Modelados de Datos (CAPÍTULO 9)

El modelado de datos representa una de las componentes esenciales de los sistemas de información. Dos de los propósitos fundamentales del modelado de datos son:

- ayudar a comprender la semántica de los datos
- facilita la comunicación de los requerimientos del sistema de información (SI)

Un modelo de datos es un conjunto de herramientas conceptuales que permiten describir la información que es necesaria administrar para un SI, las relaciones existentes entre esos datos, la semántica asociada y las restricciones de consistencia. Sirven para hacer más fácil la comprensión de los datos de una organización. Se modela para:

- obtener perspectivas de los clientes y usuarios
- obtener naturaleza y necesidad de cada dato
- obtener cómo cada actor usa cada dato

Los modelos **se utilizan para describir la realidad del problema**. Para hacer un modelado se necesitan tres clases de abstracciones: clasificación, agregación y generalización.

Abstracciones

La abstracción es un proceso que permite seleccionar algunas características de un conjunto de objetos del mundo real, dejando de lado aquellos rasgos que no nos resultan interesantes. El diseño conceptual de una BD usa tres tipos de abstracciones: clasificación, agregación y generalización.

- **Clasificación**: se utiliza para definir una clase. Una clase es la definición de un objeto. Un objeto puede pertenecer a diferentes clasificaciones. Ej días de la semana. Permite identificar los campos o atributos de los elementos individuales de datos.
- **Agregación**: define una nueva clase a partir de un conjunto de otras clases que representan sus partes. Ej partes de un automóvil . Permite agrupar los campos o atributos formando registros de datos.
- **Generalización**: define una relación de subconjunto entre los elementos de dos o más clases. En una generalización las especialidades (hijos) heredan las características del padre. Ej de las personas

Propiedades de correspondencia binaria

Las abstracciones de agregación y generalización generan correspondencia entre las clases que las conforman, no ocurre así con la clasificación.

- **Agregación binaria**: es la correspondencia existente entre dos clases diferentes.
  Por ejemplo, nacido_en es una agregación binaria entre persona y ciudad.
  Nacido_en establece una correspondencia entre una persona particular y la localidad donde esa persona nació. A partir de esa correspondencia puede definirse un nuevo concepto, la cardinalidad entre las clases. Se define la cardinalidad como el nivel de correspondencia existente entre objetos de una clase. Así, es posible definir el nivel mínimo de correspondencia (cardinalidad mínima) y el nivel máximo de correspondencia (cardinalidad máxima) entre los objetos de una clase.Cuando la cardinalidad mínima es 0 indica que en la
  agregación hay una participación opcional, ahora bien, cuando la cardinalidad mínima es 1, indica participación obligatoria de la agregación binaria.
- **Generalización**: la propiedad de cobertura define el grado de relación entre padres e hijos. Existen dos coberturas a definir. La primera de ellas determina si la relación es total o parcial. Una cobertura **es total cuando cada elemento del padre está contenido en alguno de los hijos**, en cambio, una cobertura **es parcial cuando pueden existir elementos del padre que no se instancian sobre los hijos**. Por otra parte, la cobertura puede ser exclusiva o superpuesta, en este caso se analiza si un elemento del padre puede o no estar en más de un hijo.

Técnicas de modelado (ER - OO)

Un modelo de datos es una serie de conceptos que puede utilizarse para describir un conjunto de datos y las operaciones para administrarlo. Genéricamente los modelados se dividen en tres grande grupos: modelado basado en objetos, modelado basado en registros y modelados de datos físicos (no dados en la materia)

- **Modelado basado en objetos**: se utilizan para describir los datos de acuerdo con la visión que cada usuario tiene respecto de la BD. Posteriormente, a partir de todas y cada una de las vistas producidas, es posible generar un único modelo final que considere las necesidades de cada actor. Existen diferentes modelos basados en objetos (entre los que se encuentra el modelo entidad-relación).

- **Modelado basado en registros**: nuevamente permiten describir los datos desde la perspectiva de cada usuario. La diferencia básica con los modelos basados en objetos radica en que basados en registros permiten especificar la estructura lógica completa de la BD. Este modelo lleva este nombre ya que la BD se estructura como registros de longitud fija, conformados por campos. El modelo más representativo actualmente es el modelo relacional.

Introducción modelo de entidad-relación

La técnica de modelado más usada es la entidad-relación (ER), el cual presenta las entidades de datos, sus atributos asociados y las relaciones entre estas entidades. Se basa en la concepción del mundo real como un conjunto de objetos llamados entidades y las relaciones existentes entre dichas entidades. Cada entidad está conformada por un conjunto de atributos que la caracterizan. Por último, una relación establece un nexo entre las entidades. El modelo ER no llega a tener una implementación física, ya que es un diagrama. Una vez generado el modelo de datos ER definitivo para el problema, deberá utilizarse otra forma de modelado que nos permita obtener una implementación física.

Modelado e ingeniería de Software

La ingeniería de Software (IS) es una disciplina que abarca todos los aspectos de la producción de software, desde la elicitación de requerimientos hasta la etapa del mantenimiento del sistema, conocido como ciclo de vida de desarrollo de software.
Dentro de este ciclo están incluidas la modelización de datos del problema y la generación de una BD respectiva.
Es posible resumir la IS como una serie de etapas o actividades:

- comprender un problema
- analizarlo
- diseñar una solución
- implementar dicha solución
- entregar el producto
- ajustarlo a los cambios del medio donde se haya insertado

El modelado de datos se toma como una de las actividades propias del análisis del problema.
*SGBD = sistema gestor de base de datos. Permite almacenamiento, actualización y extracción de información en una base de datos. Ej MySql.

La metodología en este libro consiste en construir un modelo de datos en tres etapas:

1. **Modelado conceptual**: es desarrollado durante la etapa de adquisición de conocimiento del programa. Se desarrolla independientemente del modelo final y de su implementación en una SGDB.
2. **Modelo lógico**: el analista debe determinar el tipo de SGBD, debido a que las decisiones que debe tomar dependen de dicha elección
3. **Modelo físico**: debe especificarse que SGBD específico se va a utilizar

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

Introducción al modelo relacional

El modelo relacional utiliza un conjunto de tablas para representar las entidades y las relaciones existentes, definidas por el modelo ER. La estructura básica del modelo relacional es la tabla, cada tabla está conformada por columnas que representan los mismos atributos definidos en el modelo ER. Cada fila representa (registro) se denomina tupla.

Modelado entidad relación conceptual (CAPÍTULO 10)

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 (propiedades básicas):

- **expresividad**: expresar de la mejor manera la semántica de los datos a resolver
- **formalidad:** requiere cada elemento representado en el modelo sea preciso y bien definido, con una sola interpretación
- **minimalidad:** cada elemento tiene una única forma de representación posible y no puede expresarse mediante otros conceptos
- **simplicidad:** el modelo debe ser fácil de entender

Componentes del modelo conceptual

Se planteó tres constructores básicos del modelo conceptual: entidades, relaciones y atributos

- **entidades**: representa un elemento u objeto del mundo real con identidad, es decir, se diferencia unívocamente de cualquier otro objeto incluso siendo del mismo tipo. Alumno se diferencia de otro alumno.
  Conjunto de entidades es una representación que, a partir de las características propias de cada entidad con propiedades comunes, se resume en un núcleo.
- **relaciones**: representan las agregaciones entre dos (binaria) o más entidades. Describen las dependencias o asociaciones entre dichas 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.
  Pueden existir más de un conjunto de relaciones entre dos entidades. Por último, pueden existir relaciones recursivas entre entidades, es decir, se denomina relación recursiva a aquella relación que une dos entidades particulares del mismo conjunto. Cada relación debe tener su cardinalidad mínima y máxima.
  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. Es el equivalente al campo de un registro. Los atributos también tienen asociado el concepto de cardinalidad, es decir, cuando se define un atributo se debe indicar si es o no obligatorio y si puede tomar más de un valor (polivalente). Hay 4 expresiones posibles.
  Cardinalidad de atributos
  - Monovalente/polivalente (máximo valor permitido, mono 1, poli n)
  - Obligatorio/opcional (el mínimo es 1 si es obligatorio, sino si es nulo 0, lo que indica opcional)

Componentes adicionales del modelo conceptual

Estos elementos son cuatro:

- **Jerarquías de generalización:** permite extraer propiedades comunes de varias entidades o relaciones y generar con ellas una superentidad que las contenga. Así, las características compartidas son expresadas una única vez en el modelo y los rasgos específicos de cada entidad quedan definidos en la subentidad. La propiedad básica de la abstracción de generalización es **el concepto de herencia**. Las subentidades o especializaciones heredan los atributos de la superentidad.
- **Subconjuntos:** Representan un caso especial de las jerarquías de generalización.
  Ocurre que se tiene una generalización de la que se desprende una sola especialización. Siempre tienen cobertura **parcial-exclusiva** (PE): la cobertura no puede ser total porque sino una especialización y una generalización estarían siendo lo mismo, y no puede ser superpuesta dado que no hay una segunda especialización para suponerse.
- **Atributos compuestos:** representan a un atributo generado a partir de la combinación de varios atributos simples. Un atributo compuesto podría ser polivalente o no obligatorio. Lo mismo podría ocurrir con los atributos simples que lo componen.
- **Identificadores**: es un atributo o un conjunto de atributos que permiten reconocer de manera unívoca a una entidad dentro del conjunto de entidades. Está ligado al concepto de claves primarias y candidatas. Los identificadores pueden ser de dos tipos:

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

Los identificadores en una jerarquía también son heredados, además, las especializaciones de la jerarquía pueden tener identificadores propios.

Mecanismos de abstracción

Los tres mecanismos de abstracción previamente explicados están presentes en el modelo conceptual. Los tres componentes básicos del modelo conceptual se basan en la abstracción de clasificación:

- entidades: son una clase de objetos del mundo real
- relaciones: son un conjunto de objetos que relacionan dos o más entidades
- atributos: representan propiedades de las entidades

La abstracción por agregación está presente en:

- entidades: son agregaciones de atributos
- relaciones: son agregaciones entre entidades
- atributos compuestos: son agregaciones de atributos simples

Por último, la abstracción de generalización sólo se ha presentado en entidades, pero es posible utilizar generalizaciones de relaciones (poco usual).

Revisiones del modelo conceptual

Decisiones

- ¿Conviene generar una entidad con un concepto nuevo? O agregar un atributo a una entidad existente?
- ¿Cuándo se debe utilizar una generalización y cuando el concepto representa una clasificación?
- ¿Convienen los atributos compuestos? ¿O se deben generar atributos simples?

Existen una serie de conceptos a revisar:

- **Compleción/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
  - Atributos derivados
- **Expresividad**: representa los requerimientos de manera natural y se puede entender con facilidad.
- **Autoexplicación**: esquema se explica a sí mismo cuando puede representarse un gran número de propiedades usando el modelo conceptual, sin otros formalismos.
  - Eliminar sub-entidades colgantes de la generalización
  - Eliminar entidades colgantes
  - Crear generalización: dos entidades similares, crea una jerarquía de generalización
  - Crear Subconjuntos
- **Extensibilidad**: un esquema se adapta fácilmente a requerimientos cambiantes cuando puede descomponerse en partes, donde se hacen los cambios
- **Legibilidad:**
  - Utilizar herramientas automatizadas
  - Estructuras simétricas
  - Se minimiza el número de cruces
  - Generalización sobre los hijos

Modelado entidad relación lógico (CAPÍTULO 11)

Características del diseño lógico

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

1. **esquema conceptual:** es el resultado tangible de la etapa anterior. El esquema conceptual es la solución del problema original. El modelo lógico debe representar la misma información.
2. **descripción del modelo lógico a obtener:** aquí se deben aplicar las reglas que se van a tener en cuenta en el proceso de conversión, eso depende del tipo de SGBD que se eligió (relacional en este caso).
3. **criterios de rendimiento de la BD:** en esta fase de diseño se consideran requerimientos del usuario que tienen que ver con requerimientos no funcionales al problema, como por ejemplo mejorar la performance de la BD.
4. **información de carga de la BD:** aparece de cierta forma ligado al concepto anterior, ya que se busca mantener la performance de la BD bajo control ya que cuando más grande es un archivo de datos, más lento el el proceso de búsqueda.

Decisiones sobre el diseño lógico

Las decisiones sobre el diseño lógico están vinculadas con un conjunto de reglas que actúan sobre el esquema conceptual que NO están presentes en los SGBD relacionales. En otras palabras, se deben tomar decisiones de cómo tratar los siguientes conceptos en caso de que aparezcan en el modelo conceptual:

- **atributos derivados:** un atributo es derivado si puede conseguirse la misma información de otra manera dentro del modelo. Una vez que se detecta hay que decidir si se deja o no. Ventaja: la disponibilidad de información. Desventaja: la necesidad de ser recalculado cada vez que se modifica la información que contiene (ejemplo de la cantidad de materias). Por lo tanto, la pauta es dejar aquellos que son muy utilizados y quitar aquellos que necesitan ser recalculados con frecuencia.
- **atributos polivalentes**: un atributo polivalente es aquel que puede tomar varios valores diferentes. Ninguna SGBD relacional permite que un atributo tenga valores múltiples determinados dinámicamente porque utiliza estructuras estáticas (vectores). Es decir, se puede tener un atributo con múltiples valores, pero la cantidad máxima debe ser prevista. El criterio que se usa para solucionar este problema es **Primera forma Normal (1FN)**. Un modelo está en 1FN si todos los atributos de entidades o relaciones son atributos simples, es decir, que no haya ningún atributo polivalente. Una posible solución a un atributo polivalente en el esquema conceptual es si no conocemos la cantidad de información que necesitamos hacer del atributo polivalente una entidad, ahora bien, si conocemos la cantidad, el diseñador podría optar por definir el atributo con una estructura de 12 elementos sin generar una nueva entidad.
- **atributos compuestos:** los atributos compuestos son un atributo compuesto por varios atributos simples. Existen tres posibles variantes para tratar un atributo compuesto:
    1. un único atributo que se convierta en la concatenación de todos los atributos simples
    2. definir los atributos simples sin un atributo compuesto que los una, en general, es la solución más indicada
    3. generar una nueva entidad que representa al atributo compuesto, conformada por los atributos simples. Se nota que esta solución capta más la esencia de lo que es un atributo compuesto pero es más complicada de implementar.
- **ciclo de relaciones:** generan repetición innecesaria de información. Nuevamente la decisión recae sobre el analista, es decir, si se busca que el esquema quede mínimo o que sea menor el tiempo de procesamiento.
- **Jerarquías:** el modelo relacional no soporta el concepto de herencia, por consecuencia, las jerarquías no pueden ser representadas. Hay 3 soluciones:
    1. eliminar a los hijos dejando solo al padre, la cual incorpora todos los atributos de los hijos y cada uno de estos atributos provenientes de los hijos deberá ser opcional.
    2. eliminar al padre dejando solo a los hijos, incluyendo a los atributos del padre en las entidades hijas
    3. dejando todas las entidades de la jerarquía convirtiéndolas en relación (1,1) entre el padre y cada uno de sus hijos (relación es_un).

  No todas las soluciones son aplicables en todos los casos, la cobertura de la jerarquía determina la solución viable en cada caso. En resumen, si la cobertura es parcial o superpuesta, la opción de eliminar al padre y quedarnos con los hijos no es viable. Por otro lado, la tercera solución es la que más se adapta al concepto de herencia, pero al final del esquema es la que más cantidad de entidades y relaciones tiene, lo que a futuro podría ser un problema de performance en la BD.

Los subconjuntos, caso especial de jerarquía, tienen una cobertura parcial-exclusiva. Tanto la primera como la tercera opción son válidas, la segunda queda descartada.

Partición de entidades

Existen dos tipos de particiones. Ambos tipos tienen como objetivo mejorar la performance sobre futuras operaciones sobre la BD, otro objetivo también es mejorar la seguridad de la BD.

- **partición horizontal**: reorganiza la distribución de entidades en un conjunto de entidades. De una entidad "persona" que almacena clientes y proveedores, se definen dos entidades diferentes, una para clientes y otra para proveedores.
- **partición vertical**: reorganiza la distribución de los atributos. De una entidad empleado se pueden agrupar los atributos de acuerdo a una característica, por ejemplo datos laborales, datos personales, datos salariales.

Modelado físico (Relacional) (CAPÍTULO 12)

El modelo relacional representa a una BD como una colección de archivos denominados tablas, las cuales se conforman por registros. Cada tabla se denomina relación, y está integrada por filas y columnas. Cada fila representa un registro y se denomina tupla, mientras que cada columna representa un atributo del registro.
Un dominio representa un conjunto de posibles valores para un atributo. Como un dominio restringe los valores del atributo, puede considerarse como una restricción.

Eliminación de identificadores externos

El primer paso en la conversión del esquema lógico hacia el esquema físico consiste en la eliminación de identificadores externos. Cada una de las entidades que conforman el esquema lógico deben tener identificadores internos. Para esto, se deberán incorporar, dentro de la entidad que contenga identificadores externos, aquellos atributos que permitan la definición del identificador de forma interna. Se deberá incluir en la entidad entonces el atributo que a futuro va a ser definido como clave primaria.

Selección de claves: primaria, candidata y secundaria

Si una entidad tiene un solo identificador, este identificador es clave primaria de la tabla. Si tuviese varios identificadores la selección de la clave primaria (CP) debería realizarse del siguiente modo:

- entre un identificador simple y compuesto, deberíamos quedarnos con el simple
- entre dos identificadores simples, se debe optar por aquel de menor tamaño físico
- entre dos identificadores compuestos, se debería optar por aquel que tenga menor tamaño en bytes. Así, al construir un índice utilizando un árbol B como estructura, podemos almacenar más claves por nodo.
  *árbol B: árbol multicamino, es decir, se mantiene siempre balanceado pero a bajo costo
  *La CP se utiliza para lograr el acceso físico al archivo de datos. El resto de las claves son tomadas como índices secundarios que no referencian físicamente al archivo, sino que al índice primario.

El resto de los identificadores serán definidos como clave candidata (CC). Sin embargo, los SGBD, ofrecen una alternativa más conveniente. Hace falta recordar que cualquier CP que sea directamente tratable con el usuario puede sufrir modificaciones y borrados, lo que influirá negativamente en la performance. La alternativa que plantean ahora las SGBD es un atributo autoincremental al que el usuario solo pueda realizar operaciones de consulta, es decir, no puede borrarla ni modificarla.

Concepto de superclave

Es un conjunto de uno o más atributos que permiten identificar de forma única a una entidad de un conjunto de entidades. Una superclave puede contener atributos innecesarios. Es decir, si un atributo es superclave, entonces también lo es cualquier conjunto que incluya a dicho atributo. Una CP o CC es una superclave que no admite un subconjunto de ella como superclave.

Conversión de entidades

En general, cada una de las entidades definidas se convierten en una tabla del modelo. Hay excepciones, por ejemplo, cuando hay una relación con cardinalidad (1,1) con cobertura total entre las dos entidades, se puede generar una única tabla que aglutina los atributos de ambas.

Conversión de relaciones. Cardinalidad

Cardinalidad	Tabla	Foreign Key
((0|1),n) → (1,1) o (1,1) → ((0|1),n)	NO	La FK va a la tabla de cardinalidad 1,1
(0,1) → (1,1)	NO	La FK va a la tabla de cardinalidad 1,1
(1,1) → (1,1)	NO	La FK va a una de las 2 tablas
(0,1) → (0,1)	SI	CP se toma de una de las 2 tablas. NO FK
(0,1) → ((0|1),n)	SI	CP se toma de tabla con (0,1). NO FK
((0|1),n) → ((0|1),n)	SI	CP son las 2 CP de cada tabla. NO FK

Integridad Referencial

(IR) Es una propiedad deseable de las BD relacionales. Esta propiedad 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 entre tuplas. Escenarios posibles:

- **restringir la operación:** es decir, si se intenta borrar o modificar una tupla que tiene IR con otra, la operación se restringe y no se puede llevar a cabo. Por ejemplo, si quiero borrar un cliente que tiene facturas primero tengo que borrar las facturas de ese cliente y después al cliente.
- **realizar la operación es cascada:** en este caso, 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 si elimino un cliente significa eliminar en la misma operación todas las facturas de dicho cliente, en caso de modificar la CP se realizarán las modificaciones en todas las CF de las tuplas de la tabla Facturas..
- **establecer la clave foránea nula**: si se borra o modifica el valor del atributo que es CP, sobre la CF se establece valor nulo.
- **no hacer nada:** en este caso se le indica al SGBD que no es necesario controlar la IR

Conceptos de normalización (CAPÍTULO 13)

El proceso de normalización de una BD consiste en aplicar una serie de reglas a las tablas obtenidas a partir del diseño físico. Una BD debe normalizarse para evitar la redundancia de datos, ya que de haber aumenta el costo de mantenimiento. **Proceso deseado, pero no necesario.**
La normalización es un mecanismo que permite que un conjunto de tablas cumpla una serie de propiedades deseables, evitando:

- redundancia de datos
- anomalías de actualización
- pérdida de integridad de datos

Redundancia y anomalías de actualización

El objetivo principal del proceso de normalización es diseñar una BD que minimice la redundancia de información. El proceso de diseño genera redundancia de datos, y en determinadas situación esa redundancia es necesaria, a eso se lo denomina redundancia deseada (por ejemplo cuando en la cardinalidad muchos a muchos se generan tablas nuevas con las CP de las entidades en cuestión) y no afecta al proceso de normalización.
Por el contrario, existen otros casos de redundancia que generan anomalías cuando se opera sobre la BD. Se clasifican en tres grupos:

- anomalías de inserción
- anomalías de borrado
- anomalías de modificación

Dependencias funcionales

Es uno de los conceptos más importantes cuando se diseña el esquema físico de una BD.
Es una restricción entre atributos de una tabla de la BD. Se dice que un atributo Y depende funcionalmente de un atributo X (X→Y) cuando para un valor de X dado, siempre se encuentra el mismo valor de Y. En una DF el atributo X se denomina determinante y el atributo Y consecuente. Cualquier atributo depende funcionalmente de la CP y algo similar ocurre con la CC

Dependencia funcional parcial

Una DF X→Y se denomina parcial cuando, además, existe otra dependencia Z→Y, siendo Z un subconjunto de X. Esta definición de DF parcial contradice la definición de conjunto mínimo de DF, por lo que trae aparejado repetición de información y consecuentemente anomalías. Se puede advertir que al definir una DF con determinante múltiple, es posible generar situaciones de DF parciales. Si el determinante es simple, no es posible que exista una DF parcial.

Dependencia funcional transitiva

Una DF X→Y se denomina transitiva cuando existe un atributo Z tal que X→Z y Z→Y . Esta DF no cumple con la definición de conjunto mínimo, generando repetición de información y anomalías.

Dependencia funcional de Boyce-Codd

La Df de Boyce-Codd tiene en cuenta las CC. Una DF X→Y se denomina Dependencia funcional de Boyce-Codd (DFBC) cuando X no es una CP o CC, e Y es una CP o CC, o parte de ella.

Resumen

- Dependencia parcial
  - Parte_clave → no_clave
- Dependencia transitiva
  - No_clave → no_clave
- Dependencia Boyce Codd (explicada más adelante)
  - No_clave → parte_clave

Formas Normales

La normalización es una técnica que permite analizar el esquema físico de datos buscando mejor representación de cada una de las tablas, evitando repetición de información y las anomalías de actualización que puedan surgir. Para el proceso de normalización se parte de un esquema no normalizado y se aplican un conjunto de reglas que permiten convertirlo en un esquema normalizado. Los niveles de normalización propuestos inicialmente fueron tres: primera, segunda y tercera forma normal. La forma normal siguiente fue planteada por Boyce-Codd. Existen dos niveles más, la cuarta y quinta forma normal propuestas posteriormente.

- **primera forma normal:** la primera forma normal plantea que todos los atributos que conformen la tabla deben ser monovalentes, es decir, la cardinalidad de los atributos debe ser 0 o 1. La solución de llevar el modelo a 1FN consistió en quitar el atributo polivalente, creando una nueva entidad. Este proceso se realiza sobre el esquema lógico, por consecuencia, todas las tablas están en 1FN.
- **segunda forma normal:** involucra el tratamiento de DF parciales. Un modelo está en 2FN si está en 1FN y además no existe ninguna tabla del modelo una DF parcial.
  La manera de tratar una DF parcial es quitar de la tabla el atributo o los atributos que generan esta DF, y situarlos en una nueva tabla. Esto significa que la DF debe mantenerse en el modelo, pero en una ubicación que no genere una DF parcial. Se debe destacar que a pesar de la división de atributos que se realice no hay pérdida de información.
- **tercera forma normal:** implica el tratamiento de DF transitivas. Un modelo está en 3FN si está en 2FN y además, no existe ninguna tabla con DF transitiva. En este caso se trata de manera similar, se busca quitar los atributos que generan DF transitiva en la tabla y colocarlos en una nueva tabla.
- **forma normal Boyce-Codd:** fue planteada como una simplificación de la 3FN, pero generó una forma normal más restrictiva. Un modelo está en FNBC (BCNF) si está en 3FN y, además, no existe en ninguna tabla del modelo una DF de BC. Al igual que las demás formas, se soluciona creando una nueva tabla, reacomodando a aquellos atributos que generan una DF BC.
- **cuarta forma normal:** una dependencia multivaluada denota 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 de Y. La 4FN busca que las dependencias multivaluadas no presenten anomalías. Un modelo está en 4FN si está en BCNF y las DM que se puedan encontrar son triviales. Una DM X→ →Y es trivial si Y está multideterminado por X y no por un subconjunto de X. Dada la DM (X,Z)→ →Y, será trivial si Y está multideterminada por el par (X,Z). Sí Y está multideterminado solamente por X o Z, la dependencia no se considera trivial.

- **Quinta forma normal:** las formas normales posteriores a 4FN representan situaciones muy particulares.

*La normalización evita la redundancia de datos pero a su vez segmenta la información.

Procesamiento de Consultas

Lenguaje de procesamiento de datos (CAPÍTULO 14)

Los lenguajes de procesamiento de datos permiten operar con la información contenida en una BD.
Una vez definido el modelo de datos (una vez que el modelo físico impacta sobre un SGBD) las operaciones posibles sobre él son cuatro: **agregar, borrar, modificar o consultar información**. Para realizar estas operaciones se debe contar con un lenguaje que permita indicar el tipo de operación deseada. Los lenguajes se catalogan en dos grande grupos:

- **Procedurales**: cuando se describe una operación se debe indicar que se requiere y cómo se debe proceder para llegar al resultado deseado.
- **No procedurales**: cuando se describe una operación, se debe indicar solo que se necesita. EL SGBD será el encargado de encontrar la mejor forma para obtener el resultado.

A primera vista, los lenguajes no procedurales parecen más sencillos, pero se complejizan a medida que las consultas van siendo más complejas. Los lenguajes no procedurales permiten al analista generar la operación e incidir en cómo resolver las consultas. Acá se describen los tres lenguajes de manipulación de datos que representan la base del modelo relacional. Estos lenguajes son: álgebra relacional (procedural) y los cálculos de tuplas y de dominio (ambos no procedurales).

Álgebra relacional

Básicamente el álgebra relacional representa un conjunto de operaciones que toman tablas (relaciones) como operandos, y regresan otras tablas como resultado. Originalmente se definieron ocho operadores, pero solo 6 se consideran como los operadores básicos, ya que los dos restantes pueden expresarse como combinaciones de los operadores básicos.

Operadores básicos

Los operadores básicos son 6 y se clasifican en dos tipos:

- **Unarios**: operan sobre una tabla o relación. Estos son: **selección, proyección, renombre**
  - **Selección**: σP (tabla) permite filtrar de manera horizontal tuplas en una tabla (corte por fila) donde tabla es una relación definida en la BD y P es una condición lógica que deben cumplir las tuplas de cada tabla para ser presentadas como resultado.
  - **Proyección**: πL (tabla) permite filtrar de manera vertical tuplas de una tabla (corte por columna) donde tabla es una relación definida en la BD y L es una lista, separada por comas, de atributos definidos en la tabla
  - **Renombre**: ρ nombre_nuevo (tabla) es la operación que permite usar la misma tabla dos veces en la misma consulta.
- **Binarios**: operan sobre dos tablas o relaciones. Estos son: **producto cartesiano, unión, diferencia.**
  - **Producto cartesiano:** tabla1 × tabla2 es equivalente al producto cartesiano entre conjuntos. Se aplica a dos tablas o relaciones de la BD, y vincula cada tupla de una tabla con cada una de las tuplas de la otra tabla. Como el producto cartesiano junta todas las tuplas de la tabla1 con cada tupla de la tabla2, si por ejemplo yo quiero saber de qué localidad es un alumno, debe usarse una condición para asegurarme que la clave foránea “localidad” de la entidad alumno coincida con la clave primaria de la entidad localidad, así conseguiré tener la información específica de a que localidad pertenece el alumno:σalumnos.idlocalidad = localidades.idlocalidad (alumnos × localidades)
  - **Unión:** tabla1 U tabla2 es equivalente a la unión matemática de conjuntos. Se aplica a dos tablas de la BD, generando una nueva tabla cuyo contenido es el contenido de cada una de las tuplas de las tablas originales. Cuando se usa la unión en AR, debe tenerse en cuenta que los dos conjuntos a unir **deben ser unión-compatibles**, es decir, que hay que unir dos conjuntos que tengan la misma cantidad de atributos, y además, el i-ésimo atributo de la primera tabla y la segunda tabla deben tener el mismo dominio, caso contrario, la información no será útil.
  - **Diferencia:** tabla1 − tabla2 es equivalente a la diferencia matemática de conjuntos. Se aplica a dos tablas, generando una nueva cuyo contenido es el contenido de cada una de las tuplas que están en

la primera tabla y que no están en la segunda tabla. **Cuando se utiliza la diferencia en AR, hay que tener en cuenta que las dos tablas deben ser unión-compatibles**

Operadores adicionales

- **Producto natural:** tabla1 |×| tabla2 el producto natural es equivalente a una selección de un producto cartesiano. El producto cartesiano, si la primera tabla tiene 1.000 tuplas y la segunda tabla tiene 100 tuplas, crea una tabla intermedia de 100.000 tuplas y luego se hace la selección deja las tuplas deseadas, en cambio el producto natural reúne directamente las tuplas de la primera tabla que se relacionan con las de la segunda tabla, descartando las tuplas que no se relacionan. Esto conlleva una gran ventaja en cuanto a performance, dado que no se generan tuplas que luego se descartan, sino que sólo se generan las tuplas necesarias. **Debe haber un atributo en común, una de las tablas debe tener definida una CF (FK) de la otra.** Caso contrario, el producto natural y el cartesiano actúan de la misma manera.
- **Intersección:** tabla1 ∩ tabla2 es equivalente a la intersección matemática de conjuntos. Se aplica a dos tablas de la BD, generando una nueva tabla con las tuplas comunes a ambas tablas tener en cuenta acá también que las tablas en cuestión deben ser unión-compatibles, sino el resultado será una tabla vacía. Es una operación adicional porque se puede obtener el mismo resultado de dos maneras diferentes:
- **Asignación:** ← no aporta una funcionalidad extra al AR, sino que tiene como objetivo mejorar la legibilidad de las consultas. Con el operador de asignación se puede generar una subconsulta y volcar el resultado de la misma a una variable temporal de tabla. Ejemplo: Materias_Segundo_Año ← πmaterias.nombre (σmaterias.año_curso = 2 (materias))
- **División:** dadas dos tablas R1 y R2, R1%R2 tiene como resultado los valores de atributos de R1 que se relacionan con todas las tuplas de la tabla R2, si y sólo si, el esquema de R2 está incluido en el esquema de R1, es decir, R2 son atributos de R1.

Actualizaciones utilizando AR

- altas: permite agregar una nueva tupla a una tabla existente. Alumnos ← Alumnos U {“Delia”, “30777987”, 2, 1}
- bajas: permite quitar una tupla de una tabla. Alumnos ← Alumnos − σnombre = ‘Lopez’ (alumnos)
- modificaciones: δatributo = valor_nuevo (tabla) donde tabla es la tabla que se modifica, atributo es el atributo de la tabla a modificar y valor_nuevo se asigna como nuevo contenido del atributo. Previo a realizar la modificación, **se debe seleccionar la tupla sobre la que se desee hacer el cambio, si no se hiciera eso antes, se modificarían el atributo de todas las tuplas**.

Cálculo relacional de tuplas

Surge como un lenguaje de consulta de BD no procedural, es decir, que se especifica el resultado al que se desea arribar pero sin definir el proceso para obtenerlo. La estructura básica de CRT es la definición de una tupla y las propiedades que esta debe cumplir para ser presentada en el resultado final de la consulta.
Tiene el formato {t / P(t)} que indica que se van a presentar las tuplas t tal que para t el predicado P sea verdadero.
Variable tupla límite: significa que una variable está ligada a una tabla, es decir, limitada a los atributos de la tabla.
Variable tupla no limitada (libre): se considera libre cuando no está ligada a una tabla, es decir, se reciben los atributos que se van a presentar como resultado.
Las expresiones en CRT deben cumplir un requisito indispensable, que se denomina seguridad en las expresiones. Para que una expresión sea válida, debe ser segura.

Cálculo relacional de dominios

Representa el segundo lenguaje de consultas de BD no procedural. En general opera de manera similar al CRT. A diferencia de CRT que trabaja sobre cada tupla, una expresión CRD tiene este formato: {(atr1 , …, atrn) / P(atr1 , …, atrn)} que indica que se presentarán aquellos dominios (atr) tal que el predicado P sea verdadero.
Las expresiones en CRD deben cumplir con los mismos requisitos de seguridad que una expresión en CRT.

SQL. y QBE. (Capítulo 15)

SQL está ligado a las BD relacionales. Los lenguajes procedurales y no procedurales creados por Codd fueron la base para la creación de SQL (Lenguaje de consultas Estructurado). Es un lenguaje de consulta de BD que está compuesto por dos submódulos fundamentales:

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

Lenguaje de definición de datos

El modelo de datos relacional utiliza los conceptos de relación, tupla y atributo, en tanto, SQL los términos corresponden a tablas, fila y columna.
*tabla=relación; tupla=fila; atributo=columna
Para administrar un modelo físico de datos, SQL presenta tres cláusulas básicas:

- Create (crear)
- Drop (borrar)
- Alter (modificar)

Crear o borrar una BD

- Para generar una BD la sentencia SQL es: CREATE DATABASE nombre_BD;
- Para eliminar una BD completa la sentencia es: DROP DATABASE nombre_BD;

Operar contra Tablas de BD

- Sentencias para crear una tabla:

```
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));
```

- Sentencias par crear una tabla pero se le agrega la definición de una clave foránea:

```
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);
```

- Para eliminar una tabla del modelo la sentencia sería:

```
DROP TABLE nombre_tabla;
```

- Para modificar una tabla del modelo la sentencia sería:

```
ALTER TABLE nombre_tabla ( ... ) ;
```

Esta sentencia debe indicar, además, que tipo de modificación se desea realizar sobre la tabla. Así entre paréntesis se pueden agregar, modificar o borrar atributos, índices o restricciones de integridad.

```
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

Estructura básica

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

```
SELECT lista_de_atributos  -- = proyección
FROM lista_de_tablas -- = producto cartesiano
WHERE predicado -- = selección
```

donde lista_de_atributos indica los nombres de los atributos que serán presentados en el resultado. Estos atributos deben estar contenidos en las tablas indicadas en la consulta. Por otro lado, lista_de_tablas indica las tablas de la BD necesarias para resolver la consulta.
Por último, el predicado indica qué condición deben cumplir las tuplas de las tablas para estar en el resultado final de la consulta.
La analogía entre SQL y AR es importante. AR representa la base teórica de SQL, por lo tanto las consultas expresadas en ambos lenguajes es similar en aspectos semánticos. Al igual que AR, SQL siempre retorna como resultado una tabla temporal

```
SELECT atr1, atr2, atr3
FROM tabla1, tabla2
WHERE atr4 = 'valor'
```

Es equivalente a : πatr1, atr2, atr3 (σatr4 = ‘valor’ (tabla1 x tabla2))

En SQL el operador ***** indica que **todos** los atributos de las tablas definidas en el FROM serán presentados en el resultado de la consulta.
Cuando una tabla resultado tiene filas repetidas, se utiliza **la cláusula DISTINCT**, en la sección del SELECT,la cual **elimina tuplas repetidas**. Además existe el operador ALL, quien muestra todas las tuplas, incluso las repetidas. SQL en caso de no indicar explícitamente la cláusula DISTINCT, asume el operador ALL.

Cuando en la sección WHERE debemos utilizar más de una condición, podemos utilizar el operador AND/OR o podemos usar el operador BETWEEN

En los atributos definidos en un SELECT se pueden realizar cualquiera de las operaciones básicas definidas para su dominio (por ejemplo si en el select tenemos un atributo que hace referencia a un integer podemos realizar una multiplicación al valor de ese atributo)

El operador AS en el SELECT permite renombrar un atributo, al igual que ocurre en el FROM que se pueden renombrar las tablas (en el FROM no hace falta usar ningún operador).

Cuando trabajamos en SQL UNION las tablas resultados no colocan en el resultado tuplas duplicadas. En ese caso, deberíamos usar la cláusula UNION ALL. La cláusula definida para la diferencia en SQL es EXCEPT, en tanto, que la cláusula para la intersección es INTERSECT

Operadores de Strings

Cuando una cadena se compara con otra cadena mediante el operador “=” el resultado será verdadero si son iguales, y falso en caso contrario. El operador LIKE se conjuga con el carácter reservado %. Es decir, si buscamos por ejemplo el nombre de un alumno que empiece con P, la sección WHERE debería ser algo así:
WHERE alumno.nombre LIKE “P%”
Se compara el atributo nombre de la tabla alumnos contra el string que empieza con P y continúa con cualquier substring (ese es el comportamiento del carácter %, el cual considera válido a partir de la posición donde aparece, cualquier cadena de caracteres, incluso la cadena vacía).
Existe además otro carácter reservado “_”, quien sustituye sólo el carácter del lugar donde aparece.

En algunas situaciones es necesario presentar las tuplas de la tabla resultado ordenada por algún criterio. La cláusula ORDER BY permite ordenar las tuplas resultantes por el atributo indicado.
Si se desea obtener el resultado de mayor a menor hay que utilizar el operador DESC (ASC por defecto)

En ORDER BY es posible indicar más de un criterio de ordenación. El segundo criterio se aplica en caso de empate en el primero y así sucesivamente.
*los criterios de ordenación DEBEN ESTAR PRESENTES EN LA SECCIÓN SELECT

Consultas con funciones de agregación

Las funciones de agregación son funciones que operan sobre un conjunto de tuplas entrada y **producen un único valor de salida**. SQL presenta 5:

- AVG: retorna como resultado el promedio del atributo indicado para todas las tupla del conjunto.
- COUNT: retorna como resultado la cantidad de tuplas involucradas en el conjunto de entrada.
- MAX: retorna el valor más grande dentro del conjunto de tuplas para un atributo indicado.
- MIN: retorna el valor más chico dentro del conjunto de tuplas para un atributo indicado.
- SUM: retorna como resultado la suma del valor del atributo indicado para todas las tuplas del conjunto

Las funciones de agregación tienen una particularidad importante. Por definición estas funciones se aplican sobre un conjunto de tuplas, por ese motivo, no pueden estar definidas dentro de una cláusula WHERE, ya que hacerlo en esta sección significa aplicar una función aplicable a conjuntos de una sola tupla.
Una función de agregación siempre **retorna un resultado**. No siempre es posible retornar además de dicho resultado, el valor de otro atributo.

Funciones de agrupación

En muchos casos es necesario agrupar las tuplas de una consulta por algún criterio, aplicando una función de agregación sobre cada grupo. Para esto se utiliza la cláusula GROUP BY. En el ejemplo planteado en el libro, la cláusula GROUP BY genera n subconjuntos de tuplas, cada uno por un nombre de alumno diferente. Luego, dentro de cada subconjunto se obtiene el promedio del atributo resultado.Cuando se genera un agrupamiento por algún criterio (atributo) es posible proyectar en el SELECT el atributo por el cual se genera el grupo.
Hay consultas que requieren obtener resultados para grupos que satisfagan determinada condición. SQL provee para estos casos la cláusula HAVING la cual actúa como filtro de grupos. Dentro de la cláusula HAVING es posible determinar la condición que debe cumplir el grupo para ser tenido en cuenta.Tener en cuenta que dentro de la cláusula HAVING se pueden usar funciones de agregación.

Consulta de subconsultas

Una consulta con subconsulta, también llamadas como subconsultas anidadas, consiste en ubicar una consulta de SQL dentro de otra. las subconsultas se puede usar en varias ocasiones:

- comprobar la pertenencia o no a un conjunto
- analizar la cardinalidad de un conjunto
- analizar la existencia o no de elementos en un conjunto

SQL define el operador IN en la sección WHERE para comprobar si un elemento es parte o no de un conjunto. Su contrario es NOT IN.
Además del operador IN, se puede usar operadores de comparación de elementos simples contra conjuntos:

- =SOME: significa igual a alguno.
- >ALL: significa mayor que todos.
- <=SOME: significa menor o igual que alguno.

La cláusula EXIST se utiliza para comprobar si una subconsulta generó o no alguna tupla como respuesta. El resultado de la cláusula es verdadero si la subconsulta tiene al menos una tupla, caso contrario retorna falso. Su contrario es NOT EXIST.

Operaciones con valores nulos

Para saber si un atributo tiene o no un valor nulo debemos consultar en la sección WHERE por IS NULL o IS NOT NULL

Productos naturales

Por las mismas cuestiones de performance que AR planteó la alternativa de producto natural en lugar del producto cartesiano, SQL implementa lo mismo. Las sentencias disponibles son:

- INNER JOIN: se utiliza como un producto natural presentado en el capítulo de AR
- LEFT JOIN
- RIGHT JOIN

- FULL JOIN

Operaciones: Insertar, Borrar y Modificar. Sobre tablas

El 80% de las operaciones que se realizan sobre una BD son de consulta. Sin embargo las restantes 3 operaciones son necesarias:

- insertar: INSERT INTO

```
INSERT INTO alumnos (nombre, dni, idlocalidad, idcarrera ) VALUES ('Julio Cesar', '12344321', 3, 1);
```

  *se agregaron 4 de los 5 valores que tiene la tabla Alumnos porque idAlumno es un autoincremental, el SGBD se encarga automáticamente de asignarle un valor. Además cuando se asigna un valor a un atributo que cumple la función de clave foránea, en caso de que la integridad referencial sea estricta, se deben indicar valores válidos, es decir, valores que existan en este caso a la tabla Localidad y Carrera (los dos últimos atributos)

- borrar: DELETE FROM

```
DELETE FROM alumnos WHERE idlocalidad = (SELECT idlocalidad FROM localidades WHERE nombre = "La Plata")
```

*tener en cuenta nuevamente la integridad referencial. Esto es, solamente pueden ser borradas de la tabla aquellas tuplas que cumplan las condiciones de integridad referencial definidas.

- modificar: UPDATE

```
UPDATE carreras SET duración_años = 5 WHERE nombre = "Contador"
```

*tener en cuenta que si varias tuplas cumplen la condición del where, todas ellas serán modificadas.

Vistas

Tienen la estructura de una tabla y almacena temporalmente una consulta. Cada vez que la vista es utilizada, el valor de la consulta se recalcula.

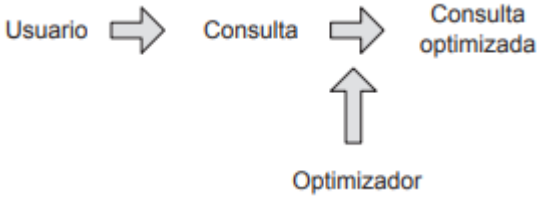
Introducción al QBE

SQL se presentó como la derivación de AR que dispone la mayoría de los SGBD. Dentro de los lenguajes no procedurales se encontraba el cálculo de tuplas y dominios. Este último es el origen teórico del denominado QBE. Es uno de los primeros lenguajes de consulta gráfico con un requerimiento mínimo de sintaxis para expresar una consulta en la BD. Se basa en expresar las necesidades del usuario a partir de plantillas, donde se seleccionan primero las tablas de la BD de donde se extrae la información para luego indicar que atributos se requiere presentar y qué condiciones deben presentar estos atributos.

Optimización de consultas (CAPÍTULO 16)

Análisis de procedimiento de consultas.

Los SGBD presentan en general un optimizador de consultas. Se denomina optimizador de consultas a un proceso del gestor de BD encargado de encontrar una consulta equivalente a la generada por el usuario, que sea de óptima en términos de performance para obtener el resultado deseado.
El proceso de optimización comienza con la consulta generada por el usuario; el optimizador de consulta la analiza y la compara contra el estado actual de la BD, y genera una consulta equivalente en cuanto a la respuesta que se obtendrá de ella, pero más eficiente en términos de procesamiento.



Seguridad e integridad de datos

Concepto de transacción. (CAPÍTULO 17)

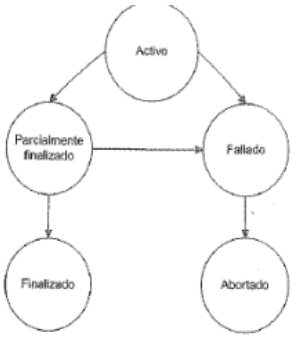
Una transacción es una unidad lógica de trabajo. Para una BD, una transacción actúa como una única instrucción que tendrá éxito si se ejecuta completamente, es decir, todas las instrucciones que la componen, o, en su defecto, nada de ella se verá reflejada en la BD.
Una transacción es un conjunto de instrucciones que actúa como unidad lógica de trabajo.
Para garantizar que una transacción mantenga la consistencia de una BD, es necesario que cumpla con cuatro propiedades básicas conocidas como **ACID**:

- **atomicidad**: garantiza que todas las instrucciones de una transacción se ejecuten sobre la BD, o ninguna de ellas se lleven a cabo.
- **consistencia**: la ejecución completa de una transacción lleva de un estado consistente a otro estado de consistencia. Para esto, la transacción debe estar escrita correctamente.
- **aislamiento**: cada transacción actúa como única en el sistema. Esto significa que la ejecución de una transacción T1, no debe afectar la ejecución simultánea de T2.
- **durabilidad**: una vez finalizada la transacción, los efectos producidos en la BD son permanentes.

Estado de las transacciones

Una transacción puede estar en alguno de los siguientes 5 estados:

- **activo**: estado desde que comienza ejecución hasta completar la última instrucción, o se produce un fallo.
- **parcialmente finalizado/parcialmente cometido:** se alcanza en el momento posterior a que la transacción ejecute su última instrucción.
- **Fallada**: luego de descubrir que no puede seguir la ejecución normal
- **abortado**: este estado garantiza que una transacción fallida no ha producido ningún cambio en la BD, manteniendo la integridad de la información.
- **finalizado/cometido**: la transacción finalizó su ejecución, y sus acciones fueron correctamente almacenadas en la memoria secundaria (porque cuando la última instrucción de la transacción se ejecuta todavía la transacción no termina, falta el paso de tirar la información de los buffers de memoria de la BD a los discos).



Suponiendo el caso de que una persona quiere realizar un pago pero ocurre un error en la transacción que finaliza con la transacción abortada ¿que hacer con dicha transacción? Las opciones son dos:

1. **reiniciar la transacción**: esta acción se puede llevar adelante cuando el error se produjo por el entorno y no por la transacción.
2. **cancelar definitivamente la transacción**: esta acción se lleva a cabo cuando se detecta un error interno en la lógica de la transacción. Este error solamente puede ser salvado si la transacción es redefinida.

Entornos de la transacción

Las transacciones pueden ser analizadas desde tres perspectivas básicas.
La primera consiste en analizar el comportamiento de las transacciones en entornos monousuario. Tienen por característica básica que admiten a un solo usuario operando con la BD y a este usuario solamente le permiten realizar una actividad por vez. De las cuatro propiedades ACID la propiedad de aislamiento se obtiene por definición, ya que si se puede ejecutar de a una transacción por vez, no hay forma que la T0 afecte la T1; la propiedad de consistencia es responsabilidad del programador, por lo tanto, para el análisis de la integridad de la BD, se asume que está bien escrita; la propiedad de durabilidad indica que una transacción cometida no puede ser retrotraída.
La segunda de las perspectivas es bajo un entorno concurrente. Estos entornos admiten varios usuarios simultáneos, cada uno de los cuales pueden generar múltiples transacciones. En este caso, el énfasis quedará

sobre la propiedad del aislamiento. El comportamiento de las otras tres propiedades es similar al del entorno monousuario.
El último escenario para las transacciones se presenta en entornos distribuidos. Estos entornos son los más comunes en la práctica actual. Una transacción generada por un usuario puede requerir modificar una BD residente en varios servidores. Los tipos de fallos que se pueden producir son mayores. Asegurar la integridad de una BD distribuida físicamente en varias computadoras es mucho más complejo.

Fallos

Si bien los fallos que se pueden producir son muy variados y algunos de ocurrencia es muy poco probable, son situaciones que deben ser consideradas a fin de asegurar la integridad de los datos.
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 (ACID).
Los fallos pueden ser clasificados en dos tipos:

- fallos que no producen pérdida de información: si una transacción que solamente está consultando a la BD falla y debe abortarse, no produce pérdida de información. Una transacción de consulta no modifica.
- fallos que producen pérdida de información: involucran transacciones que incluyen 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.

Método de recuperación de integridad de una base de datos

El problema que se genera ante un fallo es la posible pérdida de consistencia en la BD.
El primer concepto a tener en cuenta está relacionado con el tipo de almacenamiento que usa una transacción: almacenamiento volátil (RAM) y almacenamiento no volátil (disco rígido). Si el fallo se produce antes de escribir en disco, la BD queda inconsistente.
Es importante remarcar que cualquier tipo de fallo afectará a la información contenida en la RAM, debido a que esta memoria es volátil. Sin embargo, 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
Ante un fallo de transacción ¿Cuáles son las acciones que se deberían llevar a cabo para asegurar la integridad de la BD? Reejecutar o no una transacción fallida no asegura la consistencia de la BD. El motivo de la inconsistencia es que la transacción efectúa cambios sin la seguridad de finalizar correctamente. Para evitar los problemas que puedan surgir, es necesario retrotraer al estado de la BD previo al comienzo de la transacción. Estas acciones preventivas tienen que ver con dejar constancia de las actividades llevadas por la transacción, que permitirán deshacer los cambios producidos.

Existen dos métodos de recuperación: el de bitácora (log) y el doble paginación. Cualquiera de estas métodos necesita de dos algoritmos básicos:

1. 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
2. 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

El método bitácora o registro histórico plantea que todas las acciones llevadas a cabo sobre la BD deben quedar registradas en un archivo histórico de movimientos. Por lo tanto, una bitácora es un repositorio donde se registran acciones.
El método 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 misma. De esta forma, en caso de producirse un error, se tiene constancia de todos los movimientos efectuados. Con este registro de información es posible garantizar que el estado de consistencia de la BD es alcanzable en todo momento, aun ante un fallo.
La estructura de un archivo bitácora es simple. Cada transacción debe indicar su principio, su fin y cada una de las acciones llevadas a cabo sobre la BD que modifiquen su contenido. El gestor de SGBD es el responsable de controlar la ejecución de las transacciones, de administrar el archivo bitácora y de utilizar los algoritmos de recuperación cuando el SGBD se recupera de una situación de fallo. Para poder cumplir su cometido, el gestor le asigna a cada transacción un número correlativo único. Formato:

Podría ocurrir que la transacción fallara, en ese caso, se procederá a abortarla. Una vez abortada la transacción, en bitácora debe agregarse una entrada <T0 aborta>.
Según la naturaleza del fallo, puede pasar que en el medio de la transacción ocurre un problema con el suministro de energía, es decir, se registra <Ti comienza> pero no hay registro de que haya finalizado o abortado. En esa situación no es posible agregar nada a la bitácora. Para la bitácora una transacción que tiene comienzo pero no tiene fin debe abortarse.
Ejemplo de transacción en el pago de un servicio:

Solamente se registran las instrucciones que modifican la BD, por lo tanto solo se registran las operaciones de escritura.
Durante el procesamiento normal de las transacciones, se genera el registro en bitácora y luego se actualiza la BD.

Para todas las transacciones que alcanzan el estado de finalización, el trabajo registrado en bitácora es innecesario. Solamente cuando se produce un error, se deben activar los algoritmos que subsanan el error, usando el registro de bitácora.
La bitácora presenta dos alternativas para implementar el método de recuperación:

- modificación diferida de 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. Así, mientras la transacción esté en estado activo se demora cualquier escritura física en la BD. Presenta una gran ventaja: si la transacción falla, no va a haber alteraciones en la BD, por lo tanto se mantiene la integridad.
- modificación inmediata de la BD: esta solución plantea que hacer todas las escrituras una vez que finaliza la transacción implica una sobrecarga de trabajo.
  Entonces, se plantea que los cambios impactan en la BD a medida que se producen en la transacción que se está ejecutando, siempre bajo la consigna que indica un cambio se registra primero en bitácora para luego grabarse en la BD.

El método de bitácora cuenta con puntos de verificación que se agregan cuando se tiene la seguridad de que la BD está en estado consistente. La finalidad es indicar que ante un fallo, solo debe revisarse la bitácora desde el punto de verificación en adelante, dado que las transacciones anteriores finalizaron correctamente sobre la BD.

Doble Paginación

El método alternativo al de bitácora se denomina doble paginación o paginación en la sombra. Este método plantea dividir la BD en páginas que contienen determinados datos.
Se generan dos tablas en disco y cada una de las tablas direcciona a las páginas generadas. Ante una transacción que realiza una operación de escritura sobre la BD, la secuencia de pasos es la siguiente:

1. se obtiene la página sobre la cual debe realizarse la escritura (si la página está en memoria principal este paso se omite)
2. se modifica el dato en memoria principal y se graba en disco, en una nueva página, la página actual que referencie al nuevo nodo (página)
3. si la operación finaliza correctamente, se modifica la referencia de la página sombra para que apunte al nuevo nodo
4. se libera el nodo viejo

En caso de producirse un fallo, luego de la recuperación, se desecha la página actual y se toma como válida la página a la sombra.
Este método presenta algunas ventajas como por ejemplo eliminar la sobrecarga producidas por las escrituras al registro bitácora, y la recuperación ante un error es mucho más rápida. Sin embargo, presenta algunas desventajas, primeramente exige dividir la BD en páginas, en segundo lugar, la sobrecarga del método aparece en entornos concurrentes.

Transacciones en entornos concurrentes. (CAPÍTULO 18)

Ejecución concurrente de transacciones

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.
Este problema se genera debido a que un entorno concurrente debe asegurar, además, la propiedad de aislamiento de una transacción.

Un entorno concurrente puede generar situaciones de pérdida de datos, aún sin producirse fallos en el procesamiento de las transacciones.

Concepto de transacciones serializables

Planificación

En un entorno que admite varias transacciones ejecutándose en simultáneo, es necesario establecer un criterio de ejecución para que la concurrencia no afecte la integridad de la BD. A pesar de que el resultado final dependa del orden en que se ejecuten las transacciones, la integridad de la BD se respeta y esa es la propiedad deseada.
En un entorno concurrente, el orden de ejecución en serie de transacciones se denomina planificación (es decir, si tengo dos transacciones voy a tener dos planificaciones en serie, si tengo tres voy a tener tres planificaciones en serie, etc). En general, en un entorno concurrente con N transacciones en ejecución puede generar N! (N factorial) planificaciones en series diferentes.
Pero hay una desventaja con esta solución: no se aprovechan las ventajas de un entorno concurrente y el procesamiento secuencial de n transacciones demora mucho más tiempo.
Si tengo dos transacciones que trabajan sobre datos diferentes, debería poder ejecutarlas de manera simultánea, ya que se cumple la propiedad de aislamiento, porque actúan sobre diferentes datos.

Planificación serializable

Hasta el momento tenemos las siguientes conclusiones:
- se debe mantener la integridad de la BD
- la inconsistencia temporal puede causar inconsistencia en planificaciones concurrentes
- una planificación concurrente, para que sea válida, debe equivaler a una planificación en serie
- solo las instrucciones READ y WRITE son importantes y deben considerarse para cualquier análisis.
- dos instrucciones son conflictivas si operan sobre el mismo dato, y al menos una de ellas es una operación de escritura
- dos planificaciones se denominan equivalentes en cuanto a conflicto, cuando P2 se logra a partir del intercambio de instrucciones no conflictivas con P1
- una planificación 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, es decir, P1 es una planificación en serie, entonces su ejecución garantiza aislamiento, si P1 y P2 son equivalentes en cuanto a conflictos significa que obtienen un resultado similar, por lo tanto, P2 mantiene la consistencia de la BD. P2 será entonces una planificación concurrente que garantiza la integridad de la BD.

Pruebas de seriabilidad

Existen métodos para demostrar la seriabilidad de planificaciones concurrentes. Estos algoritmos están diseñados para detectar conflictos en la ejecución de dos transacciones.
Pero no se entra en profundidad con estos algoritmos (Ejemplo grafo)

Implementación de aislamiento. Control de concurrencia

Existen dos variantes para tratar de implementar el aislamiento (aunque es muy poco probable que dos transacciones quieran operar sobre el mismo dato en el mismo instante de tiempo):

- 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 (funcionamiento similar al que realiza el SO). Cada transacción debe bloquear la información que necesita para poder llevar adelante su ejecución, utilizarla y luego liberarla, para que otra transacción que necesite el dato pueda operar.
  Existen dos tipos de bloqueos sobre un elemento:
  - bloqueo compartido: debe ser realizado por una transacción para leer un dato que no modificará. Mientras se usa este bloqueo, se impide que otra transacción pueda escribir el mismo dato. Puede coexistir con otro bloqueo compartido sobre el mismo dato.
  - bloqueo exclusivo: se usa cuando una transacción necesita escribir un dato. Dicho bloqueo asegura que otra transacción no pueda usar ese dato (ya sea lectura o escritura). Es único por transacción.

  Cada transacción debe solicitar el bloqueo que necesite, si obtiene el dato lo usa, sino tiene dos posibilidades: esperar por el dato o fallar y abortar, para luego comenzar como una transacción diferente.
  Los bloqueos no siempre garantizan el aislamiento ya que si se programa mal la transacción (hacer los bloqueos al comienzo), puede ocurrir deadlock, es decir, dos transacciones toman bloquean los datos que

necesitan y ambos solicitan el elemento que tiene el otro. Como ninguno va a dejar el recurso que obtuvo, ambos impiden que su ejecución continúe.
Para asegurar la integridad de la BD con los bloqueos se deben establecer reglas.
Vamos a seguir la de la alternativa denominada dos fases. Para que una transacción sea correcta, el orden de pedidos de bloqueo debe ser:
    - fase crecimiento: se solicitan bloqueos similares o se crece de compartido a exclusivo
    - fase decrecimiento: exclusivo, compartido, librar datos
- Protocolo basado en hora de entrada: es una variante al 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 en que inicia su ejecución. Suponga que T1 y T2 modifican la BD. Si T1 comienza antes entonces HDE(T1) < HDE(T2), si T1 fuera posterior sería HDE(T1) > HDE(T2), en ningún caso puede ocurrir que HDE(T1) = HDE(T2).

El método continúa asignando la última hora de lectura (HL) y la última hora de escritura (HE).

Funcionamiento

Cualquier transacción que no pueda seguir su ejecución falla y aborta
*LOS CONFLICTOS SON EXACTAMENTE IGUAL A LOS DE CONCURRENTE, es decir, que si p1 trabaja con la variable x, p2 toma el procesador y comienza a modificar la variable x y luego p1 sigue su ejecución con la variable x, está generando una inconsistencia en la BD. La solución son los protocolos de aislamiento.

Granularidad

El concepto 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.

En general los SGBD permiten diversos niveles de bloqueo de la BD
- bloqueo completo: normalmente reservada para el diseñador de la BD para tareas de mantenimiento
- bloqueo a nivel registro: usada por transacciones en entornos concurrentes. Los bloqueos compartidos y exclusivos de datos apuntan a bloquear el uso o escritura de registros (tuplas) de la BD.
Algunos SGBD permiten generar un nivel de granularidad menor, en esos casos el bloqueo se puede realizar a nivel de campos o atributos de un registro.

Bitácora en entornos concurrentes

El protocolo de bitácora o registro histórico se aplica casi sin cambios en entornos concurrentes. La finalidad del protocolo es garantizar la atomicidad de la transacción ante posibles fallos originados en su ejecución. En el caso de entornos concurrentes se agrega una nueva situación de fallo: una transacción que no puede continuar su ejecución por problemas de bloqueos o acceso a la BD. En ese caso, como cualquier otra situación de fallo, la transacción debe abortar, dejando a la BD en el estado de consistencia anterior al comienzo de su ejecución.

Puntos de verificación de registro histórico

El único cambio que requiere el método de registro histórico con respecto de entornos monousuarios está vinculado con los puntos de verificación. En un entorno monousuario un punto de verificación se agrega periódicamente luego de un <Ti finaliza> y antes de <Tj comienza>. Un esquema concurrente no puede garantizar la existencia de un momento temporal, donde ninguna transacción se encuentre activa, por ese motivo la sentencia agrega un parámetro L. Este contiene la lista de transacciones que, en el momento de colocar el punto de verificación, se encuentran activas:

el algoritmo funciona igual que lo mencionado para entornos monousuarios, a excepción de las transacciones que están en la lista del último punto de verificación (L).

Seguridad e integridad de datos (CAPÍTULO 19)

Concepto de seguridad

Proveer seguridad a una BD consiste en protegerla de intentos de acceso malintencionados. Tiene que ver con proteger la BD, y por consiguiente, la información almacenada en ella.

Por otro lado, la integridad de la BD está ligada con la protección de los datos ante pérdidas accidentales de consistencia.
La seguridad sobre una BD se puede implantar en varios niveles. El primer nivel de seguridad sobre una BD se denomina nivel físico: para que una BD esté físicamente segura se debe resguardar el servidor que la contiene de cuestiones que tienen que ver con robo y esas cosas. Las soluciones más viables para lograr mejoras de seguridad a nivel físico son:

- replicar el hardware de los servidores, es decir, tener varios servidores de datos que funcionan como espejos entre sí. En caso de producirse algún fallo en alguno de ellos, otro toma el control de manera transparente para los usuarios.
- replicar dispositivos de almacenamiento, ya que los servidores de datos están preparados para tener varios discos rígidos. Estos discos nuevamente pueden actuar como espejos entre sí
- dotar de alarmas, personal extra de seguridad, etc que permitan detectar el acceso de personas no autorizadas.

Otro nivel de seguridad es el denominado nivel humano: los usuarios que tienen acceso a la BD deben registrar su conexión a través de una clave de usuario. Este punto representa el eslabón más débil de la seguridad en el acceso a una BD. A nivel de seguridad de SO es posible controlar la cantidad de veces que un usuario se conecta al sistema ingresando palabras claves incorrectas.
Por último, se definen dos niveles de seguridad relacionados con el SGBD y el administrador de BD.
El nivel de seguridad del SGBD resume los niveles anteriores agregando aspectos propios.
El nivel de seguridad del administrador de la BD está ligado en forma directa con las políticas que este último implementa respecto del control de accesos y con la política de actualización de datos sobre cada BD.
El concepto de auditoría de una BD está ligado al registro de todas las operaciones realizadas por cada usuario sobre la BD. Las auditorías son importantes en aquellas BD con información actualizada por múltiples transacciones y usuarios, donde se registra el responsable de cada acción. Ejemplo: BD de un banco

Autorizaciones y vistas

El concepto de autorización tiene que ver con generar cuentas de usuarios que permitan el acceso selectivo a cada tabla de la BD. Además, es posible controlar el tipo de operaciones que cada usuario puede llevar a cabo. Las autorizaciones pueden realizarse a dos niveles:

- **primer nivel**: consiste en permitir interactuar contra el modelo de datos, de esta forma el usuario podrá actualizar tanto el esquema como los índices
- **segundo nivel**: tiene que ver con el acceso a los datos. Se establecen privilegios de los usuarios para actualizar la información contenida en la BD. Los privilegios se pueden establecer a nivel de consulta, inserción, borrado y modificación de la BD.

Encriptado de información

La criptografía está vinculada con el cifrado y el descifrado de información mediante técnicas o algoritmos especiales. Cuando ese concepto se aplica a una BD, la idea es permitir que una BD solo sea accesible por los usuarios o personas autorizados.
La finalidad perseguida por la criptografía es garantizar la confidencialidad en la información y asegurar que los datos no sean corrompidos. Hay dos tipos de criptografía:

- **simétrica**: utiliza la misma clave para cifrar y descifrar la información
- **asimétrica**: utiliza dos claves diferentes para el proceso de conversión de la información.