Resumen teórico

Page • 1 enlace entrante • Tag

💡 Realizado en base a las filminas, el libro de la materia, y búsquedas de Google

Conceptos básicos

1. Un **dato** representa hechos conocidos que pueden registrarse y que tienen un resultado implícito
2. Una **Base de Datos** (BD) es una colección de *datos* relacionados, con un propósito específico vinculado a la resolución de un problema en el mundo real
3. Un **Sistema de Información** (SI) es un conjunto de agentes, códigos y procesos que interactúan coordinadamente entre sí con un fin común
4. Un componente esencial de cualquier *SI* es la *BD*
5. Un **archivo** es una estructura de *datos* que recopila una colección de elementos del mismo tipo
6. Para que una *BD* persista, se tiene que mantener, dentro de dispositivos de almacenamiento, como *archivos*
7. Además, para definir, construir y manipular una *BD*, se utilizan sistemas de software de propósito general denominados **Sistema de Gestión de Bases de Datos** (SGBD / DBMS)
8. Un *SGBD* tiene dos componentes fundamentales: un **Lenguaje de Definición de Datos** (DDL) y un **Lenguaje de Manipulación de Datos** (DML)
9. El *DDL* se encarga de especificar el esquema de la *BD*
10. El *DML* se encarga de operaciones tales como agregar, modificar o quitar información en una *BD*
11. Los actores de un *SGBD* son los siguientes: los **Administradores de la Base de Datos** (ADB / DBA), los **Diseñadores de la Base de Datos**, los **Analistas de Sistemas**, los **Programadores**, y los **Usuarios**
    - *ADB* ⟶ Autoriza accesos, coordina y vigila la utilización de recursos de hardware y software, responsable ante problemas de violación de seguridad o respuesta lenta del sistema

- *Diseñador de BD* → Definen la estructura de la *BD* de acuerdo al problema del mundo real que esté representando
- *Analista de Sistema* → Determinan los requerimientos de los *usuarios* finales, generando la información necesaria para el diseñador
- *Programador* → Implementan las especificaciones de los *analistas* utilizando la *BD* generada por el *diseñador*

Objetivos de un SGBD

- Un *SGBD* debe evitar la redundancia e inconsistencia de *datos*
- Un *SGBD* debe permitir el acceso a los *datos* en todo momento
- Un *SGBD* debe evitar anomalías en el acceso concurrente
- Un *SGBD* debe restringir los accesos no autorizados
- Un *SGBD* debe funcionar como suministro de almacenamiento persistente de *datos*
- Un *SGBD* debe mantener la integridad en los *datos*
- Un *SGBD* debe poder hacer backups

Modelado de Datos

1. Con **dominio** (del problema) nos referimos al contexto o área específica de interés que se abarca. Forma parte de la especificación de requerimientos del problema
2. Con **objeto** nos referimos a *cualquier cosa perceptible o concepto comprensible de la vida real (*un ovejero alemán*)*
3. Un **conjunto de objetos** está relacionado por algo que compartan estos *objetos* en común (perro → ovejeros, beagles...)
4. La **abstracción** es un proceso que nos permite aislar algunas características de un *conjunto de objetos* (los perros tienen edad, raza, tamaño...)
5. Hay tres tipos de *abstraccion*: **clasificación**, **agregación**, y **generalización**
    - *Clasificación* → Sirve para generar una **clase** → *abstracción* de un *conjunto de objetos*
    - *Agregación* → Define una nueva *clase* a partir de un conjunto de otras *clases* que representan sus partes componentes (ciudad → persona, perro, calle...)
    - *Generalización* → Define una nueva *clase* que extrae elementos en común de dos o más *clases* (animal → perro, gato...)
6. Un **modelo de datos** es una serie de conceptos que puede utilizarse para describir un conjunto de *datos* y las operaciones para administrarlos. Estos se

construyen utilizando mecanismos de *abstracción*, y se describen mediante representaciones gráficas que tienen una sintaxis y una semántica asociadas. Es un medio para describir la realidad

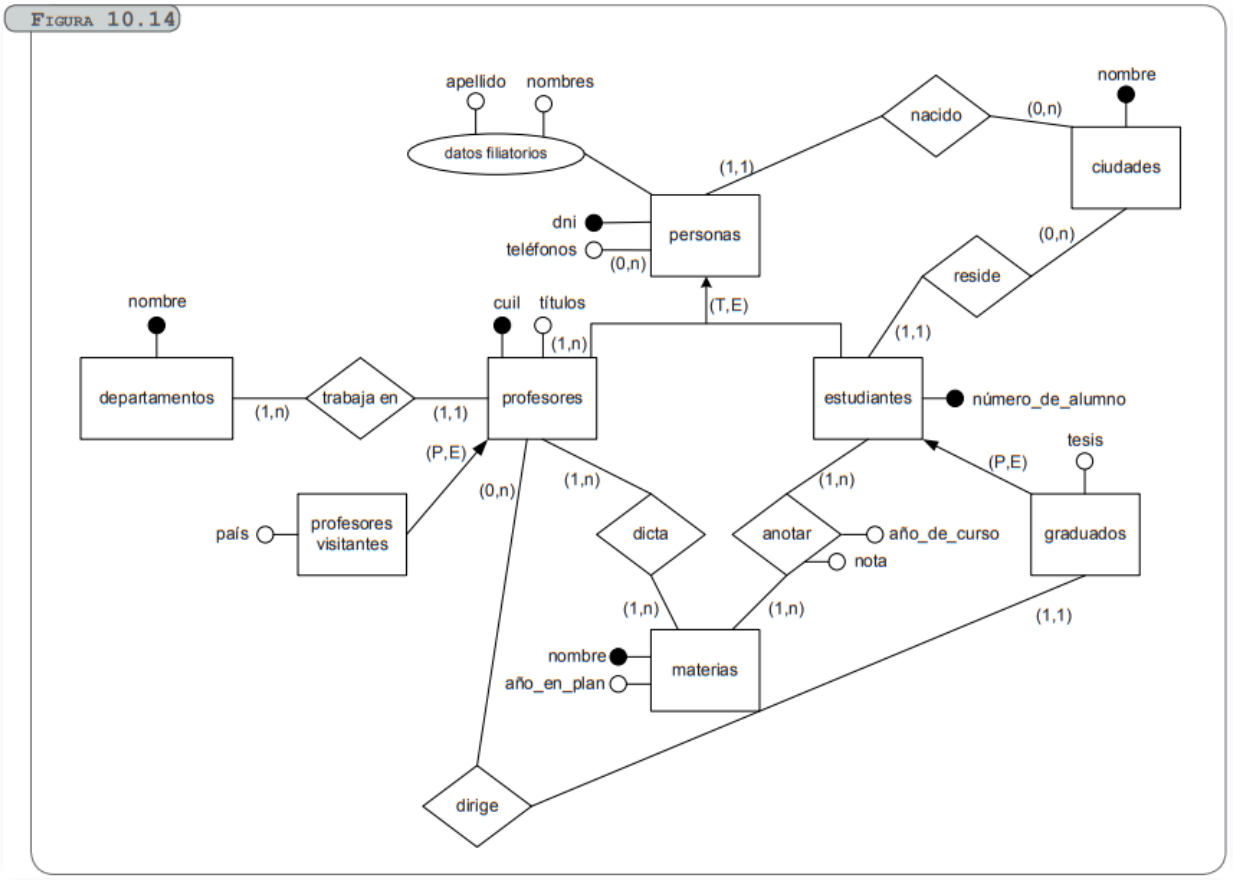
7. La construcción de un *modelo de datos*, propuesta por el libro, tiene tres etapas: el **modelado conceptual**, el **modelo lógico,** y el **modelo físico**
    - *Modelado conceptual* → El *modelo* se desarrolla independientemente de su implementación final (relacional, de red, jerárquico u OO) y del tipo de *SGBD* a utilizar. El modelo conceptual debe tener las siguientes características: **expresividad**, **formalidad**, **minimalidad**, y **simplicidad**.
    - *Modelo lógico* → El *analista* debe determinar el tipo de *SGBD*, debido a que las decisiones que debe tomar dependen de esa elección
    - *Modelo físico* → Es necesario tomar decisiones específicas. Estas últimas tienen que ver con el producto de mercado a utilizar, es decir, el *SGBD* específico
8. El **modelo Entidad Relación** (ER) es una técnica de *modelado de datos* que se basa en la concepción del mundo real como un *conjunto de objetos* llamados **entidades** y las **relaciones** existentes entre dichas *entidades*
9. El **modelo relacional** representa a una *BD* como una colección de *archivos* denominados tablas, las cuales se conforman por registros. Se lo puede obtener a partir de transformaciones del *modelo ER*

Modelo ER conceptual

> 💡 *El principal objetivo del diseño conceptual consiste en captar y representar, de la forma más clara posible, las necesidades del usuario definidas en el documento de especificación de requerimientos de un problema*

1. Cada *entidad* debe ser distinguible del resto de *entidades* → posee identidad
2. Cada *entidad* está conformada por un conjunto de propiedades básicas que la caracterizan, denominados **atributos**
3. Si un *atributo* no debe ser incluido explícitamente en el modelo, decimos que es **opcional**. Caso contrario, es **obligatorio**: (0, x); (1, x)
4. Si un *atributo* solo puede tener un valor, decimos que es **monovalente**. Caso contrario, es **polivalente**: (x, 1); (x, N)
5. Un **atributo derivado** es un *atributo* que su información, si este fuera eliminado, seguiría siendo posible obtenerla en el modelo
6. De la combinación de varios *atributos* (*simples*) obtenemos un **atributo compuesto**, que también puede ser polivalente y opcional

7. Una *relación* establece un nexo entre *entidades*
8. Una **relación recursiva** es aquella *relación* que une dos *entidades* particulares del mismo conjunto
9. Un **ciclo de relaciones** ocurre 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
10. La **cardinalidad** define el grado de *relación* existente en una *agregación*
    - **Cardinalidad mínima** → Nivel mínimo de correspondencia: (0, x); (1, x)
    - **Cardinalidad máxima** → Nivel máximo de correspondencia: (x, 1); (x, N)
11. Las *entidades* pueden compartir características de otras *entidades* → heredan
12. Si una *entidad* hereda de otra, la que hereda decimos que es una **entidad hija** (subentidad o especialización), y la otra **entidad padre** (superentidad o generalización)
13. La **cobertura** es el grado de relación entre *entidades padres* y *entidades hijas*
    - Decimos que la *cobertura* es **total** cuando, para el *dominio* de la *entidad padre*, cada elemento suyo está contenido en alguno de sus *hijos*. Caso contrario, la *cobertura* es **parcial**: (T, x); (P, x)
    - Decimos que la *cobertura* es **exclusiva** cuando un elemento del *padre* solo puede estar en un *hijo*. Caso contrario, la *cobertura* es **superpuesta**: (x, E); (x, S)
14. Si de una *generalización* se desprende solo una *especialización*, decimos que es un **subconjunto** y su cobertura es (P, E)
15. Un **identificador** es un *atributo* o conjunto de *atributos* que permite distinguir univocamente a una *entidad*
16. Si el *identificador* está compuesto solo por un *atributo*, decimos que es **simple**. Caso contrario, es **compuesto**
17. Si los *atributos* que componen al *identificador* se encuentran todos dentro de la *entidad* identificada, decimos que es **interno**. Caso contrario, es **externo**
18. Lo que se obtiene aplicando la técnica y todo lo anterior es un **esquema/modelo conceptual**



Conceptos a revisar en un modelo ER conceptual

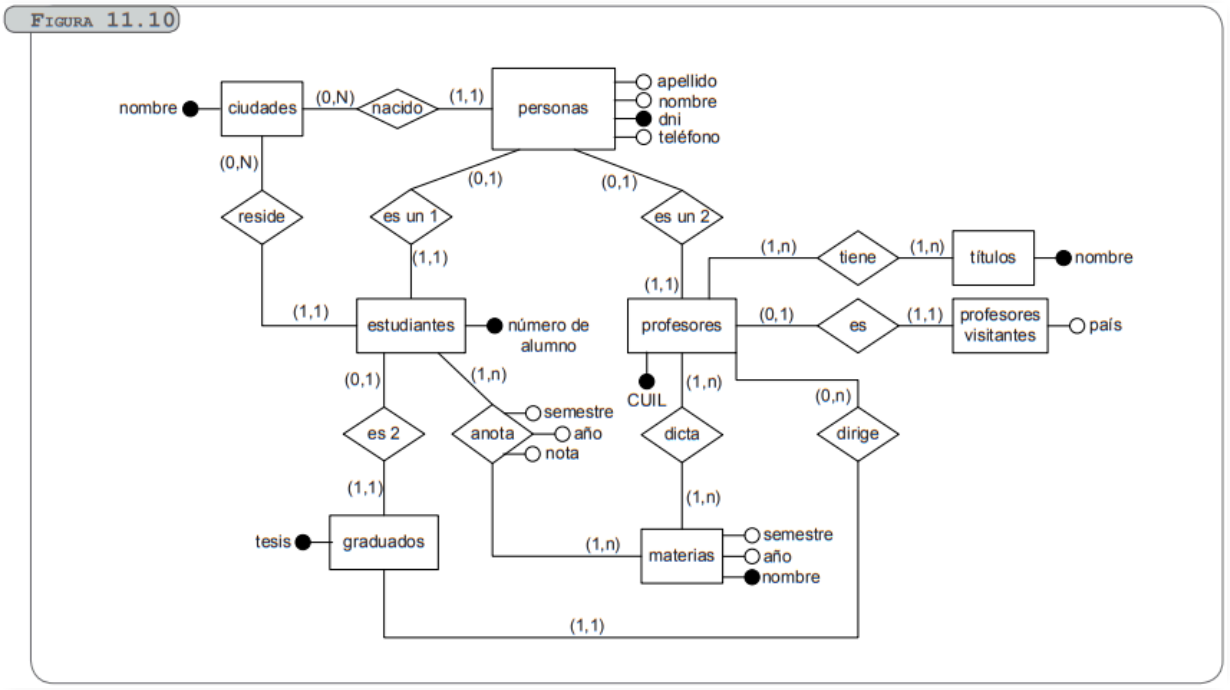
- **Completitud** —> Un *modelo* está completo cuando todas las características del problema están contempladas en él
- **Corrección** —> Un *modelo* es correcto si cada elemento en su construcción fue utilizado con propiedad (no faltan cardinalidades, coberturas, identificadores, ...)
- **Minimalidad** —> Un *modelo* es mínimo cuando cada concepto se representa una sola vez en el modelo (fijarse *ciclos de relaciones*, *atributos derivados*, ...)
- **Expresividad** —> El *modelo conceptual* resulta expresivo si a partir de su observación es posible darse cuenta de todos los detalles que lo involucran
- **Autoexplicativo** —> Un *modelo* se expresa a sí mismo si puede representarse utilizando los elementos definidos, sin necesidad de utilizar aclaraciones en lenguaje natural para expresar características
- **Extensibilidad** —> El *modelo conceptual* resulta extensible si es fácilmente modificable para incorporar nuevos conceptos en él, resultantes de cambios en los requerimientos del problema
- **Legibilidad** —> Un *modelo* es legible si la representación gráfica es adecuada

---

Modelo ER lógico

💡 El propósito de la generación de un modelo ER lógico es convertir el *esquema conceptual* en un *modelo* más cercano a la representación entendible por el *SGBD*

1. Para obtener un **esquema lógico** hay que aplicar una serie de reglas sobre el *esquema conceptual* obtenido, que van a depender del tipo de *SGBD* que se quiera utilizar (relacional, OO, ...), pero de manera que retenga la misma información
2. Si la conversión de un elemento tiene varias soluciones, se deberá elegir aquella que "permita alcanzar los estándares de rendimientos definidos para el problema"
3. La pauta sobre los *atributos derivados* y los *ciclos de relaciones* es poner en una balanza la conveniencia y el tiempo de procesamiento, y ver cual se prefiere priorizar
4. Si se quieren quitar los *atributos polivantes* (ya que ningún *SGBD* relacional permite que un *atributo* contenga valores multiples determinados dinámicamente), conviene generar una nueva *entidad* "*posee/tiene*" con una *relación* muchos a muchos, entre la *entidad* que tenía el *atributo compuesto* y la nueva *entidad*
5. Los *atributos compuestos* pueden convertirse en un único *atributo*, concatenación de todos sus atributos simples; se pueden dejar los *atributos simples* en la *entidad*; o se podría volver al *atributo compuesto* una *entidad* aparte
6. Como las jerarquías no existen en el *modelo relacional*, se puede:
    - Eliminar las *entidades hijas* y dejar la *entidad padre* con los *atributos* de estas como opcionales
    - Eliminar la *entidad padre* dejando las *entidades hijas* con los atributos del *padre* ⟶ no aplicable para los subconjuntos
    - Dejar todas las *entidades*, haciendo una *relación* es_un del *padre* con los *hijos*



Modelo ER físico

1. El *modelo físico* representa la *BD* como una colección de **tablas**, cada una conformada por registros denominados **tuplas**, y donde cada *atributo* tiene un *dominio*
2. Para obtener un **esquema físico** hay que aplicar una serie de reglas sobre el *esquema lógico*
3. Por cada *entidad* debe haber una **clave primaria** (CP), que es el *identificador* de esa *entidad*, o si hay varios *identificadores*, aquel de menor tamaño físico
4. Si un *identificador* de una *entidad* no es *clave primaria*, entonces es **clave candidata** (CC)
5. Una *clave candidata* puede transformarse en *clave primaria*
6. Decimos que un *atributo* de una *tabla* es **clave foránea** (CF/FK) cuando en otra *tabla* ese *atributo (*o grupo de *atributos)* es *clave primaria*
7. La **integridad referencial** (IR) es una propiedad deseable de la *BD* que asegura que un valor que aparece para un *atributo* en una *tabla* aparezca además en otra *tabla* para el mismo *atributo*
8. Cada *SGBD* tiene escenarios de definición de *IR* diferentes: se puede elegir restringir la operación de borrar o modificar en cualquiera de las dos *tablas*, se puede elegir realizar la operación "en cascada", establecer la clave foránea en nulo, o no hacer nada
9. Se debe realizar la conversión de las *entidades* y algunas *relaciones* a *tablas*:
    - Si entre dos *entidades* *A* y B, hay una *relación* tal que, del lado de A, la *cardinalidad* es *monovalente obligatoria*, entonces no hace falta volver

una *tabla* la *relación*, y se puede simplemente dejar la *CP* de B como *CF* en la *tabla* de A

- En cualquier otro caso con dos entidades A y B, sea porque la *cardinalidad* no es *obligatorio* desde ninguno de los dos lados, o porque ambos tienen *cardinalidad* muchos a muchos, o una mezcla de las dos, se deberá crear una *tabla* para la *relación*
- Si una *relación* es *recursiva*, al armar la *tabla* de la *relación* se toma la única *CP* disponible y se la replica

---

Normalización del Modelo

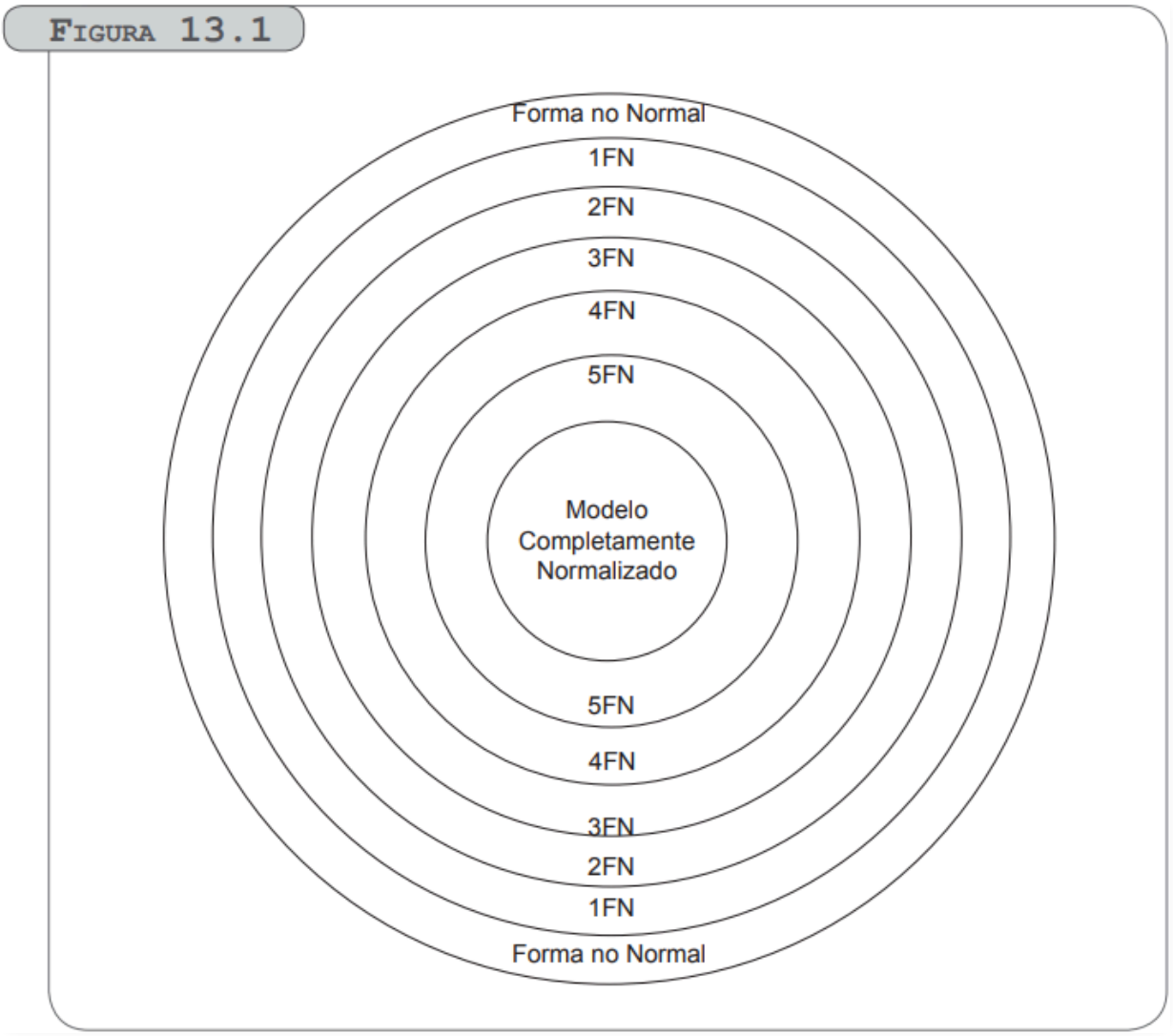
1. La **redundancia** es la repetición de *datos*
2. Cuando la *redundancia* es necesaria, como por ejemplo crear una *tabla relación* con las *claves foráneas* de otras dos *tablas*, decimos que es **deseada**. Caso contrario, es **no deseada**
3. La *redundancia no deseada* puede generar **anomalías de actualización**
4. Una **anomalía de inserción** ocurre cuando ciertos *atributos* no pueden ser insertados en la *BD* sin la presencia de otros atributos
    - Como ejemplo: imagine una *tabla Alumnos* que, en cada *tupla*, se encuentre también la información de la carrera que está cursando (IdCar, NomCar, ...). Al no haber una *tabla* Carreras aparte, si se inserta una *tupla* cuyo contenido incluya (IdCar: 1, NomCar: "Computacion y Electrónica"), y después otra que tenga (IdCar: 1, NomCar: "Electrónica y Computación"), se estaría produciendo la *anomalía*.
5. Una **anomalía de borrado** ocurre cuando ciertos *atributos* no pueden ser borrados en la *BD* sin borrar otros también
    - Al querer borrar una carrera en la *tabla* del ejemplo anterior se estaría borrando también información de un alumno
6. Una **anomalía de modificación** ocurre cuando una o mas instancias de información duplicada son actualizadas, pero no todas
    - Al querer modificar el nombre de la carrera con IdCar: 1 se van a tener que actualizar todas las *tuplas* asociadas
7. Sobre una *BD* se pueden producir las tres clases de *anomalias de actualización*: *anomalias de inserción*, *anomalías de borrado*, y *anomalías de modificación*
8. Se dice que un *atributo* (o conjunto) Y **depende funcionalmente** de un *atributo* (o conjunto) X cuando para un valor dado de X siempre se encuentra el mismo valor para el *atributo* (o conjunto) Y

    - Como ejemplo: para un ID siempre se van a encontrar los mismos valores en *nombre, apellido, edad*, por lo que ID → nombre; ID → apellido; ID → edad. Si el nombre fuera único en cada *tupla* de la *tabla*, por mas que nombre no fuera *CP*, también se estarían dando las siguientes **dependencias funcionales**: nombre → ID; nombre → edad; nombre → apellido

9. En una *DF* X→Y, al *atributo* X se lo denomina **determinante** y al Y **consecuente**
10. Una *DF* X→Y se denomina **parcial** cuando, además, existe otra *dependencia* Z→Y, siendo Z un subconjunto de X. Caso contrario, es **completa**
11. Si en una *DF* el *determinante* es simple, no es posible que la *DF* sea *parcial*
12. Una *DF* X→Y se denomina **transitiva** cuando existe un *atributo* Z, tal que X→Z y Z→Y
13. 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
14. Una **Dependencia Multivaluada** (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
    - Como ejemplo: Imagine una *tabla Canciones* donde para el ID de un artista hayan varios IDs de canciones propias
15. Una *DM* X→→Y se considera **trivial** si Y está multideterminado por el *atributo* X y no por un subconjunto de X
    - Como ejemplo: Imagine una tabla *Sucursales = (Nombre_sucursal, propietario_sucursal, empleado_sucursal)* donde se presenten las siguientes *DMs*: (Nombre_sucursal, propietario_sucursal)→→Empleado_sucursal y (Nombre_sucursal, empleado_sucursal)→→ Propietario_sucursal. En este caso hay *DM* **no triviales.** Cabe aclarar que si para un par (X, Z), y no un subconjunto de este, la Y está multideterminada, entonces sigue siendo *trivial*
16. La **normalización** es un mecanismo que permite que un conjunto de *tablas* eviten la *redundancia no deseada*, las *anomalías de actualización*, y la pérdida de integridad de *datos*
17. Es deseable que una *BD* se *normalice*, pero no es estrictamente necesario
18. Un *modelo* está en **Primera Forma Normal** (1FN) si todos los *atributos* que conforman las tablas *tablas* son *monovalentes*
19. Un *modelo* está en **Segunda Forma Normal** (2FN) si está en *1FN* y en el *modelo* no hay ninguna *DF parcial*
20. Un *modelo* está en **Tercera Forma Normal** (3FN) si está en *2FN* y en el *modelo* no hay ninguna *DF transitiva*

21. Un *modelo* está en **Forma Normal de Boyce-Codd** (FNBC) si está en *3FN* y en el *modelo* no hay ninguna *DF de BC*
22. Un *modelo* está en **Cuarta Forma Normal** (4FN) si está en *FNBC* y las *DMs* que se encuentran son *triviales*
23. Un *modelo* está en **Quinta Forma Normal** (5FN) si está en *4FN* y no hay *redundancia no deseada*

FIGURA 13.1



Procesamiento de consultas

En esta sección pongo lo mínimo indispensable para el teórico nada mas (ppts: Clase 5_1, Clase 6_1, Clase 6_2, Clase 7)

1. Una vez definido el *modelo de datos*, las operaciones posibles son cuatro: agregar, borrar, modificar, o consultar información
2. Un **lenguaje de procesamiento** de datos es aquel que descibe sintácticamente y semánticamente todas las operaciones posibles a realizar sobre una *BD*
3. Si el *lenguaje* pide el qué y el cómo (instrucciones) hacer la consulta, es **procedural**. Si solo pide el qué, es **no procedural**
4. El **algebra relacional** (AR) representa un conjunto de operadores que toman las *tablas* como operandos y regresan otra *tabla* como resultado

5. Las operaciones de AR pueden ser **unarias** o **binarias**, con respecto a si se opera sobre una o dos *tablas*
6. Las **operaciones básicas** son: selección, proyección, renombre, producto cartesiano, unión, y diferencia
7. Algunas **operaciones adicionales** son: producto natural, intersección, asignación, y división
8. Las **operaciones de actualización** son: alta, baja, y modificación
9. Operador LIKE
10. Las **funciones de agregación** son funciones que operan sobre un conjunto de tuplas de entrada y producen un único valor de salida (AVG, COUNT, MAX, MIN, SUM) (No pueden aparecer en el WHERE)
11. Las **funciones de agrupación** son GROUP BY
12. Operaciones de comparación
13. Clausula EXIST
14. Vista
15. Ejecutar una consulta tiene costo de acceso a almacenamiento secundario y costo de cómputo (también costo de comunicación en un sistema distribuido)
16. 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 *BD* hace un arbol para la consulta
17. Análisis → CTtabla [cant. tuplas], CBtabla [tam. bytes de cada tupla], CV(a, tabla) [cant. valores existentes para atributo a]
18. Las operaciones de *selección* y *proyección* se deben hacer lo antes posible para filtrar tuplas y así buscar menos
19. Resulta mas eficiente hacer *productos naturales* en pasos y no juntos
20. Eficiencia versus legibilidad

TABLA 16.1

Consulta original	Expresión equivalente más eficiente
$\sigma_{predicado}$ (tabla1 × tabla2)	($\sigma_{predicado}$ (tabla1) × $\sigma_{predicado}$ (tabla2))
$\sigma_{predicado\ 1\ AND\ predicado\ 2}$ (tabla1)	$\sigma_{predicado\ 1}$ ($\sigma_{predicado\ 2}$ (tabla1))
$\sigma_{predicado}$ (tabla1 U tabla2)	($\sigma_{predicado}$ (tabla1) U $\sigma_{predicado}$ (tabla2))
$\sigma_{predicado}$ (tabla1 – tabla2)	($\sigma_{predicado}$ (tabla1) – $\sigma_{predicado}$ (tabla2))
(tabla1 |×| tabla2 |×| tabla3)	((tabla1 |×| tabla2) |×| tabla3))
(tabla1 |×| tabla2)	(tabla2 |×| tabla1)

Transacciones, y Seguridad e Integridad de Datos

1. Una **transacción** es un conjunto de instrucciones que actúa como unidad lógica de trabajo
2. La **idempotencia** es una condición de la *transacción* que asegura que aunque se reejecute n veces, se genera siempre el mismo resultado sobre la *BD*
3. Es **atómica** si se requiere que se ejecuten todas las instrucciones para realizar la *transacción*
4. Es **consistente** si la ejecución aislada de la *transacción* conserva la consistencia de la *BD*
5. Es **aislada** si ignora el resto de las *transacciones* que se ejecutan concurrentemente en el sistema
6. Es **duradera** si realiza cambios permanentes en la *BD*, incluso si hay fallos en el sistema
7. Para mantener la consistencia de la *BD*, una *transacción* debe ser *atómica, consistente, aislada, y duradera (ACID)*
8. Si no se puede ejecutar una de las instrucciones, la *transacción* falla → atomicidad
9. Una *transacción* está **activa** desde que comienza hasta que termine o se produzca algún fallo
10. Una *transacción* está **parcialmente cometida** cuando se termina de ejecutar la última instrucción
11. Una *transacción* está **cometida** cuando la *transacción* finalizó su ejecución y sus acciones fueron almacenadas correctamente en mem. secundaria
12. Una *transacción* está **fallada** cuando no puede continuar con su ejecución normal
13. Una *transacción* está **abortada** tras haber deshecho cualquier cambio en la *BD* que pudo haber producido (fallada)
14. La **bitácora** es la secuencia de actividades realizadas sobre la *BD*
15. Las operaciones sobre la *BD* deben almacenarse luego de guardar en disco el contenido de la *Bitácora*
16. La **modificación diferida** es un método de recuperación de la *bitácora* que consiste en demorar todas las escrituras en disco de las actualizaciones de la *BD*, hasta que la *transacción* alcance el estado de *cometida* en *bitácora*
17. Como ventaja, las *transacciones fallidas* no tienen impacto sobre la *BD*. Ante un fallo, se ejecuta el algoritmo REDO para toda *transacción* que en *bitácora* haya comenzado y haya hecho "commit"
18. El método de **modificación inmediata** actualiza la *BD* mientras la *transacción* esté *activa* y se vaya ejecutando

19. **Checkpoint** → 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*
20. **Doble paginación** → plantea dividir al *BD* en nodos virtuales (páginas) que contienen determinados *datos.* Se generan dos tablas en disco y cada una de las tablas direcciona a los nodos (páginas) generados
21. La técnica de *doble paginación* tiene como ventaja menos accesos a disco, y como desventaja ser complicada de implementar en un ambiente concurrente/distribuido (pag. 428)

Preguntas del Ensayo Examen Teórico - Noviembre 2024

1. *Un modelo conceptual debe contener entidades y relaciones* (a menos que se considere la posibilidad de un modelo conceptual con una sola entidad sin relaciones)
2. *Un atributo derivado atenta contra la minimalidad del problema* (característica que tiene un elemento de tener una única forma de representación posible y no poder expresarse mediante otros conceptos)
3. *Un atributo polivalente sobre el modelo físico no existe* (es irrelevante entonces decir si puede o no tener cierta cardinalidad)
4. *Una clave candidata puede transformarse en clave primaria* (no resta destacar que en el modelo físico desaparecen los identificadores)
5. *Si una tabla se encuentra en BCNF (Boyce-Codd Normal Form) puede estar en cuarta FN* (el BCNF pide que el modelo esté en 3FN y, además, no exista en ninguna tabla del modelo una DF de BC)
6. *Una relación recursiva sobre el modelo lógico debe tener definida cardinalidad*
7. *La integridad referencial enre dos tablas controla el comportamiento de las tuplas de ambas tablas* (la IR es un concepto, una propiedad que poseen las tablas, no hace que se borren o se bloqueen el borredos de los elementos en sí)
8. *Una clave primaria de una tabla en el modelo físico puede ser un atributo simple obligatorio, y puede ser un identificador del modelo conceptual o lógico* (ya que podría crearse una clave autoincremental exclusivamente en el modelo físico)
9. *Una jerarquía cuando se pasa del modelo conceptual al lógico relacional debe quitarse* (no se puede representar el concepto de herencia en un modelo lógico)

10. *Una consulta en algebra relacional siempre devuelve un resultado* (por mas que esté vacío)
11. *Las funciones de agregación trabajan sobre un conjunto de tuplas* (no hace falta que aparezcan en el SELECT o el HAVING)
12. *La optimización de una consulta la realiza el DBMS* (aunque a veces puede ser que no quede algo por optimizar por lo que podría contemplarse elegir la opción "a veces la realiza el DBMS" y que se justifique)
13. *Una transacción que alcanzó el estado de abortada nunca alcanzó el estado de cometida* (debe haber fallado necesariamente)
14. *La modificación inmediata, sin mas datos aportados*, no es ni mas ni menos eficiente que la modificación diferida, o mejor que la doble paginación
15. *Un checkpoint puede ubicarse en cualquier lugar de la bitácora* (puede contener una lista de transacciones activas como no)