UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

Notas de clase: Fundamentos de Bases de Datos

REPORTE DE ACTIVIDAD DE APOYO A LA DOCENCIA

QUE PARA OBTENER EL TÍTULO DE:

Licenciado en Ciencias de la Computación

P R E S E N T A :

Alejandro Piña Vargas

TUTOR:
Dr. Miguel Ehécatl Morales Trujillo
2017

Datos generales:

Datos del alumno:

Piña
Vargas
Alejandro
Universidad Nacional Autónoma de México
Facultad de Ciencias
Ciencias de la Computación
300211175

Datos del tutor:
Dr.
Miguel Ehécatl
Morales
Trujillo

Datos del sinodal 1:
Dra.
Amparo
López
Gaona

Datos del sinodal 2:
M. en I.
Gerardo
Avilés
Rosas

Datos del sinodal 3:
Dr.
Javier
García
García

Datos del sinodal 4:
L. en C. C.
Marisol
Flores
Castro

Datos del trabajo escrito:
"Notas de clase: Fundamentos de Bases de Datos"
141 pág.
2017

"La ciencia es un mundo de constante aprendizaje, evolución y sorpresa; estudiar ciencia va a mostrarte que el poder de la mente humana es infinito"

Este trabajo es producto de un esfuerzo personal que después de algunos años y constantes interrupciones sale a la luz; va dedicado principalmente a mi madre que con su trabajo y paciencia procura siempre que mi formación personal y académica sea íntegra. Su ejemplo de amor por el conocimiento es lo que me ha llevado a donde estoy.

La mejor herencia de mis abuelos es su ejemplo de constancia, amor, bondad y gratitud; son y serán conceptos fundamentales para mi desarrollo. Este trabajo honra su memoria.

A mi padre que con su enseñanza sobre el infinito mundo de las ciencias y su ejemplo de constancia para superar las dificultades y aprender de ellas me ayudó a elegir mi carrera.

A mi familia, que nunca dejaron de confiar en que este día llegaría.

A mi amigo y asesor Dr. Miguel Morales, por su estímulo, disposición y ayuda para este trabajo; entramos el mismo día, al mismo grupo, con el mismo horario; es grandioso que esta etapa académica concluya con su asesoría y amistad.

A Erick, por sus precisiones para la elaboración del banco de preguntas.

Afortunadamente a lo largo de mi vida siempre me he rodeado de personas que comparten los mismos valores con los que me formaron, de cada una de ellas he aprendido muchas cosas; para mis amigos que siempre estaban ahí cuando necesitaba ayuda, a todos los que colaboraron personal y desinteresadamente para cumplir con mis objetivos profesionales también va este trabajo.

A la Universidad Nacional Autónoma de México, a la Facultad de Ciencias y a su personal académico-administrativo que forman una de las instituciones más grandes en la historia nacional y en la que he pasado experiencias que atesoro y que me formaron intelectualmente. Allí en sus aulas, pasillos, jardines y plazas fue donde conocí personas extraordinarias y ejemplos de superación personal excepcionales. Por último, espero poder retribuir a la sociedad por la oportunidad de concluir mis estudios de licenciatura en esta institución clave en la vida de nuestro país.

Tabla de contenido

Prefacio

Diariamente, en toda actividad, se toman decisiones. Éstas, invariablemente se basan en los datos que conocemos o nos son proporcionados. El efecto de calificar a una decisión como buena está intrínsecamente ligado a la calidad de la información que se tomó como argumento para finalmente decantarse por una opción en lugar de otra.

La calidad de los datos, la velocidad para acceder a ellos y el volumen de éstos son otros factores que intervienen e influyen el proceso de toma de decisiones. El avance tecnológico y el desarrollo de las disciplinas que estudian los datos o basan sus soluciones en éstos, ha ido creciendo de manera exponencial. En la actualidad, hablar de Minería de Datos, Ciencia de Datos -*Data Science*- o *Big Data* es algo habitual en el ambiente de las Tecnologías de la Información y Comunicación. Sin embargo, resulta imposible la existencia de estas disciplinas sin las Bases de Datos.

Es en las bases de datos donde reside la materia prima de la información, por tal razón es fundamental comprender la teoría detrás de éstas. Dentro del plan de estudios de la licenciatura en Ciencias de la Computación impartida en la Facultad de Ciencias de esta Universidad, la asignatura de Fundamentos de Bases de Datos, pretende proveer a los alumnos de conocimientos sólidos en esta materia. Motivado en ofrecer a los alumnos que la cursan otra alternativa para la profundización en la disciplina, este trabajo es una primera referencia que les proporciona los conceptos teóricos que dan origen a las bases de datos.

Si bien existen libros reconocidos y ampliamente consultados para la comprensión de la materia, estas notas pretenden servir como un compendio que le permita al alumno acceder de manera rápida a los conceptos teóricos de las bases de datos. Por tal razón, los objetivos principales de este trabajo de apoyo a la docencia son los siguientes:

1. Recolectar los conceptos y temas básicos referentes a las bases de datos, basados en bibliografía reconocida.

2. Proveer de una guía concentrada del tema a los alumnos que cursan materias relacionadas con las bases de datos.

Para lograrlo, se comenzó con la revisión del temario de la materia, se consultó la bibliografía recomendada y los artículos que fundamentan las bases de datos, posteriormente se formularon las preguntas que se relacionan con esos temas. Además, se observó que aunque fueron diseñadas para la asignatura de Fundamentos de Base de Datos, también resultaron ser provechosas para la asignatura optativa de Base de Datos de la carrera de Actuaría, por lo que también las notas pueden ser utilizadas como material de apoyo -en esta última se modificó la estructura del temario con el fin de enriquecer su contenido, esta modificación consistió en fusionar ambos contenidos temáticos sin afectar la secuencia didáctica de los temas-. Apoyado en la premisa de la necesidad de información, el estudiante interesado en el tema puede encontrar en este trabajo un compendio fundamental de bases de datos recolectado a partir de bibliografía reconocida.

Esta recapitulación fue creada con la colaboración del grupo 9152 en el semestre 2016-II y piloteado en un segundo grupo, el 9173 del semestre 2017-I. Si bien se obtuvieron buenos resultados, es deseable socializarlo y que sea utilizado por un mayor número de alumnos para enriquecerlo y mejorarlo.

A continuación se presentan las notas de clase para el curso de Fundamentos de Bases de Datos organizadas de acuerdo al temario oficial de la asignatura[1]. Las notas están conformadas por 9 capítulos, cada uno de ellos se corresponde con las 9 unidades temáticas previstas en el programa de la asignatura.

[1] Temario de Fundamentos de Bases de Datos, http://www.fciencias.unam.mx/asignaturas/1534.pdf (Consultada el 05/12/2016).

Notas de clase:

FUNDAMENTOS DE BASES DE DATOS

1. Introducción

1. **¿Para qué sirven las bases de datos?**

Las bases de datos sirven para almacenar información y para que los usuarios de éstas (ayudados de herramientas de software) la puedan recuperar y actualizar. Esta información puede ser cualquiera que sea significativa para el individuo u organización que la requiera, en otras palabras, la que se necesita para asistir en el proceso general del negocio de aquel individuo u organización (Date, 2004, pág. 6).

Sirven para representar algún aspecto del mundo real y almacenar datos con un propósito en específico (Elmasri, 2011, pág. 4).

Sirven para guardar información que sea importante para alguna organización, por ejemplo: en bancos, la información de los clientes, sus cuentas, préstamos y transacciones bancarias, de la misma manera en universidades, aerolíneas, telecomunicaciones, en finanzas, entre otros (Silberschatz, 2001, pág. 1-2).

1.1. Desarrollo histórico de las bases de datos y sus aplicaciones

2. **¿Cuáles son las principales aplicaciones de las bases de datos?**

Las bases de datos juegan un papel crítico en casi todas las áreas donde se utilizan las computadoras, incluyendo los negocios, el comercio electrónico, ingeniería, medicina, la genética, la ley, la educación, la ciencia y la biblioteca (Elmasri, 2011, pág. 4).

Las corporaciones mantienen toda su información importante en bases de datos (Ullman, 2009, pág. 1).

De acuerdo con (Sherman, 2014, pág. 6), los beneficios de la toma de decisiones basada en datos están presentes en una amplia gama de disciplinas, incluyendo marketing, ventas, suministro gestión de la cadena, fabricación, ingeniería, gestión de riesgos, finanzas y recursos humanos. Además, es

claro que muchas más personas de una organización necesitan la información que proviene de todos estos datos. Por ejemplo:

- Los datos de envíos se extienden mucho más allá del departamento de envíos para incluir a las empresas de transportes y, por supuesto, a los clientes.
- El análisis de sitios web que buscan "hits", que los gestores de mercado utilizan para medir el éxito de ventas y campañas en los medios sociales.
- La información médica es compartida no sólo con los médicos, sino también con redes de hospitales, los mismos pacientes y las compañías de seguros.
- Servicios de *streaming* de TV recopilan datos sobre todo lo que observamos y que usan para recomendar otras películas o programas que piensa que nos gustaría.

3. Describe algunas aplicaciones de bases de datos en la vida real.

Las bases de datos y la tecnología de bases de datos tienen un impacto mayor debido al creciente uso de las computadoras. Es justo decir que las bases de datos juegan un rol crítico en casi todas las áreas donde las computadoras son utilizadas, incluyendo negocios, comercio electrónico, ingeniería, medicina, genética, leyes, educación y ciencia bibliotecaria (Elmasri, 2011, pág. 4).

Algunas aplicaciones en la vida real son:

- “Una base de datos de un enorme tamaño y complejidad es la que posee el Internal Revenue Service (IRS) para monitorear formularios de impuesto llenados por los contribuyentes de los Estados Unidos.” (Elmasri, 2011, pág. 5).
- “Un ejemplo de una bases de datos comercial es Amazon.com. Ésta contiene los datos de más de 20 millones de libros, CDs, videos, DVDs, juegos, electrónicos, vestido y otros artículos. La base de datos ocupa 2 petabytes y se almacena en 200 servidores. Alrededor de 15 millones de visitantes acceden a Amazon.com cada día.” (Elmasri, 2011, pág. 5).

4. **¿Qué ventajas proporciona el uso de bases de datos con respecto a otros mecanismos de persistencia de información?**

Las principales ventajas de utilizar bases de datos son:

- **Compacta:** No hay necesidad de tener un volumen de papeles.
- **Rapidez:** La máquina puede recuperar y actualizar los datos más rápido que un humano.
- **Menos trabajo:** Eliminan el proceso de administrar archivos manualmente. Los procesos se automatizan.
- **Frecuencia:** La información está disponible en cualquier momento.
- **Protección:** Los datos pueden estar mejor protegidos contra la pérdida no intencional y al acceso ilícito.

Los beneficios aplican también con más fuerza en un ambiente de muchos usuarios, donde la base de datos es mucho más grande y más compleja. Aun así, existe una ventaja adicional: el sistema de bases de datos proporciona a una empresa un control centralizado de sus datos e información (Date, 2004, pág. 16-17).

5. **¿Qué desventajas conlleva el uso de bases de datos?**

La integración de los datos y la existencia del SMBD también plantean ciertos inconvenientes, como los que se citan a continuación:

- **Alta complejidad:** los SMBD son conjuntos de programas muy complejos y con una gran funcionalidad. Es preciso comprender muy bien estas funcionalidades para poder sacar un buen partido de ellas.
- **Gran tamaño:** los SMBD son programas complejos y muy extensos que requieren una gran cantidad de espacio en disco y de memoria para trabajar de forma eficiente.
- **Costo:** el costo de un SMBD varía dependiendo del entorno y de la funcionalidad que ofrece. Además, hay que considerar el costo del mantenimiento. En los últimos años han surgido SMBD libres (*open source*) que ofrecen una gran funcionalidad y muy buena eficiencia.

- **Equipamiento adicional:** tanto el SMBD, como la propia base de datos, pueden hacer que sea necesario adquirir más espacio de almacenamiento. Además, para alcanzar la eficiencia deseada, es posible que sea necesario adquirir una máquina más grande o una máquina dedicada exclusivamente al SMBD. Todo esto hará que la implantación de un sistema de bases de datos sea más costosa.
- **Conversión:** en algunas ocasiones, convertir la aplicación actual a un sistema de bases de datos tendrá alto precio. Este costo incluye la capacitación del personal para utilizar el nuevo sistema y, probablemente, la contratación de personal especializado para ayudar a realizar la conversión y poner en marcha el sistema.
- **Vulnerabilidad:** el hecho de que todos los datos estén centralizados en el SMBD hace que el sistema sea más vulnerable ante los fallos o ataques que puedan producirse. (Marqués, 2011, pág. 11-12)

6. ¿Qué tipos de usuarios interactúan con la base de datos?

Un usuario es todo aquel individuo que tenga contacto con el sistema de bases de datos. Entre los más representativos se pueden mencionar los siguientes:

- Programador.
- Diseñador de la Base de Datos (DBD).
- Administrador de Base de Datos (DBA).
- Usuario final.

7. ¿Cuáles son las funciones principales de un Diseñador de Bases de Datos (DBD)?

Los diseñadores de bases de datos (DBD) o modeladores de datos son los responsables de identificar que datos van a ser almacenados en la base de datos y de elegir las estructuras adecuadas para representar y almacenar estos datos.

Es responsabilidad de los DBD comunicarse con todos los usuarios posibles, con el fin de entender sus necesidades y opciones posibles para crear un diseño que cumpla con los requisitos deseados.

Los DBD suelen interactuar con cada grupo de usuarios potenciales y desarrollan los puntos de vista sobre la base de datos para que cumplan con los requisitos de datos y procesamiento de estos grupos. Cada vista se analiza y se integra con las opiniones de otros grupos de usuarios. El diseño de la base de datos final debe ser capaz de soportar los requerimientos de todos los grupos de usuarios (Elmasri, 2011, pág. 15).

Podemos enlistar las tareas que realiza un DBD, las cuales son[2]:

- Diseñar una estructura de base de datos (modelo de datos lógico) para hacer frente a las necesidades y expectativas de los futuros usuarios.
- Llevar a cabo un proyecto de estudio sobre TIC (Tecnologías de la Información y las Comunicaciones) para evaluar la viabilidad y/o los costes de una base de datos.
- Programar bases de datos en idiomas informáticos tales como SQL (Structured Query Language).
- Proporcionar información para la base de datos.
- Probar las bases de datos.
- Desarrollar formas de mostrar la información a los usuarios, por ejemplo programando aplicaciones de internet.
- Mantener y adaptar bases de datos existentes siguiendo las necesidades cambiantes de los usuarios, o las cambiantes posibilidades en la programación.
- Realizar informes basados en la información de la base de datos.

8. ¿Cuáles son las funciones principales de un Administrador de Bases de Datos (DBA)?

De acuerdo con (Silberschatz, 2006, pág. 27), las funciones de un Administrador de Bases de datos (DBA) son:

- **Definición de esquema.** El DBA crea el esquema base de datos original mediante la ejecución de un conjunto de instrucciones de definición de datos en el DDL (Lenguaje de Definición de Datos)
- **Definir la estructura de almacenamiento y los métodos de acceso.**

[2] Tareas de un DBD, http://www.123test.es/profesiones/profesion-disenador-de-bases-de-datos/ (Consultada el 24/02/2016).

- **Esquema y modificación-organización física.** El DBA lleva a cabo cambios en el esquema y la organización física para reflejar las cambiantes necesidades de la organización, o para modificar la organización física con el fin de mejorar el rendimiento.
- **Solicitud de autorización para el acceso a datos.** Con la concesión de diferentes tipos de autorización, el administrador de base de datos puede regular a qué partes de la base de datos pueden acceder varios usuarios.
- **Realizar mantenimiento de rutina.** Los ejemplos de las actividades de mantenimiento de rutina del administrador de base de datos son:
    - Periódicamente hacer copias de seguridad de la base de datos para evitar la pérdida de datos en caso de desastres naturales o fallas en la infraestructura.
    - Asegurar que el espacio libre en disco sea suficiente para las operaciones normales, y para actualizaciones, según sea necesario.
    - Monitoreo de trabajos ejecutándose en la base de datos y garantizar que el rendimiento no se degrada por tareas muy costosas presentadas por algunos usuarios.

En cualquier organización donde muchas personas utilizan los mismos recursos, es necesario un administrador principal para supervisar y administrar estos recursos. En un entorno de base de datos, el recurso principal es la propia base de datos, y el recurso secundario es el SMBD y el software relacionado. La administración de estos recursos es responsabilidad del DBA. El DBA es responsable de autorizar el acceso a la base de datos, coordinar y supervisar su uso y adquisición de recursos de software y hardware como sea necesario. El DBA es responsable de problemas tales como las brechas de seguridad y fallas en el tiempo de respuesta del sistema. En organizaciones grandes, el DBA es asistido por un personal que realiza estas funciones (Elmasri, 2011, pág. 15).

1.2. Ciclo de vida de una aplicación de base de datos

9. ¿En qué consiste el ciclo de vida de una aplicación de base de datos?

Para poder llegar a la consumación de una aplicación de base de datos se deben cumplir con diversas etapas que van desde la concepción y análisis hasta la implementación y liberación –todas con una

estrecha interrelación-. Generalmente, entre las etapas que componen el ciclo de vida de una base de datos se pueden mencionar: concepción, análisis de requerimientos, diseño, implementación, pruebas, liberación, operación, mantenimiento y retiro.

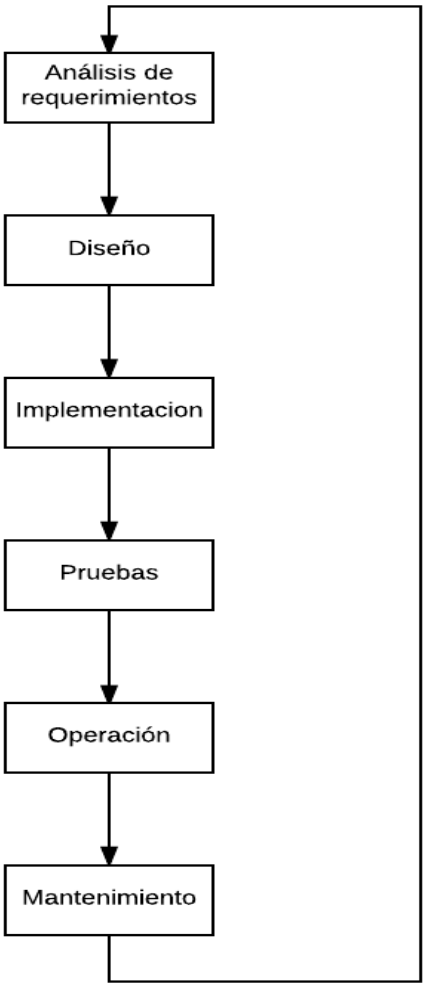


Fig. 1 – Ciclo de vida de una aplicación de base de datos

10. ¿En qué consiste el análisis de la factibilidad de una base de datos?

La fase de Análisis de factibilidad tiene que ver con el análisis de la potencial aplicación de la base de datos, identificando los aspectos económicos de la recolección y difusión de información, la realización de estudios preliminares de costo-beneficio, la determinación de la complejidad de los datos y procesos, y el establecimiento de prioridades entre las aplicaciones (Elmasri, 2011, pág. 307).

El análisis de la factibilidad consiste en determinar si el sistema solicitado es factible en función de la viabilidad de tres factores:

- **Factibilidad técnica:** la mejora puede realizarse con la tecnología, software, personal, etc. con la que se dispone.
- **Factibilidad económica:** la relación que existe entre el costo-beneficio es satisfactoria.
- **Factibilidad operacional:** la evaluación de la posible resistencia del usuario a los cambios propuestos recomienda llevar a cabo esta mejora.

Si el sistema que se quiere implementar supera el análisis de factibilidad, entonces se aprueba su realización y se presupuesta (Taboada & Cotos, 2005, pág. 10).

11. ¿En qué consiste el análisis de requerimientos de una base de datos?

El análisis de requerimientos consiste en conocer los objetivos para generar una base datos con el propósito de estructurar la información y así permitir al usuario extraer y actualizar información en demanda. La información en cuestión puede ser cualquier cosa de interés para el individuo u organización a la que concierne (Date, 2004, pág. 6).

El primer paso para el diseño es la recopilación y análisis de requerimientos. Durante este paso los diseñadores de la base de datos recopilan, entienden y documentan los requisitos de datos por parte de lo(s) usuario(s). Los requerimientos deben ser detallados y lo más específicos posible; de manera paralela se deben solicitar las funciones de la aplicación; esto es, las operaciones que la Base de datos debería de realizar.

Los requisitos son recolectados interactuando con las partes interesadas (*stakeholders*) para identificar sus problemas particulares y sus necesidades. Se identifican también las dependencias entre las aplicaciones, comunicación y procedimientos de los reportes (Elmasri, 2011, pág. 307).

12. ¿En qué consiste el diseño de la base de datos?

El diseño de una base de datos consiste en la transformación de los requerimientos a un modelo conceptual. El diseño posteriormente será trasladado a un diseño lógico el cual puede ser expresado

en un modelo de datos. La etapa final es el diseño físico, durante el cual especificaciones adicionales son dadas para almacenar y acceder a la base de datos (Elmasri, 2011, pág. 9).

13. ¿En qué consiste la implementación de una base de datos?

La implementación de una base de datos comprende el proceso de la especificación de las definiciones conceptuales externas e internas de las bases de datos, creando los archivos de la base de datos, e implementando las aplicaciones del software (Elmasri, 2011, pág. 308).

La implementación del modelo de datos es típicamente el modelo relacional de datos, y este paso consiste en la transformación del esquema conceptual definido, utilizando el modelo entidad-relación, al esquema relacional (Silberschatz, 2006, pág. 203).

Es la conversión de un diseño de alto nivel a un diseño relacional. Esta fase se produce al convertir el diseño de alto nivel en un esquema relacional de base de datos, que luego se ejecuta en un SMBD convencional (Ullman, 2009, pág. 125).

14. ¿En qué consisten las pruebas de validación y aceptación de una base de datos?

En esta parte del ciclo de vida de una aplicación de bases de datos, se valida la aceptabilidad del sistema para satisfacer los requisitos de los usuarios y los criterios de rendimiento. Además el sistema es probado contra los criterios de rendimiento y especificaciones de comportamiento (Elmasri, 2011, pág. 307).

15. ¿En qué consiste la operación de una base de datos?

De acuerdo al ciclo de vida de una aplicación de bases de datos, la operación se refiere al momento en que el sistema de base de datos y sus aplicaciones son puestos en funcionamiento. Por lo general, viejos y nuevos sistemas funcionan en paralelo durante un periodo de tiempo (Elmasri, 2011, pág. 308).

16. ¿En qué consiste el monitoreo y mantenimiento de una base de datos?

Monitorear la base de datos proporciona estadísticas para el administrador de base de datos (DBA). El DBA utiliza las estadísticas en la toma de decisiones tales como, si reorganizar o no los archivos o si desea añadir o borrar índices para mejorar el rendimiento (Elmasri, 2011, pág. 43).

Por otra parte, algunos ejemplos de las actividades de mantenimiento de rutina del DBA son:

1. Crear periódicamente copias de seguridad de la base de datos (generalmente en servidores remotos), para evitar la pérdida de datos en caso de desastres o ataques.

2. Asegurar que el espacio libre disponible en disco sea el suficiente para las operaciones normales, y actualizar el espacio en disco según sea necesario.

3. Monitoreo de los puestos de trabajo en los que se ejecutan consultas a la base de datos y asegurar que el rendimiento no se degrada por consultas redundantes presentadas por algunos usuarios.

   (Elmasri, 2006, pág. 27).

17. ¿En qué consiste la carga o conversión de datos?

La base de datos es poblada ya sea mediante la carga de los datos directamente o mediante la conversión de los archivos existentes al formato del sistema de base de datos. En pocas palabras la aplicación de conversión, es una transformación hecha a un conjunto de datos de un sistema anterior para convertirlos al formato del nuevo sistema (Elmasri, 2011, pág. 308).

18. ¿En qué consiste la conversión de aplicaciones?

La conversión de aplicaciones consiste en que cualquier aplicación de software de un sistema previo es transformada para que pueda funcionar adecuadamente con el nuevo sistema (Elmasri, 2011 pág. 308).

1.3 Conceptos básicos de bases de datos

19. ¿Qué es un dato?

Dato es la representación de la información del mundo real dentro de una base de datos.

Dato se refiere a aquello que es almacenado en una base de datos (Date, 2004, pág. 6).

Un dato es una representación simbólica (numérica, alfabética, algorítmica, espacial, entre otras) de un atributo o variable cuantitativa o cualitativa[3].

El término datos se refiere a los hechos brutos registrados en la base de datos. Pueden ser ítems acerca de personas, lugares, eventos o conceptos (Ricardo, 2009, pág. 50).

Por dato se entienden hechos conocidos que pueden ser guardados y que tienen un significado de acuerdo al contexto. Por ejemplo, considera los nombres, números de teléfono y direcciones de la gente que conoces, pudiste haber guardado esta información en una agenda o un disco duro (Elmasri, 2011, pág. 4).

Un dato debe cumplir con las siguientes propiedades (Cinco C's):

- Limpieza (Clean)
- Consistencia (Consistency)
- Conformidad (Conformity)
- Actual (Current)
- Comprensible. (Comprehensible)

(Sherman, 2014, pág. 13)

20. ¿Qué es la información?

Información es un dato que ha sido organizado, estructurado y procesado (Sherman, 2014, pág. 8).

Información es el significado de los datos que se entienden por algún usuario (Date, 2004, pág. 6).

La información es la que permite al ser humano tomar decisiones en situaciones de su contexto habitual. La información con la que se va a integrar la base de datos debe ser consistente con una característica específica, debe ser constante, es decir que, no puede cambiar el significado del dato y debe ser congruente con los requisitos del diseño de la base de datos; de esta manera se garantiza la integridad de la información almacenada en la base de datos.

[3] Definición de dato, https://es.wikipedia.org/wiki/Dato (Consultada el 05/12/2016).

21. ¿Qué es un sistema de información?

Un sistema de información (IS) es un conjunto de elementos que interactúan entre ellos, incluye todos los recursos que están involucrados en la recopilación, gestión, uso y difusión de los recursos de información de la organización. En un entorno computarizado, estos recursos incluyen los datos en sí, el SMBD, los medios de almacenamiento de hardware, el sistema operativo, el personal que usa y gestiona los datos, los programas de aplicación (software)que acceden y actualizan los datos, y los programadores de aplicaciones que desarrollan estas aplicaciones (Elmasri, 2011, pág. 307).

Conjunto de componentes relacionados que recolectan (o recuperan), procesan, almacenan y distribuyen información para apoyar la toma de decisiones y el control en una organización.[4]

Un sistema de información es un conjunto de componentes que interaccionan entre sí para satisfacer las necesidades de información de una organización. El objetivo primordial de un sistema de información es apoyar la toma de decisiones y controlar todo lo que en ella ocurre.[5]

22. ¿Qué es una base de datos?

Una base de datos es una colección de datos que es administrada por un Sistema Manejador de Base de Datos (SMBD) (Ullman, 2009, pág. 1).

Una base de datos es una colección de datos que contiene información relevante para una empresa (Silberschatz, 2006, pág. 1).

Una base de datos es una colección de datos relacionados (Elmasri, 2011, pág. 4).

Una base de datos es un depósito o contenedor para un conjunto de archivos de datos computarizados (Date, 2004, pág. 3).

[4] Definición de sistema de información, http://biblioteca.itson.mx/oa/dip_ago/introduccion_sistemas/p3.htm (Consultada el 05/12/2016).
[5] Definición de sistema de información, http://definicion.de/sistema-de-informacion/#ixzz41RMp8DMv (Consultada el 05/12/2016).

23. ¿Qué es una base de datos estática?

Una base de datos estática es una base de datos de sólo lectura. Éstas son utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.[6]

24. ¿Qué es una base de datos dinámica?

Las bases de datos dinámicas son bases de datos, en las cuales la información sufre modificaciones en el transcurso del tiempo, permitiendo actualizar, eliminar y agregar datos, además de realizar consultas.[7]

25. ¿Qué es una tupla?

Las filas de una relación, diferentes de la fila de encabezado que contiene los nombres de los atributos, son llamados tuplas (Ullman, 2009, pág. 22).

Las tuplas son las filas de una tabla o relación (Silberschatz, 2006, pág. 38).

El término tupla corresponde aproximadamente a una instancia de registro (Date, 1986, pág. 100).

Cuando una relación se piensa como una tabla de valores, cada fila en la tabla representa un conjunto de parejas (<atributo>, <valor>). Estas filas son llamadas tuplas (Elmasri, 2011, pág. 64).

1.4. Arquitectura de tres niveles

26. ¿En qué consiste la arquitectura ANSI/SPARK?

A mediados de los 70´s e inicios de los 80´s el Instituto Nacional Estadounidense de Estándares (American National Standards Institute, ANSI) y el Comité de Requisitos y Planificación de Estándares (Standards

[6] Definición de base de datos estática, https://es.wikipedia.org/wiki/Base_de_datos (Consultada el 05/12/2016).
[7] Definición de base de datos dinámica, https://es.wikipedia.org/wiki/Base_de_datos (Consultada el 05/12/2016).

Planning And Requirements Committee, SPARC), propusieron una arquitectura de tres niveles para una base de datos. Estos niveles permiten pensar la base de datos desde la abstracción, así como también la visión personalizada de la base de datos en términos de cada uno de los usuarios que intervienen en la creación, gestión y manipulación de este tipo de sistemas.[8]

27. ¿En qué consiste la arquitectura de tres niveles de una base de datos?

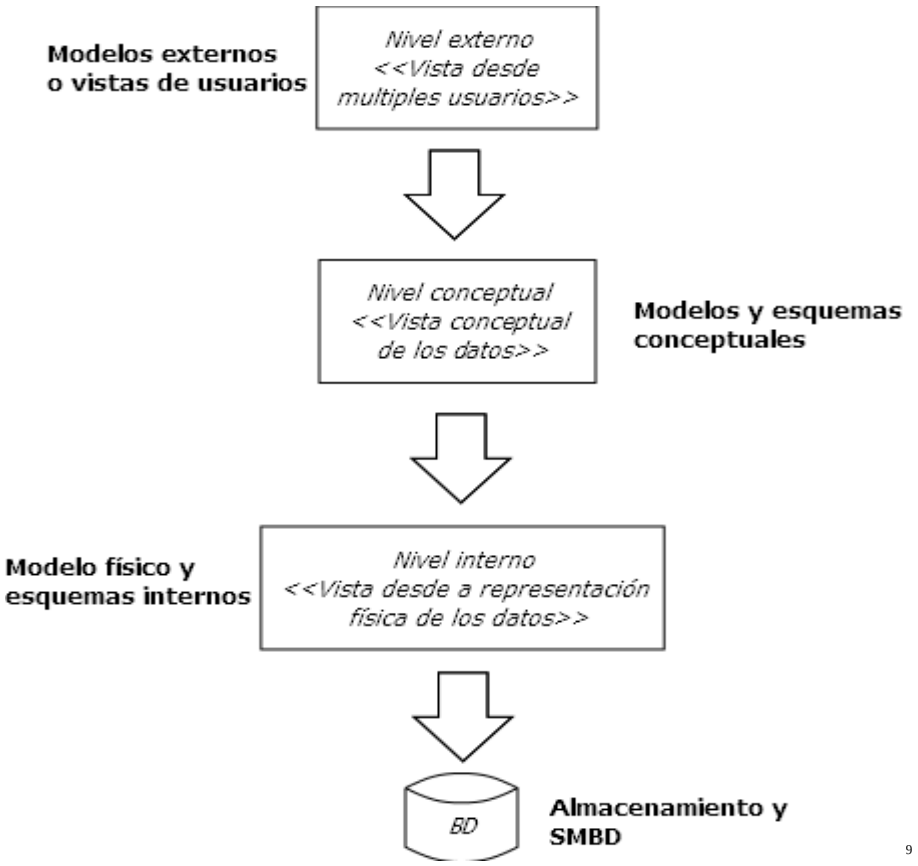
El fin de la arquitectura de tres niveles, es separar las aplicaciones del usuario de la base de datos física para ayudar en la visualización de sus características. Esta arquitectura se puede definir en los siguientes tres niveles:

- **El nivel Externo o Visual:** corresponde a la interfaz gráfica, y es como se le presentan a los usuarios los datos guardados. Sólo muestra los datos en los cuales los usuarios están interesados
- **El nivel Conceptual o Lógico:** describe la estructura de toda la base de datos a una comunidad de usuarios. Se concentra en describir entidades, tipos de datos, relaciones, usos de operaciones y límites.
- **El nivel Interno o Físico:** representa cómo se almacenan físicamente los datos.

(Elmasri, 2011, pág. 33-34).

Es un tipo de arquitectura que sirve para representar la mayoría de los sistemas de bases de datos y se divide en los niveles externo, conceptual e interno (Date, 2004, pág. 34).

[8] Arquitectura ANSI/SPARK, http://escritura.proyectolatin.org/diseno-e-implementacion-de-bases-de-datos-desde-una-perspectiva-practica/13-tipos-de-modelos-de-datos (Consultada el 22/11/2016).



[9]

Fig. 2 – Arquitectura de tres niveles de una base de datos

28. Describe el nivel físico de una base de datos.

Este nivel tiene un esquema interno, que describe la estructura de almacenamiento físico de la base de datos. El esquema interno utiliza un modelo físico de datos y describe los detalles completos de rutas de almacenamiento de datos y de acceso a la base de datos (Elmasri, 2011, pág. 34).

De acuerdo con (Ullman, 2009, pág. 369), la implementación del modelo de datos físicos, así como su almacenamiento, requiere la comprensión de las características y limitaciones de rendimiento del sistema de base de datos que se utiliza. El diseño del modelo de datos físico requiere un profundo conocimiento de los SMBD específicos que se utilizan con el fin de:

- Representar el modelo de datos lógicos en un esquema de base de datos.
- Configurar y ajustar la base de datos para los requisitos de rendimiento.

[9] Adaptada de: http://etherpad.proyectolatin.org/up/b515fe01b99052251e8d6aaff2b44524.jpg (Consultada el 05/12/2016).

La vista interna es una representación de bajo nivel de toda la base de datos; consiste en muchas ocurrencias de muchos tipos de registros internos. Registro interno es el término ANSI/SPARC para la construcción que tenemos para llamar a un registro almacenado. La vista interna está a un paso del nivel físico, ya que no se ocupa en términos de registros físicos con consideraciones específicas del dispositivo (Date, 2004, pág. 40).

29. ¿Cuál es el nivel conceptual de una base de datos?

Está destinado a ser una visión de los datos "como estos realmente son". Toda esta visión se hace mediante el esquema conceptual el cual está destinado a incluir una gran cantidad de características adicionales, ya que su objetivo final es describir completamente todo el problema y no solo los datos (Date, 2004, pág. 40).

El nivel conceptual tiene un esquema conceptual, el cual describe la estructura de toda la base de datos para la comodidad de los usuarios, se concentra en describir entidades, tipos de datos, relaciones, operaciones usadas y restricciones (Elmasri, 2011, pág. 34).

Es un nivel abstracto que describe cómo los datos son almacenados en una base de datos, y cuál es la relación que existe entre ellos, este nivel sirve para describir completamente la base en términos de una estructura simple (Silberschatz, 2006, pág. 15).

Es un nivel en el cual se pueden representar a las estructuras de bases de datos así como sus relaciones (Ullman, 2009, pág. 18).

30. Describe el nivel externo de una base de datos.

El nivel externo o de vista incluye una serie de esquemas externos o vistas de usuario. Cada esquema externo describe la parte de la base de datos en la que un grupo de usuarios particular está interesado y oculta el resto de la base de datos a partir de ese grupo de usuarios. Cada esquema externo se implementa típicamente usando un modelo de datos de representación, posiblemente basado en un esquema externo o en un modelo de datos de alto nivel (Elmasri, 2011, pág. 34).

El nivel externo es el nivel individual del usuario (Date, 2004, pág. 38).

El nivel externo es el contenido de la base de datos como lo ve algún usuario en particular (Date, 2004, pág. 39).

31. ¿En qué consiste la independencia física de datos?

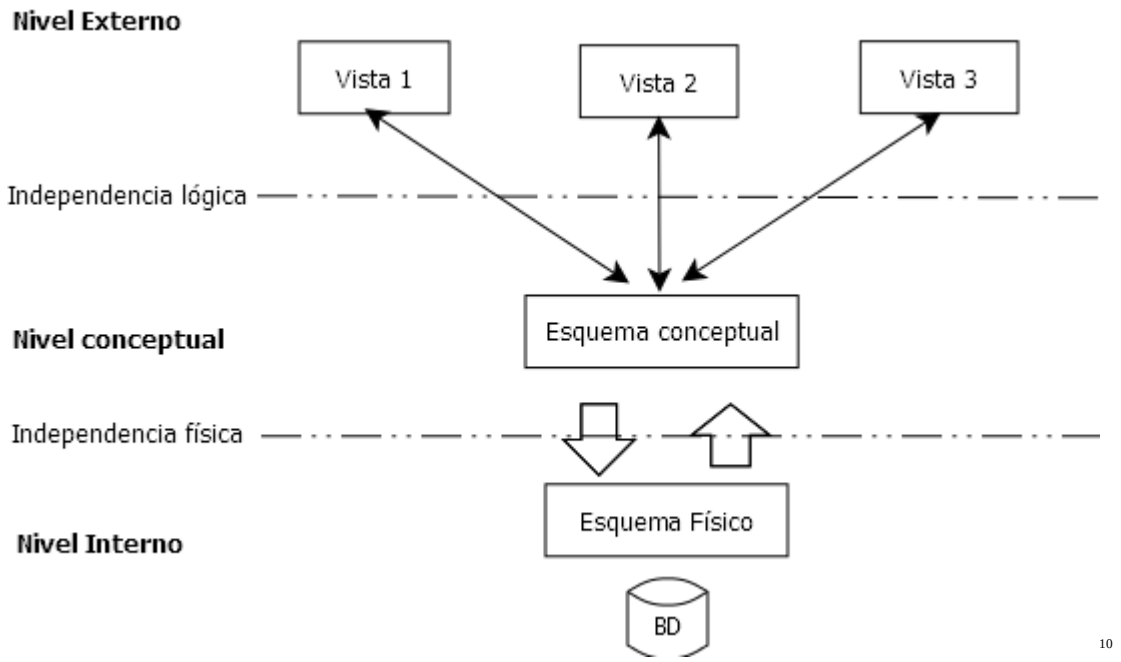
La independencia física de datos es la capacidad de cambiar el esquema interno sin tener que cambiar el esquema conceptual. Por lo tanto, el esquema externo no necesita cambiarse tampoco. Los cambios en el esquema interno pueden ser debido a que algunos archivos físicos se reorganizaron para mejorar el rendimiento de recuperación o actualización. Puede referirse también a un tipo de cambio en el hardware (Elmasri, 2011, pág. 36).

- **Independencia física de los datos:** es la capacidad de cambiar el esquema interno sin cambiar el esquema conceptual, este cambio pude ser para mejorar el rendimiento de las consultas o para actualizar el contenido de la base. Puede referirse también a un tipo de cambio en el hardware (Elmasri 2011, pág. 35-36).

32. ¿En qué consiste la independencia lógica de datos?

La independencia lógica de datos es la capacidad de cambiar el esquema en un nivel de la base de datos sin tener que cambiar el esquema en el nivel más alto, existen dos tipos de independencia:

- **Independencia lógica de datos:** que es la capacidad de cambiar el esquema conceptual sin tener que cambiar los programas de aplicación, con el cambio del esquema conceptual podemos entender la base, cambiar las constantes o reducirlos datos (Elmasri 2011, pág. 35-36).



[10]

Fig. 3 – Independencia física y lógica de datos

1.5. Tipos de arquitecturas

33. ¿En qué consiste la arquitectura centralizada?

Una base de datos centralizada es aquella que está totalmente en un solo lugar físico, es decir, está almacenada en una sola máquina y en un solo CPU, en la cual los usuarios trabajan en terminales que solo muestran resultados. Los sistemas de bases de datos centralizadas son aquellos que se ejecutan en un único sistema informático sin interactuar con ninguna otra computadora. Tales sistemas van desde los sistemas de bases de datos monousuarios ejecutándose en computadoras personales hasta los sistemas de bases de datos de alto rendimiento ejecutándose en grandes sistemas (Silberschatz, 2001, pág. 683).

Un ejemplo de arquitectura centralizada es una base de datos alojada en una sola computadora que registre los inventarios de un almacén, sus ventas, compras y devoluciones de mercancía.

[10] Adaptada de: http://slideplayer.es/slide/1371492/ diapositiva 57 (Consultada el 05/12/2016).

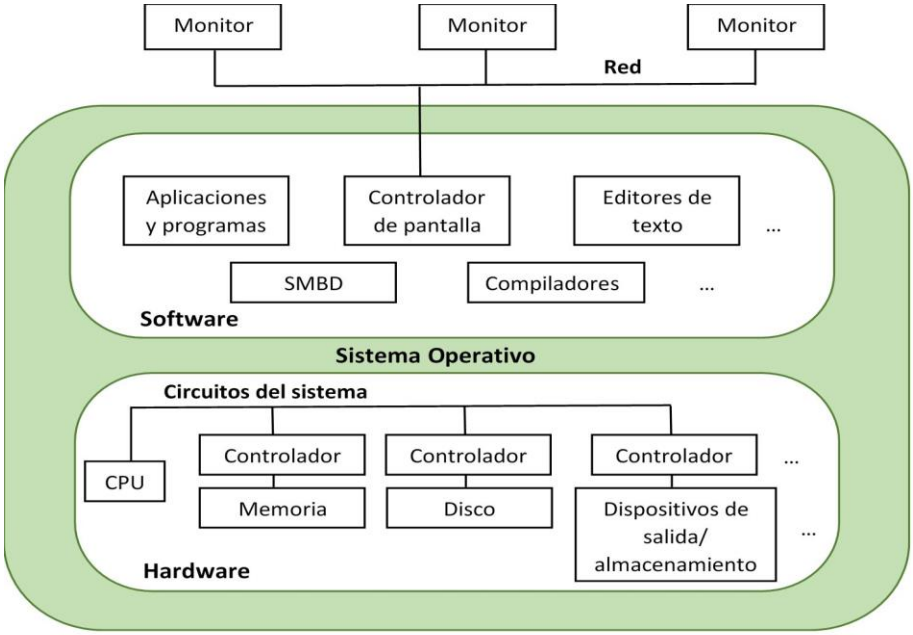


Fig. 4 – Arquitectura centralizada

(Elmasri, 2011, pág. 45)

34. ¿En qué consiste la arquitectura cliente – servidor?

La arquitectura cliente-servidor supone que la funcionalidad de la interfaz de usuario que la utiliza puede ser manejada directamente por un sistema centralizado. Como resultado, los sistemas centralizados hoy actúan como sistemas de servidor que satisfacen las peticiones generadas por los sistemas del cliente. La funcionalidad de la base de datos se puede dividir en dos partes: *front-end* y *back-end*. El *back-end* gestiona estructuras de acceso, consulta evaluación y optimización, control de concurrencia, y la recuperación. El *front-end* de un sistema de base de datos consta de herramientas tales como formularios, redactores de informes, y la facilidad que tienen los usuarios en la interfaz gráfica. La interfaz entre el *front-end* y *back-end* es a través SQL, o por medio de un programa de aplicación (Silberschatz, 2001, pág. 682-683).

Es posible conectar varias computadoras como clientes a un servidor de archivos que mantiene los archivos de las máquinas cliente. Otra máquina puede ser designada como servidor de impresión mediante su conexión con diferentes impresoras; todas las solicitudes de impresión de los clientes se envían a esta máquina. Los servidores de almacenamiento de correo del servidor web también entran en la categoría de servidor especializado. Los recursos proporcionados por servidores especializados

se pueden acceder a muchas máquinas cliente. Las máquinas de cliente proporcionan al usuario las interfaces adecuadas para utilizar estos servidores, así como con la potencia de procesamiento local para ejecutar aplicaciones locales (Elmasri, 2011, pág. 46).

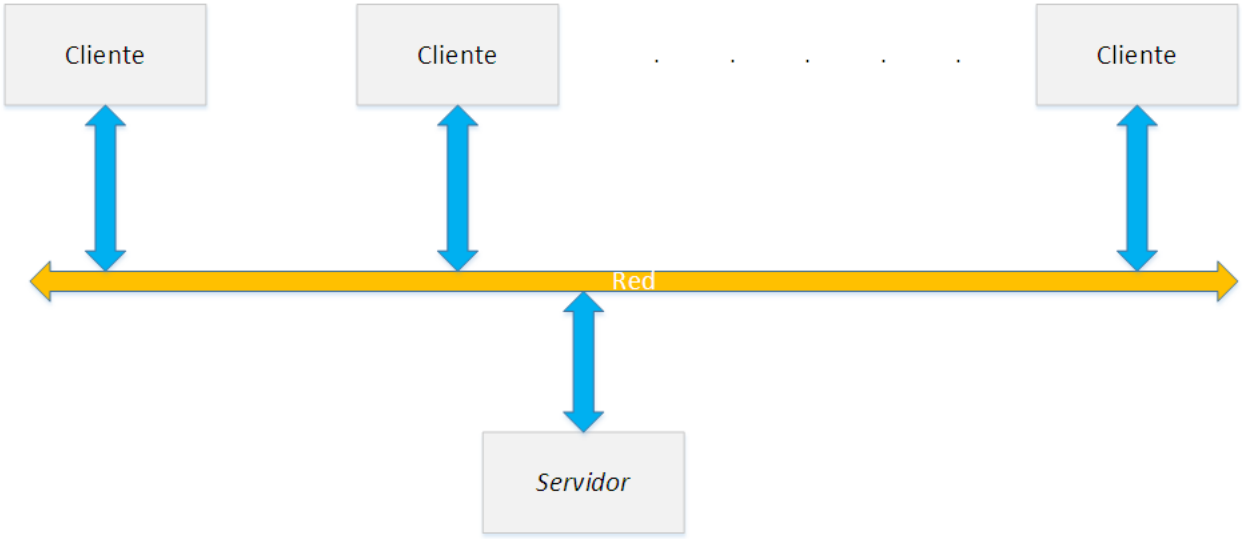


Fig. 5 – Arquitectura cliente – servidor

(Silberschatz, 2001, pág. 682-683)

1.6. Sistemas Manejadores de Bases de Datos

35. ¿Qué es un Sistema Manejador de Bases de Datos (SMBD)?

Un SMBD es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, además de proporcionar un acceso controlado a la misma (Marqués, 2011, pág. 3).

36. ¿Cuáles son los componentes principales de un SMBD? Descríbelos.

De acuerdo con (Osorio-Rivera, 2008, pág. 21-22), un SMBD se divide en una serie de módulos que se encargan de cada una de las tareas del sistema general. Básicamente, consiste en varios componentes funcionales, entre los que se cuentan:

- **El manejador de archivos:** encargado de asignar espacio en el disco y manejar las estructuras de datos que se van a emplear para representar la información almacenada en disco.

- **El manejador de base de datos:** constituye la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicaciones y las consultas que se hacen al sistema.
- **El procesador de consultas:** traduce las proposiciones en lenguaje de consulta a instrucciones de bajo nivel que pueda entender el manejador de base de datos.
- **El precompilador de DML:** convierte las proposiciones hechas en lenguaje DML en procedimientos del lenguaje huésped.
- **El compilador DDL:** convierte las proposiciones DDL en un conjunto de tablas que contienen metadatos.

37. ¿Cuáles son las ventajas de utilizar un SMBD?

Existen diversas ventajas para utilizar un SMBD, de las cuales de acuerdo con (Elmasri, 2011, pág. 15) destacan principalmente:

- **Existe un control de la redundancia:** en un SMBD, no existirán datos repetidos, esto con el fin de que haya un doble de esfuerzo al realizar una tarea determinada y a su vez, no se desperdicie el espacio de almacenamiento.
- **Se puede restringir el acceso no autorizado:** en algunos casos, cuando hay varios usuarios que comparten una base de datos, no todos pueden acceder a cierto tipo de información confidencial que otros si pueden, para estos casos, un SMBD nos permite diferenciar un perfil con mayor acceso que otro, si así se desea.
- **Existe compatibilidad entre lenguajes enfocado a objetos:** los sistemas de bases de datos orientados a objetos son compatibles (en su mayoría) con lenguajes de programación tales como C++ y Java, y el software SMBD puede realizar automáticamente las conversiones necesarias.

- **Existe un suministro de estructuras de almacenamiento para un procesamiento eficaz de las consultas:** los SMBD deben proporcionar capacidades que permitan ejecutar eficazmente consultas y actualizaciones.
- **Se puede generar una copia de seguridad y recuperación:** al existir una contingencia en el hardware o software, el SMBD es capaz de afrontar dicha situación creando previamente una copia de seguridad.
- **Representación de relaciones complejas entre datos:** un SMBD puede crear con una variedad de datos, relaciones complejas que permitan conectar la información.
- **Inferencia y acciones implementando reglas:** en determinados SMBD se pueden definir reglas de deducción para extraer información nueva a partir de datos ya ingresados.

38. ¿Cuáles son las desventajas de usar un SMBD?

A pesar de las ventajas de utilizar un SMBD, hay algunas situaciones en las que el uso de éstos puede implicar gastos innecesarios. Dichos gastos son clasificados por (Elmasri, 2011, pág. 26-27) en:

- Inversión alta en hardware, software y capacitación para emplearlo.
- La generalidad que proporciona SMBD para definir y procesar los datos
- Gastos para proporcionar seguridad, control de concurrencia, recuperación y control de integridad

Por lo tanto, de acuerdo con (Elmasri, 2011, pág. 26-27), puede ser más conveniente utilizar los archivos normales en las siguientes circunstancias:

- Cuando se tiene un sistema con limitada capacidad de almacenamiento, aquí un SMBD de propósito general no encajaría.
- Cuando no hay múltiples usuarios que tienen acceso a los datos.
- Cuando se tienen aplicaciones de bases de datos simples, que no esperan cambios en los datos.

2. Modelos de datos

39. ¿Qué es un modelo de datos?

Un modelo de datos en abstracto es la definición lógica de los objetos y sus operadores; juntos estos elementos constituyen una máquina lógica con la cual el usuario interactúa. Estos objetos permiten modelar estructuras de datos (Date, 2004, pág. 44).

Un modelo de datos es un tipo de abstracción de datos que se utiliza para proporcionar una representación conceptual. El modelo de datos utiliza conceptos lógicos, como los objetos, sus propiedades y sus interrelaciones, para que sea más fácil de comprender para la mayoría de los usuarios (Elmasri, 2011, pág. 41).

Un modelo de datos es una colección de herramientas conceptuales para describir los datos, las relaciones de datos, la semántica de los datos y las restricciones de consistencia (Silberschatz, 2006, pág. 39).

Un modelo de datos, es una especificación de las estructuras de datos y reglas que representan al objetivo de los datos (Sherman, 2014, pág. 173).

Un modelo de datos es una notación para la descripción de datos o información (Ullman, 2009, pág. 17).

40. ¿Qué características tiene un modelo de datos?

Las características más importantes de un modelo de datos son:

- **Tiene una estructura.** Los datos se describan a través de estructuras. En las bases de datos, los modelos de datos están en un nivel más alto que las estructuras de datos y son a veces usados como un modelo conceptual para enfatizar la diferencia de niveles.

- **Se pueden manipular los datos.** En los modelos de base de datos, hay un conjunto limitado de operaciones que pueden ser ejecutadas. Generalmente las bases de datos permiten un conjunto limitado de consultas (para obtener información) y modificaciones (cambios en la

base de datos). Esto permite a los programadores describir procedimientos sobre los datos a un alto nivel, mismos que son ejecutados por el SMBD de la manera más eficiente.

- **Establecer restricciones sobre los datos.** Los modelos de base de datos tienen una manera de describir las limitaciones existentes sobre los datos.

(García-Molina, 2009, pág. 18)

2.1 Modelo orientado a objetos

41. ¿En qué consiste el modelo orientado a objetos?

Los modelos orientados a objetos proporcionan un sistema de tipos de datos complejos y orientación a objetos. El modelo contiene además, un camino cómodo para migrar de las bases de datos relacionales a sistemas de bases de datos orientados a objetos.

Incorpora conceptos del paradigma de orientación a objetos -véase para mayor información, el capítulo 9 “Object-Based Databases” de (Silberschatz, 2001, pág. 337)-, como son: la herencia, encapsulación, paso de mensajes, permiten la definición de operaciones o funciones que pueden ser aplicadas a un objeto de un tipo particular.

La herencia permite especificar nuevas clases o tipos que heredan la estructura de otros tipos definidos anteriormente. Esto hace más simple su estructura y facilita la realización de operaciones desde tipos definidos previamente (Silberschatz, 2001. pág. 337).

Un problema en este modelo de bases de datos, es la relación entre objetos. La encapsulación ocasiona que en algunos casos las relaciones entre los objetos no puedan ser expresados explícitamente, en lugar de esto son descritos definiendo métodos apropiados que localizan objetos relacionados. Estas características no funcionan bien con bases de datos complejas que involucran muchas relaciones pues es difícil identificar las relaciones y hacerlas visibles a los usuarios (Elmasri, 2011. pág. 355, 356).

42. ¿Cuáles son sus principales características?

Minimiza el espacio y evita repetir información, debido a que el modelo Orientado a Objetos (OO) usa sólo una tupla por entidad, y esa tupla tiene únicamente componentes para los atributos que tienen sentido para dicha entidad. En este sentido, este tipo de modelo ofrece un menor uso de espacio (Ullman, 2009, pág. 169).

Una de las principales características de las bases de datos OO es el poder que se le da al diseñador para especificar la estructura de objetos complejos y las operaciones que le pueden ser aplicadas a estos objetos (Elmasri, 2011, pág. 316).

Otra de las razones para la creación de bases de datos OO es el incremento del uso de lenguajes de programación orientados a objetos en el desarrollo de software. Las bases de datos OO son diseñadas para que puedan ser integradas directamente con software que está siendo desarrollado con lenguajes de programación OO (Elmasri, 20, pág. 316-317).

43. ¿Qué es un objeto?

El modelo orientado a objetos permite definir a una base de datos en términos de objetos, sus propiedades y sus operaciones. Objetos con la misma estructura y funcionamiento, pertenecerán a una clase y las clases estarán organizadas en jerarquías. Las operaciones de cada clase se especifican en términos de procedimientos predefinidos llamados métodos. Los SMBD relacionales han extendido sus modelos para incorporar conceptos de bases de datos orientadas objetos y sus capacidades (Elmasri, 2011, pág. 50).

Es un principio básico del enfoque: "todo" es un objeto (concepto, abstracción o cosa) (Date, 2004, pág. 818).

Un objeto es una estructura encapsulada que tiene atributos y métodos (Kroenke, 2003, pág. 556).

Las características de un objeto son:

- **Atributos**: son los datos que caracterizan al objeto. Son variables que almacenan datos relacionados al estado de un objeto.

- **Métodos** (usualmente llamados *funciones*): caracterizan el comportamiento del objeto, es decir, son todas las acciones que el objeto puede realizar por sí mismo. Estas operaciones hacen posible que el objeto responda a las solicitudes externas (o que actúe sobre otros objetos). Además, las operaciones están estrechamente ligadas a los atributos, ya que sus acciones pueden depender de, o modificar, los valores de un atributo.

- **Identidad**: El objeto tiene una identidad, que lo distingue de otros objetos, sin considerar su estado. Por lo general, esta identidad se crea mediante un identificador que deriva naturalmente de un problema (por ejemplo: un producto puede estar representado por un código, un automóvil, por un número de modelo, etc.).[11]

44. ¿Qué es una clase? Menciona un ejemplo.

Una clase es un conjunto o colección de entidades; esto incluye cualquiera de los esquemas de grupos de entidades construidos en el Modelo E/R (Elmasri, 2011, pág. 264).

Una clase en el Lenguaje de Modelado Unificado (UML) es similar a un conjunto entidad en el modelo E/R. Una clase se representa por medio de un rectángulo dividido horizontalmente en tres partes. La parte superior contiene el nombre de la clase. En medio tiene los atributos. La parte inferior es para los métodos (Ullman, 2009, pág. 172).

Una clase es una descripción de un conjunto de objetos que comparten los mismos atributos, operaciones, métodos, relaciones y semántica. Las clases son gráficamente representadas por cajas con separaciones para:

- Nombre de la Clase
- Atributos
- Operaciones

[11] Características de un objeto, http://es.ccm.net/contents/412-el-concepto-de-objeto (Consultada el 04/12/2016).

A manera de ejemplo podemos considerar a la clase Persona con los atributos número de nómina, nombre, apellidos y fecha de nacimiento. Por otra parte la operación o método edad que se calcula mediante una operación aritmética.

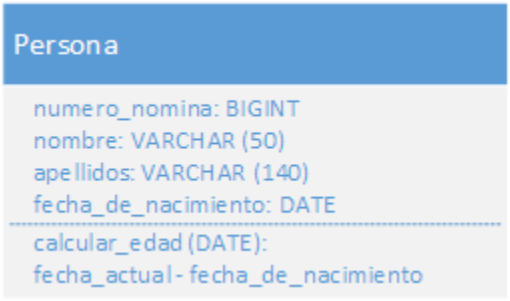


Fig. 6 – La clase persona

(Ordaz-Rosado, 2014, pág. 54)

45. ¿En qué consiste la herencia? Menciona un ejemplo.

La herencia es el mecanismo de implementación mediante el cual elementos más específicos incorporan la estructura y comportamiento de elementos más generales (Rumbaugh, 1999, pág. 299).

Gracias a la herencia es posible especializar o extender la funcionalidad de una clase, derivando de ella nuevas clases. La herencia es siempre transitiva: una clase puede heredar características de superclases que se encuentran muchos niveles más arriba en la jerarquía de herencia. Por ejemplo, si la clase *Perro* es una subclase de la clase *Mamífero*, y la clase *Mamífero* es una subclase de la clase *Animal*, entonces el *Perro* heredará atributos tanto de *Mamífero* como de *Animal*.

2.2 Modelo Relacional

46. ¿En qué consiste el Modelo Relacional?

Una teoría elemental de relaciones para sistemas de datos que proporciona un acceso compartido a bases de datos de gran tamaño. Los problemas tratados en este modelo, están relacionados con la independencia de datos y algunos casos de inconsistencia en los datos. Proporciona una manera de

describir los datos solamente con su estructura natural. Provee una base para un leguaje de datos de alto nivel que permite la independencia máxima entre los programas y su representación en los datos.

Una ventaja de este modelo es que forma una base para tratar la redundancia y la consistencia de las relaciones entre los datos. Además permite una evaluación clara del alcance y limitaciones lógicas de los modelos actuales de datos (Codd, 1970, pág. 377)

Un modelo relacional consiste en un conjunto de relaciones que se construye con una colección de tablas, a cada tabla se le asigna un nombre único. La estructura de cada tabla es similar a las tablas representadas en el modelo E/R. Una fila de la tabla representa una relación entre un conjunto de valores. Ya que una tabla es una colección de relaciones, existe una correspondencia cerrada entre los conceptos de tabla y los conceptos matemáticos de relación (Silberschatz, 2001, pág. 79).

El modelo relacional se basa sólidamente en la lógica y las matemáticas, por lo tanto proporciona un medio ideal para la enseñanza de base de datos. Este modelo consiste en los siguientes aspectos:

- Aspecto estructural: los datos en la base de datos son vistos por los usuarios como tablas.
- Aspectos de integridad: esas tablas satisfacen ciertas restricciones de integridad.
- Aspectos de manipulación: provee de los operadores necesarios para manipular las tablas.

(Date, 2004, pág. 26, 60)

2.3. Modelo de datos semiestructurados

47. ¿En qué consiste el modelo semiestructurado?

El modelo de datos semiestructurado permite establecer objetos individuales del mismo tipo con diferentes conjuntos de atributos (Silberschatz, 2006, pág. 8).

A diferencia de otros modelos, el modelo semiestructurado permite una colección de nodos, cada uno conteniendo datos, posiblemente con diferentes esquemas. El nodo en sí contiene información acerca de la estructura de sus contenidos. Las bases de datos semiestructuradas son especialmente

útiles cuando se deben integrar bases de datos existentes que tengan distintos esquemas (Ricardo, 2009, pág. 73).

De acuerdo con (Silberschatz, 2001, pág. 11), el aspecto más interesante de los datos semiestructurados es que, aunque no sigan una regla estricta de formato, mantienen una regularidad suficiente como para que se pueda extraer alguna información interesante. Ejemplos de este tipo de datos son:

- Las páginas web, que siguen ciertas pautas comunes y albergan contenido en el HTML y metadatos entre las etiquetas.
- Los documentos electrónicos que tienen etiquetas -"tags"- que brindan información acerca de las diferentes secciones que los conforman.

Dos notaciones muy utilizadas para modelar datos semi-estrucuturados son XML y JSON.

48. ¿Qué es XML?

El Lenguaje de Marcas Extensible (XML, eXtensible Markup Language) surge debido a la necesidad de definir y manejar la estructura interna de los documentos HTML. XML está basado en un estándar anterior llamado Standard Generalized Markup Language (SGML). XML fue definido por el WWW Consortium (W3C) y originalmente surge como un lenguaje de marcado para documentos, no como un lenguaje de base de datos.

El término marcado se refiere a cualquier cosa en un documento que no está destinado a formar parte de la salida impresa. Las marcas que incluye HTML, SGML, XML toman la forma de etiquetas <>. Las etiquetas delimitan una parte del documento, es por ello que se requiere indicar el inicio <etiqueta> y final </ etiqueta> de éstas.

XML no prescribe el conjunto de etiquetas restringido, el conjunto puede ser especializado según sea necesario. Esta característica es la clave para XML importante papel en la representación de datos y el intercambio (Silberschatz, 2001, pág. 361-364).

Supongamos que se quiere representar a la estructura de estas notas de clase en XML:

```
<NotasFBD>
      <Elemento>
            <Pregunta>
                  <IDPregunta>Numero de la pregunta</IDPregunta>
                  <Texto>Texto de la pregunta</Texto>
            </Pregunta>
            <Respuesta>
                  <Nombre>Nombre del alumno que la respondio</Nombre>
                  <Contenido>Texto de la respuesta</Contenido>
                  <Referencia>
                        <Autor>Autor del libro</Autor>
                        <Anyo>Anyo de publicacion del libro</Anyo>
                        <Pagina>Numero de pagina</Pagina>
                  </Referencia>
            </Respuesta>
      </Elemento>
</NotasFBD>
```

49. ¿Qué es JSON?[12]

La Notación de Objetos de JavaScript (JSON, JavaScript Object Notation) es un formato ligero de intercambio de datos. Que por una parte es fácil de leer y escribir para humanos y, por otra, fácil de interpretar y generar para máquinas. Es independiente del lenguaje pero usa convenciones conocidas por programadores de la familia de lenguajes de C, C++, Java, Python, entre otros. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

Está constituido por dos estructuras:

- Una colección de pares de nombre/valor (objeto en otros lenguajes)
- Una lista ordenada de nombres (arreglos, vectores o listas en otros lenguajes)

[12] JSON, http://www.json.org/json-es.html (Consultada el 05/12/2016).

Consideremos el siguiente ejemplo:

```
{
  "NotasFBD": {
    "Elemento": {
      "Pregunta": {
        "IDPregunta": "Numero de la pregunta",
        "Texto": "Texto de la pregunta"
      },
      "Respuesta": {
        "Nombre": "Nombre del alumno que la respondio",
        "Contenido": "Texto de la respuesta",
        "Referencia": {
          "Autor": "Autor del libro",
          "Anyo": "Anyo de publicacion del libro",
          "Pagina": "Numero de pagina"
        }
      }
    }
  }
}
```

2.4. Modelo de datos de red

50. ¿En qué consiste el modelo de red?

Las redes constituyen una manera natural de representar las interrelaciones entre los objetos. Dentro del marco de los modelos de datos, los nodos pueden considerarse como tipos de registros de los datos y las aristas pueden considerarse como la representación de las relaciones 1:1 o 1:*.

Solamente hay dos estructuras de datos fundamentales en el modelo en red, los tipos de registros y los conjuntos:

- Los tipos de registros se definen de manera usual como colecciones de elementos de los datos lógicamente relacionados.
- Un conjunto en el modelo de red expresa una relación 1:1 o 1:* entre dos tipos de registros.

Es importante notar que el modelo de redes permite solamente relaciones 1:1 y 1:* (Hansen & Hansen, 1997, pág. 489).

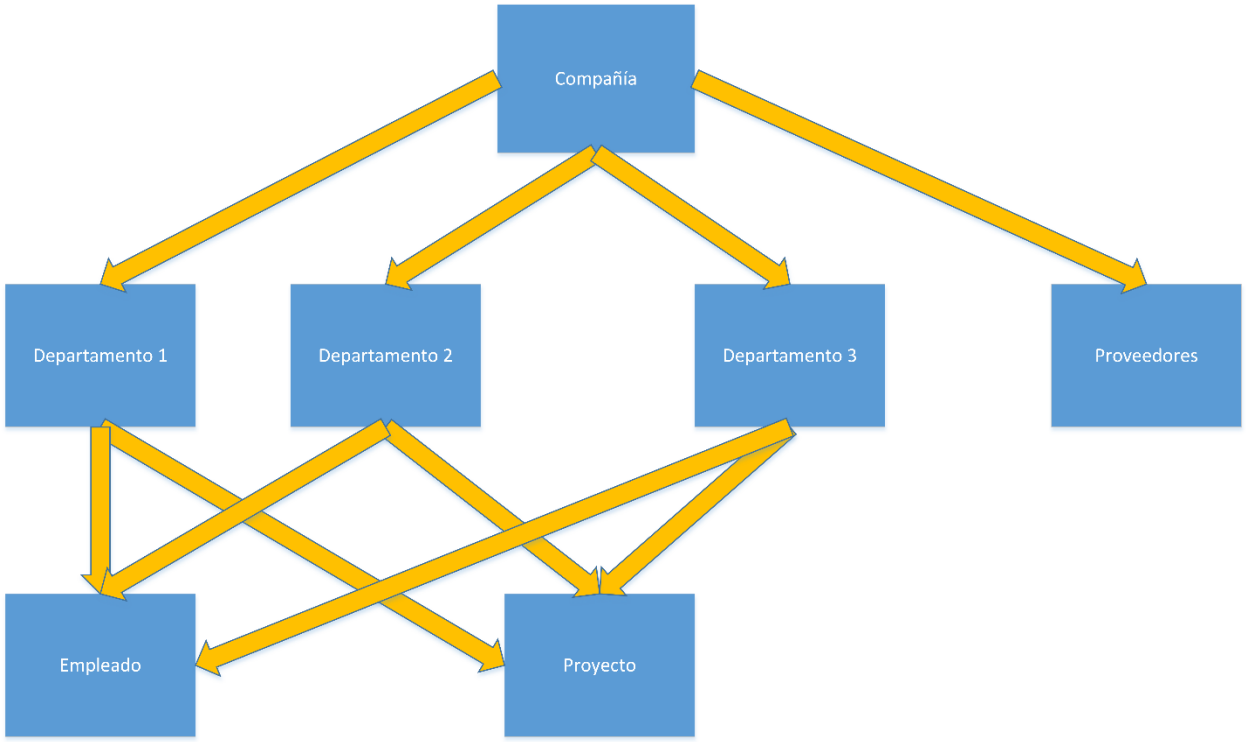


Fig. 7 – Modelo de red

2.5. Modelo de datos jerárquico.

51. ¿En qué consiste el modelo jerárquico?

Esta estructura conserva similitudes con el modelo en red, con la diferencia de que el modelo jerárquico es organizado por una estructura de árbol con raíz y no como una gráfica arbitraria.

Cada registro es una colección de atributos, los cuales contienen solo un valor de dato y la arista es una relación entre exactamente dos registros. Un diagrama de estructura de árbol es el esquema utilizado para representar bases de datos de tipo jerárquico y consiste de dos partes:

- Cajas: las cuales corresponden a los tipos de registros
- Aristas: que corresponden a las relaciones entre tipos de registros.

Adicionalmente se exige que no existan ciclos en el diagrama subyacente, las relaciones que existan entre registros deben ser de tipo 1:1 o 1:* entre el padre y el hijo.

(Elmasri, 2011, Apéndice E, pág. 1-4)

(Silberschatz, 2006, apéndice B, pág. 2 y 3)

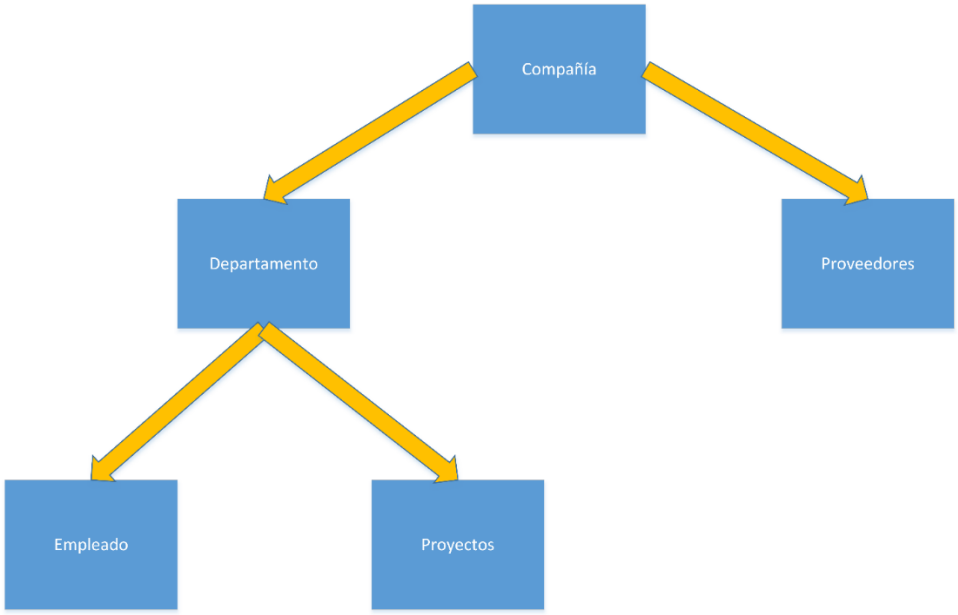


Fig. 8 – Modelo jerárquico

3. Modelo Entidad/Relación

52. ¿En qué consiste el modelo E/R?

El modelo E/R puede ser utilizado como base para la unificación de distintos modelos: el modelo de red, el modelo relacional y el modelo de conjuntos de entidades. Este modelo describe los datos mediante entidades, relaciones y atributos. Existen diferentes tipos de cada uno de estos elementos, es por ello que en las siguientes subsecciones se presentan a detalle (Chen, 1976, pág. 1).

53. ¿Quién y cuándo lo propuso?

El modelo de Entidad-Relación fue propuesto por Peter Pin-Shan Chen (1947-) en 1976 en el artículo The Entity-Relationship Model-Toward a Unified View of Data (Chen, 1976).

3.1. Elementos del modelo E/R

54. ¿Cuáles son los principales elementos del modelo E/R?

El modelo de datos E/R utiliza tres elementos básicos:

- Entidades
- Relaciones
- Atributos

(Silberschatz, 2011, pág. 36).

55. ¿Qué es y cómo se representa una entidad en el modelo E/R?

Una entidad es un objeto abstracto de algún tipo. En algunos casos se asemeja a un “objeto” en el sentido de la programación orientada a objetos (Ullman, 2009, pág. 126).

Una entidad puede ser un objeto con una existencia física (persona, auto, etc.) o un objeto con una existencia conceptual (una compañía, un trabajo, etc.) (Elmasri, 2011, pág. 203).

Es una persona, lugar, cosa, evento o concepto sobre el que una empresa mantiene los datos (Sherman, 2014, pág. 179).

Una entidad es una "cosa" u "objeto" en el mundo real que es distinguible de otros objetos. Éstas se describen en una base de datos a través de un conjunto de atributos. En el diagrama E/R, una entidad se representa mediante un rectángulo (Silberschatz, 2001, pág. 16)

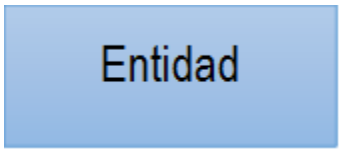


Fig. 9 – Entidad

56. ¿Qué es y cómo se representa un atributo en este modelo?

Un atributo representa algunas propiedades de interés que describen una entidad, tales como nombre del empleado o salario (Elmasri, 2011, pág. 31).

De acuerdo con (Date, 2004, pág. 415-416) y (Silberschatz, 2006, pág. 208-209), los atributos son propiedades descriptivas que posee cada miembro de una entidad.

Los atributos representan las características de las entidades. En los diagramas E/R los atributos son representados mediante elipses (Silberschatz, 2001, pág. 8).

Fig. 10 – Atributo

57. ¿Qué es y cómo se representa una relación en este modelo?

Una relación entre dos o más entidades representa una asociación entre las entidades (Elmasri, 2011, pág. 31).

Las relaciones son conexiones entre dos o más juegos de entidad (Ullman, 2009, pág. 127).

Una relación es una asociación entre varias entidades (Silberschatz, 2006, pág. 16).

Las relaciones muestran cómo las entidades se relacionan una con la otra, son los eslabones lógicos entre las entidades que representan las reglas de gestión e interacción (Sherman, 2014, pág. 180).

Una relación une entidades (Date, 2004, pág. 13).

Fig. 11 – Relación

58. ¿Qué es el dominio de un atributo?

El dominio es el conjunto de valores que pueden ser asignados al atributo de una entidad (Elmasri, 2011, pág. 59).

Para cada atributo, el conjunto de valores permitidos es llamado dominio (Silberschatz, 2006, pág. 28).

59. ¿Qué es la aridad de una relación?

Es el grado del conjunto de relaciones, esto es, el número de entidades que participan en una relación. Un conjunto de relación es de grado 2, si la relación es binaria y de grado 3, si es ternaria (Silberschatz, 2006, pág. 33).

60. ¿Qué es la aridad de una entidad?

Es el número de atributos en una entidad (Silberschatz, 2006, pág. 92).

El grado o aridad de una entidad es el número de atributos que la componen (Elmasri, 2011, pág. 62).

61. ¿Qué es la cardinalidad?

Expresa el número de entidades a las cuales otra entidad puede ser asociada, vía una relación (Silberschatz, 2001, pág. 8).

Especifica todas las combinaciones posibles de valores de los dominios subyacentes (Elmasri, 2011, pág. 63).

62. ¿Qué es un esquema?

La estructura general de la base de datos se llama esquema, este se diseña y especifica al momento de la creación de la base de datos. En la mayoría de los modelos de datos se tienen como conveniencia representar estos esquemas como diagramas (Silberschatz, 2006, pág. 7; Elmasri, 2011, pág. 32).

3.2. Restricciones del modelo E/R

63. ¿Qué es una súper-llave (SK)?

Una súper-llave (SK) especifica una restricción de unicidad en la que no existen dos tuplas distintas que puedan tener el mismo valor para la SK. Toda relación tiene al menos una SK por defecto -el conjunto de todos sus atributos- (Elmasri, 2011, pág. 69).

Una SK es un conjunto de uno o más atributos que, tomados en conjunto, nos permiten identificar de forma única una tupla en la relación (Silberschatz, 2006, pág. 43).

Al conjunto de atributos que contiene una llave, se le conoce como SK. Por lo tanto, cada llave es una SK. Sin embargo, algunas llaves no son SK mínimas. Es decir, se debe tener en cuenta que todas las SK satisfacen la primera condición de una llave: que ésta determina funcionalmente a todos los demás atributos de la relación. Sin embargo, una SK no necesariamente tendrá que cumplir la segunda condición: minimalidad (Ullman, 2009, pág. 71).

64. ¿Qué es una llave candidata (CK)?

Una llave candidata (CK) es una SK que no tenga ningún subconjunto propio que a su vez sea una SK, es decir, es un conjunto de atributos que permite identificar de manera única a una entidad y que a su vez no contiene un conjunto más pequeño que siga determinando de manera única a la entidad (Silberschatz, 2001, pág. 35).

De acuerdo con (Elmasri, 2011, pág. 69), una CK es un conjunto de atributos que cumple:

1. Dos tuplas distintas no pueden tener valores idénticos para todos los atributos de la llave.

2. Es una SK mínima, es decir, una SK a la cual no se le pueden remover atributos de tal modo que se mantenga la restricción de unicidad de la condición 1.

Por otra parte, (Ullman, 2009, pág. 70) define que un conjunto de uno o más atributos $\{A_1, A_2, \dots, A_n\}$ es una CK para una relación R si:

- Estos atributos determinan a todos los otros atributos de la relación. Es decir, es imposible para dos distintas tuplas de R el concordar en todos los atributos $A_1, A_2, \dots, A_n$.
- Ningún subconjunto propio de $\{A_1, A_2, \dots, A_n\}$ determina a todos los otros atributos de R, es decir, una CK debe ser mínima.

Un conjunto de atributos K de la relación R. Decimos que K es una CK para R si y sólo si satisface las siguientes dos propiedades:

- **Unicidad:** ningún valor permitido de R contiene dos tuplas distintas con el mismo valor para K.
- **Irreductibilidad:** ningún subconjunto propio de K posee la propiedad de unicidad.

(Date, 2004, pág. 269)

65. ¿Qué es una llave primaria (PK)?

El término de llave primaria (PK) se utiliza para denotar a una de las CK que será elegida (por el diseñador de la base de datos) como la principal representante para la identificación de las entidades dentro de un conjunto de entidades (Silberschatz, 2006, pág. 43).

La PK es el atributo o grupo de atributos que es elegido para identificar de manera única cada caso de la entidad. Si existe una lista de CK alguna debe de designarse como llave primaria (Sherman, 2014, pág. 187).

Si se tiene más de una llave candidata, alguna de ellas se designara arbitrariamente como PK. En una base de datos cada relación debe tener una PK, de lo contrario toda la relación se tratará como una SK (Elmasri, 2011, pág. 519).

Algunas veces la relación tiene más de una llave, si esto sucede se deberá designar a alguna de estas llaves como la PK (Ullman, 2009, pág. 70).

Suponiendo que tenemos más de una CK el modelo relacional requiere que exactamente una de esas llaves sea elegida como PK (Date, 2004, pág. 271).

66. ¿Qué es una llave foránea (FK)

Una llave foránea (FK) es un conjunto de atributos en una relación que constituyen una llave en alguna otra relación y es usada para indicar enlaces lógicos entre relaciones (Hansen & Hansen, 1997, pág. 143).

Una FK es un atributo o combinación de atributos de una relación que no es la llave primaria de dicha relación, pero que es la llave primaria de alguna otra relación. Son muy importantes en el modelo relacional, porque se usan para representar conexiones lógicas entre relaciones (Ricardo, 2009, pág. 130).

67. ¿Qué es una entidad débil?

Una entidad débil es una entidad cuyos atributos no la identifican completamente, sino que sólo la identifican de forma parcial. Esta entidad debe participar en una relación que ayude a identificarla (Date, 2004, pág. 419- 420)

Los tipos de entidad que no tienen atributos llave propios, se les denomina entidades débiles (Elmasri, 2007, pág. 67).

Una entidad débil es aquella cuya existencia depende de otra entidad llamada entidad identificadora; en vez de asociar una llave primaria con una entidad débil, se usa la entidad identificadora junto con atributos extra llamados discriminador para hacer única la identificación de esa entidad débil (Silberschatz, 2010, pág. 74).

68. ¿Cómo se representa visualmente una entidad débil?

Cada entidad se muestra como un rectángulo que contiene el nombre de la entidad. Para la entidad débil los bordes del rectángulo son dobles.

Fig. 12 – Entidad

(Date, 2004, pág. 418-419)

69. ¿Qué tipos de atributos existen?

- **Nulos:** atributo del cual se desconoce su valor en el momento actual o no está definido para una instancia particular. Por ejemplo, el *número interior* de un domicilio aplica solamente para departamentos o casas en fraccionamientos, y no para casas en avenidas, las cuales no tendrían el atributo *número interior* (Ricardo, 2009, pág. 89).
- **Simples:** aquel atributo que no se divide en subpartes, es decir aquellos que sólo tienen un valor para una entidad particular. Por ejemplo la edad de una persona, no se puede descomponer en más atributos (Silberschatz, 2006, pág. 57)

Fig. 13 – Atributo simple

- **Compuestos:** a diferencia del atributo simple, el compuesto es aquel que se puede descomponer en elementos más pequeños. Como lo es la dirección, la cual la podemos descomponer en *avenida*, *número interior*, *número exterior*, *delegación* y *código postal* (Ricardo, 2009, pág. 89).

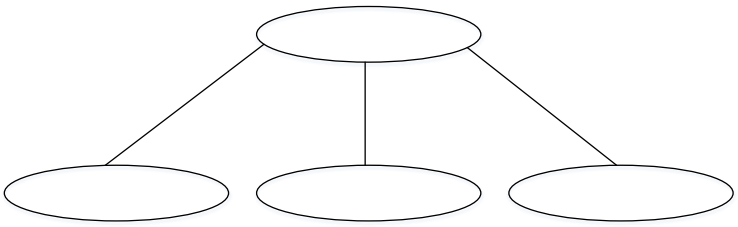


Fig. 14 – Atributo compuesto

- **Univaluados:** son atributos que sólo pueden tomar un valor para una entidad particular. La delegación en una dirección puede ser ejemplo de estos atributos, pues una persona no vive en más de una delegación a la vez (Silberschatz, 2006, pág. 57).

Fig. 15 – Atributo univaluado

- **Multivaluados:** son aquellos atributos que pueden tener valores múltiples para una instancia de entidad, como el correo electrónico para un estudiante, pues éste puede tener más de una cuenta de correo (Ricardo, 2009, pág. 89).

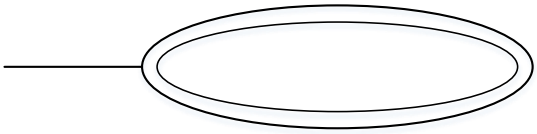


Fig. 16 – Atributo multivaluado

- **Derivados:** atributos cuyo valor puede ser calculado a través de otros. Como ejemplo está la edad de una persona, que podemos calcular si tenemos la fecha de nacimiento (Ricardo, 2009, pág. 89).

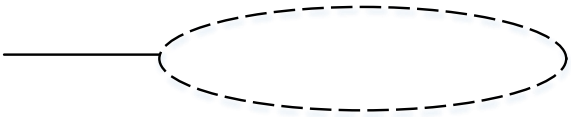


Fig. 17 – Atributo derivado

(Silberschatz, 2001, pág. 57) y (Elmasri, 2011, pág. 223)

70. ¿Qué diferencia hay entre un valor nulo 'desconocido' y un nulo 'no aplicable'?

La diferencia entre estos dos tipos de nulos es que mientras el nulo desconocido es debido a que el valor puede existir pero se desconoce, mientras que el nulo no aplicable se da debido a que el atributo en cuestión no aplica para la entidad (Date, 2004, pág. 577).

De acuerdo con (Elmasri, 2011, p 65), se tiene que:

- **Nulo desconocido:** el valor existe pero no está disponible (valor desconocido).
- **Nulo no aplicable:** el atributo no es aplicable en la tupla, también conocido como valor indefinido.

71. ¿Qué tipo de cardinalidades existen?

De acuerdo con (Elmasri, 2007, pág. 233,289-291) y (Ricardo, 2009, pág. 96-97), la cardinalidad de una relación es el número de entidades a las que otra entidad puede mapearse bajo dicha relación. Sean *X* y *Y* conjuntos de entidades y R una relación binaria de *X* y *Y*, entonces tenemos los siguientes tipos de cardinalidades:

- **Uno a uno (1:1):** una relación R de *X* a *Y* es uno a uno si cada tupla en *X* se asocia con cuando mucho una tupla en *Y* e, inversamente, cada tupla en *Y* se asocia con cuando mucho una tupla en *X*.

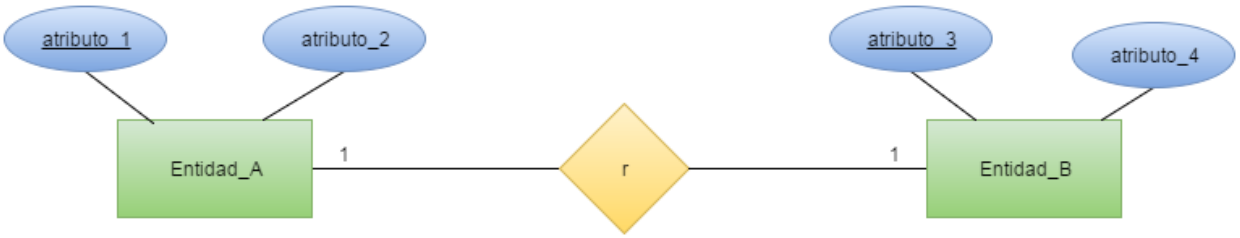


Fig. 18 – Cardinalidad uno a uno

- **Uno a muchos (1:*):** una relación *R* de *X* a *Y* es uno a muchos si cada tupla en *X* se puede asociar con muchas tuplas en *Y*, pero cada tupla en *Y* se asocia con cuando mucho una tupla en *X*.

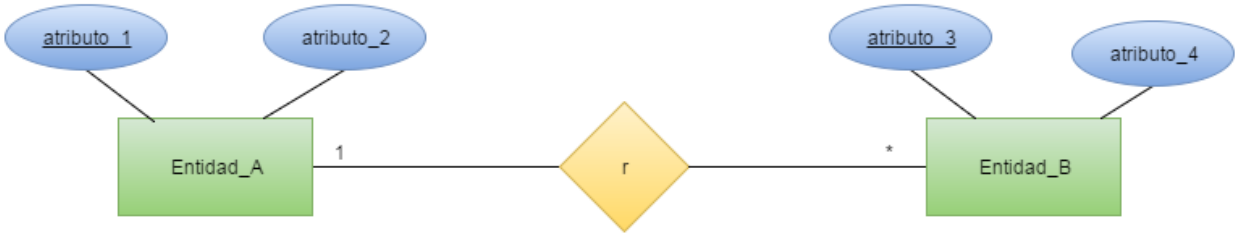


Fig. 19 – Cardinalidad uno a muchos

- **Muchos a muchos (*:*):** una relación *R* de *X* a *Y* es muchos a muchos si cada tupla en *X* se puede asociar con muchas tuplas en *Y* y cada tupla en *Y* se puede asociar con muchas tuplas a *X*.

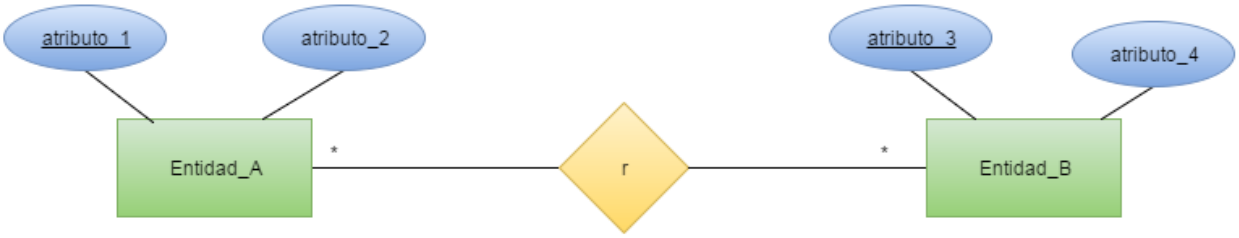


Fig. 20 – Cardinalidad muchos a muchos

3.3. Modelo E/R extendido

72. ¿En qué consiste la especialización? Menciona un ejemplo

La especialización es el proceso de definición de un conjunto de subclases de un tipo de entidad; este tipo de entidad se llama la superclase de la especialización. El conjunto de subclases que forma una especialización se define sobre la base de alguna característica distintiva de las entidades en la superclase.

Por ejemplo, el conjunto de subclases {secretaria, Ingeniero, técnico} es una especialización de la superclase Empleado que distingue entre las entidades empleados en función del tipo de trabajo de cada entidad empleado (Elmasri, 2011, pág. 248).

Es el proceso de definición de un conjunto de sub-clases de una súper-clase. La especialización es el refinamiento de arriba hacia abajo en (súper-) clases y sub-clases. El conjunto de sub-clases se basa en alguna característica distintiva de la súper-clase. Puede haber varias especializaciones de un tipo de entidad en base a diferentes características distintivas[13].

[13] Modelo Entidad-Relación extendido, http://www.slideshare.net/gerardomo/modelo-de-entidad-relacin-extendido (Consultada el 05/12/2016).

73. ¿En qué consiste la generalización? Menciona un ejemplo.

La generalización es una relación de contención que existe entre un conjunto entidad de más alto nivel y uno o más conjuntos de entidades de nivel más bajo. Para efectos prácticos, la generalización es una simple inversión de la especialización (Silberschatz, 2001, pág. 59).

Podemos pensar a la generalización como un proceso inverso de la abstracción en el que suprimimos las diferencias entre varios tipos de entidad, identificando sus características comunes, y generalizando en una sola superclase en la que los tipos de entidad originales son subclases especiales (Elmasri, 2011, pág. 250).

El reverso de la especialización es la generalización. Varias clases con características comunes se generalizan en una súper-clase. Por ejemplo, la entidad tipos de coches y camiones comparten los mismos atributos[14].

Por ejemplo, considere la entidad *Jugador* como una generalización de *Runner*, *Quarterback*, *Punter*, *Kicker* y *Receiver*. Nótese que el mismo diagrama puede leerse de manera inversa, es decir, ver a *Runner, Quarterback*, *Punter*, *Kicker* y *Receiver* como las especializaciones de *Jugador*

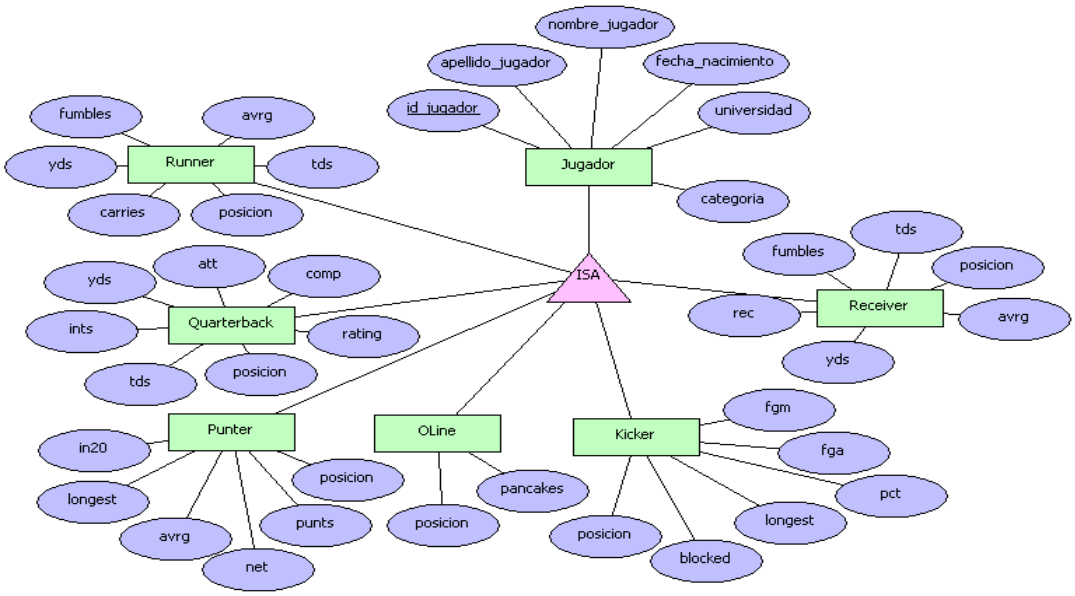


Fig. 21 – Ejemplo de generalización

(Ordaz-Rosado, 2014, pág. 57)

[14] Modelo Entidad-Relación extendido, http://www.slideshare.net/gerardomo/modelo-de-entidad-relacin-extendido (Consultada el 05/12/2016).

74. ¿En qué consiste la jerarquía?

En una jerarquía, un conjunto de entidades puede estar involucrado como una entidad de nivel inferior establecida en sólo una relación; es decir, los conjuntos de entidades sólo tienen una herencia única. Si un conjunto de entidades es una entidad de nivel inferior establecida en más de una relación, entonces el conjunto tiene herencia múltiple, y la estructura resultante se dice que es una reja. (Silberschatz, 2006, pág. 231)

4. Modelo Relacional

75. ¿En qué consiste el modelo relacional?

Un modelo relacional consiste en un conjunto de relaciones que se construye con una colección de tablas, a cada tabla se le asigna un nombre único. La estructura de cada tabla es similar a las tablas representadas en el modelo E/R. Una fila de la tabla representa una relación entre un conjunto de valores. Ya que una tabla es una colección de relaciones, existe una correspondencia cerrada entre los conceptos de tabla y los conceptos matemáticos de relación (Silberschatz, 2001, pág. 79).

76. ¿Quién y cuándo lo propuso?

Edgar Frank Codd (1923-2003) propuso en 1970 el modelo relacional en el artículo *Relational Model of Data Large Shared Data Banks* (Codd, 1970).

77. ¿Cuáles son los principales elementos del modelo relacional?

Un esquema de relación R, denotado por $R(A_1, A_2, ..., A_n)$, se compone de un nombre de relación R y una lista de atributos, $A_1, A_2, ..., A_n$. Cada atributo A_i es el nombre de una función que desempeña un dominio *D* en el esquema de la relación R. *D* es el dominio de A_i y se denota por $dom(A_i)$.

Una relación $R(A_1, A_2, …, A_n)$ es un conjunto de n tuplas $\{t_1, t_2, ... , t_m\}$. Cada n-tupla es una lista de valores, donde cada valor es un elemento de $dom(A_i)$ (Elmasri, 2011, pág. 61-62).

4.1 Estructura y restricciones del modelo relacional

78. ¿Cómo se define el concepto relación en este modelo?

El modelo relacional se basa en el concepto de relación, que se representa físicamente como una tabla o arreglo bidimensional, las tablas se usan para contener información acerca de los objetos a representar en la base de datos. Una relación se representa como una tabla bidimensional en la que las filas de la tabla corresponden a registros individuales y las columnas corresponden a atributos (Ricardo, 2009, pág. 125).

El modelo relacional representa la base de datos como una colección de relaciones. De manera informal, cada relación se asemeja a una tabla de valores o, en cierta medida, un archivo plano de registros. Es llamado un archivo plano, ya que cada registro tiene una estructura lineal o plana simple (Elmasri, 2011, pág. 61).

79. ¿Cuáles son las restricciones del modelo relacional?

El modelo relacional requiere que:

- El orden de las tuplas no aporta mayor información.
- El orden de los atributos no aporta mayor información.
- Toda tupla que vive en la relación es única.
- El valor de cada atributo es atómico.

4.2. Mapeo del esquema conceptual al esquema relacional

80. ¿Qué es UML?

El Lenguaje de Modelado Unificado (UML por sus siglas en inglés) es un lenguaje estándar desarrollado por el Object Management Group (OMG) para crear especificaciones de los diversos componentes de un sistema de software (Silberschatz, 2006, pág. 251-252).

UML es una notación para la especificación de sistemas de software que se ha propuesto como un estándar para el modelado de objetos conceptuales (Elmasri, 2011, pág. 226).

81. Describe 5 tipos de diagramas UML

Entre los principales diagramas de UML tenemos:

- **Diagramas de clases:** capturan la estructura estática del sistema y actúan como base para otros modelos. Muestran clases, interfaces, colaboraciones, dependencias, generalizaciones, asociaciones y otras relaciones. Son útiles para modelar el esquema conceptual de la base de datos (Elmasri, 2011, pág. 330).
- **Diagramas de objetos:** muestran un conjunto de objetos individuales y sus relaciones, se conocen como diagramas de instancia. Dan una visión estática de un sistema en un

momento determinado y normalmente se utilizan para probar los diagramas de clases de exactitud (Elmasri, 2011, pág. 330).

- **Diagramas de componentes:** ilustran las organizaciones y dependencias entre los componentes de software. Un diagrama de componentes normalmente consta de componentes, interfaces y relaciones de dependencia. Para bases de datos, los diagramas de componentes representan los datos almacenados tales como espacios de tabla o particiones. Las interfaces se refieren a las aplicaciones que utilizan los datos almacenados (Elmasri, 2011, pág. 330).
- **Diagramas de casos de uso:** se utilizan para modelar las interacciones funcionales entre los usuarios y el sistema. Un escenario de un caso de uso es una secuencia de pasos que describen una interacción entre un usuario y un sistema. Un diagrama de casos de uso muestra a los actores que interactúan con cada caso de uso (Elmasri, 2011, pág. 330).
- **Diagramas de actividad:** estos presentan una visión dinámica del sistema mediante el modelado del flujo de control de una actividad a otra. Son considerados como diagramas de flujo con estados. Una actividad es un estado de hacer algo, lo que podría ser un proceso en el mundo real o una operación en un objeto o una clase sobre la base de datos. Por lo general, se utilizan para modelar las operaciones de flujo de trabajo y de negocio interno para una aplicación (Elmasri, 2011, pág. 330).

82. ¿Qué es una clase?

La clase en un diagrama UML es similar a un tipo de entidad del modelo E/R que se define como una colección de entidades que tienen los mismos atributos, se muestra como un cuadro dividido en tres partes (Elmasri, 2011, pág. 227, 207).

La clase en el diagrama UML se muestra como un conjunto de entidades del mismo tipo que comparten las mismas propiedades o atributos y la representa como un cuadro (Silberschatz, 2006, pág. 27,68).

Una clase en UML es similar a un conjunto de entidades (una colección de entidades similares) en el modelo E/R. La notación para una clase es un cuadro que se divide en tres partes (Ullman, 2009, pág. 126,172).

83. ¿Qué elementos conforman una clase?

Una clase está dividida en tres partes. En la parte superior se encuentra el nombre de la clase. La parte media contiene los atributos, los cuales son como variables de instancia de una clase. La parte inferior contiene los métodos (Ullman, 2009, pág. 214).

Una clase incluye 3 secciones, la parte superior nos da el nombre de la clase; la sección media incluye los atributos y la última sección incluye las operaciones o métodos que pueden ser aplicados a objetos individuales de la clase (Elmasri, 2011, pág. 227).

84. ¿Cómo se transforma una entidad y sus atributos a una clase?

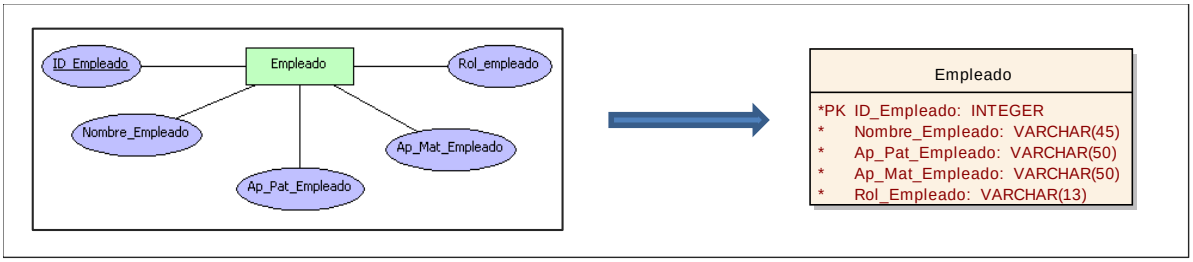


Fig. 22 – Transformación de una entidad y sus atributos a clase

(Ordaz-Rosado, 2014, Pág. 54)

85. ¿Cómo se transforma una relación 1:1 de E/R a Clases?

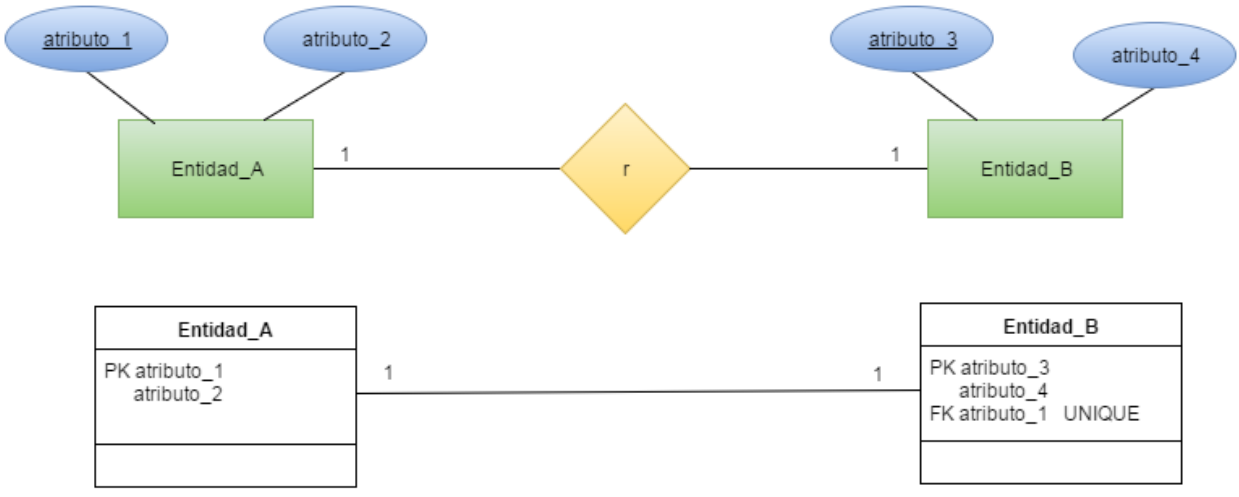


Fig. 23 – Transformación de una relación uno a uno en E/R a clases

86. ¿Cómo se transforma una relación 1:* de E/R a Clases?

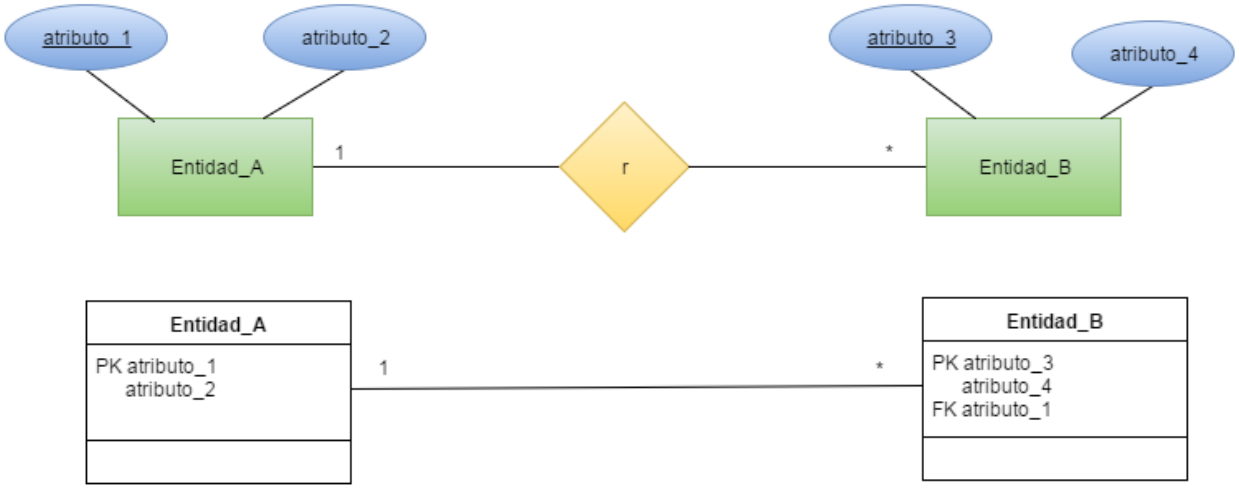


Fig. 24 – Transformación de una relación uno a muchos en E/R a clases

87. ¿Cómo se transforma una relación de *:* de E/R a Clases?

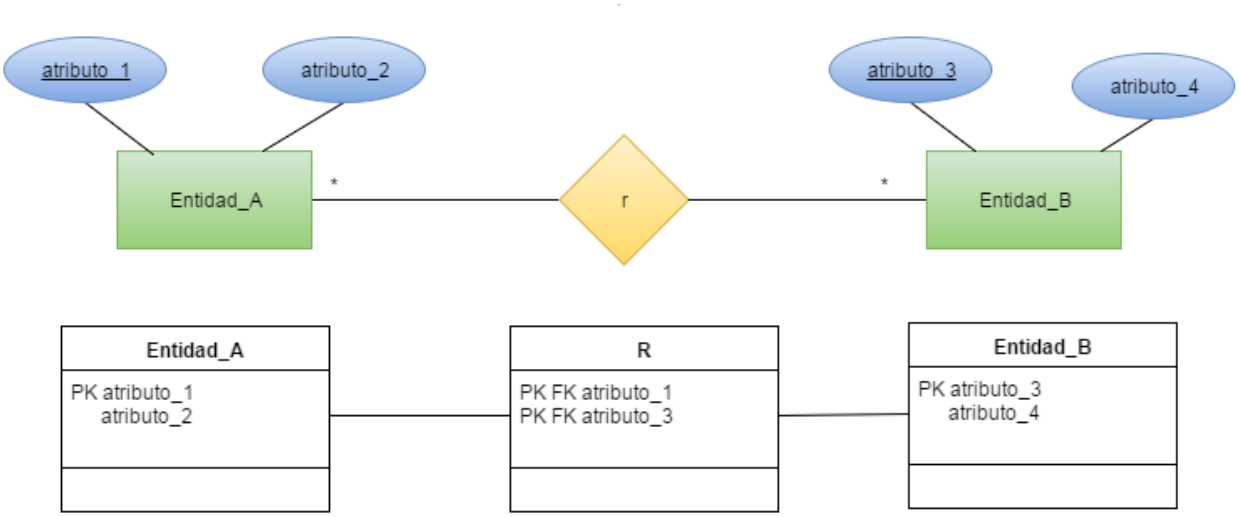


Fig. 25 – Transformación de una relación muchos a muchos en E/R a clases

88. ¿Cómo se transforma una relación con atributos de E/R a clases?

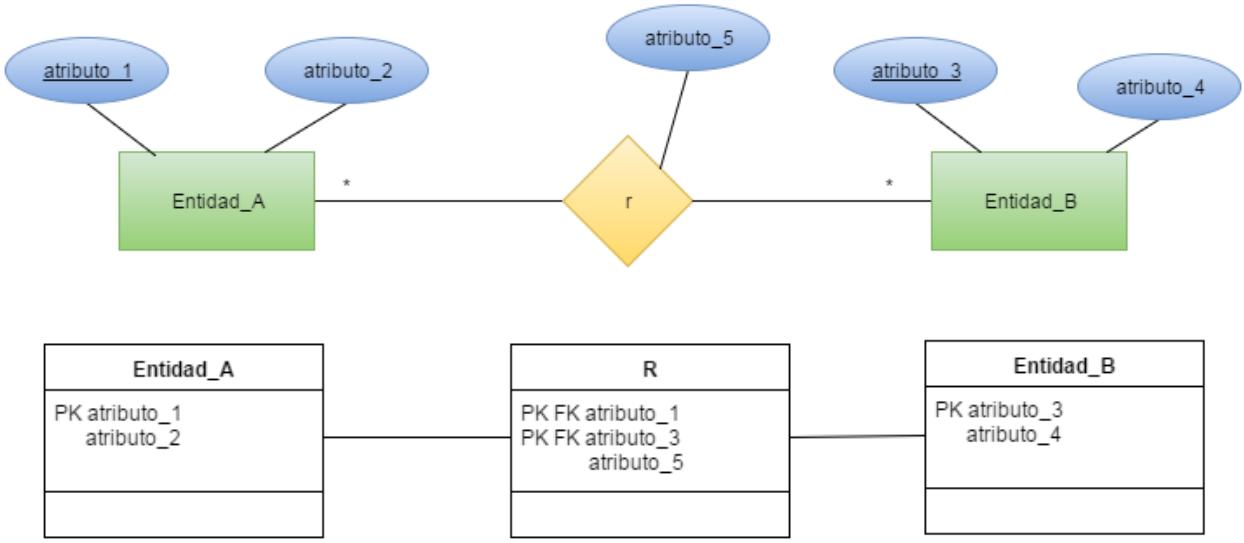


Fig. 26 – Transformación de una relación con atributos en E/R a clases

89. ¿Cómo se transforma una especialización/generalización de E/R a Clases?

No siempre es fácil elegir el diseño conceptual más apropiado para una base de datos. El diseño de una base de datos debe ser considerado un proceso de continuo refinamiento hasta que el diseño se convierta en algo adecuado. Los siguientes puntos son una guía para ayudar al proceso de especialización/generalización entre un diagrama E/R y uno de Clases:

- Muchas especializaciones y subclases pueden ser definidas para hacer el modelo conceptual más preciso. Sin embargo, el inconveniente es que el diseño se volverá desordenado. Es importante sólo representar sólo esas subclases que son necesarias para prevenir el desorden en el esquema conceptual.

- Si una subclase tiene atributos locales muy específicos y relaciones no específicas, pueden ser mezcladas o incluidas en la súper clase. Los atributos específicos pueden tener valores para entidades que no son miembros de la subclase. Un tipo de atributo puede especificar si una entidad es miembro de la subclase.

- Similarmente, si una de las subclases de una especialización/generalización tiene algunos atributos específicos y relaciones no específicas, éstas pueden ser incluidas en la superclase

y reemplazadas con una o más atributos-tipo que especifiquen la subclases a las que cada entidad pertenece.

- Los tipos de unión y categorías pueden generalmente evitarse a menos que la situación garantice este tipo de construcción.

- La elección de disjuntos/superpuestos y restricciones totales/parciales sobre la especialización/generalización, son manejadas por las reglas en el esquema que se está modelando. Si los requisitos no indican alguna restricción particular, el valor por defecto es generalmente la superposición parcial, ya que no especifica ninguna restricción como miembro de la subclase.

(Elmasri, 2011. Pág. 263)

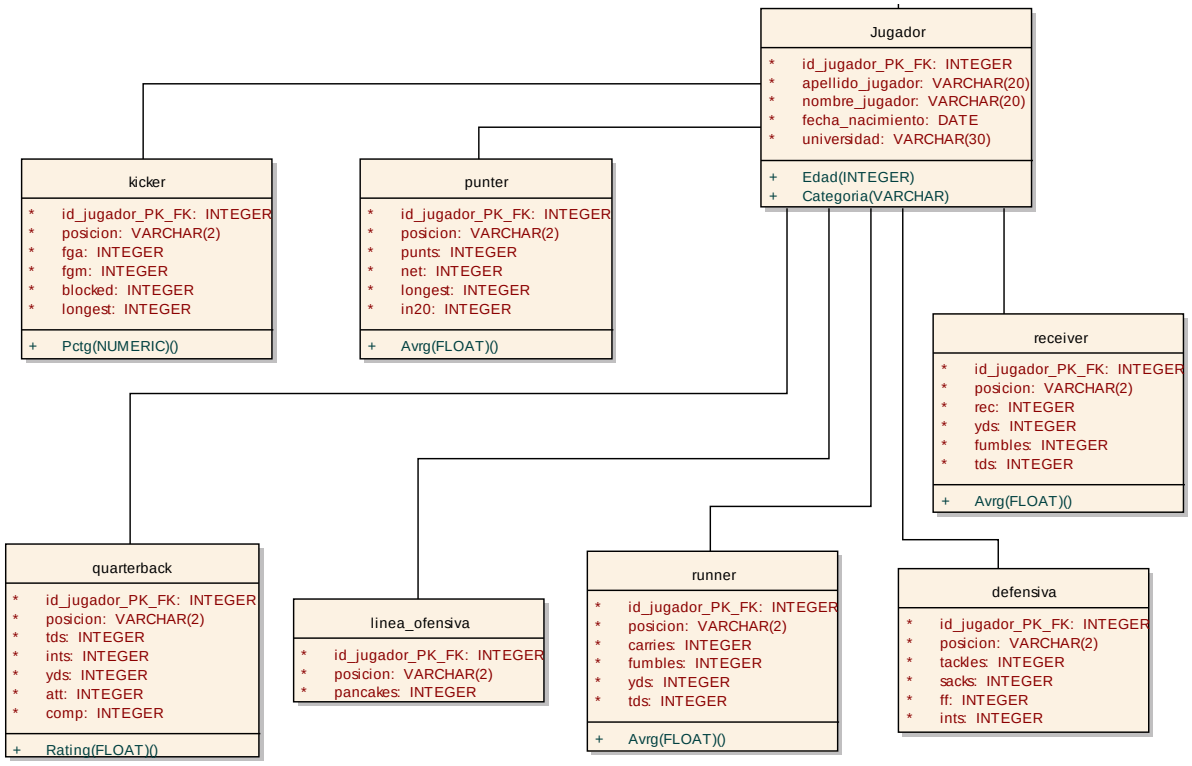


Fig. 27 – Representación de una especialización/generalización a clases

(Ordaz-Rosado, 2014, pág. 43)

4.3 Álgebra relacional

90. ¿Qué tipos de lenguajes de consulta existen?

El lenguaje de consulta es un lenguaje en el que un usuario solicita información de la base de datos. Estos lenguajes son por lo general en un nivel más alto que el de un lenguaje de programación estándar. Los lenguajes de consulta pueden ser categorizados como procedurales o no-procedurales (Silberschatz, 2006, pág. 88).

91. ¿En qué consisten los lenguajes procedurales?

Un lenguaje de manipulación de datos (DML) procedural o de bajo nivel debe ser incrustado en un lenguaje de programación de propósito general. Este tipo de lenguajes de manipulación de datos recuperan registros individuales u objetos de una base de datos y procesan cada uno por separado. Por lo tanto, necesitan usar construcciones del lenguaje de programación, tales como ciclos, para recuperar y procesar cada registro de un conjunto de registros. Debido a esto, los lenguajes procedurales también son llamados "lenguajes de un registro a la vez" (Elmasri, 2011, pág. 38).

Un lenguaje procedural es un lenguaje de manipulación de datos que requiere que el usuario especifique qué datos se necesitan y cómo obtener dichos datos a través de una secuencia de operaciones (Silberschatz, 2006, pág. 12).

El lenguaje PL/SQL de PostgreSQL es un lenguaje procedural.

92. ¿Qué es el álgebra relacional?

El conjunto básico de operadores para el modelo relacional es el álgebra relacional. Estos operadores permiten a un usuario especificar consultas de información en términos de álgebra relacional. El resultado de una consulta es una nueva relación, la cual puede estar formada de una o más relaciones, éstas también pueden ser manipuladas usando operaciones algebraicas. Una secuencia de operaciones de álgebra relacional forma una expresión de álgebra relacional cuyo resultado es también una relación que representa una consulta a base de datos (Silberschatz, 2011, pág. 145).

El álgebra relacional es una manera de construir nuevas relaciones sobre relaciones dadas, cuando las relaciones son sobre los datos almacenados, entonces podemos construir relaciones que pueden ser resultado de consultas sobre estos datos (García-Molina, 2009, pág. 38).

Es una colección de operadores que toman las relaciones como sus operandos y devuelven una relación como resultado (Date, 2004, pág. 173).

93. ¿En qué consiste el operador σ? Describe sus componentes.

La operación selección selecciona tuplas que satisfacen un predicado dado. Se utiliza la letra griega sigma minúscula (σ) para denotar la operación selección. El predicado se expresa como subíndice de σ:

$$\sigma_C (R)$$

Donde R es una expresión en álgebra relacional y c es una condición booleana (predicado) que permite expresar comparaciones utilizando: =, ≠, <, ≤, > o ≥. Además, se pueden combinar varios predicados en uno mayor utilizando los operadores de comparación "y" (∧), "o" (∨) y "no" (¬). La condición o predicado de la selección puede incluir comparaciones entre dos atributos.

Cualquier comparación que considere un valor nulo se evalúa como falsa (Silberschatz, 2009, pág. 58).

94. ¿En qué consiste el operador π? Describe sus componentes.

El operador π es utilizado para representar la operación proyección (PROJECT). La operación PROJECT selecciona ciertas columnas de una tabla y descarta las columnas restantes. Si estamos interesados en solamente algunos atributos de una relación, usamos la operación PROJECT sobre estos atributos únicamente. Su notación es:

$$\pi_L (R)$$

Donde L es la sublista de atributos deseados de la relación R (Elmasri, 2011, pág. 149-150).

Nótese que en esta operación las tuplas duplicadas se descartan.

El operador proyección es usado para producir una nueva relación a partir de una relación R que posea solamente algunas de las columnas de R. El valor de la expresión πA1,A2,...,An (R) es una relación que tiene solamente las columnas de los atributos A1, A2,..., An de R (Ullman, 2009, pág. 41).

95. ¿En qué consiste el operador ρ? Describe sus componentes.

El operador renombramiento sirve para renombrar ya sea una relación, un atributo o ambos. Cuando este operador es aplicado a una relación R de grado n es denotado por alguna de las siguientes formas:

$$\rho_{s\ (R)}$$

Donde el símbolo ρ (rho) es usado para denotar el operador renombramiento, S es el nuevo nombre de la relación, y B1, B2, …, Bn son los nuevos nombres de los atributos.

$$\rho_{s\ (B1,\ B2,\ \ldots\ ,\ Bn)}\ (R)$$

$$\rho_{s\ (R)}$$

$$\rho_{(B1,\ B2,\ \ldots\ ,\ Bn)}\ (R)$$

La primera expresión renombra la relación y sus atributos, la segunda renombra únicamente la relación, y la tercera renombra únicamente a los atributos. Si los atributos de R son A1, A2, …, An, en ese orden, entonces cada atributo A_i es renombrado como B_i (Elmasri, 2011, pág. 155).

96. ¿En qué consiste el operador U? Describe sus componentes.

Es una operación binaria, la cual es aplicada a dos relaciones que tienen el mismo tipo de tuplas, esta condición es llamada compatibilidad de unión. Dos relaciones R(A1, A2,...,An) y S(B1,B2,...,Bn) se dice que son compatibles si se llaman igual, tienen el mismo grado y se cumple que dom(Ai) = dom(Bi) para $1 \leq i \leq n$.

Esto significa que las dos relaciones tienen el mismo número de atributos y cada par de atributos correspondientes tiene el mismo dominio. El resultado de esta operación es denotado por:

$$R \cup S$$

Que es una relación que incluye todas las tuplas que están, ya sea en R o S, o en ambas R y S. Las tuplas duplicadas son eliminadas (Elmasri, 2011, pág. 153).

Dadas dos relaciones R y S del mismo tipo, la unión de estas relaciones, R unión S, es una relación del mismo tipo, con un esquema que consiste en todas las tuplas t tal que aparecen en R o S, o en ambas (Date, 2004, pág. 180).

97. ¿En qué consiste el operador -? Describe sus componentes.

La diferencia es una operación binaria, que es aplicada a dos relaciones compatibles, mismo criterio que se aplica en la unión. El resultado de la operación diferencia se denota como:

$$R - S$$

Que es una relación que incluye todas las tuplas que están en R pero no en S. El operador no es conmutativo, es decir, R - S es diferente a S – R. (Elmasri, 2001, pág. 153,154)

98. ¿En qué consiste el operador ∩? Describe sus componentes

La intersección es una operación binaria que se aplica a dos relaciones compatibles, mismo criterio que se aplica en la unión. (Elmasri, 2011, pág. 153).

La intersección de dos relaciones R y S se denota por:

$$R \cap S$$

El resultado es una relación que incluye todas las tuplas que pertenecen tanto a R como a S. La intersección de dos relaciones se puede especificar en función de otros operadores básicos:

$$R \cap S = R-(R-S)$$

(Piñeiro, 2014, pág. 11)

99. ¿En qué consiste el operador *x*? Describe sus componentes

El producto cartesiano, también conocido como PRODUCTO CRUZ o combinación cruzada (CROSS JOIN), se denota con el operador *x*. Es una operación de conjuntos binaria, pero las relaciones en las que se aplica no tienen que ser compatibles. En su forma binaria, esta operación de conjuntos produce un nuevo elemento mediante la combinación de todas las tuplas de *R* con todas las tuplas de *S*:

$$R \times S$$

En general, el resultado de *R X S* es una relación con *(n + m)* atributos y *(x * z)* tuplas, donde *n* es el número de atributos de R, m el número de atributos de S, x el número de tuplas de R y z el número de tuplas de S (Elmasri, 2011, pág. 155-157).

Este operador es conmutativo, es decir, RxS = SxR, ya que las tuplas resultantes serán las mismas, la diferencia es que en el primer caso se visualizarán primero los atributos de R seguidos por los atributos de S, mientras que en el segundo caso será de manera contraria.

100. ¿En qué consiste el operador |X|?

La operación NATURAL JOIN denotada como |X|, es usada para combinar tuplas relacionadas de dos relaciones. Esta operación es muy importante para cualquier base de datos relacional con más de una sola relación porque nos permite procesar relaciones entre relaciones. La operación |X| puede ser especificada como una operación del producto cartesiano seguida de la operación SELECT (Elmasri, 2011, pág. 157).

El NATURAL JOIN es una operación binaria que permite combinar relaciones a través de ciertas selecciones y un producto cartesiano en una operación. Esta denotado por IXI. La operación NATURAL JOIN forma un producto cartesiano de sus dos argumentos (relaciones o esquemas relacionales), realiza una selección forzando la igualdad entre los atributos que aparecen en ambos esquemas relacionales y finalmente remueve los atributos duplicados (Silberschatz, 2006, pág. 56).

Este operador es equivalente a:

$$R \,|X|\, S = \pi_L \left(\sigma_{id\_R = id\_S} \left(R \times S\right)\right)$$

101. ¿En qué consisten los operadores |X (left) y X| (right)?

Un conjunto de operadores, llamados OUTER JOINS (reuniones externas), fueron desarrollados para casos cuando el usuario desee mantener todas las tuplas de R, o todas las de S, o todas las tuplas de ambas relaciones en el resultado de una reunión R|X|S, sin importar si existen o no tuplas que coincidan en alguna relación. Esto satisface la necesidad de las consultas en las que las tuplas de dos tablas son combinadas, haciendo coincidir las filas correspondientes, pero sin perder ninguna tupla por falta de valores coincidentes. El operador LEFT OUTER JOIN (reunión externa izquierda denotada como |x) mantiene cada tupla en la primera relación R (la del lado izquierdo) en R |x S; si no hay tuplas coincidentes en S, entonces los atributos de S en el resultado de la reunión son llenados con valores nulos.

El RIGHT OUTER JOIN, denotado por x|, es similar al LEFT OUTER JOIN, pero mantiene cada tupla en la segunda relación S (la del lado derecho) en el resultado de R X| S (Elmasri, 2011, pág. 169-170).

El RIGHT OUTER JOIN es simétrico con el LEFT OUTER JOIN: llena las tuplas de la relación derecha que no coincidieron con alguna de la relación izquierda con nulos, y las agrega al resultado de la reunión natural (Silberschatz, 2006, pág. 65).

En general, un OUTER JOIN está formado por un NATURAL JOIN y la adición de cualquier tupla que no coincida con los atributos en común de alguna tupla de la otra relación de R o S (Date, 2004, pág. 590).

102. Explica, los diferentes tipos de JOINs mediante diagramas.[15]

Todos los tipos de JOINs son derivados del CROSS JOIN

ID_T1	Atrib_T1
1	A
2	B
5	C
7	D
9	E

CROSS JOIN

ID_T2	Atrib_T2
1	X
3	Y
5	Z
7	W
8	K

Recordemos que en un CROSS JOIN toma cada elemento del lado izquierdo y lo combina con cada elemento del lado derecho.

ID_T1	Atrib_T1	ID_T2	Atrib_T2
1	A	1	X
1	A	3	Y
1	A	5	Z
1	A	7	W
1	A	8	K
2	B	1	X
2	B	3	Y
2	B	5	Z
2	B	7	W
2	B	8	K
5	C	1	X
5	C	3	Y
5	C	5	Z
5	C	7	W
5	C	8	K
7	D	1	X
7	D	3	Y
7	D	5	Z
7	D	7	W
7	D	8	K
9	E	1	X
9	E	3	Y
9	E	5	Z
9	E	7	W
9	E	8	K

[15] Say NO to Venn Diagrams When Explaining JOINs, https://blog.jooq.org/2016/07/05/say-no-to-venn-diagrams-when-explaining-joins/ (Consultada el 05/12/2016).

Un INNER JOIN es un CROSS JOIN en el cual solamente se conservan aquellas combinaciones que cumplen con una característica dada.

Los siguientes diagramas explican cada tipo de JOIN:

ID_T1	Atrib_T1
1	A
2	B
5	C
7	D
9	E

INNER JOIN

ID_T2	Atrib_T2
1	X
3	Y
5	Z
7	W
8	K

Produce:

ID_T1	Atrib_T1	ID_T2	Atrib_T2
1	A	1	X
5	C	5	Z
7	D	7	W

FULL OUTER JOIN: el resultado es el conjunto total de registros de ambas tablas, coincidiendo aquellos registros cuando sea posible. Cuando las tuplas no coinciden en ambas relaciones, se llenan con valores nulos (Elmasri, 2011, pág. 170). Nótese que FULL OUTER JOIN es equivalente a FULL JOIN.

ID_T1	Atrib_T1
1	A
2	B
5	C
7	D
9	E

FULL OUTER JOIN

ID_T2	Atrib_T2
1	X
3	Y
5	Z
7	W
8	K

Produce:

ID_T1	Atrib_T1	ID_T2	Atrib_T2
1	A	1	X
2	B	NULL	NULL
5	C	5	Z
7	D	7	W
9	E	NULL	NULL
NULL	NULL	3	Y
NULL	NULL	8	K

LEFT OUTER JOIN: el resultado son todos los registros de la Tabla A (izquierda) y, si es posible, las coincidencias con la Tabla B (derecha). Nótese que LEFT OUTER JOIN es equivalente a LEFT JOIN.

ID_T1	Atrib_T1
1	A
2	B
5	C
7	D
9	E

LEFT OUTER JOIN

ID_T2	Atrib_T2
1	X
3	Y
5	Z
7	W
8	K

Produce:

ID_T1	Atrib_T1	ID_T2	Atrib_T2
1	A	1	X
2	B	NULL	NULL
5	C	5	Z
7	D	7	W
9	E	NULL	NULL

RIGHT OUTER JOIN: el resultado son todos los registros de la Tabla B (derecha), y si es posible, las coincidencias con la Tabla A (izquierda) (Silberschatz, 2006, pág. 113). Nótese que RIGHT OUTER JOIN es equivalente a RIGHT JOIN.

ID_T1	Atrib_T1
1	A
2	B
5	C
7	D
9	E

RIGHT OUTER JOIN

ID_T2	Atrib_T2
1	X
3	Y
5	Z
7	W
8	K

Produce:

ID_T1	Atrib_T1	ID_T2	Atrib_T2
1	A	1	X
5	C	5	Z
7	D	7	W
NULL	NULL	3	Y
NULL	NULL	8	K

103. ¿En qué consiste la operación DIVISIÓN? Muestra la expresión en álgebra relacional para implementarla.

La operación división, denotada por ÷, es útil para un tipo especial de consulta que a veces se produce en las aplicaciones de base de datos. En álgebra relacional la división se adapta a las consultas que incluyen la frase "para todos".

Formalmente, sean R y S relaciones, tal que $S \subseteq R$; es decir, todos los atributos del esquema S también están en el esquema R. Entonces, R ÷ S es una relación en el esquema R – S (es decir, en el esquema que contiene todos los atributos del esquema R que no están en el esquema S).

Dada una operación de división y los esquemas de las relaciones podemos, de hecho, definir la operación de división en términos de las operaciones fundamentales:

$$T1 \leftarrow \pi_y(R)$$

$$T2 \leftarrow \pi_y(S \times T1) - R)$$

$$T \leftarrow T1 - T2$$

(Silberschatz, 2001, pág. 109-111)

104. ¿Qué es una función de agregación?

Es una operación que se utiliza en consultas estadísticas sencillas para resumir información procedente de las tuplas de las bases de datos (Elmasri, 2001, pág. 166).

Una función de agregación es una operación que toma un conjunto de valores y devuelve como resultado un único valor (Silberschatz, 2006, pág. 61).

105. Describe las funciones de agregación que existen. Describe sus componentes.

Las funciones de agregación existentes son:

- **SUM:** toma una colección de valores y devuelve la suma de los valores.
- **AVG:** regresa el promedio de los valores.
- **COUNT:** regresa el número de elementos de la colección.
- **MIN/MAX:** regresa el valor mínimo/máximo de la colección.

Los componentes de una función de agregación los podemos identificar mediante el siguiente ejemplo:

$$\mathcal{G}\,\mathtt{SUM(atributo)}$$

La operación de álgebra relacional G (g caligráfica) significa que se aplica una función de agregación. El subíndice especifica la operación de agregación a aplicar sobre cierto atributo.

(Silberschatz, 2001, pág. 113)

106. ¿Cuáles son las restricciones más importantes de las funciones de agregación?

Las funciones de agregación deben cumplir ciertas restricciones:

- Regresan un valor asociado a un nombre.
- No se pueden anidar.
- Para utilizar el valor regresado por una función de agregación éste debe utilizarse a través del operador asignación.

De acuerdo con (Ullman, 2009, pág. 287), se pueden incluir las siguientes restricciones:

- El valor NULL es ignorado por las funciones de agregación. Es decir, este no contribuirá al cálculo de SUM, AVG, MIN, MAX ni COUNT(A). Sin embargo, NULL si afecta el resultado de COUNT(*), ya que esta función cuenta el número de tuplas sin importar los valores de éstas.
- Por otro lado, en las funciones de agrupación, NULL sí es tomado en cuenta, por lo que podría existir un agrupamiento donde los atributos sean etiquetados con NULL.

- Cuando se ejecuta una función de agregación sobre una relación vacía, el resultado será NULL. Salvo en el caso de COUNT(*), que el resultado será 0.

107. ¿En qué consiste la agrupación? Describe sus componentes.

La agrupación de tuplas, de acuerdo a su valor en uno o más atributos, tiene el efecto de particionar las tuplas de una relación en "grupos". El operador grouping (γ) combina la agrupación y la agregación.

El operador γ lleva un subíndice L que es una lista de elementos y puede ser:

1. **Un atributo de la relación R a la que γ es aplicada.** Este atributo es uno de los atributos por los que R será agrupada. Este elemento es conocido como *atributo de agrupación*.
2. **Una función de agregación aplicada a un atributo de la relación.** Para proveer un nombre para el atributo correspondiente a esta agregación en el resultado, se adhiere una flecha y un nuevo nombre a la agregación. Este atributo subyacente es conocido como *atributo agregado*.

$$\texttt{atributo\_agrupador}\ \mathcal{G}\ \texttt{funcion\_de\_agregacion(atributo)}$$

La relación resultante de la operación $\gamma_L(R)$ se construye como se muestra a continuación:

1. Una partición de tuplas de R en grupos.
2. Cada grupo consta de todas las tuplas que tienen una asignación particular de valores para agrupar atributos en la lista L. Si no hay atributos agrupados, la relación (R) entera es un grupo.
3. Para cada grupo, se produce una tupla que consta de:
    a. Los valores de los atributos de agrupación para ese grupo y;
    b. Las agregaciones sobre todas las tuplas de ese grupo, para los atributos anexados en la lista L.

(Ullman, 2009, pág. 216)

108. ¿Qué es el árbol de una expresión?

En el álgebra relacional, se pueden construir expresiones aplicando operaciones a otras expresiones, usando paréntesis cuando es necesario separar grupos de operandos. De igual modo, estas expresiones se pueden representar mediante un árbol de expresiones. Este tipo de diagramas facilitan la lectura pero es poco conveniente para la notación de la máquina (Ullman, 2009, pág. 47).

Considere la expresión del álgebra relacional para la consulta “Obtener los nombres de todos los clientes que tienen una cuenta en cualquier sucursal ubicada en la Ciudad de México". El árbol de la expresión sería:

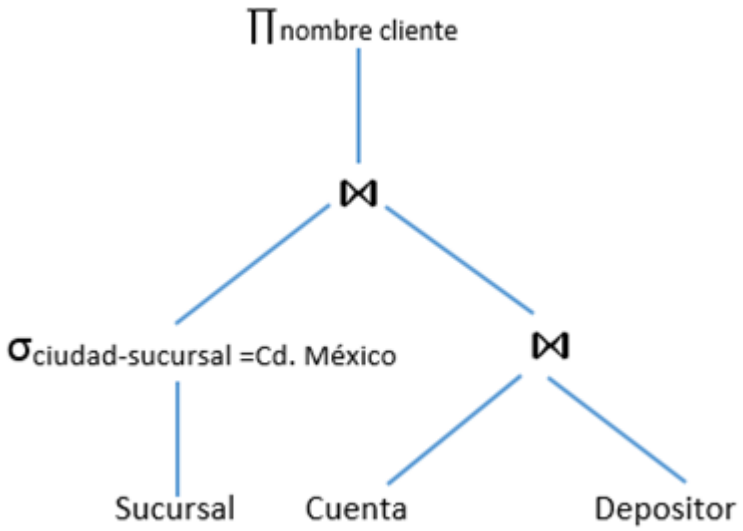


Fig. 28 – Árbol de una expresión

La consulta representada por la expresión anterior en álgebra relacional es:

$$\Pi_{\text{nombre, cliente}} ((\sigma_{\text{ciudad\_sucursal} = \text{“Cd. México”}} (\text{Sucursal})) \,|X|\, (\text{Cuenta} \,|X|\, \text{Depositor}))$$

(Silberschatz, 2011, pág. 529-530)

4.4. Cálculo relacional

109. ¿En qué consisten los lenguajes no-procedurales?

Los lenguajes no-procedurales o declarativos requieren que el usuario especifique qué datos se necesitan sin especificar cómo obtener esos datos.

Los lenguajes no-procedurales son más fáciles de aprender y usar que los procedurales. Sin embargo, como el usuario no especifica cómo conseguir los datos, el sistema de bases de datos tiene que determinar un medio eficiente para acceder a los datos. El componente DML de SQL es un lenguaje no-procedural (Silberschatz, 2001, pág. 7).

110. ¿En qué consiste el cálculo relacional?

El cálculo relacional proporciona un lenguaje declarativo de alto nivel para especificar consultas relacionales.

En una expresión de cálculo relacional, no hay orden de las operaciones para especificar cómo recuperar el resultado de la consulta, sólo la información que el resultado debe contener. Ésta es la principal diferencia entre el álgebra relacional y el cálculo relacional.

El cálculo relacional es importante porque tiene bases en la lógica matemática y porque el lenguaje de consulta estándar (SQL) para SMBDR tiene algunos de sus fundamentos en una variación del cálculo relacional conocido como el cálculo relacional de tuplas (Elmasri, 2011, pág. 146).

Cuando escribimos una expresión en el álgebra relacional, proporcionamos una secuencia de procedimientos que genera la respuesta a nuestra consulta. El cálculo relacional de tuplas, por el contrario, es un lenguaje de consulta no procedimental. En él se describe la información deseada sin describir un procedimiento específico para la obtención de esta información.

Una consulta en el cálculo relacional de tuplas se expresa como { t lP (t) }, es decir, es el conjunto de todas las tuplas t tal que el predicado P es cierto para t.

Siguiendo nuestra notación anterior, usamos t [A] para indicar el valor de la tupla t en el atributo A, y usamos t ∈ r para denotar que la tupla t está en la relación r (Silberschatz, 2001, pág. 118).

La principal diferencia entre el cálculo relacional y el álgebra relacional es la siguiente: el álgebra da un conjunto explícito de operadores (como intersección, unión, etc.) que pueden ser usados para indicar a un sistema como construir alguna relación deseada a partir de ciertas relaciones dadas. El cálculo, por el contrario, se limita a establecer una anotación para indicar la definición de la relación deseada en términos de las relaciones dadas.

El cálculo relacional se basa en una rama de la lógica matemática llamada cálculo de predicados. La idea de utilizar el cálculo de predicados como base para un lenguaje de consulta parece tener su origen en (Kuhns, 1967).

El concepto de un cálculo específicamente relacional, es decir, una forma aplicada de cálculo de predicados adaptado específicamente a las bases de datos relacionales, fue propuesto por primera vez por Codd (Date, 2004, pág. 214).

4.5 Reglas de Codd[16]

111. ¿Qué son las reglas de Codd?

En dos artículos de 1985, Codd publicó reglas o principios que debe usar un sistema de gestión de base de datos para ser considerado “completamente relacional” (Codd, 1985a; 1985b). Codd quería mantener la integridad del modelo relacional y dejar en claro que colocar una interfaz de usuario relacional por encima de un sistema que utilizaba algún otro modelo como su modelo de datos básico no era suficiente para hacer verdaderamente relacional un SMBD.

Él identificó 12 reglas, junto con una regla abarcadora fundamental que llamó Regla Cero. Las reglas proporcionan un conjunto de estándares para juzgar si un SMBD es completamente relacional. (Ricardo, 2009, pág. 156).

[16] Reglas de Codd, http://www.galeon.com/nevifi/Archivos/Codd.pdf (Consultada el 05/12/2016).

112. ¿En qué consiste la regla 1: De la información?

Todos los datos de una base de datos relacional se representan explícitamente (al nivel lógico) como valores de tablas (Osorio Rivera, 2008, pág. 71).

Toda la información de una base de datos relacional debe estar representada lógicamente como valores en columnas y renglones dentro de tablas (Coronel, 2011, pág. 88).

113. ¿En qué consiste la regla 2: Acceso garantizado?

Para todos y cada uno de los datos (valores atómicos) de una base de datos relacional se garantiza que son accesibles a nivel lógico utilizando una combinación del nombre de la tabla, el valor de la clave primaria y el nombre de la columna.

Cualquier dato almacenado en una base de datos relacional tiene que poder ser direccionado unívocamente, para ello hay que indicar en qué tabla está, cuál es la columna y cuál es la fila (mediante la clave primaria).

114. ¿En qué consiste la regla 3: Tratamiento sistemático de valores nulos?

Se debe disponer de valores nulos (distintos de la cadena vacía, blancos, 0, etc.) para representar información desconocida o no aplicable de manera sistemática e independientemente del tipo de datos.

115. ¿En qué consiste la regla 4: Catálogo dinámico en línea basado en el modelo relacional?

La descripción de la base de datos se representa a nivel lógico de la misma manera que los datos normales, de modo que los usuarios autorizados pueden aplicar el mismo lenguaje relacional a su consulta, igual que lo aplican a los datos normales.

Los metadatos se almacenan y se manejan usando el modelo relacional, con todas las consecuencias.

116. ¿En qué consiste la regla 5: Sublenguaje completo de datos?

Un sistema relacional debe soportar varios lenguajes y varios modos de uso. Debe proveer al menos un lenguaje cuyas sentencias sean expresables, mediante una sintaxis bien definida y que soporte:

- Definición de datos.
- Definición de vistas
- Manipulación de datos
- Restricciones de integridad
- Autorización
- Transacciones.

117. ¿En qué consiste la regla 6: De la actualización?

Todas las vistas que sean teóricamente actualizables, son también actualizables por el sistema (Nevado, 2010, pág. 69).

Las vistas deben mostrar la última información contenida, por lo tanto, deben ser actualizables por el sistema. Las vistas de la base de datos deben estar siempre actualizadas con los últimos datos de la base de datos (Jiménez, 2014, pág. 6).

118. ¿En qué consiste la regla 7: Alto nivel de inserción, actualización y borrado?

Se refiere a la capacidad de manejar una relación base o derivada como un solo operando, que se aplica no sólo a la recuperación de los datos (consultas), sino también a la inserción, actualización y borrado de datos.

El alto nivel de inserción, actualización y borrado, permite al sistema realizar manipulación de datos de alto nivel, es decir, sobre conjuntos de tuplas. Esto significa que los datos no solo se pueden recuperar de una base de datos relacional de filas múltiples y/o de tablas múltiples, sino también pueden realizarse inserciones, actualización y borrados sobre varias tuplas y/o tablas al mismo tiempo (no sólo sobre registros individuales).

119. ¿En qué consiste la regla 8: Independencia física de los datos?

Los programas de aplicación permanecen inalterados a nivel lógico cuando se realicen cambios en las representaciones de almacenamiento o a los métodos de acceso (Jiménez, 2014, pág. 6).

120. ¿En qué consiste la regla 9: Independencia lógica de los datos?

Los programas de aplicación permanecen inalterados a nivel lógico cualesquiera que sean los cambios que se realicen a las tablas que almacenan la información.

121. ¿En qué consiste la regla 10: Independencia de integridad?

Las restricciones de integridad específicas para una base de datos relacional podrán definirse a través del sublenguaje de datos relacional y almacenarse en el catálogo, no en los programas de aplicación.

Por tanto en una base de datos relacional se deben poder definir restricciones de integridad. Como parte de las restricciones inherentes al modelo relacional:

- Integridad de Entidad
- Integridad de Dominio
- Integridad Referencial

122. ¿En qué consiste la regla 11: Independencia de distribución?

La distribución de las porciones de la base de datos debe ser invisible a los usuarios de ésta.

123. ¿En qué consiste la regla 12: No Subversión?

Si el sistema proporciona una interfaz de bajo nivel de registro, a parte de una interfaz relacional, que esa interfaz de bajo nivel no se pueda utilizar para subvertir el sistema.

Es decir, no es posible saltar las restricciones de seguridad ni de integridad definidas con anterioridad.

5. Diseño de bases de datos

124. ¿En qué consiste el diseño de una base de datos?

El diseño de una base de datos consiste en definir la estructura de los datos que debe tener la base de datos de un sistema de información determinado. En el caso relacional, esta estructura será un conjunto de esquemas de relación con sus atributos, dominios de atributos, claves primarias, claves foráneas, etc.

La tarea de crear una aplicación de base de datos es compleja, envuelve varias tareas como el diseño del esquema de la base, el diseño de los programas que acceden a modificar los datos y el diseño de un esquema de seguridad para controlar el acceso a los datos. Las necesidades de los usuarios juegan un papel central en el proceso de diseño.

El diseñador de una aplicación de base de datos debe conocer las necesidades de la empresa y debe comenzar por centrar la atención en los problemas que debe resolver. Estos aspectos adicionales involucran varios aspectos de diseño en los niveles físico, lógico y vista de los datos (Silberschatz, 2006, pág. 201).

5.1. Dependencias funcionales

125. ¿Cuál es la definición formal de dependencia funcional?

Sean A y B atributos de R, se dice que B depende funcionalmente de A si para todas las tuplas t_i y t_j en R se cumple que:

$$t_i[A] = t_j[A] \Rightarrow t_i[B] = t_j[B]$$

Si R es un esquema relacional y A y B son conjuntos de atributos no vacíos en R, se dice que B es funcionalmente dependiente de A si y sólo si cada valor de A en R tiene asociado exactamente un valor de B en R.

Esto se denota como:

$$A \rightarrow B$$

y se lee como: “A determina funcionalmente a B”.

(Ricardo, 2009, pág. 168)

126. ¿Cuál es la definición informal de dependencia funcional?

En una relación R con esquema R(A, B), se dice que A determina funcionalmente a B si para cada valor del dominio de A, éste tiene asociado exactamente un valor del dominio de B.

127. ¿En qué consiste una dependencia funcional trivial?

Una dependencia funcional trivial tiene un lado derecho que es un subconjunto de su lado izquierdo (Ullman, 2009, pág. 74-75).

La regla reflexiva establece que un conjunto de atributos siempre se determina a sí misma o cualquiera de sus subconjuntos, lo que es obvio. Debido a que la reflexividad genera dependencias que son siempre ciertas, dichas dependencias se denominan triviales. Formalmente, una dependencia funcional $X \rightarrow Y$ es trivial si $X \supseteq Y$; de lo contrario, no es trivial (Elmasri, 2011, pág. 546).

128. ¿En qué consiste una dependencia funcional transitiva?

Sean A, B y C tres atributos o conjuntos de atributos de una relación R. Si A determina funcionalmente a B, y B determina funcionalmente a C, se dice que C depende de manera transitiva de A.

129. ¿En qué consiste la cerradura de un atributo?

Sea F el conjunto de las dependencias funcionales que son especificadas en el esquema relacional R.

Formalmente, el conjunto de todas las dependencias que incluye F, así como todas las dependencias que pueden inferirse a partir de F se llama cerradura de F, y se denota por F^+ (Elmasri, 2011, pág. 548)

La cerradura de F, denotado por F^+, es el conjunto de todas las dependencias funcionales lógicamente implicadas por F (Silberschatz, 2001, pág. 265).

130. ¿Qué es un atributo superfluo (raro o extraño)?

Se dice que un atributo de una dependencia funcional es superfluo o "extraño" si se puede eliminar sin modificar la cerradura del conjunto de dependencias funcionales. La definición formal de atributos superfluos es la siguiente. Considérese un conjunto F de dependencias funcionales y la dependencia funcional $\alpha \rightarrow \beta$ de F.

- El atributo A es superfluo en α si $A \in \alpha$ y F implica lógicamente a:

$$(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}.$$

- El atributo A es superfluo en β si $A \in \beta$ y el conjunto de dependencias funcionales

$$(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\} \text{ implica lógicamente a F.}$$

Por ejemplo, supóngase que se tienen las dependencias funcionales $AB \rightarrow C$ y $A \rightarrow C$ en F. Entonces, B es superfluo en $AB \rightarrow C$.

131. ¿Qué es la cobertura canónica?

Se puede reducir el esfuerzo de revisar las violaciones a las dependencias funcionales comprobando un conjunto simplificado de dependencias funcionales que tenga la misma cerradura. Cualquier base de datos que satisfaga el conjunto simplificado de dependencias funcionales satisfará también el conjunto original y viceversa, ya que los dos conjuntos tienen la misma cerradura.

La cobertura canónica F^c de F es un conjunto de dependencias funcionales mínimo tal que F implica lógicamente todas las dependencias de F^c y F^c implica lógicamente todas las dependencias de F. Además, F^c debe tener las siguientes propiedades:

1. Ninguna dependencia funcional de F^c contiene atributos superfluos.
2. El lado izquierdo de cada dependencia funcional de F^c es único.

   Es decir, no hay dos dependencias $\alpha 1 \rightarrow \beta 1$ y $\alpha 2 \rightarrow \beta 2$ de F^c tales que $\alpha 1 = \alpha 2$.

La cobertura canónica del conjunto de dependencias funcionales F puede calcularse de la siguiente manera:

1. $F^c := F$
2. REPEAT
    1. Utilizar la regla de unión para sustituir las dependencias de F^c de la forma $\alpha 1 \rightarrow \beta 1$ y $\alpha 1 \rightarrow \beta 2$ con $\alpha 1 \rightarrow \beta 1\ \beta 2$.
    2. Hallar una dependencia funcional $\alpha \rightarrow \beta$ de F^c con un atributo superfluo en α o en β. Nota: la comprobación de los atributos superfluos se lleva a cabo empleando F^c, no F.
    3. Si se halla algún atributo superfluo, hay que eliminarlo de $\alpha \rightarrow \beta$.
3. UNTIL F^c ya no cambie.

(Silberschatz, 2001, pág. 268)

132. ¿Cuáles son los axiomas de Armstrong?

Los axiomas de Armstrong se utilizan para encontrar dependencias funcionales lógicamente implicadas. Mediante la aplicación de estos axiomas, podemos encontrar a F^+. Los axiomas de Armstrong son

- **Regla de reflexividad.**

  Si α es un conjunto de atributos y $\beta \subseteq \alpha$, entonces $\alpha \rightarrow \beta$ se cumple.

- **Regla de aumento.**

  Si $\alpha \rightarrow \beta$ se cumple y γ es un conjunto de atributos, entonces $\gamma\alpha \rightarrow \gamma\beta$ se cumple.

- **Regla de transitividad.**

  Si $\alpha \rightarrow \beta$ se cumple y $\beta \rightarrow \gamma$ se cumple, entonces $\alpha \rightarrow \gamma$.

- **Regla de unión.**

  Si $\alpha \rightarrow \beta$ se cumple y $\beta \rightarrow \gamma$ se cumple, entonces $\alpha \rightarrow \beta\gamma$ se cumple.

- **Regla de pseudotransitividad.**

  Si $\alpha \rightarrow \beta$ se cumple y $\gamma\beta \rightarrow \delta$ se cumple, entonces $\alpha\gamma \rightarrow \delta$ se cumple.

- **Regla de descomposición.**

  Si $\alpha \rightarrow \beta\gamma$ se cumple, entonces $\alpha \rightarrow \beta$ se cumple y $\alpha \rightarrow \gamma$ se cumple.

(Silberschatz, 2006, pág. 279)

133. ¿En qué consiste el axioma de aumento?

El axioma de aumento dice que al adicionar el mismo conjunto de atributos a ambos lados de una dependencia funcional dará como resultado otra dependencia funcional válida (Elmasri, 2011, pág. 546).

134. ¿En qué consiste el axioma de reflexión?

Si $\{A_1, A_2, ..., A_n\}$ es un conjunto de atributos y $\{B_1, B_2, ..., B_m\}$ está contenido en $\{A_1, A_2, ..., A_n\}$ entonces se cumple que $A_1A_2....A_n$ implica $B_1B_2...B_m$. Esto es lo que se llama Dependencia Funcional Trivial (Ullman, 2009, pág. 81).

135. ¿En qué consiste el axioma de transitividad?

El axioma de Transitividad nos permite conectar en cascada dos dependencias funcionales:

Si $\{A_1,...,A_n\}$,$\{B_1,...,B_n\}$ y $\{C_1,...,C_n\}$ son conjuntos de atributos y $A_1A_2 \dots A_n \rightarrow B_1B_2 \dots B_n$, y $B_1B_2 \dots B_n \rightarrow C_1C_2 ...C_n$, entonces se cumple que $A_1A_2 \dots A_n \rightarrow C_1C_2 ...C_n$.

(Ullman, 2009, pág. 80-81)

5.2. Formas Normales

136. ¿Cuál es la definición de la Primera Forma Normal (1FN)?

Todos sus atributos son atómicos.

Alternativamente, se puede definir de la siguiente manera:

- No hay orden de arriba-abajo en las tuplas.
- No hay orden de izquierda-derecha en atributos.
- No hay filas duplicadas.
- Cada intersección fila – columna contiene exactamente un valor del dominio.

137. ¿Cuál es la definición de la Segunda Forma Normal (2FN)?

Todo atributo que no es llave en R depende de manera completa de la llave primaria de R.

Para las entidades que tienen una llave primaria que es una llave compuesta, habrá que preguntarse si los atributos que no son llave dependen de algún subconjunto de la llave primaria, es decir, que no dependen de manera completa de la llave, sino de manera parcial.

NOTA: Si la llave primaria de una relación R está compuesta por un único elemento, automáticamente R está en 2FN (Sherman, 2014,192).

138. ¿Cuál es la definición de la Tercer Forma Normal (3FN)?

Ningún atributo no llave depende transitivamente de la llave primaria.

Una base de datos está en tercera forma si satisface las siguientes condiciones:

- Está en 2FN
- No hay dependencia funcional transitiva.

Por dependencia funcional transitiva entendemos que tenemos las siguientes relaciones en las tablas: A es funcionalmente dependiente de B y B es funcionalmente dependiente de C. En este caso, C es transitivamente dependiente de A vía B.[17]

139. ¿Cuál es la definición de la Forma Normal Boyce-Codd (FNBC)?

Todo determinante es llave.

Una relación R está en FNBC si y sólo si siempre que hay una dependencia funcional no trivial A_1, A_2, …,$A_n \rightarrow B_1$, B_2, …,B_m para R, se cumple que {A_1, A_2, ..., A_n} es una superllave de R.

Es decir, el lado izquierdo de cada dependencia funcional no trivial debe ser una superllave. Recordemos que una superllave no tiene que ser mínima. Por lo tanto, un enunciado equivalente de

[17] Tercera Forma Normal, http://www.1keydata.com/database-normalization/third-normal-form-3nf.php (Consultada el 05/12/2016).

la condición FNBC es que el lado izquierdo de cada dependencia funcional no trivial debe contener una llave (Ullman, 2009, pág. 88).

La FNBC fue propuesta como una forma más simple de la 3FN pero se encontró que era más estricta que la 3FN. Es decir, cada relación en la FNBC también está en 3FN. Sin embargo, una relación en 3FN no está necesariamente en FNBC (Elmasri, 2011, pág. 529).

140. Ejemplifica una descomposición de una relación.

Sea R (A, B, C) una relación y DF = {A → B, B → C} su conjunto de dependencias funcionales.

Supongamos que vamos a insertar las siguientes tuplas: (α1, β1, γ1), (α2, β2, γ2), (α5, β4, γ1) y (α6, β2, γ3). En el esquema de R no existe ninguna restricción que impida realizar dichas inserciones.

A	B	C
α1	β1	γ1
α2	β2	γ2
α5	β4	γ1
α6	β2	γ3

Sin embargo existen tuplas que no cumplen con el conjunto de DF, en particular (α2, β2, γ2) y (α6, β2, γ3)

Dada B → C tenemos que {β2} → {γ2, γ3} lo cual rompe la definición de DF. Supongamos que {β2} →{γ2,} es la correcta, por lo que habrá que evitar la inserción de (α6, β2, γ3)

A	B	C
α1	β1	γ1
α2	β2	γ2
α5	β4	γ1
α6	β2	γ3

Es por eso que debemos normalizar a R para así conservar las DF.

Descomponiendo a R en R1 y R2 como sigue:

R1 (A, B) con DF1 = {A → B}

R2 (B, C) con DF2 = {B → C}

A	B
α1	β1
α2	β2
α5	β4

B	C
β1	γ1
β2	γ2
β4	γ1

Veamos que en este nuevo esquema no es posible insertar a (β2, γ3) quien rompía la DF.

141. ¿Qué establece el teorema de Heath?

Sea R(A, B, C) tal que A→B y C es el resto de los atributos de R entonces R_1(A, B) y R_2(A, C) es una descomposición sin pérdida de R (Heath, 1971).

142. ¿Qué establece el Teorema de Rissanen?

La descomposición en R_1 y R_2 de R es sin pérdida si y sólo si:

- Toda dependencia funcional en R se puede deducir lógicamente de las dependencias funcionales en R_1 y R_2.
- Los atributos comunes de R_1 y R_2 componen una clave candidata de al menos una de las relaciones.

(Date, 2004, pág. 366)

5.3 Dependencias multivaluadas

143. ¿Qué es una dependencia multivaluada?

La dependencia multivaluada α→→β indica que la relación entre α y β es independiente de la relación de α y R-β.

Ésta se cumple en R si para todo par de tuplas t_1 y t_2 de r tales que $t_1[\alpha]= t_2[\alpha]$, existen unas tuplas t_3 y t_4 de r tales que:

$$t_1[\alpha]= t_2[\alpha]= t_3[\alpha]= t_4[\alpha]$$

$$t_3[\beta]= t_1[\beta]$$

$$t_3[R-\beta]= t_2[R-\beta]$$

$$t_4[\beta]= t_2[\beta]$$

$$t_4[R-\beta]= t_1[R-\beta]$$

(Silberschatz, 2002, pág. 180)

Veamos el siguiente ejemplo de una descomposición para respetar una dependencia multivaluada:

Restaurante	Variedad de Pizza	Área de envío
Vincenzo's Pizza	Corteza gruesa	Springfield
Vincenzo's Pizza	Corteza gruesa	Shelbyville
Vincenzo's Pizza	Corteza fina	Springfield
Vincenzo's Pizza	Corteza fina	Shelbyville
Elite Pizza	Corteza fina	Capital City
Elite Pizza	Corteza rellena	Capital City
A1 Pizza	Corteza gruesa	Springfield
A1 Pizza	Corteza gruesa	Shelbyville
A1 Pizza	Corteza gruesa	Capital City
A1 Pizza	Corteza rellena	Springfield
A1 Pizza	Corteza rellena	Shelbyville
A1 Pizza	Corteza rellena	Capital City

Restaurante	Variedad de pizza
Vincenzo's Pizza	Corteza gruesa
Vincenzo's Pizza	Corteza fina
Elite Pizza	Corteza fina
Elite Pizza	Corteza rellena
A1 Pizza	Corteza gruesa
A1 Pizza	Corteza rellena

Restaurante	Área de envío
Vincenzo's Pizza	Springfield
Vincenzo's Pizza	Shelbyville
Elite Pizza	Capital City
A1 Pizza	Springfield
A1 Pizza	Shelbyville
A1 Pizza	Capital City

5.4. Otras formas normales

144. ¿Cuál es la definición de la Cuarta Forma Normal (4FN)?

Un esquema de relación R está en 4FN con respecto a un conjunto de dependencias F (que incluye dependencias funcionales y multivaluadas) si, por cada dependencia multivaluada no trivial $X \rightarrow Y$ en F^+, X es una superllave de R (Elmasri, 2007, pág. 335).

145. ¿Cuál es la definición de la Quinta Forma Normal (5FN)?

Un esquema de relación R está en quinta forma normal (5FN) respecto a un conjunto F de dependencias de reunión (JD), funcionales y multivaluadas si para cada dependencia de reunión no trivial JD (R_1, R_2, …, R_n) implicada por F, cada R_i es superllave de R (Elmasri, 2011, pág. 535).

146. ¿Cuál es la definición de la forma normal de dominio/llave (FNDK)?

Una relación R se dice que está en FNDK si y sólo si cada restricción en R es una consecuencia lógica de las restricciones de dominio y de las restricciones de la llave (Date, 2004, pág. 399)

Se dice que una relación R está en FNDK si todas las restricciones y dependencias de la relación pueden ser forzadas simplemente mediante la aplicación de las restricciones de dominio y de las restricciones de llave en la relación (Elmasri, 2011, pág. 574).

6. Lenguaje de Consulta Estructurado (SQL)

147. ¿En qué consiste SQL?

SQL es un lenguaje declarativo de alto nivel que proporciona una interfaz en la que el usuario solamente especifica qué resultado quiere obtener, dejando la optimización y la decisión de como ejecutar la consulta al SMBD.

El nombre SQL es la abreviación de Structured Query Language. Originalmente fue llamado SEQUEL y fue diseñado e implementado por IBM como la interfaz para un sistema experimental de base de datos llamado SYSTEM R. SQL. (Elmasri, 2011, pág. 87-88).

148. ¿Quién y cuándo lo propuso?

SEQUEL, la primera versión de SQL, fue propuesta por Donald Chamberlin y Raymond Boyce a principios de los 70's. En 1986 se publicó el estándar de SQL, el SQL_86 (Silberschatz, 2010, pág. 57).

La historia de SQL comienza en 1974 con la definición en los laboratorios de investigación de IBM, como un lenguaje para la especificación de las características de las bases de datos que adoptaban el modelo relacional. Este lenguaje se llamaba SEQUEL (Structured English QUEry Language) y se implementó en un prototipo llamado SEQUEL-XRM entre 1974 y 1975. Los experimentos con ese prototipo condujeron, entre 1976 y 1977, a una revisión del lenguaje (SEQUEL/2), que a partir de ese momento cambió de nombre por motivos legales, convirtiéndose en SQL. El prototipo (System R), basado en este lenguaje, se adoptó y utilizó internamente en IBM y lo adoptaron algunos de sus clientes elegidos. Gracias al éxito de este sistema, que no estaba todavía comercializado, también otras compañías empezaron a desarrollar sus productos relacionales basados en SQL. En el curso de los años ochenta, compañías como Oracle y Sybase, comercializaron productos basados en SQL, que se convierte en el estándar industrial de hecho por lo que respecta a las bases de datos relacionales.[18]

[18] Historia de SQL, http://www.htmlpoint.com/sql/sql_04.htm (Consultada el 05/12/2016).

6.1. Sublenguajes de SQL

149. ¿Qué es el Lenguaje de Definición de Datos (DDL)?

El Lenguaje de Definición de Datos (DDL) se utiliza para definir el esquema conceptual de base de datos. En la mayoría de los SMBD, el DDL también define vistas de usuario y, a veces, las estructuras de almacenamiento; en otros SMBD, existen lenguajes o funciones particulares para especificar estructuras de almacenamiento (Elmasri, 2011, pág. 53).

Un sistema de base de datos proporciona un lenguaje de definición de datos para especificar el esquema de base de datos (Silberschatz, 2001, pág. 21).

Un DDL define la estructura y las restricciones de la información del esquema de la base de datos (Ullman, 2009, pág. 5).

150. ¿Cuáles son los comandos principales de DDL?

Algunos ejemplos de comandos DDL son:

- **`CREATE DATABASE`:** crea una nueva base de datos.
- **`CREATE TABLE`:** crea una nueva tabla.
- **`CREATE INDEX`:** crea un índice en una tabla determinada.
- **`ALTER TABLE`:** modifica la definición de una tabla alterando, agregando o eliminando columnas y restricciones.
- **`DROP INDEX`:** elimina uno o varios índices de la base de datos actual.
- **`DROP TABLE`:** elimina la definición de una tabla y todos sus datos, índices y restricciones.

151. ¿Qué es el lenguaje de manipulación de datos (DML)?

El Lenguaje de Manipulación de Datos (DML) es un lenguaje que permite a los usuarios acceder o manipular datos organizados por el modelo de datos apropiado. Hay básicamente dos tipos de DML´s:

- **Procedurales:** requieren que un usuario especifique qué datos son necesarios y cómo obtener tales datos.

- **Declarativos (o no procedurales):** estos requieren que el usuario indique qué datos son necesitados sin tener que especificar cómo obtenerlos.

(Silberschatz, 2001, pág. 22)

El uso de DML no afecta el esquema de la BD pero afecta su contenido (por ejemplo si la acción es un comando de modificación) o permite extraer información de ella (si la acción es una consulta) (Ullman, 2009, pág. 6-7).

152. ¿Cuáles son los comandos principales de DML?

Algunos ejemplos de comandos DML son:

- **`SELECT`:** corresponde a la operación de proyección del álgebra relacional. Se utiliza para enumerar los atributos deseados en el resultado de una consulta.
- **`FROM`:** corresponde a la operación producto-cartesiano del álgebra relacional. En él se enumeran las relaciones a las que se accesará.
- **`WHERE`:** corresponde a la operación selección del álgebra relacional. Se compone de una condición para los atributos de las relaciones que aparecen en la cláusula FROM.
- **`ORDER BY`:** permite que las tuplas en el resultado de una consulta aparezcan en un orden establecido.
- **`INSERT`:** se usa para insertar datos en una relación. Los valores de los atributos de tuplas insertadas deben ser miembros del dominio del atributo y las tuplas.
- **`DELETE`:** se expresa como una consulta y elimina las tuplas que cumplen con una condición dada.
- **`UPDATE`:** se usa para cambiar un valor en una tupla sin cambiar todos los valores de ésta. Dicha actualización se realiza sobre las tuplas que cumplen con una condición dada.
- **`GROUP BY`:** genera valores de resumen para un solo campo en combinación con las funciones de agregación.
- **`HAVING`:** permite seleccionar o rechazar un grupo de registros de acuerdo a una condición.

(Silberschatz, 2011, pág. 163, 165, 166)

153. ¿Qué es el Lenguaje de Control de Datos (DCL)?

El Lenguaje de Control de Datos (DCL) es un conjunto de comandos que generalmente controla permisos sobre los datos, así como la definición de los derechos de acceso (Matthew, 2005, pág. 9).

Es un lenguaje que permite controlar quién o qué (usuarios y/o aplicaciones) tiene acceso a objetos específicos en la base de datos. Las instrucciones DCL también permiten controlar el tipo de acceso que cada usuario tiene a los objetos de una base de datos (Oppel, 2010, pág. 19).

154. ¿Cuáles son los principales comandos de DCL?

Algunos ejemplos de comandos incluidos en el DCL son los siguientes:

- **GRANT:** otorga permisos a uno o varios usuarios o roles para realizar tareas determinadas.
- **REVOKE:** quita permisos que previamente se han concedido con la cláusula GRANT.

(Silberschatz, 2002, pág. 153-154)

Las tareas sobre las que se pueden conceder o anular permisos son las siguientes:

- **CONNECT:** referente al permiso de conexión a la base de datos.
- **SELECT:** referente al permiso de realizar consultas sobre una tabla o base de datos.
- **INSERT:** referente al permiso de realizar inserciones sobre una tabla o base de datos.
- **UPDATE:** referente al permiso de realizar actualizaciones sobre una tabla o base de datos.
- **DELETE:** referente al permiso de realizar eliminaciones sobre una tabla o base de datos.

155. ¿Qué es el lenguaje de Control de transacciones (TCL)?

Es un lenguaje de programación y un subconjunto de SQL, que se utiliza para controlar el procesamiento de transacciones en una base de datos.[19]

Las operaciones en la base de datos son ejecutadas una a la vez, y las acciones de la base de datos se ejecutan hasta que la siguiente operación actúa. Sin embargo, esto no quiere decir que la base de

[19] TCL, http://robertodiazg.blogspot.mx/2014/03/sentencias-sql-ddl-dml-dcl-y-tcl.html (Consultada el 05/12/2016).

datos actúe paso a paso. El lenguaje de control de transacciones garantiza que cada operación en la base de datos sea ejecutada una a la vez (Ullman, 2009, pág. 296-299).

156. ¿Cuáles son los comandos principales de TCL?

Algunos ejemplos de comandos incluidos en el TCL son los siguientes:

- **`COMMIT`:** se utiliza para guardar de forma permanente cualquier transacción en la base de datos.
- **`ROLLBACK`:** restaura la base de datos al último estado *commit*.
- **`SAVEPOINT`:** se utiliza para guardar temporalmente una transacción de modo que se pueda revertir a ese punto siempre que sea necesario.

(Ullman, 2009, pág. 300)

6.2. Consultas en SQL

157. Muestra la sintaxis básica en SQL de una consulta.

La estructura básica de una expresión SQL consta de tres cláusulas: `SELECT`, `FROM` y `WHERE`.

```
SELECT <listado_de_atributos>
FROM <listado_de_tablas>
WHERE <condición>
```

La inclusión de la cláusula `WHERE` es opcional.

158. Muestra la sintaxis en SQL del comando `DISTINCT`.

Si no se desea obtener duplicados en el resultado de una consulta, se deberá incluir, después del comando `SELECT`, la palabra clave `DISTINCT`. Esa palabra indica a SQL que debe producir una sola copia de cualquier tupla (Ullman, 2009, pág. 282).

La cláusula DISTINCT nos sirve para recuperar los valores distintos de una tabla. Su sintaxis es la siguiente:

```
SELECT DISTINCT <listado_de_atributos>
FROM <listado_de_tablas>
WHERE <condición>
```

La inclusión de la cláusula WHERE es opcional.

159. Muestra la sintaxis en SQL del comando AS.

El comando AS se puede utilizar para cambiar el nombre de una relación en la cláusula FROM o de alguno de sus atributos si se usa en la cláusula SELECT. Por ejemplo:

```
SELECT <lista_de_atributos> ,<atributo_a_renombrar>
AS <alias_del_atributo>
FROM <expresion_a_renombrar> AS
<alias_de_la_expresion>
WHERE <condicion>
```

La inclusión de la cláusula WHERE es opcional.

(Elmasri, 2011, pág. 124-125)

160. Muestra la sintaxis en SQL del comando LIKE. Incluye _ y %.

LIKE es un operador que compara si un valor cumple con un patrón que el usuario puede definir.

La sintaxis genérica se expresa de la siguiente forma:

```
SELECT <lista_de_atributos>
FROM <tablas>
WHERE <atributo_tipo_cadena> LIKE '<cadena>%<cadena>'
```

El % es usado para definir un comodín y puede colocarse antes o después de un patrón; en este caso el patrón es <cadena>. El % se puede sustituir por cero o más caracteres.

```
SELECT <lista_de_atributos>
FROM <tablas>
```

```
WHERE <atributo_tipo_cadena> LIKE '<cadena>_<cadena>'
```

Al igual que %, el _ también es un comodín, sin embargo, _ se sustituye por exactamente un caracter.

161. Muestra la sintaxis de SQL del comando ORDER BY. Incluyendo los comandos ASC y DESC.

Por default se ordenan de forma ascendente, pero también se puede ordenar de forma descendente mediante el comando DESC.

```
ORDER BY <lista_de_atributos> DESC
```

Similarmente se puede ordenar de forma ascendente mediante el comando ASC, aunque este no es necesario.

```
ORDER BY <lista_de_atributos> ASC
```

(Ullman, 2009, pág. 255)

162. Muestra la sintaxis en SQL de los comandos INTERSECT, UNION y EXCEPT.

Las operaciones UNION, INTERSECT y EXCEPT de SQL operan con relaciones y corresponden a las operaciones unión, intersección y resta del álgebra relacional. Al igual que en el álgebra relacional las relaciones que participan deben ser compatibles, es decir, deben tener el mismo número de atributos.

```
(SELECT atributo.tabla1 FROM Tabla1) UNION (SELECT atributo.tabla2 FROM Tabla2);
(SELECT atributo.tabla1 FROM Tabla1) INTERSECT (SELECT atributo.tabla2 FROM Tabla2);
(SELECT atributo.tabla1 FROM Tabla1) EXCEPT (SELECT atributo.tabla2 FROM Tabla2);
```

(Silberschatz, 2006, pág. 88-89)

163. Muestra la sintaxis en SQL de los comandos AVG, MIN, MAX, SUM, COUNT.

Las funciones SUM, MAX, MIN y AVG se pueden aplicar a un conjunto o conjunto múltiple de valores numéricos, son respectivamente la suma, valor máximo, valor mínimo y el promedio de esos valores.

Estas funciones se pueden utilizar en una cláusula SELECT o en una cláusula HAVING. Las funciones MAX y MIN también se puede utilizar con atributos que tienen dominios no numéricos si los valores del dominio tienen un ordenamiento total entre unos y otros.

- La sintaxis de `AVG` es:

```
AVG(atributo)
```

- La sintaxis para `COUNT` es:

```
COUNT(atributo)
```

- La sintaxis para `MIN` y `MAX` es:

```
MIN(atributo)
MAX(atributo)
```

- La sintaxis para `SUM` es:

```
SUM(atributo)
```

Por ejemplo, si se desea encontrar la suma de los salarios de todos los empleados, el salario máximo, el salario mínimo y el salario promedio se puede utilizar una expresión como la siguiente:

```
SELECT SUM (salario), MAX (salario), MIN (salario), AVG (salario)
FROM empleado;
```

(Elmasri, 2011, pág. 124-125)

164. Muestra la sintaxis en SQL del comando `GROUP BY`.

La sintaxis genérica de este operador es:

```
SELECT <atributo_agrupador>, <funcion_agregacion>
FROM <tabla>
GROUP BY <atributo_agrupador>
```

La `<funcion_agregacion>` de la cláusula `SELECT` puede omitirse.

Si se requiere saber cuál es el saldo promedio de cada tipo de cuenta se obtendría de la siguiente manera:

```
SELECT tipo_cuenta, AVG(saldo)
FROM cuenta
GROUP BY tipo_cuenta
```

(Silberschatz, 2006, pág. 90)

165. Muestra la sintaxis en SQL del comando `HAVING`.

La función del comando `HAVING` es la de especificar una condición sobre los grupos, formados por el comando `GROUP BY`, que habrán de seleccionarse.

La sintaxis de una consulta que hace uso del comando `HAVING` es la siguiente:

```
SELECT <lista de atributos y funciones>
FROM <lista de tablas>
*WHERE <condición>
GROUP BY <atributo(s) por el(los) cual(es) agrupar>
HAVING <condición del grupo>;
```

*Es posible que en una consulta donde se haga uso del comando `HAVING` no sea necesario utilizar antes el comando `WHERE`.

A continuación se presenta un ejemplo del uso de este comando en una consulta donde una empresa constructora desarrolla varios proyectos, cada uno de ellos adquiere el nombre de la calle y número donde se ubica la construcción. La empresa desea saber en cuáles de sus proyectos hay trabajando más de 20 personas.

```
SELECT id_proyecto, nombre_proyecto, COUNT (*)
FROM Proyecto, Trabaja_en
WHERE Proyecto.id_proyecto = Trabaja_en.id_proyecto
GROUP BY id_proyecto, nombre_proyecto
HAVING COUNT (*) > 20;
```

(Elmasri, 2011, pág. 128)

6.3. Subconsultas en SQL

166. ¿Qué es una subconsulta?

Una subconsulta es una expresión SELECT-FROM-WHERE que se anida dentro de otra consulta. Un uso común de subconsultas es llevar a cabo comprobaciones sobre pertenencia a conjuntos, comparación de conjuntos y cardinalidad de conjuntos (Silberschatz, 2002, pág. 95).

167. Menciona que operadores son aplicables a una subconsulta.

La sintaxis de una subconsulta es la siguiente:

```
SELECT <lista_de_atributos>
FROM <tabla>
WHERE <atributo> <operador> (SELECT <lista_de_atributos>
                             FROM <otra_tabla>);
```

Donde <operador> puede ser sustituido por alguno de los siguientes: IN, NOT IN, ANY, ALL, EXISTS y NOT EXISTS.

168. Muestra el ejemplo de una subconsulta en SQL.

Considérese una base de datos que modela una tienda. El siguiente ejemplo devuelve todos los productos cuyo precio unitario es mayor que el de cualquier producto vendido con un descuento igual o mayor al 25 por ciento:

```
SELECT *
FROM Productos
WHERE precio_unitario > ANY (SELECT precio_unitario
                             FROM DetallePedido
                             WHERE descuento >= 0.25)
```

(Silberschatz, 2006, pág. 93)

169. ¿Qué es una subconsulta correlacionada?

Una consulta correlacionada es una subconsulta que contiene una referencia a una tabla que también aparece en la consulta exterior, por ejemplo:

```
SELECT *
FROM tabla1
WHERE atributo1 = ANY (SELECT atributo1
                       FROM tabla2
                       WHERE tabla2.atributo2 = tabla1.atributo2);
```

Este ejemplo de subconsulta contiene una referencia a una columna de la `tabla1`, aunque la cláusula `FROM` de la subconsulta no menciona a la `tabla1`. Por lo tanto, el SMBD busca fuera de la subconsulta y encuentra a `tabla1` en la consulta externa.

Las subconsultas correlacionadas no pueden referirse a los resultados de funciones de agregación de la consulta exterior.[20]

6.4. Actualizaciones sobre tablas y tuplas

170. Muestra la sintaxis en SQL para modificar una tabla.

A través del comando `ALTER` es posible modificar una tabla. Las posibles acciones de `ALTER TABLE` incluyen el agregar o eliminar un atributo o cambiar la definición de éste.

Para agregar un nuevo atributo a una tabla empleamos la siguiente sintaxis básica:

```
ALTER TABLE nombre_tabla
ADD nuevo_atributo tipo_de_dato;
```

(Elmasri, 2011, pág. 138)

[20] Subconsultas correlacionadas, http://download.nust.na/pub6/mysql/doc/refman/5.0/es/correlated-subqueries.html (Consultada el 19/11/2016).

Para eliminar un atributo de una tabla:

```
ALTER TABLE nombre_tabla
DROP COLUMN nombre_ atributo;
```

Mediante el uso del comando ALTER, es posible agregar restricciones mediante ADD o eliminarlas utilizando DROP, por ejemplo, para agregar la llave primaria:

```
ALTER TABLE nombre_tabla
ADD PRIMARY KEY <nombre_atributo>;
```

O para eliminar una llave foránea:

```
ALTER TABLE nombre_tabla
DROP FOREIGN KEY <nombre_atributo>;
```

171. Muestra la sintaxis en SQL para insertar una tupla.

El comando `INSERT` se utiliza para añadir una tupla a una relación. Hay que especificar el nombre de la relación y una lista de valores para la tabla

```
INSERT INTO tabla VALUES (valor_atributo1, … , valor_atributoN);
```

A manera de ejemplo insertaremos una tupla a la relación *persona*

```
INSERT INTO persona VALUES (76, 'Richard', 'Marini', '1962-12-30', '98 Oak Forest', 'TX');
```

(Elmasri, 2011, pág. 108)

172. Muestra la sintaxis en SQL para eliminar una tupla.

El comando `DELETE` se utiliza para eliminar tuplas de una relación que cumplan con la condición dada. No se pueden eliminar atributos por separado, solo tuplas completas.

```
DELETE FROM tabla
WHERE condicion;
```

(Silberschatz, 2001, pág. 158)

A manera de ejemplo eliminaremos una tupla a la relación *persona*

```
DELETE FROM persona
WHERE id_persona = 76;
```

En caso de no incluir la cláusula WHERE, el SMBD borrará todas las tuplas de la tabla.

173. Muestra la sintaxis en SQL para actualizar una tupla.

El comando `UPDATE` se utiliza para actualizar los valores de una o más tuplas. Al igual que en el comando `DELETE`, el comando `WHERE` en una cláusula `UPDATE` selecciona las tuplas a modificar a partir de una única relación.

```
UPDATE nombre_tabla
SET atributo_a_actualizar = nuevo_valor
WHERE condicion;
```

(Elmasri, 2011, pág. 110)

A manera de ejemplo actualizaremos algunas tuplas de la relación *persona*

```
UPDATE persona
SET estado_residencia = 'CA'
WHERE id_persona > 113;
```

6.5. Definición de esquemas

174. Muestra la sintaxis en SQL para crear una base de datos.

La sintaxis básica para crear una base de datos es:

```
CREATE DATABASE nombre_de_la_base;
```

175. Muestra la sintaxis en SQL para eliminar una base de datos.

La sintaxis básica para eliminar una base de datos es:

```
DROP DATABASE nombre_de_la_base;
```

176. Muestra la sintaxis en SQL para crear una tabla.

La sintaxis básica para crear una tabla es:

```
CREATE TABLE nombre_tabla (
                atributo_1 tipo_dato,
                atributo_2 tipo_dato
                );
```

177. Muestra la sintaxis en SQL para eliminar una tabla.

La sintaxis básica para eliminar una tabla es:

```
DROP TABLE nombre_tabla;
```

(Ordaz-Rosado, 2014, pág. 176)

178. Muestra la sintaxis en SQL para definir una llave primaria.

La sintaxis básica para definir una llave primaria de una tabla es:

```
PRIMARY KEY (<listado_de_atributos>)
```

Cabe mencionar que ésta debe incluirse dentro de una expresión `CREATE TABLE` o `ALTER TABLE` (Ordaz-Rosado, 2014, pág. 86).

179. Muestra la sintaxis en SQL para definir una llave foránea.

La sintaxis básica para definir una llave foránea en una tabla es:

```
FOREIGN KEY (atributo) REFERENCES tabla_referenciada atributo_referenciado)
```

Cabe mencionar que ésta debe incluirse dentro de una expresión `CREATE TABLE` o `ALTER TABLE` (Ordaz-Rosado, 2014, pág. 88).

180. Muestra la sintaxis en SQL para indicar que un atributo no puede ser nulo.

La sintaxis básica para restringir que un atributo no pueda ser nulo es:

```
atributo tipo_atributo NOT NULL
```

Cabe mencionar que ésta debe incluirse dentro de una expresión `CREATE TABLE` o `ALTER TABLE` (Ordaz-Rosado, 2014, pág. 92).

181. Muestra la sintaxis en SQL para definir una restricción de dominio.

Para restringir los valores del atributo o del dominio se usa `CHECK`, seguido de un atributo o dominio de definición (Elmasri, 2011, pág. 94).

La sintaxis básica para restringir el dominio de un atributo es:

```
CHECK (atributo IN (<listado_de_valores_validos>))
```

Cabe mencionar que ésta debe incluirse dentro de una expresión `CREATE TABLE` o `ALTER TABLE` (Ordaz-Rosado, 2014, pág. 90).

La cláusula *CHECK* actúa revisando si la condición es verdadera o falsa, es decir, si el valor que se pretende insertar en esa columna cumple o no con la condición descrita anteriormente. Al resultar verdadera la condición, la cláusula CHECK permite la inserción, en otro caso la restringe.

Existen otros casos de condiciones como:

```
CHECK (atributo <desigualdad_aritmetica> valor)
      CHECK (atributo LIKE cadena)
    CHECK (atributo <desigualdad_aritmetica> otro_atributo)
    CHECK (atributo BETWEEN valor_inferior AND valor_superior)
```

182. Describe los principales tipos de datos en SQL.

Los tipos de dato varían de acuerdo al SMBD, en PostgreSQL se tienen los siguientes:

Tipo de dato	Alias	Descripción
bigint	int8	*Entero con signo de 8 bytes*
bigserial	serial8	*Autoincremento entero de 8 bytes*
bit		*Cadena de bit de longitud fija*
bit varying(n)	varbit(*n*)	*Cadena de bit de longitud variable*
boolean	Bool	*Lógico (true/false)*
box		*Rectángulo en el plano*
bytea		*Datos binarios*
character varying(n)	varchar(*n*)	*Cadena de caracteres de longitud variable*
character(n)	char(*n*)	*Cadena de caracteres de longitud fija*
cidr		*Dirección IP de red (IPv4 ó IPv6)*
circle		*Círculo en el plano*
date		*Fecha (año, mes, día)*
double precision	float8	*Número de punto flotante de precisión doble*

Tipo de dato	Alias	Descripción
double precision	float8	*Número de punto flotante de precisión doble*
inet		*Dirección de un host de red (IPv4 or IPv6)*
integer	int, int4	*Entero con signo de 4 bytes*
interval(p)		*Intervalo de tiempo*
line		*Línea infinita en el plano (no se aplica completamente)*
lseg		*Segmento de línea en el plano*
macaddr		*Dirección MAC de tarjeta o dispositivo de red*
money		*Moneda*
numeric [(p, s)]	decimal [(*p*, *s*)]	*Numérico exacto con precisión modificable*
path		*Trazado geométrico abierto y cerrado en el plano*
point		*Punto geométrico en el plano*
polygon		*Polígono cerrado geométrico en el plano*
real	float4	*Número de punto flotante de precisión simple*
smallint	int2	*Entero con signo de 2 bytes*
serial	serial4	*Autoincremento entero de 4 bytes*
text		*Cadena de caracteres de longitud variable*
time [(p)] [sin zona horaria]		*Hora del día*
time [(p)] con zona horaria	timetz	*Hora del día, incluyendo la zona horaria*
timestamp [(p)] [sin zona horaria]	timestamp	*Fecha y hora*
timestamp [(p)] con zona horaria	timestamptz	*Fecha y hora incluyendo la zona horaria*

(Ordaz-Rosado, 2014, pág. 53)

6.6. Optimización de consultas

183. ¿En qué consiste el concepto de optimización de consultas?

La optimización de consultas consiste en reducir el tiempo total de una consulta, así como el consumo de los recursos del SMBD. Toma en cuenta los siguientes puntos:

1. Evaluar aquellas condiciones cuya probabilidad de ser verdaderas sea menor.
2. Asociar primero las tablas que contengan un menor número de registros o, dependiendo de la naturaleza de la consulta, que arrojen un menor número de registros como resultado.
3. En consultas que tomen un tiempo considerable en ejecutarse, será de gran ayuda indexar aquellas columnas que sean utilizadas como criterio de discriminación en la cláusula WHERE.

(Ordaz-Rosado, 2014, pág. 128, 134).

184. ¿Qué es el plan de ejecución (evaluación) de una consulta?

El plan de ejecución consiste en la serie de instrucciones de bajo nivel requeridas para realizar una consulta dada.

Una consulta por lo general se puede traducir en una serie de alternativas de evaluación. Estas alternativas son llamadas planes de ejecución y todos dan el mismo resultado. Sin embargo, el compilador DML del SMBD realiza la optimización de la consulta, es decir, que elige el plan de evaluación con menor costo de entre las alternativas (Silberschatz, 2001, pág. 28).

185. Muestra la sintaxis en SQL para mostrar el plan de ejecución de una consulta (EXPLAIN).

El comando en SQL para mostrar el plan de ejecución de consulta es `EXPLAIN`. Este comando es usado para obtener la descripción de la estrategia o el plan que el SMBD utiliza para ejecutar una consulta específica en SQL.

Su sintaxis es la siguiente:

```
EXPLAIN consulta_a_evaluar;
```

186. ¿Qué es y para qué sirve un índice?

Un índice es una estructura que ayuda a localizar registros de una relación de forma rápida, sin examinar todos los registros (Silberschatz, 2006, pág. 439).

Un índice es cualquier estructura de datos que toma el valor de uno o más campos y encuentra los registros con ese valor "rápidamente". En particular, un índice permite encontrar un registro sin tener que mirar más que una pequeña fracción de todos los posibles archivos (Ullman, 2009, pág. 619).

Un índice es algo similar a un “hashing” dinámico y a las estructuras de directorios utilizadas para “hashing” extensibles. Ambos son sondeados para encontrar un apuntador al bloque de datos que contiene el registro deseado (Elmasri, 2011, pág. 636).

187. Muestra la sintaxis en SQL para crear un índice.

La sintaxis básica para crear un índice es:

```
CREATE INDEX nombre_del_indice ON nombre_tabla (atributo_a_indexar);
```

(Ordaz-Rosado, 2014, pág. 133)

188. ¿Qué es y para qué sirve un trigger?

Los disparadores o *triggers* permiten al SMBD monitorear la base de datos. Los *triggers* se aplican a un rango amplio de situaciones y permiten una mayor variedad de acciones. Un disparador consiste en tres partes:

- Un **evento**, que normalmente es algún cambio hecho a la base de datos.
- Una **condición**, que es un predicado lógico que se evalúa a verdadero o falso.
- Una **acción**, que es un procedimiento que se realiza cuando ocurre el evento y la condición se evalúa a verdadero.

Un disparador tiene acceso a los datos insertados, borrados o actualizados que causan su disparo. Los valores de los atributos se pueden usar en el código tanto para la condición como para la acción (Ricardo, 2009, pág. 240-241).

189. ¿Qué es y para qué sirve una función definida por usuario?

Una función definida por usuario es un conjunto de instrucciones que se almacenan físicamente en el SMBD y que en consecuencia tienen acceso directo a los datos. Siendo esta una ventaja con respecto a funciones similares pero implementadas de manera ajena al SMBD con otros lenguajes de programación.

En particular, en PostgreSQL, las funciones definidas por usuario se implementan utilizando PL/pgSQL. Este lenguaje de programación imperativo permite la construcción de funciones más complejas al incluir validaciones, cálculos y procedimientos integrados con sentencias SQL. Las funciones definidas por el usuario cuentan con la estructura de un programa al cuál se le pueden definir parámetros de entrada y salida, constantes, reglas de validación y por supuesto sentencias SQL de cualquier tipo (Ordaz-Rosado, 2014, pág. 149).

7. Vistas

190. ¿Qué es una vista?

Una vista en SQL es una tabla que se deriva de otras tablas. Estas otras tablas pueden ser tablas base o vistas previamente definidas. Una vista se considera como una tabla virtual, en contraste a las tablas base, cuyas tuplas siempre se almacenan físicamente en la base de datos. Esto limita las posibles operaciones de actualización que pueden ser aplicadas a las vistas, pero no proporciona ningún tipo de limitaciones en la consulta de una vista.

Podemos pensar en una vista como una forma de especificar una tabla a la que tenemos que hacer referencia con frecuencia, a pesar de que no puede existir físicamente (Elmasri, 2011 pág. 133).

Las vistas son relaciones que se definen por una consulta sobre otras relaciones, no se almacenan en la base de datos, pero se pueden consultar como si existieran (Ullman, 2009, pág. 341).

191. ¿Para qué se utilizan las vistas?

Hay muchas razones por lo que las vistas son de gran apoyo, algunas de ellas son:

- Proporcionan una capacidad de "macro" o abreviada.
- Permite no sobrecargar el rendimiento en tiempo de ejecución asignado en la utilización de una vista, ya que sólo hay una pequeña sobrecarga en el momento de la visualización de procesamiento.
- Permiten que los mismos datos puedan ser vistos por diferentes usuarios en diferentes formas al mismo tiempo.
- Permiten a los usuarios centrarse en solo la parte de la base de datos que es de interés para ellos e ignorar el resto. Esta consideración es obviamente importante cuando hay muchos usuarios diferentes, con necesidades diferentes donde todos interactúan simultáneamente con una sola base de datos integrada.
- Proporcionan seguridad al permitir ocultar datos. Por datos ocultos se refiere a los datos que no son visibles a través de alguna vista determinada. Por lo tanto, obligar a los usuarios a

acceder a la base de datos a través de vistas constituye un mecanismo de seguridad simple pero eficaz.

- Actúan como un atajo y por lo tanto para hacer la vida más fácil para el usuario.

(Date, 2004, pág. 341)

7.1. Declaración de vistas

192. Muestra la sintaxis en SQL para crear una vista.

La forma más simple para crear una vista es:

```
CREATE VIEW OR REPLACE VIEW nombre_de_la_vista AS definicion_de_la_vista
```

Donde `definicion_de_la_vista` es una consulta. Este caso sirve para cualquier consulta siempre y cuando ésta no tenga parámetros.

7.2. Consultas sobre vistas

193. ¿Cómo se utilizaría una vista en una consulta?

Supóngase que la vista `mi_vista` existe, ésta puede ser utilizada en una consulta de la misma manera como se utiliza una tabla cualquiera.

194. Muestra la sintaxis en SQL para usar una vista.

La vista `mi_vista` se utilizaría en una consulta de la siguiente manera:

```
SELECT <lista de atributos>
FROM mi_vista
WHERE <condicion>
```

7.3. Actualización de vistas

195. Muestra la sintaxis en SQL para modificar una vista[21].

Una vista puede ser modificada a través del comando `ALTER`. Entre los elementos a modificar de una vista se incluyen:

- El propietario de ésta:

```
ALTER VIEW nombre_de_la_vista OWNER TO nuevo_propietario
```

- El nombre de ésta:

```
ALTER VIEW nombre_de_la_vista RENAME TO nuevo_nombre
```

- El esquema de ésta:

```
ALTER VIEW nombre_de_la_vista SET SCHEMA nuevo_esquema
```

Una inserción en la vista se puede aplicar directamente a la relación subyacente R:

```
INSERT INTO nombre_de_la_vista VALUES (atributo1, …, atributoN);
```

Existen algunas reglas de SQL que permiten modificaciones a las tuplas que una vista arroja. Para que esto sea posible, la vista debe cumplir lo siguiente:

- La cláusula `WHERE` no debe contener una subconsulta.
- La cláusula `FROM` sólo puede aparecer una vez.
- La lista en la cláusula `SELECT` debe incluir suficientes atributos de modo tal que, para cada tupla insertada en la vista, podamos llenar los otros atributos con valores `NULL` o el valor por defecto apropiado.

(Ullman, 2009, pág. 345-347)

196. Muestra la sintaxis en SQL para eliminar una vista.[22]

Es posible eliminar una vista de una base de datos con el comando `DROP VIEW`. La sintaxis para ello sería:

```
DROP VIEW nombre_de_la_vista;
```

[21] PosgreSQL Documentation, http://www.postgresql.org/docs/9.5/static/sql-alterview.html (Consultada el 05/12/2016).
[22] PosgreSQL Documentation, http://www.postgresql.org/docs/9.5/static/sql-dropview.html (Consultada el 05/12/2016).

8. Integridad

197. ¿Qué se entiende por integridad?

Es la exactitud o corrección de los datos en la base de datos (Date, 2004, pág. 249).

Una restricción de integridad es una condición booleana que está asociada con una base de datos y que se requiere evaluar en todo tiempo como verdadera (Date, 2004, pág. 254).

198. ¿Por qué es importante la integridad en una base de datos?

Las restricciones de integridad garantizan que los cambios realizados en la base de datos por los usuarios autorizados no den lugar a una pérdida de consistencia de los datos. Por lo tanto, las restricciones de integridad ayudan a proteger contra daños accidentales a la base de datos (Silberschatz, 2006, pág. 229).

8.1. Integridad de entidad

199. ¿En qué consiste la integridad de entidad?

La restricción de integridad de entidad establece que el valor de una llave primaria no puede ser nulo; esto debido a que el valor de la llave primaria es usado para identificar de forma única a una tupla en una relación. El hecho de que la llave primaria asumiera valores nulos implicaría el no poder identificar algunas tuplas. Por ejemplo, si el valor de la llave primaria de dos o más tuplas fuera nulo, no seríamos capaces de distinguirlas si tratáramos de hacer referencia a ellas desde otras relaciones (Elmasri, 2011, pág. 73).

Para implementar en SQL la integridad de la entidad se utilizan las palabras reservadas `PRIMARY KEY`.

La cláusula `PRIMARY KEY` permite especificar uno o más atributos que conforman la llave primaria de una relación (Elmasri, 2011, pág. 95).

8.2. Integridad de dominio

200. ¿En qué consiste la integridad de dominio?

Las restricciones de dominio especifican el conjunto de valores posibles que pueden ser asociados a un atributo. Dichas restricciones también pueden prohibir el uso de valores nulos para atributos particulares.

Es el tipo más simple de integridad que restringe el tipo de dato para cada dato almacenado en la base. (Elmasri, 2011, pág. 21)

La integridad de dominio puede restringir los valores de un atributo, acotando el tipo de dato. Esto se logra utilizando la palabra reservada CHECK.

La cláusula CHECK evalúa si la condición es verdadera o falsa, si el valor que se inserta en la columna cumple con las condiciones, entonces permitirá la inserción de dicho valor (Elmasri, 2011, pág. 94).

8.3. Integridad referencial

201. ¿En qué consiste la integridad referencial?

Es un tipo de restricción que involucra específicamente que un registro en una relación se debe relacionar con registros en otras relaciones. Ésta se especifica entre dos relaciones y es usada para mantener la consistencia entre tuplas dentro de las dos relaciones (Elmasri, 2011, pág. 73).

Es una condición que sirve para asegurarnos que un valor que aparece en una relación de un conjunto de atributos dados también aparece para un cierto conjunto de atributos en otra relación. A menudo se tiene la necesidad de que un valor en una relación aparezca también para un conjunto determinado de atributos en otra relación, esta condición se denomina integridad referencial (Silberschatz, 2006, pág. 227).

Requiere que un valor que aparece en la columna de una relación, también aparezca en alguna otra columna de la misma o de una diferente relación (Ullman, 2009, pág. 58).

La base de datos no debe contener valores de llave foránea sin correspondencia (Date 2000, pág. 263).

Para vigilar la integridad referencial se hace uso de las llaves foráneas, las cuales permiten relacionar dos tablas. Las llaves foráneas pueden especificarse mediante la cláusula `FOREIGN KEY`.

8.4. Integridad de usuario

202. ¿En qué consiste la integridad de usuario?

Las reglas de integridad de usuario son aquellas reglas propias de la realidad que se intenta representar en la BD que se desea crear (Marco-Galindo, 2010, pág. 152).

Denominamos integridad a la propiedad de los datos de corresponder a representaciones plausibles del mundo real. Las restricciones de integridad de usuarios son condiciones específicas de una base de datos concreta; es decir, son las que se deben cumplir en una base de datos particular con unos usuarios concretos, pero no necesariamente relevantes en otra base de datos (Costal-Costa, 2009, pág. 21).

203. ¿Cómo se podría implementar en SQL la integridad del usuario?

La integridad de usuario es posible implementarla a través de lenguajes de programación de propósito general. Dentro del SMBD se pueden utilizar lenguajes como PL/SQL para construir funciones o procedimientos almacenados encargados de vigilar el cumplimiento de las reglas de negocio.

8.5. Integridad de no nulidad.

204. ¿En qué consiste la integridad de no nulidad?

La integridad de no nulidad actúa sobre un atributo. Se encarga de rechazar las tuplas en las cuales dicho atributo es nulo. El efecto es no permitir tuplas en la que dicho atributo sea nulo (Ullman, 2009, pág. 319).

La restricción es declarada mediante las palabras reservadas `NOT NULL` que siguen a la declaración del atributo en el que se aplicará.

8.6. Seguridad de la base de datos

205. ¿En qué consiste la seguridad en una base de datos?

La gran mayoría de los datos sensibles del mundo están almacenados en sistemas gestores de bases de datos, atacar bases de datos es uno de los objetivos para obtener información y darle mal uso.

Estas son aspectos sobre seguridad en bases de datos:

- Identificar la sensibilidad de los datos y sus tablas. Elaborar una clasificación de datos confidenciales.
- Evaluar la vulnerabilidad y la configuración de la base de datos: verificar la configuración de bases de datos, para asegurarse que no tiene huecos de seguridad, así como revisar la forma en que se instaló la base de datos y su sistema operativo.
- Revisar que la ejecución de la base de datos no sea con versiones que incluyen vulnerabilidades conocidas.
- Por parte del DBA estas tareas aseguran la integridad de la base de datos: Limitar el acceso a los procedimientos a ciertos usuarios; delimitar el acceso a los datos para ciertos usuarios, procedimientos y/o datos, declinar la coincidencia de horarios entre usuarios que coincidan.
- Eliminación de todas las funciones y opciones que no se utilicen. Aplicar una política estricta sobre que se puede y que no se puede hacer, pero asegurarse de desactivar lo que no necesita.
- Automatizar el control de la configuración de tal forma que se registre cualquier cambio en la misma. Implementar alertas sobre cambios en la configuración. Cada vez que un cambio se realice, éste podría afectar a la seguridad de la base de datos.
- Monitoreo en tiempo real de la actividad de base de datos es clave para limitar su exposición.

- Aplicar pistas de auditoría y generar seguimiento de las actividades que afectan la integridad de los datos, o la visualización de los datos sensibles.
- No todos los datos y no todos los usuarios son creados iguales. Se deben autenticar a los usuarios, garantizar la rendición de cuentas por usuario, y administrar los privilegios para delimitar el acceso a los datos.
- Utilizar cifrado para hacer ilegibles los datos confidenciales, es una buena herramienta de prevención.[23]

206. Muestra la sintaxis en SQL para crear un usuario.

El comando `CREATE USER` permite crear usuarios.

```
CREATE USER nombre_usuario;
```

(Ordaz-Rosado, 2014, pág. 167)

207. Muestra la sintaxis en SQL para asignar un rol a un usuario.

El comando `WITH` permite asignar un rol a un usuario.

```
CREATE USER nombre_usuario WITH
SUPERUSER
PASSWORD ´clave_para_ingresar´
VALID UNTIL ´2099-07.04´;
```

(Ordaz-Rosado, 2014, pág. 168)

208. Muestra la sintaxis en SQL para otorgar privilegios a un usuario.

Los SMBD permiten otorgar permisos específicos para las bases de datos y las tablas que contengan, así como sobre las acciones que se pueden realizar en estas. Es decir, los permisos específicos limitarán las posibilidades de un usuario para operar dentro del SMBD. Algunos de estos permisos son:

- **`SELECT`:** Consultas a tablas.

[23] Seguridad en bases de datos, http://revista.seguridad.unam.mx/numero-12/principios-basicos-de-seguridad-en-bases-de-datos (Consultada el 05/12/2016).

- **INSERT:** Inserción de nuevos registros
- **UPDATE:** Actualización o modificación de los registros existentes.
- **DELETE:** Eliminación de registros existentes.

Para poder definir estos permisos, utilizamos la siguiente sintaxis:

```
GRANT <lista_de_permisos_a_otorgar>
ON <lista_de_tablas_sobre_las_que_aplica>
TO <lista_de_usuarios>;
```

En esta sintaxis encontramos:

- `GRANT` Palabra reservada para comenzar a definir los permisos. Se traduce cómo conceder u otorgar.
- `<lista_de_permisos_a_otorgar>` Es la lista de los permisos que se quieren otorgar.
- `ON` Palabra reservada para indicar sobre cuáles tablas o bases de datos serán válidos los permisos que se están definiendo. Si se quiere definir una base de datos completa se deberá de agregar la palabra reservada `DATABASE`.
- `<lista_de_tablas_sobre_las_que_aplica>` Es la lista de las tablas sobre las que serán válidos los permisos.
- `TO` Palabra reservada para indicar a qué usuario, usuarios o grupo de usuarios se les otorgarán los permisos que se están definiendo.
- `<lista_de_usuarios>` Es la lista de los usuarios a los que se le otorgarán los permisos.

(Ordaz-Rosado, 2014. PÁG. 169)

209. Muestra la sintaxis en SQL para revocar privilegios a un usuario.

Para revocar una autorización, usamos la declaración `REVOKE`. Toma una forma casi idéntica a la declaración `GRANT`:

```
REVOKE <lista de privilegios> ON <nombre_del_objeto> FROM <usuario>
```

La revocación de un privilegio de un usuario/rol puede provocar que otros usuarios/ roles también pierdan esos privilegios. Este comportamiento es llamado revocación en cascada. La declaración revocar alternativamente puede especificar la restricción para prevenir la revocación en cascada (Silberschatz, 2001, pág. 250-251).

210. Muestra la sintaxis en SQL para crear un rol.

La sintaxis general para crear un rol es la siguiente:

```
CREATE ROLE nombre_usuario WITH <privilegios>;
```

Por ejemplo:

```
CREATE ROLE nombre_rol WITH CREATEDB;
```

(Ordaz-Rosado, 2014, pág. 167)

8.7. Respaldo de la base de datos

211. ¿En qué consiste el respaldo de una base de datos?

Consiste en mantener una copia de la base de datos separada de la base de datos. Diseñar un plan de respaldo y recuperación involucra decidir qué bases de datos respaldar, con qué frecuencia, dónde almacenar los respaldos, qué tan frecuentemente sobre-escribirlos y qué tan rápido necesitas recuperar la base de datos (Ullman, 2009, pág. 875).

212. Muestra la sintaxis en SQL para respaldar una base de datos

La sintaxis para respaldar una base de datos en PostgreSQL es la siguiente:

```
pg_dump nombre_bd_a_respaldar > respaldo.backup
```

8.8. Restauración de la base de datos

213. ¿En qué consiste la restauración de una base de datos?

La restauración de la base de datos es, por ejemplo, cuando el servidor falla en medio de una transacción de actualización compleja y el subsistema regresa la base de datos al estado en que se encontraba antes de que la operación comenzará a ejecutarse, esto garantiza que la operación se reanudará desde el punto en el que se interrumpió, de forma que su efecto se registrará en la base de datos (Elmasri, 2001 pág. 20).

La restauración de una base de datos consiste en que se debe llevar a cabo la recuperación de errores, es decir, detectar fallos del sistema y restaurar la base de datos al estado que existía antes de la aparición de la falla (Silberschatz, 2001 pág. 16).

214. Muestra la sintaxis en SQL para restaurar una base de datos.

La sintaxis para restaurar una base de datos en PostgreSQL es la siguiente:

```
pg_restore -d nombre_bd_nueva respaldo.backup
```

9. Procesamiento de transacciones

215. ¿Para qué sirve una transacción?

Una transacción es una unidad de la ejecución de un programa. Puede consistir en varias operaciones de acceso a la base de datos. Está delimitada por constructoras como `begin-transaction` y `end-transaction`. Los SMBD son sistemas concurrentes, su utilidad reside en permitir la ejecución concurrente de consultas.[24]

Una transacción proporciona un mecanismo para describir las unidades lógicas de los procesos de la base de datos. Este mecanismo debe ser concluido por completo para asegurar su precisión (Elmasri, 2001, pág. 743).

9.1. Concepto y problemas de una transacción

216. ¿Qué es una transacción?

Una transacción es una unidad lógica de trabajo en la base de datos. Puede ser un programa completo, fragmento de programa o comando único. Involucra cualquier número de operaciones en la base de datos. El concepto de transacción es fundamental para entender tanto la recuperación como el control de la concurrencia (Ricardo, 2009, pág. 364).

Las transacciones son una colección de operaciones que forman una sola unidad lógica de trabajo. Una transacción es una unidad de ejecución de programa que accede y posiblemente actualiza varios datos dentro de la BD (Silberschatz, 2006, pág. 565).

217. ¿Por cuáles estados puede pasar una transacción?

Los estados por los que puede pasar una transacción son:

- Activa

[24] Transacciones, https://www.fdi.ucm.es/profesor/fernan/DBD/apuntestema07.pdf (Consultada el 05/12/2016).

- Parcialmente llevada a cabo
- Llevada a cabo
- Fallida
- Terminada

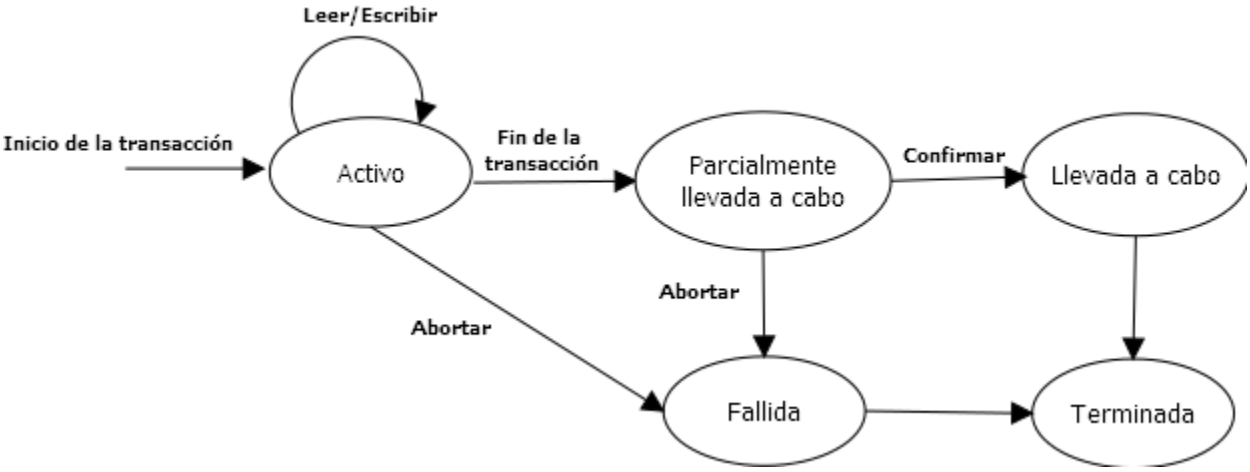


Fig. 29 – Estados de una transacción

218. ¿En qué consiste el estado Activa?

El estado *Activa* es en el cual la transacción permanece durante su ejecución (Silberschatz, 2006, pág. 613).

El estado activo de una transacción comienza con el enunciado `BEGIN TRANSACTION` y continúa hasta que el programa de aplicación aborta o concluye con éxito (Ricardo, 2009, pág. 365).

Una transacción pasa a estado activo inmediatamente después que inicia su ejecución, donde puede iniciar las operaciones de lectura o escritura. Cuando la transacción termina, pasa a un estado parcialmente llevada a cabo (Elmasri, 2001, pág. 752).

219. ¿En qué consiste el estado Parcialmente llevada a cabo?

También conocida como Parcialmente Comprometida, la transacción pasa a este estado cuando acaba de realizar la última instrucción[25].

[25] Estados de una transacción, https://chargers090187.wordpress.com/2011/06/25/estados-de-una-transaccion (Consultada el 05/12/2016).

Cuando una transacción acaba (END) pasa al estado de Parcialmente llevada a cabo. En este punto, algunas técnicas de control de concurrencia requieren la realización de ciertos chequeos para asegurar que la transacción no interfiere con otras transacciones que se ejecutan al mismo tiempo. Además, algunos protocolos de recuperación necesitan comprobar que un fallo del sistema no inhabilitará la grabación de los cambios de la transacción de forma permanente.[26]

Cuando la transacción termina de leer (READ) o escribir (WRITE) pasa al estado parcialmente completada (Elmasri, 200, pág. 752).

220. ¿En qué consiste el estado Llevada a cabo?

El estado *Llevada a cabo* se da cuando todas las operaciones que tienen acceso a la base de datos se han ejecutado con éxito y el efecto de todas las operaciones de la transacción en la base de datos se han registrado en la bitácora de transacciones (Elmasri, 2011, pág. 754).

221. ¿En qué consiste el estado Fallida?

Una transacción entra en el estado *Fallida* después de que el sistema determina que la transacción ya no puede procesarse más con su ejecución normal (por ejemplo, debido a errores de hardware o lógicos). Tal transacción debe ser abortada (Silberschatz, 2006, pág. 613).

Una transacción puede ir al estado fallida si alguna de las verificaciones falla o si la transacción es abortada durante su estado activo (Elmasri, 2011, pág. 752).

222. ¿En qué consiste el estado Terminada?

Una transacción está en estado *Terminada* tras completarse con éxito. En una transacción, cuando la última instrucción se escribe en el SMBD, la operación pasa a este estado.

De la misma forma, se dice que una transacción ha concluido si se encuentra en estado *Fallida* o *Terminada* (Silberschatz, 2006, pág. 613).

[26] Estados de una transacción, http://8team8a-blog.tumblr.com/post/45378981289/estados-de-una-transacción (Consultada el 05/12/2016).

9.2. Propiedades de una transacción

223. ¿Cuáles son las propiedades de una transacción?

Para asegurar la integridad de los datos, se requiere que el SMBD siga las siguientes propiedades de las transacciones, llamadas ACID por su acrónimo en inglés:

- Atomicity (atomicidad)
- Consistency (consistencia)
- Isolation (aislamiento)
- Durability (persistencia)

(Silberschatz, 2006, pág. 610)

224. ¿En qué consiste la propiedad de atomicidad?

La atomicidad consiste en que cualquier cambio de estado que produce una transacción es atómico, es decir, ocurren todos o no ocurre ninguno. Cualquiera de las operaciones de transacción deberán reflejarse correctamente en la base de datos, o ninguna lo hará. La atomicidad requiere que si una transacción se interrumpe por una falla, sus resultados parciales deben ser deshechos (Silberschatz, 2006, pág. 609).

225. ¿En qué consiste la propiedad de consistencia?

Una transacción debe preservar la consistencia, lo que significa que si es ejecutada completamente de inicio a fin sin interferencia de otras transacciones, debe llevar a la base de datos de un estado consistente a otro (Elmasri 2011 pág. 754).

226. ¿En qué consiste la propiedad de aislamiento?

Es la propiedad de una transacción para parecer que se ejecuta de manera aislada de otras transacciones a pesar, de que muchas de las transacciones se ejecuten concurrentemente. Esto es, la ejecución de una transacción no debe ser interferida por ninguna transacción que se ejecutan simultáneamente (Elmasri 2011 pág. 754).

227. ¿En qué consiste la propiedad de persistencia?

La propiedad de persistencia asegura que después de que una transacción se completa exitosamente, los cambios que se hayan hecho a la base de datos persisten, aún si hay fallas en el sistema (Silberschatz 2011pág. 628).

9.3 Control de concurrencia

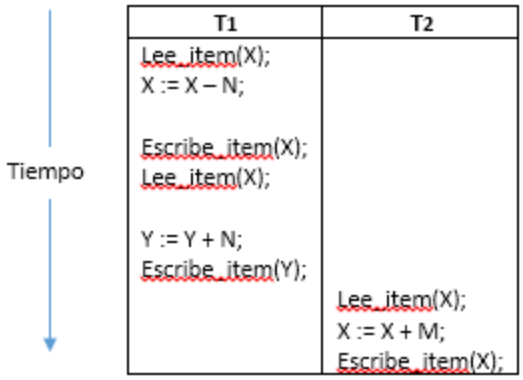
228. Qué es la planeación (schedule) de una transacción.

Es una secuencia de las acciones realizadas por una o más transacciones. Cuando se estudia el control de concurrencia, las acciones de escritura y de lectura tienen lugar en los buffers de la memoria principal, no del disco. Es decir, un elemento de base de datos A, el cual se llevó a algún buffer por alguna transacción T puede ser leído o escrito en ese buffer no solo por la transacción T, sino también por otras transacciones que tienen acceso al elemento de la base de datos A (Ullman, 2009, pág. 884).

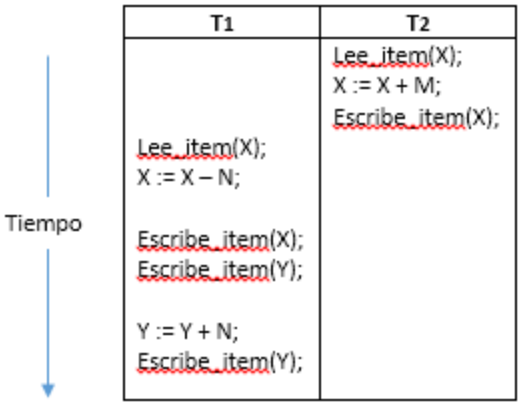
Una secuencia (*S*) de n transacciones T_1, T_2, T_3,…T_n es un ordenamiento de operaciones de las transacciones. Las operaciones de diferentes transacciones pueden ser intercalados en la secuencia S. sin embargo para cada transacción T_i en *S* deben aparecer en el mismo orden en el cual ocurren en T_i. El orden de las operaciones en *S* se considera que es un orden total, lo que significa que para dos operaciones en la secuencia, una ocurre antes que otra. Es posible -teóricamente-, manejar secuencias cuyas operaciones son órdenes parciales (Elmasri, 2011, pág. 756).

Ejemplos de planeaciones seriales y no seriales que involucran a la transacción T1 y T2.

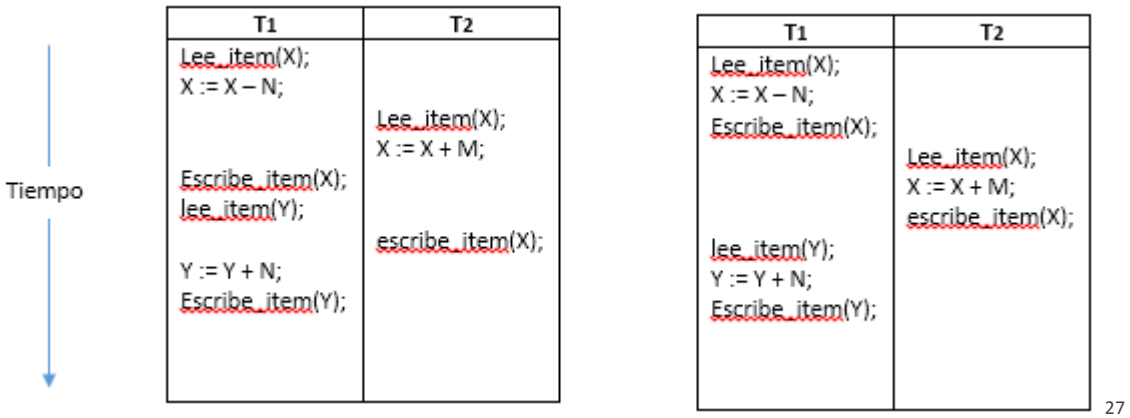
a) Planeación serial A: T1 seguido de T2:



b) Planeación serial B: T2 seguido de T1:



c) Dos planeaciones no-seriales C y D con operaciones entrelazadas:



[27]

[27] Ejemplos de transacciones, http://pld.cs.luc.edu/database/images/fig21.5.png (Consultada el 5/12/2016).

229. ¿Qué significa que una operación de una transacción esté en conflicto?

Se dice que dos operaciones de una planificación entran en conflicto si satisfacen estas tres condiciones:

1. Pertenecen a transacciones diferentes.
2. Acceden al mismo elemento.
3. Al menos una de las operaciones es de escritura.

(Date, 2007, pág. 536)

230. ¿Qué tipos de conflictos existen?

Dos operaciones son contradictorias si un cambio en su orden, producen resultados diferentes:

- **Conflicto de lectura-escritura: s**i cambiamos el orden de dos operaciones $r_1(X)$; $w_2(X)$ por $w_2(X)$; $r_1(X)$, entonces el valor de *X* que es leído por la transacción T_1 cambia, porque en el segundo orden el valor de *X* es cambiado por $w_2(X)$ antes de que sea leído por $r_1(X)$, mientras que en el primer orden el valor es leído antes de que cambie.

- **Conflicto escritura-escritura:** cuando se cambia el orden de dos operaciones como $w_1(X)$; $w_2(X)$ *a* $w_2(X)$; $W_1(X)$. Para este conflicto el último valor de *X* es diferente porque en un caso es escrito por T_2 y en otro caso por T_1. Observemos que dos operaciones de escritura no están en conflicto pues al cambiar su orden no hacen ninguna diferencia en el resultado obtenido.

(Elmasri, 2011, pág. 756)

231. ¿En qué consiste el conflicto escritura-escritura?

Ocurre cuando dos transacciones actualizan el valor de un mismo dato. Y por tanto una actualización sobrescribe a la otra (Shiva, 2006. pág. 98).

232. ¿En qué consiste una transacción serial?

Formalmente, una secuencia *S* es serial si, para cada transacción *T* que participan en el programa, todas las operaciones de *T* se ejecutan consecutivamente en la ejecución. De otra manera la transacción es no-serial (Elmasri, 2011, pág. 761).

233. ¿En qué consiste la violación de lectura sucia?

Este problema se produce cuando una operación de una transacción realiza una actualización y la transacción no llega a completarse con éxito por algún problema (caída del sistema, problemas en la red, etc.), y otra transacción utiliza el valor actualizado antes de que el elemento actualizado por la transacción fallida se restaure a su valor original. Se le denomina dato sucio, porque está creado por una transacción que no ha finalizado aún, dando lugar al problema de la lectura sucia[28].

Una transacción T_1 puede leer la actualización de la transacción T_2, la cual todavía no termina. Si T_2 falla y se aborta, entonces T_1 podría haber leído un valor que no existe y es incorrecto (Elmasri, 2011, pág. 770).

Por ejemplo, supongamos que A y B son dos transacciones, A escribe un registro y B quiere leerlo. Si B está autorizada para hacer la lectura del registro y éste no está siendo utilizado por otra transacción entonces realiza la lectura. En el caso que el registro este siendo utilizado por otra transacción y ésta no ha sido finalizada con COMMIT, entonces B estaría leyendo un dato erróneo.

En otras palabras, una lectura sucia, sucede cuando una transacción lee la información que está siendo modificada por otra transacción pero que todavía no ha sido finalizada con el comando COMMIT.[29]

234. ¿En qué consiste la violación de Lectura No Repetible?

Una transacción T_1 lee un valor dado de una tabla. Si otra transacción T_2 actualiza posteriormente ese valor y T_1 lee otra vez el mismo valor; T_1 puede ver un valor diferente (Elmasri, 2011, pág. 771).

Por ejemplo, si se lee un registro en una transacción R1 en un tiempo t_1 mientras otra transacción R2 actualiza el registro que ha sido leído por R1 en t_2, mostrando un resultado diferente cuando R1 vuelve a consultar el mismo registro en un tiempo t_3[30].

[28] Lectura sucia, http://indalog.ual.es/mtorres/BD/bdt6.pdf (Consultada el 05/12/2016).
[29] Lectura sucia, http://www.elconspirador.com/2014/05/29/problemas-tradicionales-de-los-bloqueos-de-transacciones-y-concurrencia-en-la-multiversion (Consultada el 05/12/2016).
[30] Lectura no repetible, http://www.elconspirador.com/2014/05/29/problemas-tradicionales-de-los-bloqueos-de-transacciones-y-concurrencia-en-la-multiversion (Consultada el 05/12/2016).

235. ¿En qué consiste la violación de Lectura Fantasma?

Una lectura fantasma ocurre cuando, durante el curso de una transacción, se ejecutan dos consultas idénticas y la colección de filas retornada por la segunda es diferente a la primera. Esto puede ocurrir cuando no se adquieren bloqueos de rango al ejecutar una operación `SELECT….WHERE.`

Las lecturas fantasma son un caso especial de las lecturas no-repetibles, cuando la transacción 1 repite la consulta rango `SELECT….WHERE` y, entre ambas consultas, la transacción 2 crea `(INSERT)` nuevas filas que cumplen con la condición `WHERE.` (Date, 2004, pág. 482).

Por ejemplo, si dos consultas iguales en una misma transacción devuelven información diferente, esto significa que se realizó una inserción por otra transacción en el intervalo de tiempo entre una y otra consulta[31]

9.4 Manejo de transacciones en SQL

236. Muestra la sintaxis en SQL para comenzar una transacción.

`BEGIN` permite iniciar un bloque de instrucciones, es decir, todas las declaraciones después de un comando `BEGIN` serán ejecutadas en una sola transacción hasta que aparezca de manera explícita `COMMIT` o `ROLLBACK.`

237. Muestra la sintaxis en SQL para que los cambios hechos por una transacción sean permanentes.

Usando la sentencia `COMMIT` la transacción es completada y los cambios hechos se guardan de manera permanente en la base de datos (Silberschatz, 2006, pág. 110).

`COMMIT_TRANSACTION` señala la finalización exitosa de una transacción de modo que los cambios ejecutados por la transacción son guardados en la base de datos y no se pueden deshacer (Elmasri, 2011, pág. 752).

[31] Lectura fantasma, http://www.elconspirador.com/2014/05/29/problemas-tradicionales-de-los-bloqueos-de-transacciones-y-concurrencia-en-la-multiversion (Consultada el 05/12/2016).

La sentencia `COMMIT` en SQL finaliza con éxito la transacción y después de dicha sentencia los cambios dejan de ser provisionales y se vuelven permanentes en la base de datos (Ullman, 2009, pág. 300).

238. Muestra la sintaxis en SQL para cancelar los cambios hechos por una transacción.

Para borrar todas las modificaciones de datos realizadas desde el inicio de la transacción o hasta un punto de retorno se usa `ROLLBACK`. Este comando también libera los recursos usados durante la transacción.

Conclusiones

Las presentes notas de clase, forman una guía de conceptos que, basados en el plan de la materia de Fundamentos de Base de Datos, pretende orientar al estudiante en la búsqueda de la información fundamental del tema.

Para elaborar este trabajo, primero tuve que comprender los temas que iba a incluir de acuerdo al temario de la materia, hacer un consenso de las preguntas y descartar muchas que estaban fuera del alcance o eran redundantes. Durante el desarrollo de estas notas no solamente mejoré el conocimiento que tenía sobre las bases de datos, además me involucré en el proceso de producción de material didáctico desde el inicio: buscar la bibliografía, desarrollar las preguntas, buscar y analizar las respuestas. Ello me llevó a elegir el esquema de pregunta-respuesta, que creo que ayudará al lector a comprender mejor cada término teórico, además de que facilita la búsqueda de términos muy específicos, ya que se incluyen las referencias para que el lector -si es de su interés- consulte la fuente directa y complemente su conocimiento. Incluir diversas respuestas a una misma pregunta permite al alumno comparar las respuestas entre diferentes autores.

La necesidad de aprender y comprender las bases de datos se hace cada día más importante, por lo que es necesario aprenderlo, incluso de manera autodidacta, por tal razón espero que este compendio de conocimiento sustentado en fuentes reconocidas sea útil para aquellos que se interesen en conocer más sobre la teoría de las bases de datos.

Con la evolución de la tecnología y otras alternativas didácticas, se puede hacer mejoras a este manual, que incluyan nueva información, como Minería de Datos, Inteligencia de Negocios, entre otros temas. En cuanto a las alternativas didácticas este trabajo podría convertirse en un objeto de aprendizaje portable que pueda ser visualizado por otros medios más dinámicos y de mayor alcance. Vale la pena agregar que, durante la etapa de revisión por parte de los sinodales de este trabajo, se hizo notoria la necesidad de una herramienta que permitiera almacenar las preguntas y respuestas de este trabajo. Por tal razón se construyó un sistema web[32] que sirviera como plataforma

[32] Accesible a través de 132.247.127.131:8080/Examenes/ con el usuario “apv” (Consultada el 09/01/2017).

de autoevaluación para los alumnos. En dicho sistema los alumnos pueden practicar respondiendo las preguntas que el sistema va presentando y de esta manera conocer el avance de su aprendizaje.

Este trabajo, tiene mucha de la formación intelectual que obtuve dentro de la carrera, sobre todo la manera en como el origen de problemas matemáticos se pueden resolver por medio de su aplicación a la ciencia de la computación. A pesar de ser un trabajo que está dedicado a la materia de Bases de Datos, también utilicé conocimientos adquiridos en otras materias que van desde el Análisis de Algoritmos hasta la Ingeniería de Software.

Espero que los estudiantes encuentren en este material una fuente de información que los estimule a conocer más específicamente la ciencia de los datos y sobretodo de sus aplicaciones. Recuerden que la necesidad de información es y será infinita; el perfeccionamiento de las técnicas de administración de datos, proporcionará bases para que las decisiones de cualquier tipo sean tomadas con fundamentos y de esta manera sean funcionales.

Referencias

(Codd, 1970) Codd, E.: *A relational Model of Data for Large Shared Data Banks.* Communications of the ACM, Vol. 13, No. 6 (1970)

(Codd, 1985) Codd, E.: *Is Your DBMS really relational?* Computerworld, Octubre 1985. http://each.uspnet.uspág. br/sarajane/wp-content/uploads/2015/08/appendix-A-Codds-12-rules-for-an-rdbms-1-7.pdf

(Codd, 1990) Codd, E.: *The relational model for database management*. Addison-Wesley. Reading, MA, EUA. ISBN: 0-201-14192-2 (1990)

(Coronel, 2011) Coronel, C., Morris, S., Rob, P.: *Bases de Datos: diseño, implementación y administración*. CENGAGE Learning, México. ISBN: 978-607-481-618-1 (2011)

(Costal-Costa, 2009) Costal-Costa, D.: *El modelo relacional y el álgebra relacional.* UOC Barcelona (2009)

(Date, 2004) Date, C. J., *An introduction to database systems.* Boston; México: Pearson Education. ISBN: 032-119-784-4 (2004)

(Elmasri, 2011) Elmasri, R., Navathe, S.: *Fundamentals of database systems.* Boston: Addison-Wesley. ISBN: 978-013-608-620-8 (2011)

(García-Molina, 2009) García-Molina, H., Ullman, J., Widom, J.: *Database systems: the complete book*. Pearson Prentice Hall. ISBN: 978-013-187-325-4 (2009)

(Hansen & Hansen, 1997) Hansen, G., Hansen, J.: *Diseño y Administración de Bases de Datos.* Prentice-Hall, España. ISBN: 978-848-3220-02-3 (1997)

(Heath, 1971) Heath, I.: *Unacceptable File Operations in a Relational Data Base*. In Proceedings ACM SIGFIDET Workshop on Data Description, Access and Control. ACM New York, NY, EUA, pp.: 19-33 (1971)

(Jiménez, 2014) Jiménez, M.: *Bases de datos relacionales y modelado de datos.* IC Editorial, España. ISBN: 978-84-16433-30-8 (2014)

(Kroenke, 2003) Kroenke, D.: *Procesamiento de bases de datos: fundamentos, diseño e implementación*. Prentice-Hall, México. ISBN: 970-26-0325-0 (2003)

(Kuhns, 1967) Kuhns, J.*: Answering Questions by Computer: A Logical Study.* RAND Corporation. RM-5428-PR (1967)

(Marco-Galindo, 2010) Marco-Galindo, J.: *Escaneando la informática*. Editorial UOC Barcelona, ISBN: 978-84-9788-110-4 (2010)

(Marqués, 2011) Marqués, M.: *Bases de Datos.* Departament d'Enginyeria i Ciència dels Computadors, Universitat Jaume I, España. ISBN: 978-84-693-0146-3 (2011)

(Matthew, 2005) Matthew, N., Stones, R.: *Beginning Databases with PostgreSQL*. Springer, NY, EUA. ISBN: 159-059-478-9 (2005)

(Nevado, 2010) Nevado, M.: *Introducción a las bases de datos relacionales.* Visión Libros, España. ISBN: 978-84-9886-809-8 (2010)

(Oppel, 2010) Oppel, A., Sheldon, R.: *Fundamentos de SQL*. McGraw-Hill Interamericana. México. ISBN: 978-607-150-254-4 (2010)

(Ordaz-Rosado, 2014) Ordaz-Rosado, D.: *Prácticas para el curso de Bases de Datos: Análisis, Diseño, Construcción, Explotación y Mantenimiento.* Facultad de Ciencias, UNAM (2014)

(Osorio-Rivera, 2008) Osorio-Rivera, F.: *Bases de Datos Relacionales Teoría y Práctica.* Fondo Editorial ITM, Colombia. ISBN: 978-958-8351-42-1 (2008)

(Piñeiro, 2014) Piñeiro, J.: *Definición y manipulación de datos.* Ediciones Paraninfo, España. ISBN: 978-84-283-9824-4 (2014)

(Ricardo, 2009) Ricardo, C*.: Bases de Datos.* McGraw-Hill, México. ISBN 13: 978-970-10-7275-2 (2009)

(Rumbaugh et al., 1999) Rumbaugh, J., Jacobson, I., Booch, G.: *The Unified Modeling Language Reference Manual.* Addison Wesley. ISBN: 0-201-30998-X (1999)

(Sánchez, 2004) Sánchez, J.: *Principios sobre Bases de Datos Relacionales.* http://www.jorgesanchez.net/ (2004)

(Sherman, 2015) Sherman, R.: *Business Intelligence Guidebook: From Data Integration to Analytics.* El Sevier – Morgan Kauffman. Waltham, MA, EUA. ISBN 978-0-12-411461-6 (2015)

(Shiva, 2006) Shiva, S.: *Advanced computer architectures.* Boca Raton: CRC Press/Taylor & Francis. ISBN: 084-933-758-5 (2006)

(Silberschatz, 2002) Silberschatz, A., Korth, F., Sudarshan, S.: *Database system concepts*. McGraw-Hill. New York, NY, EUA. ISBN: 0-07-228363-7 (2002)

(Silberschatz, 2006) Silberschatz, A., Korth, F., Sudarshan, S.: *Database system concepts.* Boston: McGraw-Hill Higher Education. ISBN: 007-124476-X (2006)

(Silberschatz, 2011) Silberschatz, A., Korth, F., Sudarshan, S.: *Database system concepts.* McGraw-Hill. New York, NY, EUA. ISBN: 978-0-07-352332-3 (2011)

(Taboada & Cotos, 2005) Taboada-González, J., Cotos-Yáñez, J.: *Sistemas de Información Medioambiental.* Netbiblo. España. ISBN: 84-9745-056-6 (2005)

(Ullman, 2002) Ullman, J., Widom, J.: *A first course in database systems*. Upper Saddle River, NJ, EUA. ISBN: 013600637X Prentice Hall (2002)