de los atributos de la tabla
- Lo mismo con las CC

X -> Y

- El atributo Y depende del atributo X, ó
- El atributo X determina el valor único del valor Y, ó
- El valor del atributo Y está determinado por el valor del atributo X, ó
- Y depende funcionalmente de X.
- X = determinante
- Y = consecuente

Conjunto mínimo de DF

- Es fundamental definirlo
- Características
    - El consecuente de la DF debe estar formado por un sólo atributo. Si se define la DF X → Y, Y debe ser un sólo atributo
    - Si se define la DF X → Y, no es posible encontrar otra dependencia Z → Y donde Z sea un subconjunto de atributos de X. En este caso se define la DF como completa
    - No se puede quitar de un conjunto de DF alguna de ellas, y seguir teniendo un conjunto equivalente

Dependencia funcional completa

- Si A y B son atributos de una relación r, B depende funcionalmente de manera completa de A, si B depende de A pero de ningún subconjunto de A.

Dependencia funcional parcial

- A -> B es una dependencia funcional parcial si existe algún atributo que puede eliminarse de A y la dependencia continúa verificándose.
- Contradice la definición de conjunto mínimo de DF

Dependencia funcional transitiva

- Una condición en la que A, B y C son atributos de una relación tales que A -> B y B -> C entonces C depende transitivamente de A a través de B.
- Contradice la definición de conjunto mínimo de DF

Dependencia Boyce Codd

- Una DF X → Y se denomina Dependencia Funcional de Boyce-Codd (**DPBC**) cuando X no es una CP o CC, e Y es una CP o CC o parte de ella
- Genera repetición innecesaria de información

Normalización

Definición

- Técnica de diseño de BD que comienza examinando las relaciones que existen entre los atributos (dependencias funcionales).
- La normalización identifica el agrupamiento óptimo de estos atributos, con el fin de identificar un conjunto de relaciones que soporten adecuadamente los requisitos de datos de la organización.

Propósito

- Producir un conjunto de relaciones (tablas) con una serie de propiedades deseables partiendo de los requisitos de datos de una organización.

Otras definiciones

- La normalización es una técnica formal que puede utilizarse en cualquier etapa del diseño de BD.

- La **redundancia** de datos en un modelo es la causa primaria de posibles inconsistencias.
- Primer paso para un proceso de normalización
    - Identificar la CP y las CC de cada relación (tabla) del modelo.

Proceso

- Incremental -> cada vez más restrictivo.
- Comienza con BD en forma NO normal.
- A medida que se avanza las relaciones (tablas) tiene un formato cada vez más restringido y son menos vulnerables a anomalías de actualización.
- En general, 1NF es muy restrictiva (se aplica siempre).
- El resto puede ser opcional, de hecho 2NF y 3NF normalmente se aplican siempre.

Primera Forma Normal (1NF)

- Una tabla que contiene uno o más grupos repetitivos no está en 1NF, o sea una tabla que tenga atributos polivalentes.
- Un modelo estará en 1NF si para toda relación r del modelo (tabla) cada uno de los atributos que la forman es si y sólo sí monovalente

Segunda Forma Normal (2NF)

- Una tabla que tenga atributos que dependan parcialmente de otro no está en 2NF.
- Un modelo está en 2NF si y sólo sí está en 1NF y para toda relación r del mismo (tabla) no existen dependencia parciales.
- Deben quitarse estas DF parciales de la tabla que las presenta, moviendo los atributos que las generan a otras tablas, ya existentes o nuevas

Tercera Forma Normal (3NF)

- Una tabla que tenga atributos que dependan transitivamente de otro no está en 3NF.
- Un modelo está en 3NF si y sólo sí está en 2NF y para toda relación r del mismo (tabla) no existen dependencia transitivas.
- Deben quitarse estas DF transitivas de la tabla que las presenta, moviendo los atributos que las generan a otras tablas, ya existentes o nuevas

Boyce Codd Forma Normal (BCNF)

- Una tabla que tenga atributos que dependan de acuerdo a la definición de Boyce Codd de otro no está en BCNF.
- Un modelo está en BCNF si y sólo sí está en 3NF y para toda relación r del mismo (tabla) no existen dependencia de Boyce Codd.
- Algunos comentarios
    - Fue propuesta como una “suavización” de 3NF.

    - Pero resultó ser más restrictiva.
- Otra acepción de Boyce Codd
    - Una relación (tabla) está en BCNF si y sólo sí todo determinante es una clave candidata.
- Puede producir pérdida de Claves Candidatas.
- La decisión de si es mejor detener el proceso en 3NF o llegar a BCNF depende de
    - La cantidad de redundancia que resulte de la presencia de una DF de Boyce Codd.
    - De la posibilidad de perder una CC con la cual se podrían realizar muchos más controles sobre los datos.
- Se suele resolver al mover la DFBC a otra tabla
    - Para evitar perder información se pueden agregar más tablas (siempre que estas cumplan con FNBC

Dependencias Multivaluadas

- La posible existencia de DM en una relación se debe a 1NF, que impide que una tupla tenga un conjunto de valores diferentes.
- Así, si una tabla tiene dos atributos multivaluados, es necesario repetir cada valor de uno de los atributos con cada uno de los valores del otro. Así se garantiza la coherencia en la BD.
- En DM, denotada como X → → Y, siendo X e Y conjuntos de atributos en una tabla, indica que para un valor determinado de X es posible determinar múltiples valores para el atributo Y
- **En general, una DM se da entre atributos A, B y C en una relación de modo que para cada valor de A hay un conjunto de valores de B y un conjunto de valores de C, sin embargo los conjuntos B y C no tienen nada entre sí.**
- Se dice que A ->-> B y que A ->-> C.
- Se lee A multidetermina B y A multidetermina C.
- No es un problema el hecho que un atributo esté multideterminado
- Solución
    - Tabla 1 con A y B
    - Tabla 2 con A y C

Cuarta Forma Normal (4NF)

- Un modelo está en 4NF si y sólo sí está en BCNF y para toda relación r del mismo (tabla) sólo existen dependencia multivaluadas triviales.
- ¿Cuáles son triviales?
    - A ->-> B
    - A ->-> C
- ¿Cuáles no?
    - (A,B) ->-> C
    - (A,C) ->-> B

Quinta Forma Normal (5NF)

- Un modelo está en 5NF si y sólo sí está en 4NF y no existen relaciones con dependencias de combinación.
- Una dependencia de combinación es una propiedad de la descomposición que garantiza que no se generen tuplas espurias al volver a combinar las relaciones mediante una operación del álgebra relacional.
- En otras palabras
    - Se tiene A,B y C
    - (A,B,C) es CP
    - Ninguna combinación de conjuntos multidetermina otra.
- Se soluciona
    - t1 = (A,B)
    - t2 = (A,C)
    - t3 = (B,C)

Consultas

Lenguajes de consulta

- Utilizados para operar con la BD.
    - Procedurales: (Instrucciones para realizar secuencia de operaciones) (qué y cómo).
    - No Procedurales: (solicita directamente la información deseada) (que).
- Las consultas representan el 80% de las operaciones registradas sobre una BD.

Álgebra Relacional

- Lenguaje de consultas procedural.
- Operaciones de una o dos relaciones de entrada que generan una nueva relación como resultado.

Operaciones fundamentales

- Unitarias.
    - Selección: $\boldsymbol{\sigma_p}$**(Tabla)**.
        - Produce una tabla que contiene únicamente aquellas tuplas de Tabla que satisfacen el predicado p
    - Proyección: $\boldsymbol{\pi_L}$**(Tabla)**.
        - Produce una tabla que tiene un subconjunto de atributos(L) de Tabla eliminando tuplas duplicadas.
    - Renombre: **ρT(Tabla)**.
        - Renombra la tabla “Tabla” a “T”
- Binarias.
    - Producto Cartesiano: $\boldsymbol{T1 \times T2}$.
        - Produce una tabla concatenando cada tupla de T1 con todas las tuplas de T2.
    - Unión: **T1** ∪ **T2**.
        - Produce una tabla que contiene todas las tuplas de T1 más todas las de T2, eliminando las tuplas duplicadas. T1 y T2 deben ser compatibles (sus esquemas deben ser equivalentes en la cantidad, posición y dominio de los atributos, aunque sus nombres sí pueden ser distintos).
    - Diferencia: **T1** - **T2**.
        - Produce una tabla que contiene todas las tuplas de T1 que no se encuentran en T2. T1 y T2 deben tener esquemas compatibles.
- Adicionales.
    - Intersección: **T1** ∩ **T2**.
        - Produce una tabla que contiene todas las tuplas que se encuentran tanto en T1 como en T2. T1 y T2 deben tener esquemas compatibles.
    - Producto Natural: $\boldsymbol{T1 |\times| T2}$.

        - Produce una tabla concatenando tuplas de ambas tablas que tengan valores iguales en atributos con igual nombre (equi combinación). Se elimina uno de los ejemplares de cada atributo común.
    - División: **T1 % T2**.
        - Produce una tabla con los campos de T1-T2 (están en T1 y no en T2), donde los valores en esos campos de T1 se corresponden con TODAS las tuplas en T2. El esquema de T2 debe estar incluido en T1.
    - Asignación Temporal: **A** ⇐ **Consulta**.
        - Vuelca a A los resultados de CONSULTA. Luego puede utilizar A.

Operaciones de Updates

- Agregar tupla
    - Tabla ⇐ Tabla ∪ {(valor_atributo1, valor_atributo2, … ,valor_atributoN)}
    - Producto ⇐ Producto ∪ {(1235, “tuerca de 9 mm”, 10, 50,$10)}
- Eliminar tupla
    - Tabla ⇐ Tabla - Consulta
    - Producto ⇐ Producto – $\sigma_{codProd=893}$(Producto)
- Actualización de datos
    - $\delta_{atributo \Leftarrow valor\_nuevo}$ (Tabla)
    - $\delta_{saldo \Leftarrow saldo * 1.05}$ (Depósito)

Definición Formal

- Una expresión básica en AR consta de
    - Una relación o tabla de una BD
    - Una relación o tabla constante (puede ser un conjunto de tuplas expresadas literalmente cada una de ellas)
- Una expresión (consulta) general E en AR se construye a partir de subexpresiones (subconsultas) E1, E2, … ,En
- Las expresiones posibles son:
    - σ_p (E1) P Predicado con atributos en E1
    - π_S(E1) S Lista de atributos de E1
    - ρ_Z(E1) Z Nuevo nombre de E1
    - $E1 \times E2$
    - E1 ∪ E2
    - E1 - E2

SQL

Introducción

- Es un lenguaje de consultas de BD, que está compuesto por dos submódulos fundamentales

- Módulo para definición del modelo de datos, denominado DDL (Data Definition Language) con el cual se pueden definir las tablas, atributos, integridad referencial, índices y restricciones del modelo
- Módulo para la operatoria normal de la BD, denominado DML (Data Manipulation Language) con el cual se pueden realizar consultas, altas, bajas y modificaciones de datos sobre las tablas del modelo

Lenguaje de definición de datos (DDL)

- Es muy amplio.
- Operaciones más comunes.
  - CREATE DATABASE.
  - DROP DATABASE.
  - CREATE TABLE.
  - ALTER TABLE.
  - DROP TABLE.

Crear tabla

```
CREATE TABLE nombre_tabla ( nombre_atributo1 DOMINIO CARACTERÍSTICAS,
                            nombre_atributo2 DOMINIO CARACTERÍSTICAS,
                            …
                            nombre_atributon DOMINIO CARACTERÍSTICAS,
                            OTRAS CARACT);
```

Ej:

```
CREATE TABLE empresas
    ( idempresa INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    empresa VARCHAR(100) NOT NULL,
    abreviatura VARCHAR(10) NULL,
    cuit VARCHAR(13) NULL,
    direccion VARCHAR(100) NULL,
    observaciones TEXT NULL,
    PRIMARY KEY(idempresa),
    UNIQUE INDEX empresas_index19108(empresa));

CREATE TABLE pacientesempresas
    ( idpacienteempresa INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
    idpaciente_os INTEGER UNSIGNED NOT NULL,
    idempresa INTEGER UNSIGNED NOT NULL,
    fecha_desde DATE NOT NULL,
    fecha_hasta DATE NULL,
    PRIMARY KEY(idpacienteempresa),
    INDEX empleadosempresas_FKIndex1(idempresa),
    INDEX pacientesempresas_FKIndex2(idpaciente_os),
    FOREIGN KEY(idempresa)REFERENCES empresas(idempresa)
        ON DELETE RESTRICT
        ON UPDATE NO ACTION,
```

FOREIGN KEY(idpaciente_os) REFERENCES pacientes_os(idpaciente_os)
ON DELETE RESTRICT
ON UPDATE NO ACTION);

Modificar tabla

ALTER TABLE nombre_tabla (acción1, acción2, … ,acciónn);

Ej:
ALTER TABLE empresas
(Add column razon_social VARCHAR(100) NOT NULL,
Drop column cuit,
Alter column direccion VARCHAR(50) NULL);

Lenguaje de manipulación de datos (DML)

- Operadores
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
- Estructura básica
  - SELECT lista_de_atributos
    FROM lista_de_tablas
    WHERE condición
  - Equivale a $\pi_{lista\_de\_atributos}(\sigma_{condición}$ (tabla1 x tabla2 x … x tablan))
- Cláusula WHERE
  - Pueden usarse operadores lógicos: AND, OR, NOT, etc
  - Operador **BETWEEN** x and y
  - Operador **IS NULL** (su negación **IS NOT NULL**): Verifica si un atributo contiene el valor de NULL, valor que se almacena por defecto si el usuario no define otro
  - Operador LIKE
    - Brinda gran potencia para aquellas consultas que requieren manejo de Strings
    - %: Representa cualquier cadena de caracteres, inclusive la cadena vacía
    - _: Sustituye sólo el carácter del lugar donde aparece
    - descripcion LIKE “%pas”
    - nombre LIKE “zapa__”
    -
- Cláusula FROM
  - Podemos renombrar las tablas poniendo su nuevo nombre posterior al nombre de la tabla
    - FROM tabla1 t1, tabla2 t2
  - Producto Cartesiano = lista de tablas separadas por comas

    - Producto Natural = tabla1 INNER JOIN tabla2 ON (condición de igualdad entre los atributos correspondientes)
    - Producto Natural por equicombinación de FK y CP = tabla1 NATURAL JOIN tabla2
    - LEFT JOIN: Producto Natural pero conservando todos los registros de la tabla de la izquierda, completando con NULL los caso de no correspondencia
    - RIGHT JOIN: La inversa del LEFT JOIN
- Operadores para atributos
    - *: Indica todos los atributos de las tablas definidas en el FROM de la consulta
        - SELECT *
    - DISTINCT: Elimina tuplas repetidas
        - SELECT DISTINCT(atributo)
    - Los atributos utilizados en el SELECT de una consulta SQL pueden tener asociados operaciones válidas para sus dominios
    - Se puede utilizar AS para renombrar atributos
- Cláusula ORDER BY
    - Especifica el atributo por el cual las tuplas serán ordenadas
    - Por defecto se ordena de forma ascendente (asc), pero puede especificarse descendente (desc)
    - ORDER BY atributo1, atributo2 DESC, atributo3
- Otras operaciones de conjuntos
    - UNION: Agrupa las tuplas resultantes de dos subconsultas
    - UNION ALL: Union, pero conserva duplicados
    - INTERSECT: Intersección entre dos subconsultas
    - EXCEPT: Diferencia entre dos subconsultas
- Funciones de agregación
    - Promedio (avg): Aplicable a atributos numéricos, retorna el promedio de la cuenta
    - Mínimo (min): Retorna el elemento más chico dentro de las tuplas para ese atributo
    - Máximo (max): Retorna el elemento más grande dentro de las tuplas para ese atributo
    - Total (sum): Aplicable a atributos numéricos, realiza la suma matemática
    - Cuenta (count): Cuenta las tuplas resultantes
- Operaciones de Agrupamientos (GROUP BY)
    - Permite agrupar un conjunto de tuplas por algún criterio
    - HAVING: Permite aplicar condiciones a los grupos
    - GROUP BY atributo1, atributo2
- Operadores entre subconsultas
    - =: Cuando una subconsulta retorna un único resultado, es posible compararlo contra un valor simple
    - IN: Comprueba si un elemento es parte o no de un conjunto. Negación (NOT IN)
    - =SOME: Igual a alguno (<, >, >=, <=, <>)
    - >ALL: Mayor que todos (<, =, >=, <=, <>)
    - EXIST: Es verdadero si la subconsulta tiene al menos una tupla, y falso en caso contrario. Negación (NOT EXIST)
- ABM

    - INSERT INTO: Agrega tuplas a una tabla
        - INSERT INTO tabla (nombre_atributo1, nombre_atributo2, … ,nombre_atributon) VALUES (valor1, valor2, … , valorn);
        - INSERT INTO producto (nombre, descripcion, precio_unitario) VALUES (‘Sal Gruesa’, ‘Sal Gruesa marca Y x 20gr',20);
    - DELETE FROM: Borra una tupla o un conjunto de tuplas de una tabla
        - DELETE FROM tabla WHERE condición
        - DELETE FROM detalle WHERE idFactura>1000;
    - UPDATE … SET: Modifica el contenido de uno o varios atributos de una tabla
        - UPDATE tabla SET asignaciones
        - UPDATE factura SET total = total + total*0.21
- Creación de vistas
    - Tienen la estructura de una tabla y almacena temporalmente una consulta
    - Cada vez que la vista es utilizada, la consulta almacenada es “recalculada”
    - CreateView nombre as <expresion>
    - CreateView Judo as (select nombre from …)

Optimización de consultas

Optimizador de Consultas

- Los SGBD presentan en general un optimizador de consultas
- Proceso del gestor de BD, 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:
    - Analizador sintáctico (parser)
        - Genera una expresión manipulable por el optimizador de consultas. El análisis sintáctico convierte al texto de entrada, en este caso, la consulta del usuario, en una estructura tipo árbol, que resulta más útil para su análisis
    - Obtener una consulta equivalente más eficiente
        - Evalúa costos
    - Elección de índices
        - Considera el estado actual de la BD y los índices definidos, para resolver la consulta que se tiene hasta el momento, con el acceso a disco posible

Costo

- Componentes del “costo” de ejecución de una consulta:
    - Costo de acceso a almacenamiento secundario
        - Acceder al bloque de datos que reside en disco.
    - Costo de cómputo
        - Costo de realizar operaciones sobre memoria RAM.
    - Costo de comunicación

        - Costo de enviar la consulta y los resultados (si es un sistema Distribuido).

Optimización Lógica

- Expresiones equivalentes -> Álgebra Relacional
    - Existe una secuencia de resolución.
    - Se puede encontrar una expresión más eficiente que otra.

Selección: Personas del género masculino que sean solteros.

- $\sigma_{Genero='M' \wedge ECivil='Soltero'}(Persona)$
    - Se aplican 2 condiciones a todas las tuplas.
- $\sigma_{Genero='M'}(\sigma_{ECivil='Soltero'}(Persona))$
    - Se aplica 1 condición a todas las tuplas y 1 condición a menos tuplas.
- Conclusión: **Conviene realizar la selección lo antes posible**.

Proyección: DNI de las personas que vivan en la ciudad de Junín.

- $\pi_{dni}(Persona \mid\times\mid \sigma_{nombre='Junín'}(Ciudad))$
- $\pi_{dni}(\pi_{dni,\ idCiudad}(Persona) \mid\times\mid \pi_{idCiudad}(\sigma_{nombre='Junín'}(Ciudad)))$
- Conclusión: Conviene realizar la proyección para disminuir la cantidad de información que se almacena en buffers de memoria.
- **Lo mismo aplica respecto a otras operaciones binarias (Unión, Intersección, Diferencia).**

Producto Natural

- Es más eficiente resolver productos naturales entre más de dos tablas en distintos pasos
    - T1 |x| T2 |x| T3
    - (T1 |x| T2) |x| T3
- No es lo mismo T1 |x| T2 que T2 |x| T1 desde un punto de vista de performance
    - Mismo resultado, distintos tiempos de respuesta
    - Si no tienen atributos en común, son lo mismo ambas consultas
    - Si una de las tablas tiene el atributo en común como CP, conviene realizar la operación entre la tabla con CF sobre la tabla con CP
        - $T_{CF} \mid x \mid T_{CP}$
    - Si en ninguna el atributo en común es CP, entonces el producto debe realizarse sobre aquella tabla que tenga mayor cantidad de elementos diferentes
        - Para poder resolver estas situaciones, el optimizador dispone de estadísticas de la BD, sobre la cantidad de tuplas de las tablas y la distribución de valores para los atributos
        - Estas estadísticas se mantienen parcialmente actualizadas
            - El costo de tenerlas todo el tiempo actualizadas es alto
            - Cada cierto tiempo el gestor de BD ejecuta un proceso de actualización de las estadísticas

Algunos valores:

- CT tabla (cantidad de tuplas de la **tabla**).
- CB tabla (cantidad de bytes que ocupa cada tupla de la **tabla**).
- CV (a, tabla) (cantidad de ocurrencias distintas del atributo **a** en la **tabla**)
- Costo en Bytes selección: $\sigma_{(at = \text{“valor”})}$(Tabla)
    - (CT tabla / CV (at, tabla)) * CB tabla
- Costo en bytes proyección: $\pi_{at1, at2, ..atn}$(Tabla)
    - (CB at1 + CB at2 + .. + CB atn) * CT tabla
- Costo en bytes producto cartesiano: T1 X T2
    - (CT T1 * CT T2) * (CB T1 + CB T2)
- Costo producto natural: T1 |x| T2
    - Sin atributos en común
        - T1 X T2
    - Con atributo “a” en común
        - (CT T1 * CT T2) / MAX(CV (a, T1), CV (a, T2))

Transacciones

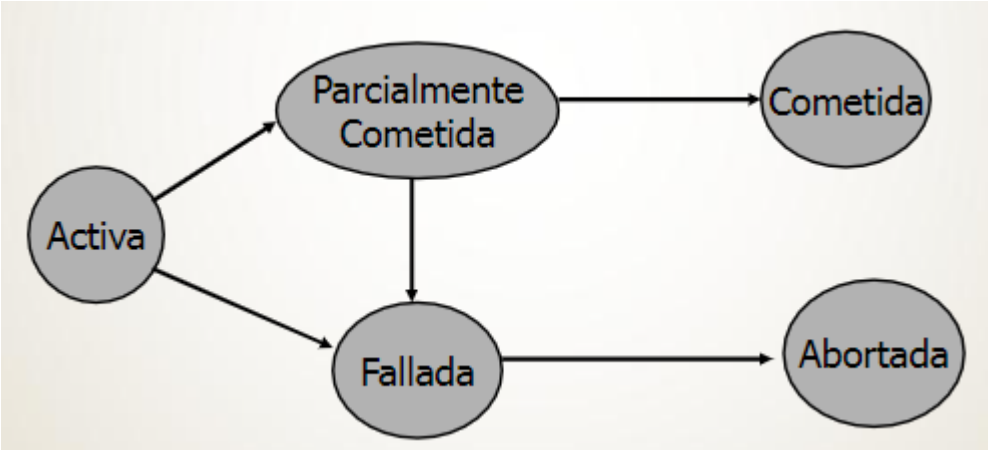
- Colección de operaciones que forman una unidad lógica de trabajo.

Propiedades ACID

- **Atomicidad**: Todas las operaciones de la transacción se ejecutan o no lo hacen ninguna de ellas.
    - Responsabilidad del gestor del SGBD implementar los algoritmos necesarios para que se controle la atomicidad de una transacción
- **Consistencia**: La ejecución aislada de la transacción conserva la consistencia de la BD.
    - Responsabilidad del programador
- **Aislamiento (isolation)**: Cada transacción ignora el resto de las transacciones que se ejecutan concurrentemente en el sistema, actúa como si fuera única.
- **Durabilidad**: Una transacción terminada con éxito realiza cambios permanentes en la BD, incluso si hay fallos en el sistema.

Estados

- **Activa**: Estado inicial, estado normal durante la ejecución.
- **Parcialmente Cometida**: Después de ejecutarse la última instrucción.
- **Fallada**: Luego de descubrir que no puede seguir la ejecución normal.
- **Abortada**: Después de haber retrocedido la transacción y restablecido la BD al estado anterior al comienzo de la transacción.
- **Cometida**: Tras completarse con éxito.



Modelo de transacción

- READ (A, a1)
- a1 := a1 - 100;
- WRITE (A, a1)
- READ (B, b1)
- b1 := b1 + 100;
- WRITE (B, b1)

Diferencia entre READ, WRITE y INPUT, OUTPUT.

Uso

- En sistemas monousuario
- En sistemas concurrentes
- En sistemas distribuidos

Fallos

- Los protocolos de trabajo tienen por objetivo asegurar la integridad y la consistencia de la BD, ante cualquier situación que lleve a no cumplir alguna de las cuatro propiedades definidas para una transacción
- Ej: rotura de disco o fuente de alimentación, errores de software (que deje de funcionar el SO o que falle el SGBD), etc
- Pueden clasificarse en dos tipos
    - Fallos que no producen pérdida de información
        - Si una transacción que solamente está consultando la BD falla y debe abortarse, no produce pérdida de información
        - Una transacción de consulta no modifica, borra ni agrega datos en la BD; por lo tanto, si se produjese un fallo en su ejecución, no se alteraría el contenido de la BD
    - Fallos que producen pérdida de información

        - Involucran transacciones que incluyen las cláusulas SQL: INSERT, DELETE o UPDATE
        - Estas transacciones afectan el contenido de la BD y debe asegurarse que la integridad de los datos se mantenga
- Los que producen pérdida de información son los más complejos de tratar, y los métodos de recuperación, tienen por objetivo salvar las situaciones de pérdida de integridad generadas por éstos
- Hay fallos con baja probabilidad de ocurrencia en los entornos informáticos actuales
    - La pérdida de energía eléctrica es salvada con UPS (Uninterruptible Power Supply, o Sistema de Alimentación Ininterrumpida) o grupos electrógenos
    - La rotura de disco rígido también puede ocurrir, pero para asegurar un sistema estable es probable que se disponga de al menos dos discos trabajando en espejo
        - RAIDs, manejados por el SO y el hardware
- Otros fallos: La CPU de la computadora puede fallar, el SO puede dejar de funcionar, el SGBD puede interrumpir su ejecución por errores internos, etc, y estas situaciones pueden llevar a la pérdida de consistencia de la BD
    - Necesitan métodos de recuperación de integridad de la BD

Métodos de recuperación de integridad de una base de datos

- Ponen especial énfasis en tratar la información contenida en memoria RAM y disco rígido
- Cualquier tipo de fallo producido afectará a la información en RAM, debido a que esta memoria es volátil
- Salvo la rotura o robo de disco rígido, los fallos no afectan a los datos contenidos en memoria no volátil

Acciones para asegurar la integridad de los datos. Atomicidad

- Acciones ante un fallo:
    - Volver a ejecutar completamente la transacción fallida
        - Al empezar desde una BD inconsistente, el resultado también lo será
    - No hacer nada
        - La BD permanecerá inconsistente
- Para evitar los problemas que puedan surgir, es necesario realizar acciones que permitan retrotraer el estado de la BD al instante anterior al comienzo de ejecución de la transacción
- Solamente se deben registrar los cambios en la BD ante la seguridad de que la transacción no va a fallar o, en su defecto, se deben tomar todas las acciones preventivas posibles para que, ante un fallo, se pueda mantener la consistencia
- Deben dejar constancia de las actividades llevadas a cabo por la transacción, que permitirán deshacer los cambios producidos
- Existen dos métodos de recuperación: el de **bitácora** (log) y el de **doble paginación**
- Cualquiera de estos métodos de recuperación necesita de dos algoritmos básicos:

    - Algoritmos que se ejecutan durante el procesamiento normal de las transacciones, que realizan acciones que permiten recuperar el estado de consistencia de la BD ante un fallo
    - Algoritmos que se activan luego de detectarse un error en el procesamiento de las transacciones, que permiten recuperar el estado de consistencia de la BD

Bitácora (log)

- También llamado Registro Histórico
- Todas las acciones llevadas a cabo sobre la BD deben quedar registradas en un archivo histórico de movimientos
- Repositorio donde se registran acciones
- 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
- Garantiza el estado de consistencia de la BD en todo momento, aún ante un fallo
- Cada transacción debe indicar su comienzo, su finalización y cada una de las acciones llevadas a cabo sobre la BD que modifiquen su contenido
- El gestor del SGBD es el responsable de controlar la ejecución de las transacciones, de administrar el archivo de bitácora y de utilizar los algoritmos de recuperación cuando el SGBD se recupera de una situación de fallo
- El gestor identifica cada transacción asignándole un número correlativo único a cada una de ellas.
- Entradas:
    - <T iniciada>
        - La entrada comienza su ejecución
    - <T, E, Va, Vn>
        - Operación de escritura
        - T: Identificador de la transacción
        - E: Identificador del elemento de datos
        - Va: Valor anterior
        - Vn: Valor nuevo
    - <T Commit>
        - Transacción T terminó sin fallos
    - <T Abort>
        - Transacción abortada
- Formato:
    - <T0 iniciada>
    - <T0, dato1, valor viejo, valor nuevo>
    - <T0, dato2, valor viejo, valor nuevo>
    - ...
    - <T0, daton, valor viejo, valor nuevo>
    - <T0 Commit>
- Si una transacción registra un inicio pero no un fin (por un fallo como un corte abrupto de suministro), debe abortarse

- Se debe garantizar que primero se graben los buffers del archivo de bitácora en memoria secundaria, antes de grabar en disco los buffers de la BD
    - Si no se respeta, se corre riesgo de perder la integridad de la BD
    - El gestor tendría que ir en contra del SO si este quisiese hacer lo contrario a esta condición
- Las operaciones sobre la BD deben almacenarse luego de guardar en disco el contenido de la Bitácora.
- Presenta dos alternativas para implementar el método de recuperación:
    - Modificación diferida de la BD.
    - Modificación inmediata de la BD.

Modificación diferida

- Las operaciones write se aplazan hasta que la transacción esté parcialmente cometida, en ese momento se actualiza la bitácora y la BD.
- Demora todas las escrituras en disco de las actualizaciones de la BD, hasta que la transacción alcance el estado de finalizada en bitácora
- Ventaja
    - Si una transacción activa falla, se puede asegurar que los cambios producidos por la transacción no tuvieron impacto sobre la BD
    - El fallo de la transacción es ignorado y la BD permanecerá integra
- No es necesario guardar en bitácora el valor viejo del dato
    - <Ti, daton, valor nuevo>
- Condición bajo la cual la BD puede perder la integridad
    - Se produce un fallo luego de que se guarda en disco la bitácora con <Ti commit> y que se haya actualizado parcialmente la BD.
    - Cuando se recupera del fallo, el algoritmo de recuperación ignoraría la transacción, ya que aparece como finalizada en la bitácora
    - Sin embargo, se activa el algoritmo REDO, que vuelve a ejecutar todas las transacciones que en bitácora tengan un start y un commit, devolviendo la integridad
        - Este algoritmo se ejecuta, incluso si la BD esta en estado de consistencia
- Las transacciones tienen una condición llamada **idempotencia**
    - Aunque se reejecute n veces una transacción, se genera siempre el mismo resultado sobre la BD
- Dada la siguiente transacción
    - <T0 Start>
    - <T0, A, 900>
    - <T0, B, 2100>
    - <T0 Commit>
    - Recién con T0 parcialmente cometida, entonces se actualiza la BD.
    - Ante un fallo, y luego de recuperarse:
        - REDO (Ti), para todo Ti que tenga un Start y un Commit en la Bitácora.
        - Si no tiene Commit entonces se ignora, dado que no llegó a hacer algo en la BD.

Modificación inmediata

- La actualización de la BD se realiza mientras la transacción está activa y se va ejecutando.
- Reducen la sobrecarga de trabajo al pasar muchas transacciones al disco a la vez, distribuyendolo en el tiempo
    - Las escrituras se siguen guardando primero en la bitácora y luego en disco
- Ahora ante un fallo, la BD puede estar parcialmente modificada, por lo que se necesita otro algoritmo, uno de UNDO, para retrotraer la BD al estado que tenía antes de comenzar la transacción
    - Para esto necesitamos los valores viejos de cada dato modificado por Ti registrados en bitácora
    - El algoritmo restaura estos valores en la BD
- Entonces ante un fallo, y luego de recuperarse:
    - REDO (Ti), para todo Ti que tenga un Start y un Commit en la Bitácora.
    - UNDO(Ti), para todo Ti que tenga un Start y no un Commit.

Buffers de Bitácora

- Grabar en disco c/registro de bitácora insume gran costo de tiempo -> se utilizan buffer. Proceder:
    - Transacción está parcialmente cometida después de grabar en memoria no volátil el Commit en la Bitácora.
    - Un Commit en la bitácora en memoria no volátil, implica que todos los registros anteriores de esa transacción ya están en memoria no volátil.
    - Siempre graba primero la Bitácora y luego la BD.

Puntos de Verificación

- Cuando ocurre un fallo, sin importar el tipo de modificación, todas las transacciones que tengan un start y un commit deberían rehacerse, lo que sería un trabajo lento e innecesario, ya que la gran mayoría no habrían presentado fallos
- Para evitar la revisión de toda la bitácora ante un fallo, el gestor de BD agrega en forma periódica, al registro histórico, una entrada <checkpoint> o punto de verificación
- Se agregan cuando se tiene la seguridad de que la BD está en estado consistente
- Ahora ante un fallo, sólo debe revisarse la bitácora desde el punto de verificación en adelante
- En entornos monousuario, siempre es posible colocar el punto de verificación en un instante del tiempo sin transacciones activas
    - Un poco más compleja para entornos concurrentes
- Periodicidad
    - Colocarlos “muy cerca”
        - Rehacer poco trabajo en el caso de producirse un fallo
        - Sobrecarga a la escritura de la bitácora de la BD
    - Colocarlos “lejanos”
        - Menos sobrecarga de escrituras en bitácora
        - Necesidad de rehacer mayor cantidad de trabajo ante un fallo

Doble paginación

Paginación en la sombra

- Plantea dividir a la 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
- Pasos ante operación de escritura
    - 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)
    - Se modifica el dato en memoria principal y se graba en disco, en un nuevo nodo, la página actual que referencia al nuevo nodo
    - Si la operación finaliza correctamente, se modifica la referencia de la página a la sombra para que apunte al nuevo nodo
    - Se libera el nodo viejo
- Ventaja: Menos accesos a disco
- Desventaja: Complicada en un ambiente concurrente/distribuido.
- N páginas equivalente a páginas del SO
    - Tabla de páginas actual
    - Tabla de páginas sombra

En caso de fallo

- Luego de la recuperación
    - Copia la tabla de páginas sombra en memoria principal.
    - Abort automáticos, se tienen la dirección de la página anterior sin las modificaciones
- Recuperación
    - Ventajas:
        - Elimina la sobrecarga de escrituras del log
        - Recuperación más rápida (no existe el REDO o UNDO)
    - Desventajas:
        - Es complejo dividir la BD en nodos o páginas
        - Sobrecarga en el compromiso: La técnica de paginación es por cada transacción.
        - Fragmentación de datos: Cambia la ubicación de los datos continuamente.
        - Garbage Collector: Ante un fallo queda una página que no es más referenciada.

Entornos concurrentes

- Los SGBD, a través de los gestores de BD, permiten que varias transacciones operen simultáneamente sobre la BD, situación que puede generar problemas de consistencia

- Dos transacciones, T1 y T2, son consistentes ejecutadas en entornos monousuarios, pero pueden no serlo si se ejecutan en simultáneo
    - Esto ocurre por la propiedad de aislamiento
- Una solución simple para el problema consiste en secuencializar la ejecución de las transacciones
    - Mantiene el aislamiento de las transacciones y la consistencia de la BD, ejecutando como monousuario
    - No es factible para entornos en los que se tendrían miles de usuarios simultáneos
- Se debería utilizar la concurrencia y mantener consistencia de datos
- Transacciones correctas, en ambientes concurrentes pueden llevar a fallos.

Seriabilidad

- Garantiza la consistencia de la BD
    - T0 Read(a)
      a := a - 50
      Write (a)
      Read (b)
      b := b + 50
      Write (b)
    - T1 Read(a)
      temp := a * 0,1
      a := a - temp
      Write (a)
      Read (b)
      b := b + temp
      Write (b)
    - Resolver T0, T1 o T1, T0 se respeta A+B
    - Ahora bien T0 T1 <> T1 T0

Planificación

- Secuencia de ejecución de transacciones
- Características
    - Involucra todas las instrucciones de las transacciones
    - Conservan el orden de ejecución de las mismas
- Cada transacción debe ser analizada como un todo, sin quitarle instrucción alguna, y que, cuando una transacción comienza, hasta que no finalice, ninguna puede iniciar
    - Garantiza la secuencialidad en la ejecución y, por ende, el cumplimiento del aislamiento para cada transacción
    - No aprovecha el entorno concurrente
- Un conjunto de m transacciones generan m! planificaciones en serie
- La ejecución concurrente no necesita una planificación en serie.
- Cuando la ejecución de una transacción no afecta el desempeño de otra transacción sobre la BD, ambas pueden ejecutarse concurrentemente sin afectar la propiedad de aislamiento

- READ(A)
  A := A –50
  WRITE(A)
  READ(A)
  TEMP := A * 0.1
  A := A –TEMP
  WRITE(A)
  READ(B)
  B := B + 50
  WRITE(B)
  READ(B)
  B := B + TEMP
  WRITE(B)
  A + B se conserva

- READ(A)
  A := A –50
  READ(A)
  TEMP := A * 0.1
  A := A –TEMP
  WRITE(A)
  READ(B)
  WRITE(A)
  READ(B)
  B := B + 50
  WRITE(B)
  B := B + TEMP
  WRITE(B)
  A + B no se conserva

Conclusiones

- El programa debe conservar la consistencia.
- La inconsistencia temporal puede ser causa de inconsistencia en planificaciones en paralelo.
- Una planificación concurrente debe equivaler a una planificación en serie.
- Solo las instrucciones READ y WRITE son importantes y deben considerarse.

Conflicto en planificaciones serializables

- I1, I2 instrucciones de T1 y T2
  - Si operan sobre datos distintos. No hay conflicto.
  - Si operan sobre el mismo dato
    - I1 = READ(Q) = I2, no importa el orden de ejecución. No hay conflicto
    - I1 = READ(Q), I2 = WRITE(Q) depende del orden de ejecución (I1 leerá valores distintos). Se genera conflicto
    - I1 = WRITE(Q), I2 = READ(Q) depende del orden de ejecución (I2 leerá valores distintos). Se genera conflicto
    - I1 = WRITE(Q) = I2, depende el estado final de la BD. Se genera conflicto
- I1, I2 está en conflicto si actúan sobre el mismo dato y al menos una es un write

Definiciones

- Una Planificación S se transforma en una S’ mediante intercambios de instrucciones no conflictivas, entonces S y S’ son equivalentes en cuanto a conflictos.
- Esto significa que si
  - S’ es consistente, S también lo será
  - S’ es inconsistente, S también lo será
- S’ es serializable en conflictos si existe S/ son equivalentes en cuanto a conflictos y S es una planificación serie.

- Una planificación concurrente debe ser equivalente a una planificación en serie, para que se mantenga la integridad de la BD

Pruebas de seriabilidad

- Uno de los algoritmos para demostrar la seriabilidad de planificaciones concurrentes, genera un grafo de precedencia entre las transacciones involucradas en una planificación.
- En dicho grafo,
    - Cada vértice representa una transacción de la planificación,
    - y una arista dirigida entre Ti y Tj significa que Ti realiza una operación de escritura antes que Tj realice una de lectura o escritura.
- El grafo puede tener ciclos
    - En caso de tener al menos uno, la planificación no es serializable en cuanto a conflictos

Control de Concurrencia

- El SGBD debe constar de algo más que de una definición para saber cuándo una planificación concurrente es válida o no
- Para poder ejecutar de manera concurrente y correcta una planificación que involucre múltiples transacciones simultáneas
- Como primera medida, es de notar que para que dos transacciones generen conflicto, estas deben operar sobre los mismos datos, aunque, generalmente, al mismo tiempo, utilicen la misma información en forma completa
- Existen dos variantes para lograr implementar el aislamiento
    - Protocolo de bloqueo
    - Protocolo basado en hora de entrada

Protocolo de bloqueo

- Se basa en que una transacción debe solicitar el dato con el cual desea operar, y tenerlo en su poder hasta que no lo necesite más
- Similar al bloqueo que realiza el SO
    - Por ejemplo, en un red de computadoras cuando varias computadoras quieren acceder a una misma impresora, el SO decide cuál imprime primero y el resto espera su turno en una cola
- En el caso de las transacciones, la BD ,y cada dato contenido en ella, actúa como un recurso compartido
- Cada transacción debe bloquear la información que necesita para poder llevar adelante su ejecución, utilizarla y luego liberarla, para que otra transacción necesite el dato pueda operar
- Existen dos tipos de bloqueos
    - Bloqueo compartido
    - Bloqueo exclusivo
- Un 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 mismo dato
- Un bloqueo exclusivo se genera cuando una transacción necesita escribir un dato
    - Garantiza que otra transacción no pueda utilizar ese dato (ya sea lectura o escritura de ese dato)
- Un bloqueo compartido puede coexistir con otro bloqueo compartido sobre el mismo dato
- Por el contrario, un bloqueo exclusivo es único
- Cada transacción debe solicitar el tipo de bloqueo que necesite
    - Si obtiene el dato, debe utilizarlo y luego liberarlo, cuando la transacción finaliza
    - Si el dato está siendo usado en ese momento, la transacción tiene dos posibilidades: esperar por el dato, o fallar y abortar, para luego comenzar como una transacción diferente
- Una transacción que solicita un bloqueo y no lo obtiene, como primera alternativa, puede quedar en situación de espera.
    - El programador de la transacción decide su comportamiento
    - Si se posee certeza de que el dato bloqueado será rápidamente liberado, es posible esperar
    - En caso contrario, resulta conveniente que la transacción falle y que el usuario sea el encargado de decidir si reinicia o no una nueva transacción
- En general, cuando se opera contra un SGBD, el porcentaje de transacciones que fallan por problemas de bloqueo es menor
    - Las transacciones suelen obtener los datos o esperar por ellos
- Se pueden mejorar las condiciones de aislamiento, llevando la ejecución de transacciones que pueden generar conflicto (dado que operan sobre el mismo dato) a una ejecución en serie
    - No soluciona el problema en un 100%
    - Puede plantear situaciones de deadlock (abrazo mortal)
- Deadlock:
    - Situación en que dos procesos obtienen un elemento y ambos solicitan el elemento que tiene el otro.
    - Como ninguno de los dos procesos puede obtener el segundo elemento solicitado, se produce deadlock, que impide continuar con la ejecución de los procesos
- Una transacción que traslada sus bloqueas al comienzo puede generar deadlock
- Si los bloqueos se ejecutan cuando son necesarios, las posibilidades de deadlock disminuyen, pero el riesgo de pérdida de consistencia aumenta
- Es preferible generar situaciones de deadlock, dado que estas no conducen a la pérdida de consistencia en los datos
- Para asegurar la integridad de la BD, el protocolo de bloqueo necesita reglas bien definidas, por ejemplo el protocolo de dos fases
- El protocolo de bloqueo de dos fases garantiza aislamiento en la ejecución de las transacciones, utilizando el principio de las dos fases: crecimiento y decrecimiento
- Para que una transacción sea correcta, el orden de pedidos de bloqueos debe ser como sigue
    - Fase crecimiento: Se solicitan bloqueos similares o se crece de compartido a exclusivo

    - Fase decrecimiento: exclusivo, compartido, liberar datos
- Se necesitan también políticas de detección de deadlock, inanición, protocolo de dos fases, etc
- Inanición: Cuando una transacción espera la liberación de un dato pero nunca lo obtiene, ya sea porque una o varias lo tienen bloqueado, o porque cuando se libera se le da a otras transacciones

- Conclusiones
    - Si los datos se liberan pronto -> se evitan posibles deadlock
    - Si los datos se mantienen bloqueados -> se evita inconsistencia
    - Inanición
- Dos fases
    - Requiere que las transacciones hagan bloqueos en dos fases:
        - Crecimiento: Se obtienen datos.
        - Decrecimiento: Se liberan los datos
    - Garantiza seriabilidad en conflictos, pero no evita situaciones de deadlock.
    - Como se consideran operaciones
        - Fase crecimiento: Se piden bloqueos en orden: compartido, exclusivo
        - Fase decrecimiento: Se liberan datos o se pasa de exclusivo a compartido.

Protocolo basado en hora de entrada

- Es una variante del protocolo de bloqueo, donde la ejecución exitosa de una transacción se establece de antemano, en el momento en que la transacción fue generada
- Cada transacción recibe una Hora de Entrada (HDE) en el momento que inicia su ejecución
- La HDE es una marca única asignada a una transacción
    - Puede ser el valor de un contador o la hora del reloj interno del servidor de la BD
    - Lo importante es que nunca dos transacciones tengan la misma HDE
- El método continúa asignando a cada elemento de dato de la BD dos marcas temporales: la hora de última lectura y la hora de última escritura
    - Corresponden a la HDE de la última transacción que leyó o escribió el dato
    - Se denomina HL(D) a la hora en que el dato fue leído por última vez, y HE(D) a la hora en que el dato fue escrito por última vez
- Para decidir si una transacción puede o no ejecutar su operación de lectura/escritura sobre la BD, el protocolo toma la decisión basado en el siguiente cuadro
    - Una operación de lectura
        - Para poder leer el dato, se debe cumplir que HDE(Ti) > HE(D)
            - Se asegura que cualquier transacción que intente leer un dato debe haber iniciado en un momento posterior a la última escritura del dato
        - Si HDE(Ti) < HE(D), el dato D es demasiado nuevo para que Ti pueda leerlo

                - Una transacción posterior a Ti pudo escribir el dato
                - Si se acepta la lectura del dato D por parte de Ti, es posible que Ti plantee una situación de pérdida de consistencia de información
            - Para leer un dato de la BD, el protocolo basado en HDE no necesita chequear la HL(D)
                - Las lecturas pueden ser compartidas
            - En caso de que la operación de lectura del dato D tenga éxito, se debe establecer la HL(D) como máximo entre HDE(Ti) y HL(D)
        - Una operación de escritura necesita chequear más información
            - Si HDE(Ti) < HL(D), la operación falla
                - No es posible que Ti escriba el dato D si fue leído por una transacción posterior a ella
                - Si se aceptara esa operación, la transacción que leyó resultaría incorrecta
            - Si HDE(Ti) < HE(D), la operación nuevamente falla
                - No es posible que Ti escriba el dato D, cuando dicho dato ya fue escrito por una transacción posterior
            - En cualquier otro caso, Ti puede ejecutar la operación de escritura, y establece HE(D) con el valor de HDE(Ti)
- Cualquier transacción que no puede seguir su ejecución falla y aborta su trabajo
    - Posteriormente, se puede generar una nueva transacción con una nueva HDE, e intentar ejecutar la operación deseada

Granularidad

- El concepto de granularidad en el acceso a los datos está vinculado con el tipo de bloqueo que se puede realizar sobre la BD
- La granularidad gruesa permite solamente bloquear toda la BD, y a medida que la granularidad disminuye, es posible bloquear a nivel tabla, registro o hasta campos que la componen
- En general, los SGBD permiten diversos niveles de bloqueo sobre los datos
- El nivel mayor de bloqueo es sobre la BD completa
    - Está normalmente reservada para el diseñador de la BD,
    - Se produce en actividades de mantenimiento
    - Durante este bloqueo, ningún usuario puede acceder a la BD
- Las transacciones en entornos concurrentes necesitan bloqueos a nivel de registros
    - Los bloqueos compartidos y exclusivos de datos apuntan a bloquear el uso o la escritura de registros (tuplas) de la BD
- Algunos SGBD permiten generar un nivel de granularidad aún menor
    - A nivel de campos o atributos de un registro o tupla

Otras operaciones concurrentes

- Operaciones como insertar o borrar tuplas de una tabla también necesitan garantizar su operación en forma aislada
- Una instrucción SQL INSERT necesita acceso único a operación

    - La escritura de una tupla requiere atomicidad
    - Otra transacción no puede leer, escribir ni borrar los datos que se están insertando
- Lo mismo al borrar una tupla, no es posible leer o modificar dicha tupla
- Cumplen los mismos requisitos que las operaciones lectura/escritura
    - Necesitan tener el control exclusivo de los datos o, en su defecto, con el protocolo basado en HDE deben cumplir las mismas condiciones que una operación de escritura

Bitácora en entornos concurrentes

- Se aplica sin cambios en entornos concurrentes
    - Garantiza la atomicidad de la transacción ante posibles fallos originados en su ejecución
- En entornos concurrentes, se agrega un nuevo tipo de fallo, una transacción que no puede continuar su ejecución por problemas de bloqueos o acceso a la BD
    - Como cualquier otro fallo, la transacción debe abortar, dejando la BD en el estado de consistencia anterior al comienzo de su ejecución
- Análogo a entornos monousuarios, con funciones REDO y UNDO, dependiendo el tipo de modificación
- Consideraciones
    - Existe un único buffer de datos tanto para la BD 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
- 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
    - Protocolo de bloqueo de dos fases: los bloqueos exclusivos deben conservarse hasta que Ti termine.
    - HDE, agrega un bit, para escribir el dato, además de lo analizado, revisar el bit si está en 0 proceder, si está en 1 la transacción anterior no termino, esperar....

Puntos de verificación de registro histórico

- No se puede garantizar la existencia de un momento temporal en que ninguna transacción se encuentre activa
- La sentencia <punto de verificación (L)> agrega un parámetro L
    - L contiene la lista de transacciones que, en el momento de colocar el punto de verificación, se encuentran 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