ANTOLOGIA

BASE DE DATOS I

INGENIERIA EN SISTEMAS COMPUTACIONALES

SEPTIMO CUATRIMESTRE

Marco Estratégico de Referencia

ANTECEDENTES HISTORICOS

Nuestra Universidad tiene sus antecedentes de formación en el año de 1979 con el inicio de actividades de la normal de educadoras "Edgar Robledo Santiago", que en su momento marcó un nuevo rumbo para la educación de Comitán y del estado de Chiapas. Nuestra escuela fue fundada por el Profesor de Primaria Manuel Albores Salazar con la idea de traer Educación a Comitán, ya que esto representaba una forma de apoyar a muchas familias de la región para que siguieran estudiando.

En el año 1984 inicia actividades el CBTiS Moctezuma Ilhuicamina, que fue el primer bachillerato tecnológico particular del estado de Chiapas, manteniendo con esto la visión en grande de traer Educación a nuestro municipio, esta institución fue creada para que la gente que trabajaba por la mañana tuviera la opción de estudiar por las tarde.

La Maestra Martha Ruth Alcázar Mellanes es la madre de los tres integrantes de la familia Albores Alcázar que se fueron integrando poco a poco a la escuela formada por su padre, el Profesor Manuel Albores Salazar; Víctor Manuel Albores Alcázar en septiembre de 1996 como chofer de transporte escolar, Karla Fabiola Albores Alcázar se integró como Profesora en 1998, Martha Patricia Albores Alcázar en el departamento de finanzas en 1999.

En el año 2002, Víctor Manuel Albores Alcázar formó el Grupo Educativo Albores Alcázar S.C. para darle un nuevo rumbo y sentido empresarial al negocio familiar y en el año 2004 funda la Universidad Del Sureste.

La formación de nuestra Universidad se da principalmente porque en Comitán y en toda la región no existía una verdadera oferta Educativa, por lo que se veía urgente la creación de una institución de Educación superior, pero que estuviera a la altura de las exigencias de los jóvenes que tenían intención de seguir estudiando o de los profesionistas para seguir preparándose a través de estudios de posgrado.

Nuestra Universidad inició sus actividades el 18 de agosto del 2004 en las instalaciones de la 4ª avenida oriente sur no. 24, con la licenciatura en Puericultura, contando con dos grupos de cuarenta alumnos cada uno. En el año 2005 nos trasladamos a nuestras propias instalaciones en la carretera Comitán – Tzimol km. 57 donde actualmente se encuentra el campus Comitán y el Corporativo UDS, este último, es el encargado de estandarizar y controlar todos los procesos operativos y Educativos de los

diferentes Campus, Sedes y Centros de Enlace Educativo, así como de crear los diferentes planes estratégicos de expansión de la marca a nivel nacional e internacional.

Nuestra Universidad inició sus actividades el 18 de agosto del 2004 en las instalaciones de la 4ª avenida oriente sur no. 24, con la licenciatura en Puericultura, contando con dos grupos de cuarenta alumnos cada uno. En el año 2005 nos trasladamos a nuestras propias instalaciones en la carretera Comitán – Tzimol km. 57 donde actualmente se encuentra el campus Comitán y el corporativo UDS, este último, es el encargado de estandarizar y controlar todos los procesos operativos y educativos de los diferentes campus, así como de crear los diferentes planes estratégicos de expansión de la marca.

MISIÓN

Satisfacer la necesidad de Educación que promueva el espíritu emprendedor, aplicando altos estándares de calidad Académica, que propicien el desarrollo de nuestros alumnos, Profesores, colaboradores y la sociedad, a través de la incorporación de tecnologías en el proceso de enseñanza-aprendizaje.

VISIÓN

Ser la mejor oferta académica en cada región de influencia, y a través de nuestra Plataforma Virtual tener una cobertura Global, con un crecimiento sostenible y las ofertas académicas innovadoras con pertinencia para la sociedad.

VALORES

- Disciplina
- Honestidad
- Equidad
- Libertad

ESCUDO

El escudo de la UDS, está constituido por tres líneas curvas que nacen de izquierda a derecha formando los escalones al éxito. En la parte superior está situado un cuadro motivo de la abstracción de la forma de un libro abierto.

ESLOGAN

"Mi Universidad"

ALBORES

Es nuestra mascota, un Jaguar. Su piel es negra y se distingue por ser líder, trabaja en equipo y obtiene lo que desea. El ímpetu, extremo valor y fortaleza son los rasgos que distinguen.

BASE DE DATOS I

Objetivo de la materia:

Objetivo: Identificar los fundamentos generales de las bases de datos, y la importancia de la seguridad de los datos en los sistemas de información, así como su adecuada administración

UNIDAD I. CONCEPTOS Y OBJETIVOS DE LA BASE DE DATOS

1.1. Antecedentes
1.2. Objetivos de las bases de datos
1.3 Áreas de aplicación de los sistemas de bases de datos
1.4- Elementos de una base de datos
1.5 Concepto de datos
1.6 Concepto de campo, registro y archivo
1.7 Clasificación de bases de datos
1.8 Arquitectura del sgbd
1.9 Tipos de usuarios
1.10 Tipos de lenguajes
1.11 Tópicos selectos de base de datos
1.12 Mostrar Propietario de Tablas

UNIDAD II. DISEÑO DE BASES DE DATOS Y EL MODELO E-R

2.1 El proceso de diseño
2.2. El modelo entidad relación
2.3. Tipos de relaciones
2.4. Conjunto de entidades débiles
2.5. La notación e-r con uml
2.7 Otros aspectos del diseño de bases de datos
2.8 Restricciones
2.9 Diseño con Diagrama E-R
2.10 Modelos de Datos Semánticos
2.11 Intervalos de Valores

UNIDAD III. NORMALIZACIÓN DE BASES DE DATOS Y ALGEBRA RELACIONAL

3.1. Normalización de base de datos
3.2 Primera forma normal 1fn
3.3. Segunda forma normal. (2fn)
3.4. Tercera forma normal. (3fn)
3.5. Forma normal boyce-codd. / Cuarta forma normal (4fn)
3.6. Tercera forma normal 3fn en relaciones transitivas
3.7 La tabla o relación

3.8. Operaciones fundamentales del álgebra relacional
3.9. Valores nulos.
3.10. Operaciones de modificación a la base de datos
3.11.- tipos de almacenamiento de datos.
3.12. Seguridad, integridad y confidencialidad de datos.
3.13 Valores Nulos
3.14 Tablas de Verdad
3.15 Integridad de entidad

UNIDAD IV. LENGUAJE SQL

4.1. Introducción
4.2. Lenguaje de manipulación de datos
4.3. Lenguaje de manipulación de datos (dml)
4.4. Operaciones con conjuntos
4.5 Valores nulos
4.6. Consultas anidadas
4.7 Componentes de SQL
4.8 SQL Server
4.9 Ventajas de SQL
4.10 Componentes del SQL
4.11 DDL(Data Definition Language)

Indice

UNIDAD 1. CONCEPTOS Y OBJETIVOS DE LA BASE DE DATOS

1.1. ANTECEDENTES

Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada base de datos, contiene información relevante para una empresa.

El término de bases de datos fue escuchado por primera vez en 1963, en un simposio celebrado en California, USA. Una base de datos se puede definir como un conjunto de información relacionada que se encuentra agrupada o estructurada.

Desde el punto de vista informático, la base de datos es un sistema formado por un conjunto de datos almacenados en discos que permiten el acceso directo a ellos y un conjunto de programas que manipulen ese conjunto de datos.

Cada base de datos se compone de una o más tablas que guarda un conjunto de datos. Cada tabla tiene una o más columnas y filas. Las columnas guardan una parte de la información sobre cada elemento que queramos guardar en la tabla, cada fila de la tabla conforma un registro.

1.2. OBJETIVOS DE LAS BASES DE DATOS

Disminuir la redundancia e inconsistencia de los datos: Puesto que los archivos y los programas de aplicaciones fueron creados por distintos programadores en un periodo largo,

es posible que un mismo dato esté repetido en varios sitios (archivos). Esta redundancia aumenta los costos de almacenamiento y acceso, además de incrementar la posibilidad de que exista inconsistencia en la información.

Reducir la dificultad para tener acceso a los datos: Supóngase que uno de los gerentes del banco necesita averiguar los nombres de todos los clientes que viven en cierta parte de la ciudad. El gerente llama al departamento de procesamiento de datos y pide que generen la lista correspondiente. Como ésta es una solicitud fuera de lo común no existe un programa de aplicaciones para generar semejante lista. Lo que se trata de probar aquí es que este ambiente no permite recuperar la información requerida en forma conveniente o eficiente.

Evitar el aislamiento de los datos: Puesto que los datos están repartidos en varios archivos, y éstos pueden tener diferentes formatos, es difícil escribir nuevos programas de aplicaciones para obtener los datos apropiados.

Corregir anomalías en el acceso concurrente: Para mejorar el funcionamiento del sistema y tener un tiempo de respuesta más corto, muchos sistemas permiten que varios usuarios actualicen la información simultáneamente. En un ambiente de este tipo, la interacción de las actualizaciones concurrentes puede resultar en información inconsistente. Para prevenir estas situaciones debe mantenerse alguna forma de supervisión en el sistema.

Disminuir los problemas de seguridad: No es recomendable que todos los usuarios del sistema de base de datos puedan tener acceso a toda la información. Por ejemplo, en un sistema bancario, una persona que prepare los cheques de nómina sólo debe poder ver la parte de la base de datos que contenga información de los empleados. No puede consultar información correspondiente a las cuentas de los clientes.

Disminuir los problemas de integridad: Los valores que se guardan en la base de datos debe satisfacer ciertos tipos de *limitantes de consistencia*. El sistema debe obligar al cumplimiento de estas limitantes. Esto puede hacerse agregando el código apropiado a los distintos programas de aplicaciones. El problema se complica cuando las limitantes implican varios elementos de información de distintos archivos.

· Acceso concurrente por parte de múltiples usuarios.

· Integridad de los datos.

· Consultas complejas optimizadas.

· Seguridad de acceso y auditoria.

· Respaldo y recuperación.

· Acceso a través de lenguaje de programación estándar.

1.3 ÁREAS DE APLICACIÓN DE LOS SISTEMAS DE BASES DE DATOS

Las bases de datos son ampliamente usadas. Las siguientes son algunas de sus aplicaciones más representativas:

• Banca. Para información de los clientes, cuentas y préstamos, y transacciones bancarias.

• Líneas aéreas. Para reservas e información de planificación. Las líneas aéreas fueron de los primeros en usar las bases de datos de forma distribuida geográficamente (los terminales situados en todo el mundo accedían al sistema de bases de datos centralizado a través de las líneas telefónicas y otras redes de datos).

• Universidades. Para información de los estudiantes, matrículas de las asignaturas y cursos.

•Transacciones de tarjetas de crédito. Para compras con tarjeta de crédito y generación mensual de extractos.

•Telecomunicaciones. Para guardar un registro de las llamadas realizadas, generación mensual de facturas, manteniendo el saldo de las tarjetas telefónicas de prepago y para almacenar información sobre las redes de comunicaciones.

• Finanzas. Para almacenar información sobre grandes empresas, ventas y compras de documentos formales financieros, como bolsa y bonos.

• Ventas. Para información de clientes, productos y compras.

• Producción. Para la gestión de la cadena de producción y para el seguimiento de la producción de elementos en las factorías, inventarios de elementos en almacenes y pedidos de elementos.

• Recursos humanos. Para información sobre los empleados, salarios, impuestos y beneficios, y para la generación de las nóminas. Como esta lista ilustra, las bases de datos forman una parte esencial de casi todas las empresas actuales.

A lo largo de las últimas cuatro décadas del siglo veinte, el uso de las bases de datos creció en todas las empresas. En los primeros días, muy pocas personas interactuaron directamente con los sistemas de bases de datos, aunque sin darse cuenta interactuaron con bases de datos indirectamente (con los informes impresos como extractos de tarjetas de crédito, o mediante agentes como cajeros de bancos y agentes de reserva de líneas aéreas). Después vinieron los cajeros automáticos y permitieron a los usuarios interactuar con las bases de datos. Las interfaces telefónicas con los computadores (sistemas de respuesta vocal interactiva) también permitieron a los usuarios manejar directamente las bases de datos. Un llamador podía marcar un número y pulsar teclas del teléfono para introducir información o para seleccionar opciones alternativas, para determinar las horas de llegada o salida, por ejemplo, o para matricularse de asignaturas en una universidad. La revolución de Internet a finales de la década de 1990 aumentó significativamente el acceso directo del usuario a las

bases de datos. Las organizaciones convirtieron muchas de sus interfaces telefónicas a las bases de datos en interfaces Web, y pusieron disponibles en línea muchos servicios. Por ejemplo, cuando se accede a una tienda de libros en línea y se busca un libro o una colección de música se está accediendo a datos almacenados en una base de datos. Cuando se solicita un pedido en línea, el pedido se almacena en una base de datos. Cuando se accede a un banco en un sitio Web y se consulta el estado de la cuenta y los movimientos, la información se recupera del sistema de bases de datos del banco. Cuando se accede a un sitio Web, la información personal puede ser recuperada de una base de datos para seleccionar los anuncios que se deberían mostrar. Más aún, los datos sobre Considérese parte de una empresa de cajas de ahorros que mantiene información acerca de todos los clientes y cuentas de ahorros. Una manera de mantener la información en un computador es almacenarla en archivos del sistema operativo. Para permitir a los usuarios manipular la información, el sistema tiene un número de programas de aplicación que manipula los archivos, incluyendo:

• Un programa para efectuar cargos o abonos en una cuenta.

• Un programa para añadir una cuenta nueva.

• Un programa para calcular el saldo de una cuenta.

• Un programa para generar las operaciones mensuales.

Estos programas de aplicación se han escrito por programadores de sistemas en respuesta a las necesidades de la organización bancaria. Si las necesidades se incrementan, se añaden nuevos programas de aplicación al sistema. Por ejemplo, supóngase que las regulaciones de un nuevo gobierno permiten a las cajas de ahorros ofrecer cuentas corrientes. Como resultado se crean nuevos archivos permanentes que contengan información acerca de todas las cuentas corrientes mantenidas por el banco, y puede ser necesario escribir nuevos programas de aplicación para tratar situaciones que no existían en las cuentas de ahorro,

tales como manejar descubiertos. Así, sobre la marcha, se añaden más archivos y programas de aplicación al sistema. Este sistema de procesamiento de archivos típico que se acaba de describir se mantiene mediante un sistema operativo convencional. Los registros permanentes son almacenados en varios archivos y se escriben diferentes programas de aplicación para extraer registros y para añadir registros a los archivos adecuados. Antes de la llegada de los sistemas de gestión de bases de datos (SGBDs), las organizaciones normalmente han almacenado la información usando tales sistemas. los accesos Web pueden ser almacenados en una base de datos. Así, aunque las interfaces de datos ocultan detalles del acceso a las bases de datos, y la mayoría de la gente ni siquiera es consciente de que están interactuando con una base de datos, el acceso a las bases de datos forma una parte esencial de la vida de casi todas las personas actualmente. La importancia de los sistemas de bases de datos se puede juzgar de otra forma: actualmente, los vendedores de sistemas de bases de datos como Oracle están entre las mayores compañías software en el mundo, y los sistemas de bases de datos forman una parte importante de la línea de productos de compañías más diversificadas, como Microsoft e IBM.

1.4- ELEMENTOS DE UNA BASE DE DATOS

Los principales elementos de una base de datos son los siguientes:

Tablas: Es el elemento principal de la base de datos, ya que allí se registra la información que se quiere administrar. Está compuesta, como una hoja de cálculo, por filas y columnas. Cada archivo de una base de datos puede contener una o millones de tablas como sea necesario.

Formularios: La información que se introduce a la base de datos puede introducirse directamente en las tablas, pero también pude ser a través de un formulario, esto lo que resulta práctico, Los formularios hacen que sea más fácil introducir los datos.

Consultas: Este elemento que se emplea para buscar y seleccionar la información que requiere el usuario del interior de la base de datos. La consulta, nos permite establecer los criterios de búsqueda para que el software que administra la información seleccione, dentro de las tablas, aquellos datos que se quieren conocer.

Informes: Es te elemento se utilizan para que la información que nos arrojan las búsquedas nos aparezca ordenada y bien presentada para cuando el usuario demande una impresión del documento. Gracias a los informes, el usuario puede seleccionar que información, de la que se registró en las tablas de una base de datos, desea imprimir y con qué formato.

1.4.1 CONCEPTO DE DATOS

Cifra, letra o palabra que se suministra a la computadora como entrada y la máquina almacena en un determinado formato

En informática, los datos son representaciones simbólicas (vale decir: numéricas, alfabéticas, algorítmicas, etc.) de un determinado atributo o variable cualitativa o cuantitativa, o sea: **la descripción codificada de un hecho empírico**, un suceso, una entidad.

Los datos son, así, la información (valores o referentes) que recibe el computador a través de distintos medios, y que es manipulada mediante el procesamiento de los algoritmos de programación. **Su contenido puede ser prácticamente cualquiera**: estadísticas, números, descriptores, que por separado no tienen relevancia para los usuarios del sistema, pero que en conjunto pueden ser interpretados para obtener una información completa y específica.

Tipos de datos

En la informática, cuando hablamos de tipos de datos (o simplemente "tipo") nos referimos a un atributo que se indica al computador respecto a la naturaleza de los datos que se dispone a procesar. Esto incluye delimitar o restringir los datos, definir los valores que pueden tomar, qué operaciones se puede realizar con ellos, etc.

Algunos tipos de datos son:

- **Caracteres**. Dígitos individuales que se pueden representar mediante datos numéricos (0-9), letras (a-z) u otros símbolos.
- **Caracteres unicode**. Unicode es un estándar de codificación que permite representar más eficazmente los datos, permitiendo así hasta 65535 caracteres diferentes.
- **Numéricos**. Pueden ser números reales o enteros, dependiendo de lo necesario.
- **Booleanos**. Representan valores lógicos (verdadero o falso)

Campo: En informática, un campo es un espacio de almacenamiento para un dato en particular. En las bases de datos, un campo es la mínima unidad de información a la que se puede acceder; un campo o un conjunto de ellos forman un registro, donde pueden existir campos en blanco, siendo éste un error del sistema operativo. En las hojas de cálculo los campos son llamados celdas. Podemos agregar muchos tipos de campos que formarán parte de nuestra base de datos. Cada campo admite distintos tipos de datos, con diferentes interfaces. Después de crear los campos deseados podemos añadir información en la base de datos.

En las bases de datos, un campo es la mínima unidad de información a la que se puede acceder; un campo o un conjunto de ellos forman un registro, donde pueden existir campos

en blanco, siendo este un error del sistema operativo. Aquel campo que posee un dato único para una repetición de entidad, puede servir para la búsqueda de una entidad en específico.

Tipos de campos informáticos

- **Alfanumérico**: contiene cifras numéricas y caracteres alfabéticos.
- **Numérico**: existen de varios tipos principalmente como enteros y reales.
- **Auto-incrementable**: son campos numéricos enteros que incrementan en una unidad su valor para cada registro incorporado. Su utilidad resulta más que evidente: servir de identificador registro.
- Booleano: admite dos valores, «verdadero» ó «falso».
- Fechas: almacenan fechas facilitando posteriormente su explotación. Almacenar fechas de esta forma posibilita ordenar los registros por fechas o calcular los días entre una fecha y otra.
- **Memo:** son campos alfanuméricos de longitud ilimitada. Presentan el inconveniente de no poder ser indexados.

1.4.2 CONCEPTO DE CAMPO, REGISTRO Y ARCHIVO

Registro: Un registro informático es un tipo o conjunto de datos almacenados en un sistema**.**

En primer lugar, un registro del sistema viene a ser una base de datos que tiene el fin de almacenar configuración, opciones y comandos propios del sistema operativo. Un registro de sistema puede contener información y configuraciones del hardware y software en uso, preferencias del usuario, asociaciones de archivos y ficheros, usos de sistema, cambios y

modificaciones, etcétera. Estos registros son conservados dentro del sistema con denominaciones como "User.dat" o "System.dat" y pueden ser recuperados por el usuario para su transporte a otro sistema.

Otro tipo de registro es el de programación. Este tipo de dato está formado por varios elementos en asociación que responden a una misma estructura. Los registros de programación pueden ser elementales o complejos y guardan información sobre cómo el software o aplicación en particular funcionará o actuará en cada momento.

Por otro lado, en una base de datos también se hace uso de registros. Cada registro representa un ítem o elemento único que se encuentra en una tabla, hoja o base. Así, el registro está configurado por el conjunto de datos que pertenecen a una entidad en particular.

En todos estos casos y otros, el empleo de registros tiene el fin de almacenar información y datos, ponerla en relación y colocarla al alcance bajo un índice o sistema de orden que permita su acceso y uso en cualquier momento. Los registros son el método que tanto el usuario como el sistema informático utilizan para acceder y utilizar toda la información.

Archivo: En el campo de la informática, se llama "archivo" al **elemento de información compuesto por una suma de registros** (combinaciones de bytes). Llevan este nombre por ser los equivalentes digitalizados de los archivos antes descriptos. Tanto es así que muchos de los archivos "en papel" se están actualmente digitalizando, para reducir su tamaño físico y facilitar su organización y búsqueda. Los archivos informáticos, en general, tienen algunas características en común:

- **Nombre.** Cada archivo es identificable con un nombre, que no puede coincidir con otro que esté en la misma ubicación.

- **Extensión.** Los archivos llevan una extensión opcional, que muchas veces indica su formato.

- **Tamaño.** Como se dijo, están compuestos por una serie de bytes que determinan su tamaño. Puede alcanzar kilobytes, megabytes, gigabytes.

- **Descripción.** Además del nombre y la extensión, suelen tener otras características. Dentro de estas características puede aparecer la protección del archivo, lo que significa el permiso limitado para la lectura o modificación.

- **Ubicación.** Todos los archivos pertenecen a determinado lugar en la computadora (o circunstancialmente fuera de ella), el llamado espacio de almacenamiento. La mayoría se encuentran almacenados en discos rígidos, que están ordenados. jerárquicamente en carpetas y subcarpetas. Existe necesariamente una ruta de acceso hacia ese lugar, que comienza con el disco al que se hace referencia (C:, D:).

- **Formato.** El modo en que el archivo será interpretado depende de su formato, entre los que están los formatos de texto, ejecutable, de datos, de imagen, de audio, de video, entre muchísimos otros.

La forma en la que las computadoras organizan los archivos suelen llamarse sistemas de archivos y dependen del sistema operativo con el que se esté trabajando. Los archivos pueden ser ejecutables o no ejecutables, según si funcionan en sí mismos (como puede ser un juego de PC) o si necesitan de otra aplicación que los cargue (como un documento de Word).

Definición de Archivos de Bases de Datos

Los archivos de bases de datos almacenan datos en formato estructurado, organizados en tablas y campos que permiten su localización y acceso más rápido.

Las entradas individuales dentro de una base de datos se llaman registros. Una base de datos es una colección de registros relacionados lógicamente, las bases de datos se utilizan para el almacenamiento de datos referenciados por aplicaciones de software o por sitios web dinámicos.

En informática, se conoce como archivo o fichero a un **conjunto organizado de unidades de información (bits) almacenados en un dispositivo**. Se les denomina de esa manera como metáfora a partir de los archivos tradicionales de oficina, escritos en papel, ya que vendrían a ser su equivalente digital.

Cada archivo posee una identificación única o nombre, la cual puede ser modificada o asignada a voluntad del usuario o del programador, y una extensión que determina qué tipo de archivo es y qué funciones cumple. **Usualmente ambos términos de su nombre están separados por un punto**, por ejemplo: *Command.com*

Dentro de los archivos existen paquetes pequeños de datos expresados en bits (la unidad informática más pequeña que existe) y que se ordenan en registros o líneas, siendo individualmente distintos pero con algún rasgo común. El modo de agrupación de esta información depende de quién haga el archivo, por lo que **existen numerosas estructuras de archivo**, más simples y más complejas, que están más o menos estandarizadas hoy día.

Estas unidades mínimas de operación y organización de un Sistema Operativo que son los archivos, entonces, **se pueden crear, eliminar, reubicar, comprimir, renombrar y activar** (*ejecutar*, en lenguaje informático), junto con otras operaciones básicas de organización.

1.5 CLASIFICACIÓN DE BASES DE DATOS

Las bases de datos se pueden clasificar de acuerdo a los tipos de contenido: bibliográficas, texto, numéricos, imágenes. Los datos de una base de datos se organizan en base a un esquema o modelo, que es el formato o estructura de la misma. Los modelos de bases de datos comunes incluyen jerárquico, red, relacional, entidad-relación, objeto-relacional, o modelo de objetos.

Bases de Datos más conocidas: Los tipos más conocido de bases de datos para servidores son: Microsoft SQL Server, MySQL, PostgreSQL, Firebird, SQLite, Oracle e IBM DB2. El otro grupo común de base de datos que pueden utilizarse localmente son: Microsoft Access, Microsoft Visual FoxPro, dBASE y FileMaker.

Extensiones comunes: Extensiones de los archivos de bases de datos comunes incluyen .DB, .accdb, NSF, y .fp7.

Lista de extensiones: A continuación, una lista de las extensiones de archivos de base de datos:

Los archivos pueden tener numerosas funciones. Desde simplemente contener información de manera ordenada, como los archivos de texto, y permitir el acceso a ella por parte de programas determinados, hasta **archivos ejecutables que desencadenan cierta secuencia de acciones** (y de otros archivos) que tienen como resultado una acción concreta.

Desde apagar el computador hasta iniciar un videojuego, todo lo que ocurre en un sistema informático ocurre a través de archivos interconectados ejecutándose por turno en la memoria del computador.

Características generales de un archivo

En líneas generales, los archivos de un sistema informático son:

- **Representables**. Los archivos suelen tener un nombre de máximo 255 caracteres y suelen ser representados en sistemas operativos de interfaz gráfica (como Windows) por un ícono determinado.
- **Únicos por directorio**. En una misma carpeta o directorio no pueden existir dos archivos idénticos con el mismo nombre. Cuando ello ocurra alguno de los dos habrá de cambiar levemente su nombre o en todo caso será remplazado uno por otro.
- **Modificables**. Excepto aquellos que expresamente hayan sido protegidos contra modificación, como es el caso de los archivos vitales del sistema informático, que no deben sufrir cambios pues éste se desestabilizaría, lo común es que los archivos puedan borrarse, crearse, modificarse, renombrarse a voluntad o necesidad.
- **Poseen un tamaño**. De acuerdo a la cantidad de información que un archivo contenga, éste tendrá un tamaño o "peso", mensurable en Kb, Mb o incluso Gb. Mientras más grande sea el archivo, más capacidad deberá tener el soporte donde se encuentre.

Algunos ejemplos típicos de archivos son:

- **Archivos de texto**. Usualmente identificados con extensiones .doc, .txt, .rtf o .odt, contienen secuencias de caracteres alfanuméricos dispuestos en secuencias específicas, que denominamos "documentos".
- **Archivos ejecutables**. Usualmente terminados en .exe (*executable*, "ejecutable" en inglés), .com (*command*, "comando") o .bat (*batch*, "lote"), son aquellos que disparan las acciones, como ejecutar una aplicación o un videojuego.
- **Archivos de imagen**. Apellidados .jpg, .gif o .tiff normalmente, son imágenes cuya información recompuesta se traduce en una imagen, ilustración o fotografía.

Bases de datos estáticas

Base de datos dinámicas

Base de datos bibliográficos

Base de datos jerárquicos

Base de datos red

Base de datos transacciones

Base de datos documentales

Sistema de datos deductivos

Base de datos estáticas: Fundamentada solo en leyendas o escritos para ser leídos, además, son datos significativos para proyectos y tomas de decisiones que a futuro puedan

ser necesarios. En efecto, son datos que carecen de dinamismo o mutaciones, esto para tener una perspectiva clara en cuanto a tomar alguna decisión importante, tomando en consideración hechos del pasado para mejoras del futuro.

Base de datos dinámicas: Como su mismo nombre lo indica están en constante movimiento, recopilando datos nuevos y suprimiendo los más antiguos. Ahora bien, con estos datos que constantemente están en afán de renovarse permitirá que alguna empresa pueda tener actualizadas sus operaciones de consulta. Siendo útil para tiendas de constantes cambios de productos.

Según su contenido

Base de datos bibliográficos: Cabe señalar que en estos datos encontrara referencias del publicador como autor intelectual del contenido de las publicaciones consultadas.

Base de datos jerárquicos: Según la selección este va a ser descendente o descendente, en esta base de dato encontrar coordinado los datos almacenados dependiendo la cantidad de búsquedas a través de un caudal de información. Base de datos red: Debido a la retórica y la enorme cantidad de datos encontrados en una base de datos esta es solo recomendada o mejor utilizada por los profesionales en la programación.

Base de datos transacciones: En este campo se maneja la velocidad, eficacia y eficiencia, asimismo estos datos deben ser procesos rápidos y neutrales, verbigracia, las transacciones manejadas por los bancos, los cuales, solo maneja dos puntos o líneas como lo son, un emisor y un receptor.

Ahora bien, entre ellos van a surgir movimientos de tipo transacción, por lo tanto, como se ha mencionado, esta deberá ser generada eficaz y rápidamente, no obstante, si surgen

inconvenientes, estas deben tener solo dos opciones, transacción fallida o finalizada, nunca en proceso, o de reanudación de transacción.

Base de datos documentales: En esta clasificación encontramos al más potente de los datos, ya que este busca la información de forma excautiva y con alta probabilidad de resultados positivos en la búsqueda de información.

Sistema de datos deductivos: La lógica matemática es utilizada en este tipo de sistemas, permitiendo así deducir y relacionar información buscada por el usuario.

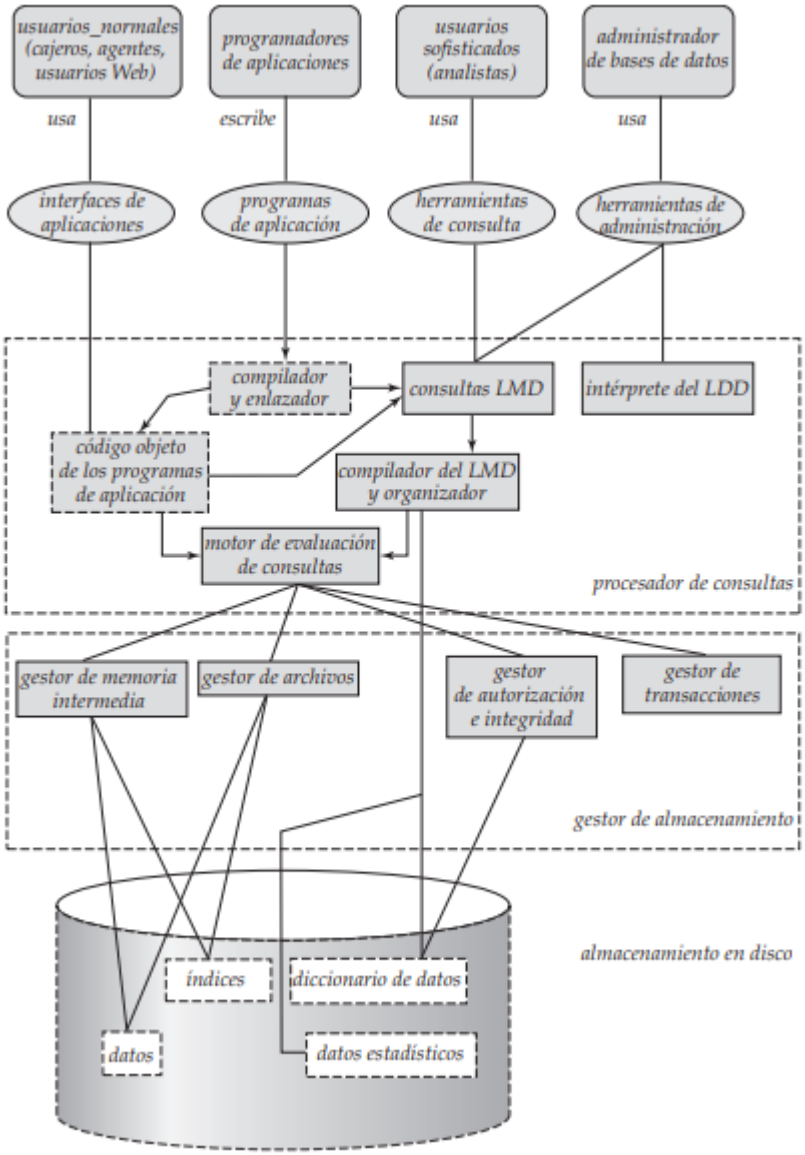
1.6 ARQUITECTURA DEL SGBD

Ahora es posible ofrecer una visión única de los diversos componentes de los sistemas de bases de datos y de las conexiones existentes entre ellos.

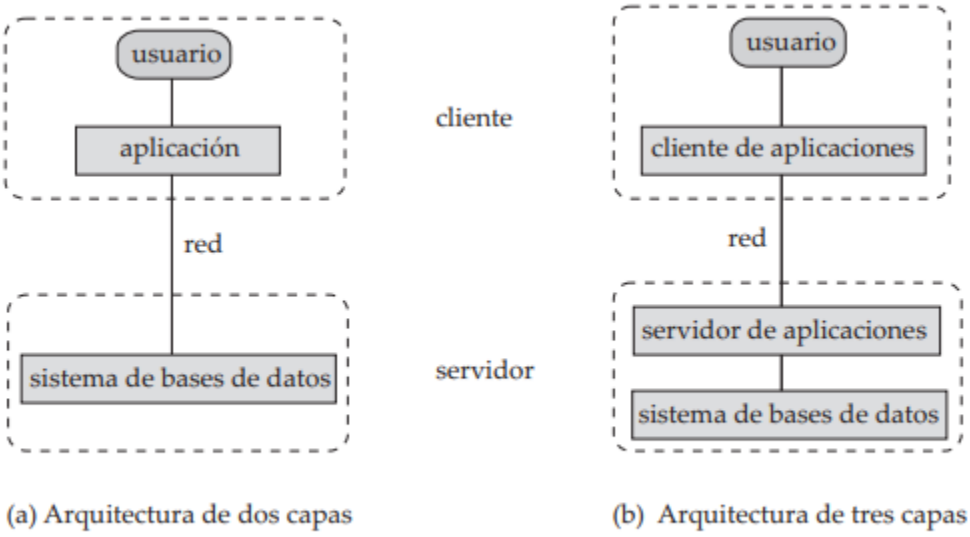
La arquitectura de los sistemas de bases de datos se ve muy influida por el sistema informático subyacente sobre el que se ejecuta el sistema de bases de datos. Los sistemas de bases de datos pueden estar centralizados o ser del tipo cliente-servidor, en los que una máquina servidora ejecuta el trabajo en nombre de multitud de máquinas clientes. Los sistemas de bases de datos pueden diseñarse también para aprovechar las arquitecturas de computadoras paralelas.

Entre los problemas se encuentran el modo de almacenar los datos, la manera de asegurar la atomicidad de las transacciones que se ejecutan en varios sitios, cómo llevar a cabo controles de concurrencia y el modo de ofrecer alta disponibilidad en presencia de fallos. El procesamiento distribuido de las consultas y los sistemas de directorio también se describen en ese capítulo. Hoy en día la mayor parte de los usuarios de los sistemas de bases de dato no está presente en el lugar físico en que se encuentra el sistema de bases de datos, sino que se conectan a él a través de una red. Por tanto, se puede diferenciar entre los sistemas

clientes, en los que trabajan los usuarios remotos de la base de datos, y los sistemas servidores, en los que se ejecutan los sistemas de bases de datos. Las aplicaciones de bases de datos suelen dividirse en dos o tres partes, como puede verse en la Figura 1.7.



En una arquitectura de dos capas, la aplicación se divide en un componente que reside en la máquina cliente, que llama a la funcionalidad del sistema de bases de datos en la máquina servidora mediante instrucciones del lenguaje de consultas. Los estándares de interfaces de programas de aplicación como ODBC y JDBC se usan para la interacción entre el cliente y el servidor. En cambio, en una arquitectura de tres capas, la máquina cliente actúa simplemente como una parte visible al usuario y no contiene ninguna llamada directa a la base de datos. En vez de eso, el extremo cliente se comunica con un servidor de aplicaciones, generalmente mediante una interfaz de formularios. El servidor de aplicaciones, a su vez, se comunica con el sistema de bases de datos para tener acceso a los datos. La lógica de negocio de la aplicación, que establece las acciones que se deben realizar según las condiciones reinantes, se incorpora en el servidor de aplicaciones, en lugar de estar distribuida entre múltiples clientes. Las aplicaciones de tres capas resultan más adecuadas para aplicaciones de gran tamaño y para las aplicaciones que se ejecutan en World Wide Web.



Arquitectura de dos y tres capas

NOTA: Para mayor información consultar: Capítulo 20 se trata la arquitectura general de los sistemas informáticos modernos. El Capítulo 21 describe el modo en que diversas acciones de las bases de datos, en especial el procesamiento de las consultas, pueden implementarse para aprovechar el procesamiento paralelo. El Capítulo 22 presenta del libro Funamentos de bases de datos, Stlberschatz, Korth, Sudarshan, Mc Graw Hill, 5ª. Edición

http://mateo.pbworks.com/w/file/fetch/122276985/Fundamentos_de_Bases_de_Datos_5a_Ed.-_Si.pdf

1.7 TIPOS DE USUARIOS

Un usuario es todo aquel que tenga contacto con el sistema de bases de datos. Se tienen 3 clases generales de usuarios:

1. Programador de aplicaciones
2. Usuario final
3. Administrador de bases de datos o DBA

1. **Programador de aplicaciones:** son aquellos profesionales en informática que interactúan con el sistema a través del DML (Lenguaje de Manipulación de Datos), los cuales se encuentran en un lenguaje de programación (Pascal, Cobol, etc.) Es el encargado de escribir programas de aplicación que usen Bases de Datos.

2. **Usuario Final:** accede a la base de datos desde un equipo en el cual puede utilizar lenguaje de consulta generado como parte del sistema o acude a un programa de aplicación suministrado por un programador.

3. **Administrador de Bases de Datos:** es el encargado del control general del sistema.

Todo usuario que ingrese o consulte una base de datos puede clasificarse:

Programador de Aplicaciones.

Usuario sofisticado: interactúa con el sistema sin escribir programas. Generan consultas en un lenguaje de bases de datos.

Usuario Especializado: algunos usuarios sofisticados desarrollan aplicaciones de bases de datos especializadas. Entre estas aplicaciones se encuentran los sistemas de diseño asistido por computador.

Usuarios ingenuos: es el usuario final que utiliza bases de datos sin saberlo, para él es totalmente transparente como se generan las consultas de la información.

Quienes diseñan y participan en el mantenimiento de un BD se les clasifica como Actores en el escenario y Trabajadores tras bambalinas

Actores en el escenario: personas que su trabajo depende del uso constante una base de datos.

DataBase Administrators(DBA): administran 2 recursos 1. la base de datos y 2. es el SGBD y el software con el relacionado. El Administrador de Base de Datos (DBA) es quien autoriza el acceso a la base de datos, vigilar el uso y adquirir hardware y software necesarios para su uso. También es el responsable de velar por la seguridad y lentitud en el sistema.

Diseñador de Base de Datos: es el encargado de estructurar la arquitectura para representar y almacenar los datos. Él debe atender a los usuarios de Bases de Datos para comprender sus necesidades presentando un diseño que de respuesta a sus necesidades.

Usuarios Finales: son quienes requieren acceso a la base de datos para generar consultas e informes. Hay varios usuarios finales como son:

Usuarios finales esporádicos: acceden de vez en cuando, pero esto no significa que siempre requieran la misma información.

Usuarios finales simples o paramétricos: su función gira en torno a consultas y actualizaciones de la base de datos. Todos estamos acostumbrados a tratar con estos usuarios, como los cajeros bancarios al revisar los saldos, al generar retiros y depósitos.

Usuarios finales avanzados: estos son ingenieros, analistas de negocios, científicos, son quienes conocen los recursos del SGBD para satisfacer requerimientos complejos.

Usuarios Autónomos: utilizan bases de datos personalizadas basadas en programas comerciales que cuentas con interfaces de fácil uso.

Analista de sistemas y programadores de aplicaciones: determinan los requerimientos de los usuarios finales.

Trabajadores tras bambalinas: están para mantener el sistema de base datos.

Diseñadores e implementadores del SGBD: se encarga de diseñar e implementar los módulos e interfaces de SGBD. Un Sistema de Gestión de Base de Datos consta de varios componentes y módulos.

1.8. TIPOS DE LENGUAJES

Tipos de lenguajes de Programación

Existen tres tipos de lenguajes claramente diferenciados; el lenguaje máquina y los lenguajes de bajo nivel y los de alto nivel.

1° **El Lenguaje Máquina:** es el lenguaje de programación que entiende directamente la máquina (computadora). Este lenguaje de programación utiliza el alfabeto binario, es decir, el 0 y el 1.

2° **Lenguajes de programación de bajo nivel:** Son mucho más fáciles de utilizar que el lenguaje máquina, pero dependen mucho de la máquina o computadora como sucedía con el lenguaje máquina.

3° **Lenguajes de programación de alto nivel.** Los lenguajes de programación de alto nivel son más fáciles de aprender porque se usan palabras o comandos del lenguaje natural, generalmente del inglés. Este es el caso del BASIC, el lenguaje de programación más conocido.

Tipos de lenguajes de programación de alto nivel según el punto de vista de trabajar los programas y la filosofía de sus creaciones:

- Lenguaje imperativo: entre ellos tenemos el Cobol, Pascal, C y Ada.
- Lenguaje declarativo: el Lisp y el Prolog.
- Lenguaje de programación orientado a objetos: el Smalltalk y el C++.
- Lenguaje orientado al problema: son aquellos lenguajes específicos para gestión.
- Lenguaje de programación natural: son los nuevos lenguajes que pretender aproximar el diseño y la construcción de programas al lenguaje de las personas.

Otra clasificación de los lenguajes de programación de alto nivel, es teniendo en cuenta el desarrollo de las computadoras según sus diferentes generaciones:

- Lenguaje de programación de primera generación: el lenguaje máquina y el ensamblador.
- Lenguaje de segunda generación: los primeros lenguajes de programación de alto nivel imperativo (FORTRAN, COBOL).
- Lenguaje de tercera generación: son lenguajes de programación de alto nivel imperativo, pero mucho más utilizados y vigentes en la actualidad (ALGOL 8, PL/I, PASCAL, MODULA).
- Lenguaje de cuarta generación: usados en aplicaciones de gestión y manejo de bases de datos (NATURAL, SQL).
- Lenguaje de quinta generación: creados para la inteligencia artificial y para el procesamiento de lenguajes naturales (LISP, PROLOG).
- Lenguajes de bases de datos Los sistemas de bases de datos proporcionan un lenguaje de definición de datos para especificar el esquema de la base de datos y un lenguaje

de manipulación de datos para expresar las consultas y las modificaciones de la base de datos. En la práctica, los lenguajes de definición y manipulación de datos no son dos lenguajes diferentes; en cambio, simplemente forman parte de un único lenguaje de bases de datos, como puede ser el muy usado SQL.

- Lenguaje de manipulación de datos Un lenguaje de manipulación de datos (LMD) es un lenguaje que permite a los usuarios tener acceso a los datos organizados mediante el modelo de datos correspondiente o manipularlos.

- Lenguaje de definición de datos Los esquemas de las bases de datos se especifican mediante un conjunto de definiciones expresadas mediante un lenguaje especial denominado lenguaje de definición de datos (LDD). El LDD también se usa para especificar más propiedades de los datos. La estructura de almacenamiento y los métodos de acceso usados por el sistema de bases de datos se especifican mediante un conjunto de instrucciones en un tipo especial de LDD denominado lenguaje de almacenamiento y definición de datos. Estas instrucciones definen los detalles de implementación de los esquemas de las bases de datos, que suelen ocultarse a los usuarios. Los valores de los datos almacenados en la base de datos deben satisfacer ciertas restricciones de consistencia.

1.9. TÓPICOS SELECTOS DE BASE DE DATOS

Un tópico es una idea o un tema en específico, en este caso bases de datos, así que resumiremos los tópicos más importantes, o selectos de bases de datos.

Una base de datos es una biblioteca donde a de mantener listas de los libros que posee, de los usuarios que tiene de sus productos, ventas y empleados.

A este tipo de información se le llama datos.

Un gestor de base de datos es un programa que permite introducir y almacenar datos, ordenarlos y manipularlos. Organizarlos de manera significativa para que se pueda obtener información no visible como totales, tendencias o relaciones de otro tipo.

Debe permitir:

-Introducir datos

-Almacenar datos

-Recuperar datos y trabajar con ellos

Tabla o fichero, registro y campo, un programa de base de datos almacena la información que introducimos en forma de tablas como las que podemos ver, por ejemplo, en una lista telefónica.

Registro: es el concepto básico en el almacenamiento de datos. El registro agrupa la información asociada a un elemento de un conjunto y está compuesto por campos.

Tabla: conjunto de registros homogéneos con la misma estructura.

Tipos de base de datos: planas y relacionales, para hacer una base de datos que cumpla las funciones de listín telefónico necesitamos una sola tabla, pero puede haber casos en los que necesitemos más de una.

A esta forma de organizar la base de datos mediante distintas tablas relacionadas por campos comunes se le llama base de datos relacional.

No todos los programas de gestión de base de datos tienen esta capacidad de manejar bases de datos relacionales, por eso, antes de elegir uno deberemos considerar si necesitamos o no esta capacidad.

Esta base de datos relacional estará formada por tablas. Con la característica de que las mismas se relacionan entre sí mediante uno o más campos. Se puede decir que cada objeto de la realidad será una tabla en nuestra base de datos y que hay que buscar la manera de reflejar las relaciones antes mencionadas.

Para este tipo de bases de datos con múltiples usuarios aparecieron las llamadas bases de datos de red. Estas están situadas en un único ordenador –llamado servidor (generalmente ordenadores de gran potencia) y se puede acceder a ellas desde terminales u ordenadores con un programa que permita el acceso a ella –los llamados clientes–.

Los Gestores de bases de datos de este tipo permiten que varios usuarios hagan operaciones sobre ella al mismo tiempo: uno puede hacer una consulta al mismo tiempo que otro, situado en un lugar diferente, está introduciendo datos en la base.

Utilidad de una base de datos: Las tres cosas básicas que debe permitir un gestor de base de datos son: introducir datos, almacenarlos y recuperarlos.

Al mismo tiempo permiten otra serie de funciones que hacen de ellos herramientas incomparablemente superiores a los métodos tradicionales de almacenamiento de datos: archivadores, carpetas, etc.

Cualquier gestor debe permitir: ordenar los datos, realizar búsquedas, mostrar distintas vistas de los datos, realizar cálculos sobre ellos, resumirlos, generar informes a partir de ellos, importarlos y exportarlos.

Búsquedas

En los antiguos sistemas de archivo de datos si se quería buscar un conjunto determinado de registros era necesario tener los datos ordenados previamente por un criterio determinado

(por ejemplo, en los ficheros de biblioteca, por materia o autor). Luego visualmente y a mano, a menudo con gran trabajo y pérdida de tiempo, ir extrayendo los registros de uno en uno.

Al terminar de usarlos se tenía que seguir el proceso contrario. En el caso de que se quisiera hacer una búsqueda por un criterio diferente al del orden del archivo.

1.12 MOSTRAR PROPIETARIO DE TABLA.

Indique si un origen de datos necesita que se especifique el propietario de una tabla antes de acceder a ella. Anule la selección de esta opción para los orígenes de datos que no tienen este requisito.

Mostrar. Muestra las columnas del origen de datos a la que está conectado actualmente. Pulse en una de las opciones siguientes para personalizar la vista de las tablas disponibles:

- Pulse en **Tablas de usuario** para ver las tablas normales de la base de datos creadas por los usuarios de la base de datos.
- Pulse en **Tablas de sistema** para ver las tablas de la base de datos que posee el sistema (por ejemplo, las tablas que ofrecen información acerca de la base de datos, como detalles de índices). Esta opción puede utilizarse para ver las pestañas utilizadas en las bases de datos de Excel. (Tenga en cuenta que también existe un nodo de origen de Excel independiente. Consulte el tema Nodo de origen de Excel si desea más información.)

UNIDAD II. DISEÑO DE BASES DE DATOSY EL MODELO E-R.

2.1 EL PROCESO DE DISEÑO

Fases del diseño Para aplicaciones pequeñas puede resultar factible para un diseñador de bases de datos que comprenda los requisitos de la aplicación decidir directamente sobre las relaciones que hay que crear, sus atributos y las restricciones sobre las relaciones. Sin embargo, un proceso de diseño tan directo resulta difícil para las aplicaciones reales, ya que a menudo son muy complejas. Frecuentemente no existe una sola persona que comprenda todas las necesidades de datos de la aplicación. El diseñador de la base de datos debe interactuar con los usuarios para comprender las necesidades de la aplicación, realizar una representación de alto nivel de esas necesidades que pueda ser comprendida por los usuarios y luego traducir esos requisitos a niveles inferiores del diseño. Los modelos de datos de alto nivel sirven a los diseñadores de bases de datos ofreciéndoles un marco conceptual en el que especificar de forma Diseño de bases de datos y el modelo E-R sistemática los requisitos de datos de los usuarios de la base de datos y una estructura para la base de datos que satisfaga esos requisitos.

• La fase inicial del diseño de las bases de datos es la caracterización completa de las necesidades de datos de los posibles usuarios de la base de datos. El diseñador de la base de datos debe interactuar intensamente con los expertos y los usuarios del dominio para realizar esta tarea. El resultado de esta fase es una especificación de requisitos del usuario.

• A continuación, el diseñador elige el modelo de datos y, aplicando los conceptos del modelo de datos elegido, traduce estos requisitos en un esquema conceptual de la base de datos. El esquema desarrollado en esta fase de diseño conceptual proporciona una visión detallada de la empresa. Se suele emplear el modelo entidad-relación,

Generalmente, la fase de diseño conceptual da lugar a la creación de un diagrama entidad-

relación que ofrece una representación gráfica del esquema. El diseñador revisa el esquema

para confirmar que realmente se satisfacen todos los requisitos y que no entran en conflicto entre sí. También puede examinar el diseño para eliminar características redundantes. Su atención en este momento se centra en describir los datos y sus relaciones, más que en especificar los detalles del almacenamiento físico.

• Un esquema conceptual completamente desarrollado indica también los requisitos funcionales de la empresa

• El proceso de paso desde el modelo abstracto de datos a la implementación de la base de datos se divide en de dos fases de diseño finales.

¤ **En la fase de diseño lógico** el diseñador traduce el esquema conceptual de alto nivel al modelo de datos de la implementación del sistema de bases de datos que se va a usar. El modelo de implementación de los datos suele ser el modelo relacional, y este paso suele consistir en la traducción del esquema conceptual definido mediante el modelo entidad-relación en un esquema de relación.

¤ Finalmente, **el diseñador usa el esquema de base de datos resultante** propio del sistema en la siguiente fase de diseño físico, en la que se especifican las características físicas de la base de datos. Entre estas características están la forma de organización de los archivos y las estructuras de almacenamiento interno

2.2. EL MODELO ENTIDAD RELACION

El modelo de datos entidad–relación (E-R) se desarrolló para facilitar el diseño de bases de datos permitiendo la especificación de un esquema de la empresa que representa la estructura lógica global de la base de datos. Es uno de los diferentes modelos de datos

semánticos; el aspecto semántico del modelo radica en la representación del significado de los datos, resulta muy útil para relacionar los significados e interacciones de las empresas reales con el esquema conceptual. Debido a esta utilidad, muchas herramientas de diseño de bases de datos se basan en los conceptos del modelo E-R. El modelo de datos E-R emplea tres conceptos básicos:

Los conjuntos de entidades, Una entidad es una "cosa" u "objeto" del mundo real que es distinguible de todos los demás objetos

Los conjuntos de relaciones: Una relación es una asociación entre varias entidades

Los atributos: son propiedades descriptivas que posee cada miembro de un conjunto de entidades

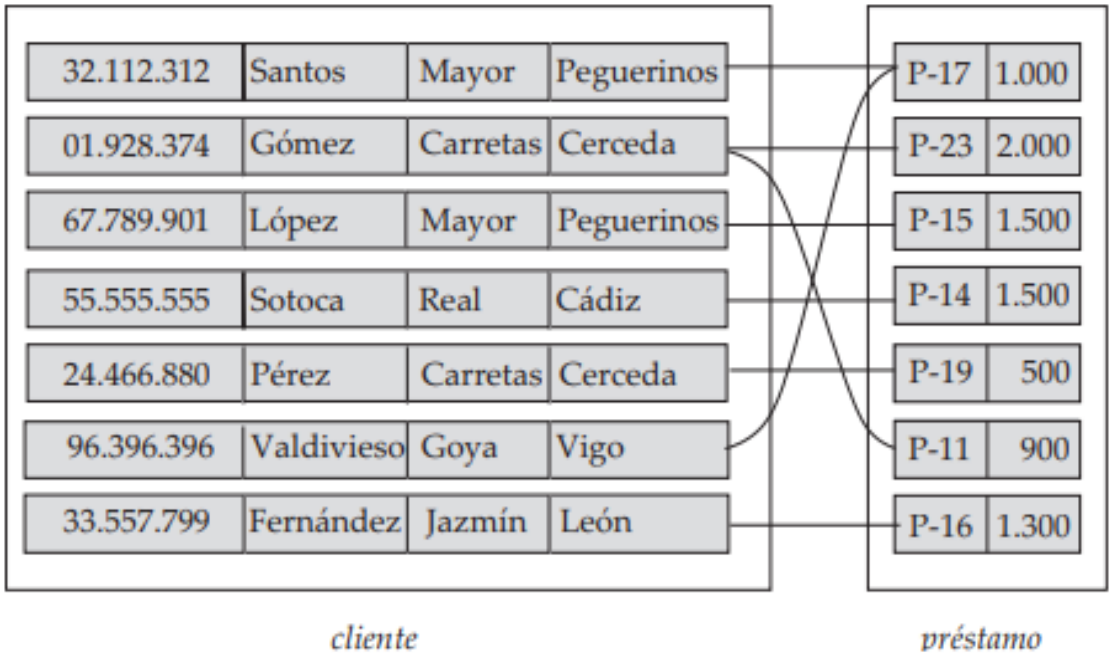
32.112.312	Santos	Mayor	Peguerinos
1.928.374	Gómez	Carretas	Cerceda
67.789.901	López	Mayor	Peguerinos
55.555.555	Sotoca	Real	Cádiz
24.466.880	Pérez	Carretas	Cerceda
96.396.396	Valdivieso	Goya	Vigo
33.557.799	Fernández	Jazmín	León

cliente

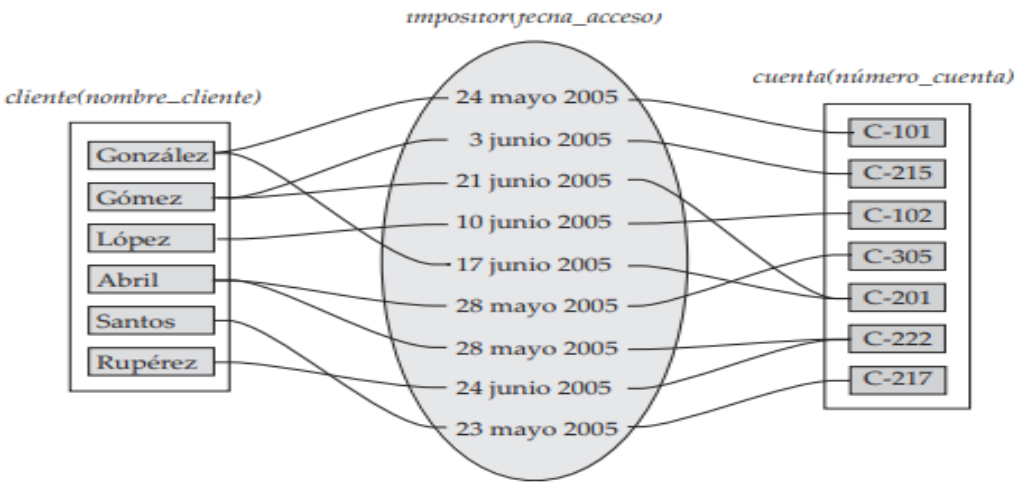
P-17	1.000
P-23	2.000
P-15	1.500
P-14	1.500
P-19	500
P-11	900
P-16	1.300

préstamo

Ejemplo de conjuntos de entidades cliente y préstamo



Ejemplo de conjunto de relaciones prestatario



Ejemplo de atributo como Fecha acceso del conjunto de entidades impositor

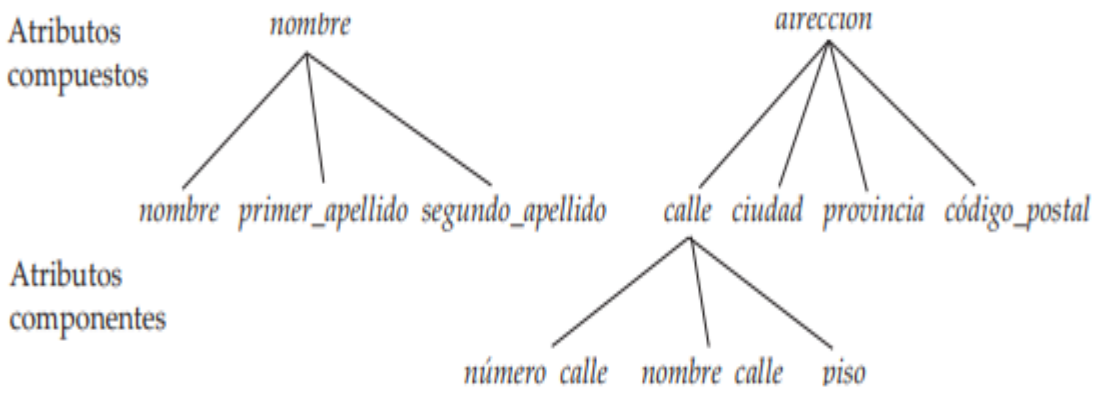
En nuestro ejemplo, los conjuntos de entidades cliente y préstamo participan en el conjunto de relaciones prestatario. Además, supóngase que cada préstamo deba tener otro cliente que sirva como avalista del préstamo. Entonces los conjuntos de entidades cliente y préstamo pueden participar en otro conjunto de relaciones: avalista.

Los conjuntos de relaciones prestatario y sucursal préstamo proporcionan un ejemplo de conjunto de relaciones binario—es decir, uno que implica dos conjuntos de entidades. La mayor parte de los conjuntos de relaciones de los sistemas de bases de datos son binarios.

A veces, no obstante, los conjuntos de relaciones implican a más de dos conjuntos de entidades. Por ejemplo, considérense los conjuntos de entidades empleado, sucursal y trabajo. Ejemplos de entidades trabajo pueden ser director, cajero, interventor, etc. Las entidades trabajo pueden tener los atributos puesto y nivel. El conjunto de relaciones trabaja en entre empleado, sucursal y trabajo es un ejemplo de relación ternaria. Una relación ternaria entre Santos, Navacerrada y director indica que Santos trabaja de director de la sucursal de Navacerrada. Santos también podría actuar como interventor de la sucursal de Centro, lo que estaría representado por otra relación. Podría haber otra relación entre Gómez, Centro y cajero, que indicaría que Gómez trabaja de cajero en la sucursal de Centro. El número de conjuntos de entidades que participan en un conjunto de relaciones es también el grado de ese conjunto de relaciones.

Los conjuntos de relaciones binarios tienen grado 2; los conjuntos de relaciones ternarios tienen grado 3.

La Figura de abajo muestra estos ejemplos de atributos compuestos para el conjunto de entidades cliente.

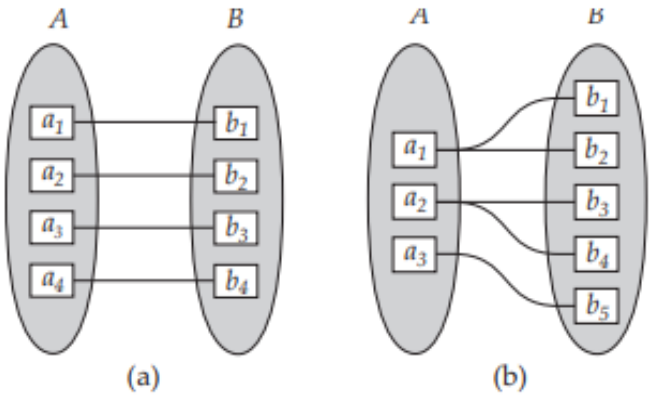


Atributos compuestos nombre y dirección

• **Atributos mono valorados y multivaluados**. Todos los atributos que se han especificado en los ejemplos anteriores tienen un único valor para cada entidad concreta. Por ejemplo, el atributo número préstamo para una entidad préstamo específica hace referencia a un único número de préstamo. Se dice que estos atributos son mono valorados. Puede haber ocasiones en las que un atributo tenga un conjunto de valores para una entidad concreta. Considérese un conjunto de entidades empleado con el atributo número teléfono. Cada empleado puede tener cero, uno o varios números de teléfono, y empleados diferentes pueden tener diferente cantidad de teléfonos. Se dice que este tipo de atributo es multivaluado. Como ejemplo adicional, el atributo nombre _subordinado del conjunto de entidades empleado es multivaluado, ya que cada empleado podría tener cero, uno o más subordinados. Si resulta necesario, se pueden establecer apropiadamente límites inferior y superior al número de valores en el atributo multivaluado. Por ejemplo, el banco puede limitar a dos el número de números de teléfono que se guardan por cliente. El establecimiento de límites en este caso expresa que el atributo número teléfono del conjunto de entidades cliente puede tener entre cero y dos valores.

• **Atributos derivados**. El valor de este tipo de atributo se puede obtener a partir del valor de otros atributos o entidades relacionados. Por ejemplo, supóngase que el conjunto de entidades cliente que tiene un atributo préstamos que representa el número de préstamos que cada cliente tiene concedidos en el banco. Ese atributo se puede obtener contando el número de entidades préstamo asociadas con cada cliente. Como ejemplo adicional, supóngase que el conjunto de entidades cliente tiene el atributo edad, que indica la edad del cliente. Si el conjunto de entidades cliente tiene también un atributo fecha de _nacimiento, se puede calcular edad a partir de fecha_de_nacimiento y de la fecha actual. Por tanto, edad es un atributo derivado. En este caso, fecha_de_nacimiento puede considerarse un atributo básico, o almacenado. El valor de los atributos derivados no se almacena, sino que se calcula cada vez que hace falta. Los atributos toman valores nulos cuando las entidades no tienen ningún valor para ese atributo. El valor nulo también puede indicar "no aplicable"—es decir, que el valor no existe para esa entidad. Por ejemplo, una persona puede no tener un segundo nombre de pila. Nulo puede también designar que el valor del atributo es desconocido. Un valor desconocido puede ser falta (el valor existe, pero no se tiene esa información) o desconocido (no se sabe si ese valor existe realmente o no). Por ejemplo, si el valor nombre de un cliente dado es nulo, se da por supuesto que el valor es falta, ya que todos los clientes deben tener nombre. Un valor nulo para el atributo piso puede significar que la dirección no incluye un piso (no aplicable), que existe el valor piso, pero no se conoce cuál es (falta), o que no se sabe si el valor piso forma parte o no de la dirección del cliente (desconocido).

Restricciones Un esquema de desarrollo E-R puede definir ciertas restricciones a las que el contenido de la base de datos se debe adaptar. En este apartado se examinan la correspondencia de cardinalidades, las restricciones de claves y las restricciones de participación.



Correspondencia de cardinalidades. (a) uno a uno. (b) Uno a varios

2.3. TIPOS DE RELACIONES

La correspondencia de cardinalidades, o razón de cardinalidad, expresa el número de entidades a las que otra entidad se puede asociar mediante un conjunto de relaciones.

La correspondencia de cardinalidades resulta muy útil para describir conjuntos de relaciones binarias, aunque pueda contribuir a la descripción de conjuntos de relaciones que impliquen más de dos conjuntos de entidades.

En este apartado se centrará la atención únicamente en los conjuntos de relaciones binarias. Para un conjunto de relaciones binarias R entre los conjuntos de entidades A y B, la correspondencia de cardinalidades debe ser una de las siguientes:

• Uno a uno Cada entidad de A se asocia, a lo sumo, con una entidad de B, y cada entidad en B se asocia, a lo sumo, con una entidad de A (véase la Figura 6.5a).

• Uno a varios Cada entidad de A se asocia con cualquier número (cero o más) de entidades de B. Cada entidad de B, sin embargo, se puede asociar, a lo sumo, con una entidad de A (véase la Figura 6.5b). • Varios a uno Cada entidad de A se asocia, a lo sumo, con una entidad de B. Cada entidad de B, sin embargo, se puede asociar con cualquier número (cero o más) de entidades de A

• Varios a varios Cada entidad de A se asocia con cualquier número (cero o más) de entidades de B, y cada entidad de B se asocia con cualquier número (cero o más) de entidades de A

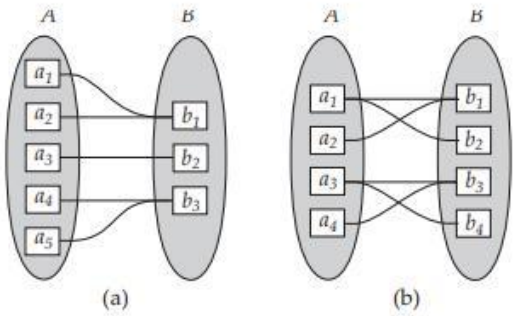
La correspondencia de cardinalidades adecuada para un conjunto de relaciones dado depende, obviamente, de la situación del mundo real que el conjunto de relaciones modele.

Como ilustración, considérese el conjunto de relaciones prestatario. Si, en un banco dado, cada préstamo sólo puede pertenecer a un cliente y cada cliente puede tener varios préstamos, entonces el conjunto de relaciones de cliente a préstamo es uno a varios. Si cada préstamo puede pertenecer a varios clientes (como los préstamos solicitados conjuntamente por varios socios de un negocio) el conjunto de relaciones es varios a varios.

Claves Es necesario tener una forma de especificar la manera de distinguir las entidades pertenecientes a un conjunto de entidades dado.

Conceptualmente cada entidad es distinta; desde el punto de vista de las bases de datos, sin embargo, la diferencia entre ellas se debe expresar en términos de sus atributos. Por lo tanto, los valores de los atributos de cada entidad deben ser tales que permitan identificar

unívocamente a esa entidad. En otras palabras, no se permite que ningún par de entidades de un conjunto de entidades tenga exactamente el mismo valor en todos sus atributos.



Correspondencia de cardinalidades (a) Varios a uno. (b) Varios a varios

Las claves permiten identificar un conjunto de atributos que resulta suficiente para distinguir las entidades entre sí. Las claves también ayudan a identificar unívocamente las relaciones y, por tanto, a distinguir las relaciones entre sí.

Conjuntos de entidades Una superclave es un conjunto de uno o más atributos que, tomados conjuntamente, permiten identificar de forma unívoca una entidad del conjunto de entidades. Por ejemplo, el atributo id_cliente del conjunto de entidades cliente es suficiente para distinguir una entidad cliente de las demás. Por tanto, id _cliente es una superclave. Análogamente, la combinación de nombre_cliente e id_cliente es una superclave del conjunto de entidades cliente. El atributo nombre_cliente de cliente no es una superclave, ya que varias personas pueden tener el mismo nombre.

El concepto de superclave no es suficiente para lo que aquí se propone ya que, como se ha visto, las superclaves pueden contener atributos innecesarios. Si C es una superclave, entonces también lo es cualquier superconjunto de C. A menudo interesan las superclaves

para las que ningún subconjunto propio es superclave. Esas superclaves mínimas se denominan claves candidatas.

Es posible que conjuntos distintos de atributos puedan servir como clave candidata. Supóngase que una combinación de nombre_cliente y calle_cliente sea suficiente para distinguir entre los miembros del conjunto de entidades cliente. Entonces, tanto id_cliente como nombre_cliente, calle_cliente son claves candidatas. Aunque los atributos id_cliente y nombre_cliente juntos puedan diferenciar las entidades cliente, su combinación no forma una clave candidata, ya que el atributo id_cliente por sí solo es una clave candidata. Se usa el término clave primaria para denotar la clave candidata elegida por el diseñador de la base de datos como elemento principal de identificación de las entidades pertenecientes a un conjunto de entidades.

Las claves (primarias, candidatas y superclaves) son propiedades del conjunto de entidades, más que de cada una de las entidades.

Dos entidades cualesquiera del conjunto no pueden tener el mismo valor de los atributos de su clave al mismo tiempo. La designación de una clave representa una restricción de la empresa real que se está modelando.

Las claves candidatas se deben escoger con cuidado. Como se ha indicado, el nombre de cada persona es obviamente insuficiente, ya que puede haber muchas personas con el mismo nombre.

Conjuntos de relaciones La clave primaria de cada conjunto de entidades permite distinguir entre las diferentes entidades del conjunto. Se necesita un mecanismo parecido para distinguir entre las diferentes relaciones de cada conjunto de relaciones. Sea R un conjunto de relaciones que involucra los conjuntos de entidades E1, E2, . . . , En. Sea clave _primaria (Ei) el conjunto de atributos que forman la clave primaria del conjunto de entidades

La estructura de la clave primaria para el conjunto de relaciones depende de la correspondencia de cardinalidades del conjunto de relaciones. Como ejemplo, considérense los conjuntos de entidades cliente y cuenta y el conjunto de relaciones impositor, con el atributo fecha_acceso Supóngase que el conjunto de relaciones es varios a varios. Entonces, la clave primaria de impositor consiste en la unión de las claves primarias de cliente y de cuenta. Sin embargo, si un cliente sólo puede tener una cuenta—es decir, si la relación impositora es varios a uno de cliente a cuenta—entonces la clave primaria de impositor es simplemente la clave primaria de cliente. Análogamente, si la relación es varios a uno de cuenta a cliente—es decir, cada cuenta pertenece, a lo sumo, a un cliente— entonces la clave primaria de impositor es simplemente la clave primaria de cuenta.

Para relaciones uno a uno se puede usar cualquiera de las claves primarias. Para las relaciones no binarias, si no hay restricciones de cardinalidad, la superclave formada como se ha descrito anteriormente en este apartado es la única clave candidata, y se elige como clave primaria. La elección de la clave primaria resulta más complicada si hay restricciones de cardinalidad.

Restricciones de participación Se dice que la participación de un conjunto de entidades E en un conjunto de relaciones R es total si cada entidad de E participa, al menos, en una relación de R. Si sólo algunas entidades de E participan en relaciones de R, se dice que la participación del conjunto de entidades E en la relación R es parcial. Por ejemplo, se puede esperar que cada entidad préstamo esté relacionada, al menos, con un cliente mediante la relación

prestatario. Por tanto, la participación de préstamo en el conjunto de relaciones prestatario es total. En cambio, un individuo puede ser cliente del banco tenga o no tenga concedido algún préstamo en el banco. Por tanto, es posible que sólo algunas de las entidades cliente estén relacionadas con el conjunto de entidades préstamo mediante la relación prestatario, y la participación de cliente en la relación prestatario es, por tanto, parcial.

Diagramas entidad-relación pueden expresar gráficamente la estructura lógica general de las bases de datos. Son sencillos y claros—cualidades que pueden ser responsables en gran parte de la popularidad del modelo E-R.

Estos diagramas constan de los siguientes componentes principales:

• Rectángulos, que representan conjuntos de entidades.

• Elipses, que representan atributos.

• Rombos, que representan conjuntos de relaciones.

• Líneas, que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con los conjuntos de relaciones.

• Elipses dobles, que representan atributos multivalorados.

• Elipses discontinuas, que denotan atributos derivados.

• Líneas dobles, que indican participación total de una entidad en un conjunto de relaciones.

• Rectángulos dobles, que representan conjuntos de entidades débiles

El conjunto de relaciones prestatario puede ser varios a varios, uno a varios, varios a uno o uno a uno. Para distinguir entre estos tipos, se dibuja o una línea dirigida (→) o una línea no dirigida (—) entre el conjunto de relaciones y el conjunto de entidades en cuestión.

• Una línea dirigida desde el conjunto de relaciones prestatario al conjunto de entidades préstamo especifica que prestatario es un conjunto de relaciones uno a uno o varios a uno desde cliente a préstamo; prestatario no puede ser un conjunto de relaciones varios a varios ni uno a varios desde cliente a préstamo.

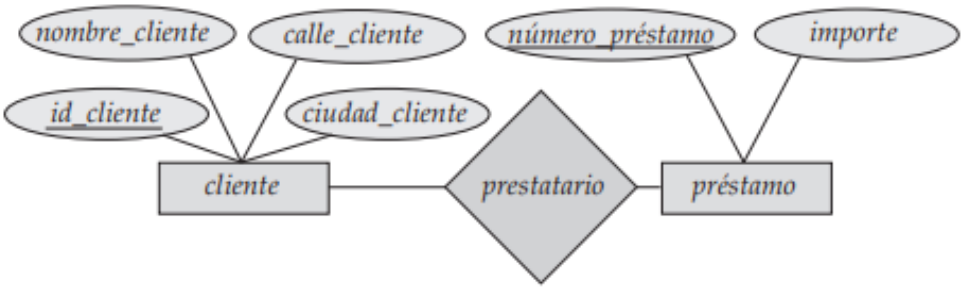
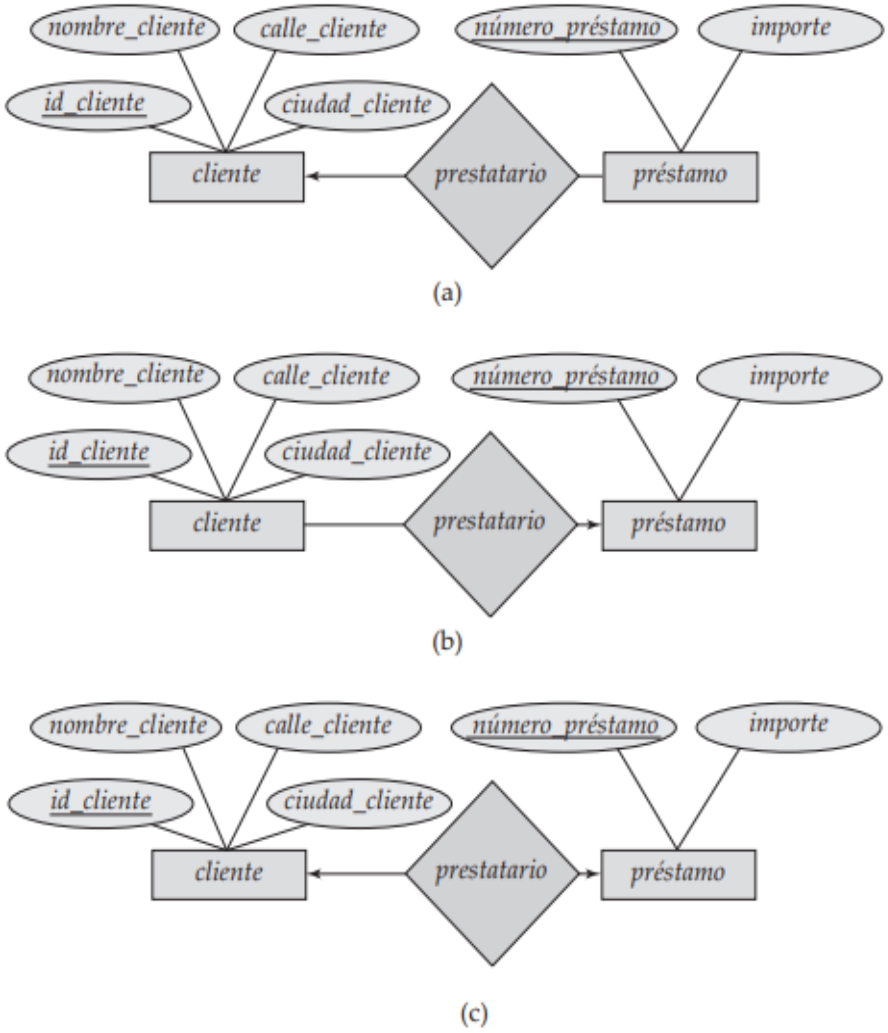


Diagrama E-R correspondiente a clientes y préstamos



Relaciones a.- uno a varios b.- Varios a uno c- uno a uno

• Una línea no dirigida desde el conjunto de relaciones prestatario al conjunto de relaciones préstamo especifica que prestatario es un conjunto de relaciones varios a varios o uno a varios desde cliente a préstamo.

El conjunto de relaciones prestatario es varios a varios. Si el conjunto de relaciones prestatario fuera uno a varios, desde cliente a préstamo, entonces la línea desde prestatario a cliente sería dirigida, con una flecha que apuntaría al conjunto de entidades cliente.

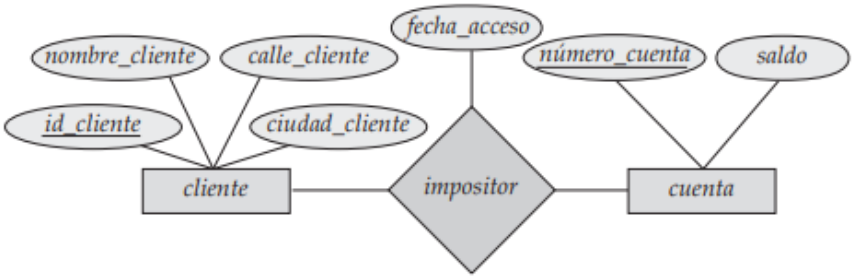


Diagrama E-R con un atributo unido a un conjunto de relaciones

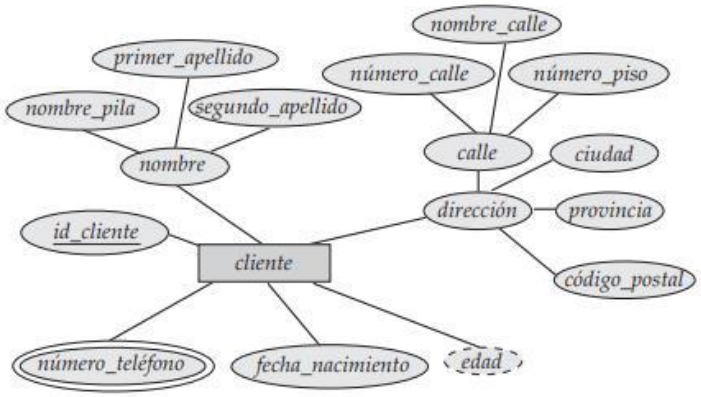


Diagrama E-R con atributos compuestos, multivalorados y derivados

Análogamente, si el conjunto de relaciones prestatario fuera varios a uno desde cliente a préstamo, entonces la línea desde prestatario a préstamo tendría una flecha que apuntaría al conjunto de entidades préstamo, Finalmente, si el conjunto de relaciones prestatario fuera uno a uno, entonces las dos líneas que salen de prestatario tendrían flecha: una que apuntaría al conjunto de entidades préstamo y otra que apuntaría al conjunto de entidades cliente

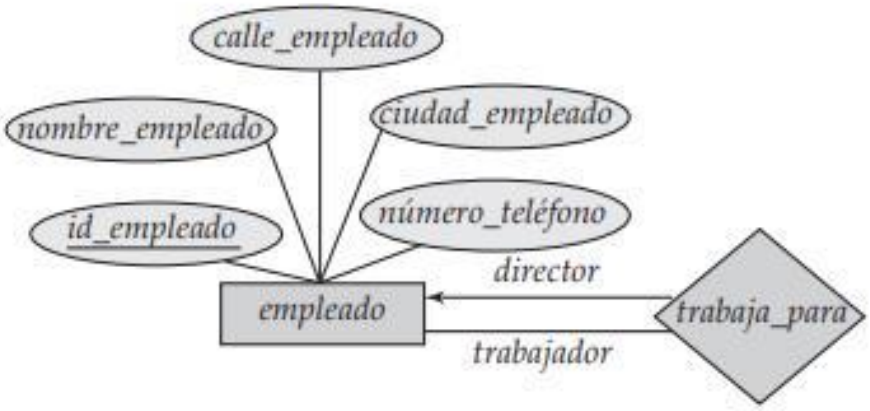


Diagrama E-R con indicadores de roles

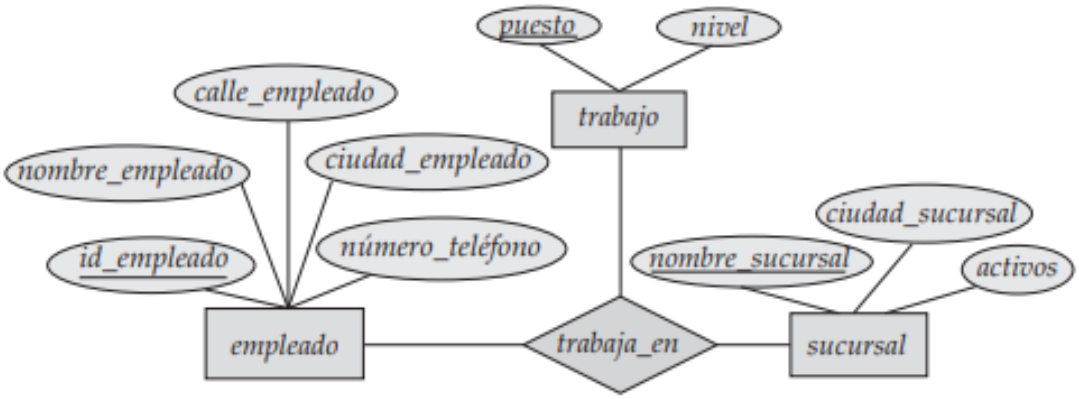


Diagrama E-R con una relación ternaria

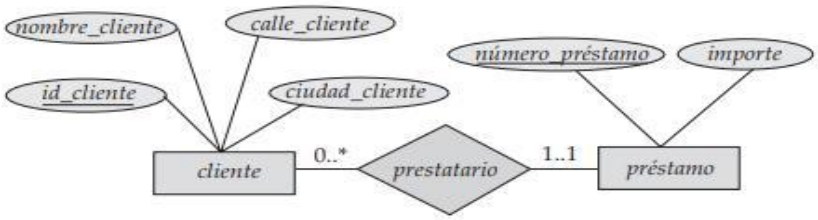
En el caso de conjuntos de relaciones no binarias se pueden especificar algunos tipos de relaciones varios a uno. Supóngase que un empleado puede tener, a lo sumo, un trabajo en cada sucursal (por ejemplo, Santos no puede ser director e interventor en la misma sucursal). Esta restricción se puede especificar mediante una flecha que apunte a trabajo en el arco de trabaja_en. Como máximo se permite una flecha desde cada conjunto de relaciones, ya que

los diagramas E-R con dos o más flechas salientes de cada conjunto de relaciones no binarias se pueden interpretar de dos formas.

Los diagramas E-R también ofrecen una manera de indicar restricciones más complejas sobre el número de veces que cada entidad participa en las relaciones de un conjunto de relaciones.

Un segmento entre un conjunto de entidades y un conjunto de relaciones binarias puede tener unas cardinalidades mínima y máxima asociadas, que se muestran de la forma min..Max, donde min es la cardinalidad mínima y Max es la máxima. Un valor mínimo de 1 indica una participación total del conjunto de entidades en el conjunto de relaciones. Un valor máximo de 1 indica que la entidad participa, a lo sumo, en una relación, mientras que un valor máximo de $*$ indica que no hay límite. Téngase en cuenta que una etiqueta 1..$*$ en un segmento es equivalente a una línea doble.

Participación total de un conjunto de entidades en un conjunto de relaciones



Límites de cardinalidad en los conjuntos de relaciones

Si los dos segmentos de una relación binaria tienen un valor máximo de 1, la relación es uno a uno. Si se hubiese especificado un límite de cardinalidad de 1..∗ en el segmento entre cliente y prestatario, se estaría diciendo que cada cliente debe tener, al menos, un préstamo.

Aspectos del diseño entidad-relación Los conceptos de conjunto de entidades y de conjunto de relaciones no son precisos, y es posible definir el conjunto de entidades y las relaciones entre ellas de diferentes formas. En este apartado se examinan aspectos básicos del diseño de esquemas de bases de datos E-R.

Uso de conjuntos de entidades y de atributos Considérese el conjunto de entidades empleado con los atributos nombre_empleado y número_teléfono. Se puede argumentar que un teléfono es una entidad en sí misma con los atributos número_teléfono y ubicación; la ubicación puede ser la sucursal o el domicilio donde el teléfono está instalado, y se pueden representar los teléfonos móviles (celulares) mediante el valor "móvil". Si se adopta este punto de vista, hay que redefinir el conjunto de entidades empleado como:

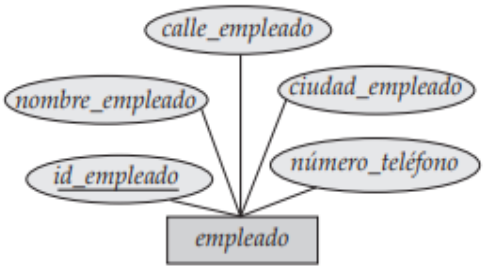
• El conjunto de entidades empleado con los atributos id_empleado y nombre_empleado.

• El conjunto de entidades teléfono con los atributos número_teléfono y ubicación.

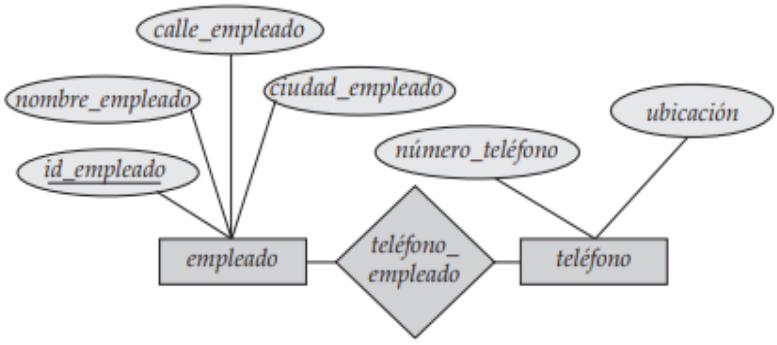
• La relación teléfono_empleado, que denota la asociación entre los empleados y sus teléfonos.

Un error común es usar la clave primaria de un conjunto de entidades como atributo de otro conjunto de entidades en lugar de usar una relación.

La relación prestataria es la forma correcta de representar la conexión entre los préstamos y los clientes, ya que hace explícita su conexión, en lugar de dejarla implícita mediante un atributo. Otro error relacionado con éste que se comete a veces es escoger los atributos de clave primaria de los conjuntos de entidades relacionados como atributos del conjunto de relaciones.

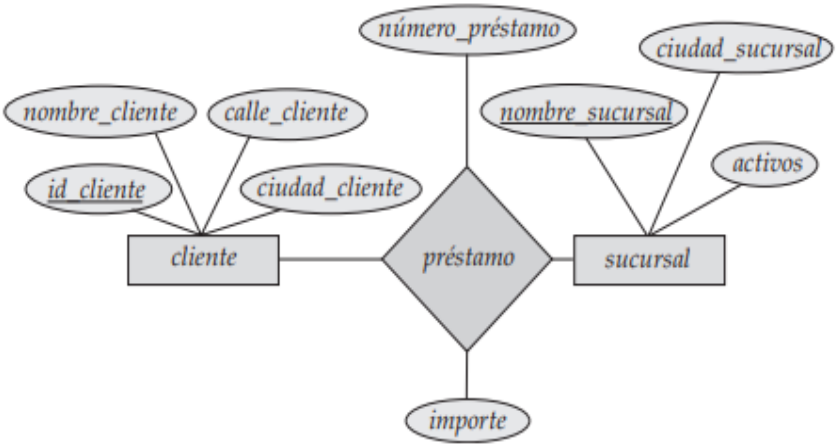


(a)



(b)

Ejemplo de alterativas para empleado y teléfono



Ejemplo de préstamo como conjunto de relaciones

Ubicación de los atributos de las relaciones La razón de cardinalidad de una relación puede afectar a la ubicación de sus atributos. Por tanto, los atributos de los conjuntos de relaciones uno a uno o uno a varios puede estar asociados con uno de los conjuntos de entidades participantes, en lugar de con el conjunto de relaciones. Por ejemplo, especifiquemos que impositor es un conjunto de relaciones uno a varios tal que cada cliente puede tener varias cuentas, pero cada cuenta sólo puede tener un cliente como titular.

Los atributos de un conjunto de relaciones uno a varios sólo se pueden recolocar en el conjunto de entidades de la parte "varios" de la relación. Por otra parte, para los conjuntos de entidades uno a uno, los atributos de la relación se pueden asociar con cualquiera de las entidades participantes. La decisión de diseño sobre la ubicación de los atributos descriptivos en estos casos—como atributo de la relación o de la entidad—debe reflejar las características de la empresa que se modela.

El diseñador puede elegir mantener un atributo de impositor para expresar explícitamente que se produce un acceso en el punto de interacción entre los conjuntos de entidades cliente

y cuenta. La elección de la ubicación del atributo es más sencilla para los conjuntos de relaciones varios a varios.

Cuando un atributo se determina mediante la combinación de los conjuntos de entidades participantes, en lugar de por cada entidad por separado, ese atributo debe estar asociado con el conjunto de relaciones varios a varios.

2.4. CONJUNTO DE ENTIDADES DÉBILES.

Conjuntos de entidades débiles Puede que un conjunto de entidades no tenga suficientes atributos para formar una clave primaria. Ese conjunto de entidades se denomina conjunto de entidades débiles.

Los conjuntos de entidades que tienen una clave primaria se denominan conjuntos de entidades fuertes.

Para que un conjunto de entidades débiles tenga sentido, debe estar asociado con otro conjunto de entidades, denominado conjunto de entidades identificadoras o propietarias.

Cada entidad débil debe estar asociada con una entidad identificadora; es decir, se dice que el conjunto de entidades débiles depende existencialmente del conjunto de entidades identificadoras. Se dice que el conjunto de entidades identificadoras posee el conjunto de entidades débiles al que identifica.

La relación que asocia el conjunto de entidades débiles con el conjunto de entidades identificadoras se denomina relación identificadora. La relación identificadora es varios a uno del conjunto de entidades débiles al conjunto de entidades identificadoras y la participación del conjunto de entidades débiles en la relación es total.

El discriminante de un conjunto de entidades débiles es un conjunto de atributos que permite que se haga esta distinción.

El discriminante del conjunto de entidades débiles se denomina clave parcial del conjunto de entidades. La clave primaria de un conjunto de entidades débiles se forma con la clave primaria del conjunto de entidades identificadoras y el discriminante del conjunto de entidades débiles.

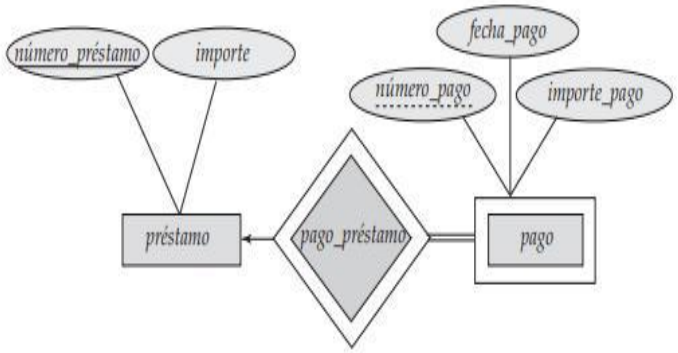
Los conjuntos de entidades débiles pueden participar en otras relaciones aparte de la relación identificadora.

Los conjuntos de entidades débiles pueden participar como propietario de una relación identificadora con otro conjunto de entidades débiles. También es posible tener conjuntos de entidades débiles con más de un conjunto de entidades identificadoras.

Cada entidad débil se identificaría mediante una combinación de entidades, una de cada conjunto de entidades identificadoras. La clave primaria de la entidad débil consistiría de la unión de las claves primarias de los conjuntos de entidades identificadoras y el discriminante del conjunto de entidades débiles.

En los diagramas E-R los rectángulos con líneas dobles indican conjuntos de entidades débiles, mientras que un rombo con líneas dobles indica la correspondiente relación de identificación.

Los conjuntos de entidades débiles se pueden modelar mejor como atributos si sólo participan en la relación identificadora y tienen pocos atributos. A la inversa, las representaciones de los conjuntos de entidades débiles modelarán mejor las situaciones en las que esos conjuntos participen en otras relaciones aparte de la relación identificadora y tengan muchos atributos. Como ejemplo adicional de conjunto de entidades que se puede modelar como conjunto de entidades débiles, considérese las ofertas de asignaturas en una universidad. La misma asignatura se puede ofrecer en diferentes cursos y, dentro de cada curso, puede haber varios grupos para la misma asignatura. Por tanto, se puede crear el conjunto de entidades débiles oferta_asignatura, que depende existencialmente de asignatura; las diferentes ofertas de la misma asignatura se identifican mediante curso y número_grupo, que forman un discriminante pero no una clave primaria.



Ejemplo de diagrama E-R con un conjunto de entidades débiles

Características del modelo E-R extendido, Aunque los conceptos básicos del modelo E-R pueden modelar la mayor parte de las características de las bases de datos, algunos aspectos

de las bases de datos se pueden expresar mejor mediante ciertas extensiones del modelo E-R básico.

Las características E-R extendidas de especialización, generalización, conjuntos de entidades de nivel superior e inferior, herencia de atributos y agregación.

Especialización Los conjuntos de entidades pueden incluir subgrupos de entidades que se diferencian de alguna forma de las demás entidades del conjunto. Por ejemplo, un subconjunto de entidades de un conjunto de entidades puede tener atributos que no sean compartidos por todas las entidades del conjunto de entidades.

El modelo E-R ofrece un medio de representar estos grupos de entidades diferentes.

Como ejemplo, considérese el conjunto de entidades persona con los atributos id_persona, nombre, calle y ciudad. Cada persona puede clasificarse además en una de las categorías siguientes: • cliente • empleado Cada uno de estos tipos de persona se describen mediante un conjunto de atributos que incluye todos los atributos del conjunto de entidades persona más otros posibles atributos adicionales.

Por ejemplo, las entidades cliente se pueden describir además mediante el atributo calificación_crediticia, mientras que las entidades empleado se pueden describir además mediante el atributo sueldo. El proceso de establecimiento de subgrupos dentro del conjunto de entidades se denomina especialización.

La especialización de persona permite distinguir entre las personas basándonos en si son

empleados o clientes: en general, cada persona puede ser empleado, cliente, las dos cosas o ninguna de ellas. Como ejemplo adicional, supóngase que el banco desea dividir las cuentas en dos categorías: cuentas corrientes y cuentas de ahorro. Las cuentas de ahorro necesitan un saldo mínimo, pero el banco puede establecer diferentes tasas de interés para cada cliente y ofrecer mejores tasas a los clientes preferentes. Las cuentas corrientes tienen una tasa de interés fija, pero permiten los descubiertos; hay que registrar el importe de los descubiertos de las cuentas corrientes. Cada uno de estos tipos de cuenta se describe mediante un conjunto de atributos que incluye todos los atributos del conjunto de entidades cuenta más otros atributos adicionales. El banco puede crear dos especializaciones de cuenta, por ejemplo, cuenta_ahorro y cuenta_corriente. Como ya se ha visto, las entidades cuentan se describen mediante los atributos número_cuenta y saldo. El conjunto de entidades cuenta_ahorro tendría todos los atributos de cuenta y el atributo adicional tasa_interés. El conjunto de entidades cuenta_corriente tendría todos los atributos de cuenta y el atributo adicional importe_descubierto.

La especialización se puede aplicar repetidamente para refinar el esquema de diseño. Por ejemplo, los empleados del banco se pueden clasificar también en alguna de las categorías siguientes: • oficial • cajero • secretaria Cada uno de estos tipos de empleado se describe mediante un conjunto de atributos que incluye todos los atributos del conjunto de entidades empleado y otros adicionales. Por ejemplo, las entidades oficiales se pueden describir además por el atributo número_despacho, las entidades cajero por los atributos número _caja y horas_semana, y las entidades secretaria por el atributo horas_semana. Además, las entidades secretaria pueden participar en la relación secretaria_de, que identifica a los empleados a los que ayuda cada secretaria. Cada conjunto de entidades se puede especializar en más de una característica distintiva. En este ejemplo, la característica distintiva entre las entidades empleado es el trabajo que desempeña cada empleado. Otra especialización coexistente se puede basar en si cada persona es un trabajador temporal o fijo, lo que da lugar a los conjuntos de entidades empleado_temporal y empleado_fijo. Cuando se forma más de una especialización en un conjunto de entidades, cada entidad concreta puede pertenecer a varias especializaciones. Por ejemplo, un empleado dado puede ser un empleado temporal y una secretaria.

Los conjuntos de entidades de nivel superior e inferior se representan como conjuntos de entidades regulares—es decir, como rectángulos que contienen el nombre del conjunto de entidades.

Generalización El refinamiento a partir del conjunto de entidades inicial en sucesivos niveles de subgrupos de entidades representa un proceso de diseño descendente en el que las distinciones se hacen explícitas. El proceso de diseño también puede proceder de forma ascendente, en la que varios conjuntos de entidades se sintetizan en un conjunto de entidades de nivel superior basado en características comunes. El diseñador de la base de datos puede haber identificado primero el conjunto de entidades cliente con los atributos id_cliente, nombre_cliente, calle_cliente, ciudad_cliente y calificación_crediticia, y el conjunto de entidades empleado con los atributos id_empleado, nombre_empleado, calle_empleado, ciudad_empleado y sueldo_empleado. Existen analogías entre el conjunto de entidades cliente y el conjunto de entidades empleado en el sentido de que tienen varios atributos que, conceptualmente, son iguales en los dos conjuntos de entidades: los atributos para el identificador, el nombre, la calle y la ciudad. Esta similitud se puede expresar mediante la generalización, que es una relación de contención que existe entre el conjunto de entidades de nivel superior y uno o varios conjuntos de entidades de nivel inferior. En este ejemplo, persona es el conjunto de entidades de nivel superior y cliente y empleado son conjuntos de entidades de nivel inferior. En este caso, los atributos que son conceptualmente iguales tienen nombres diferentes en los dos conjuntos de entidades de nivel inferior. Para crear generalizaciones los atributos deben tener un nombre común y representarse mediante la entidad de nivel superior persona.

Los conjuntos de entidades de nivel superior e inferior también se pueden denominar con los términos superclase y subclase, respectivamente. El conjunto de entidades persona es la superclase de las subclases cliente y empleado. A todos los efectos prácticos, la generalización es una inversión simple de la especialización. Se aplicarán ambos procesos, combinados, en el transcurso del diseño del esquema E-R de una empresa. En términos del propio diagrama E-R no se distingue entre especialización y generalización. Los niveles

nuevos de representación de las entidades se distinguen (especialización) o sintetizan (generalización) cuando el esquema de diseño llega a expresar completamente la aplicación de la base de datos y los requisitos del usuario de la base de datos.

Las diferencias entre los dos enfoques se pueden caracterizar mediante su punto de partida y su objetivo global.

La especialización parte de un único conjunto de entidades; destaca las diferencias entre las entidades del conjunto mediante la creación de diferentes conjuntos de entidades de nivel inferior.

NOTA: Para mayor detalles sobre el tema consultar :
http://mateo.pbworks.com/w/file/fetch/122276985/Fundamentos_de_Bases_de_Datos_5a_Ed.-_Si.pdf
PAG 197 – 209

2.5. LA NOTACIÓN E-R CON UML.

El lenguaje de modelado ubicado UML** Los diagramas entidad-relación ayudan a modelar el componente de representación de datos de los sistemas de software. La representación de datos, sin embargo, sólo forma parte del diseño global del sistema. Otros componentes son los modelos de interacción del usuario con el sistema, la especificación de los módulos funcionales del sistema y su interacción, etc.

El lenguaje de modelado unificado (Unified Modeling Language, UML) es una norma desarrollada bajo los auspicios del Grupo de Administración de Objetos (Object Management Group, OMG) para la creación de especificaciones de diferentes componentes de los sistemas de software.

Algunas de las partes de UML son:

• Diagramas de clase.

• Diagramas de caso de uso.

• Diagramas de actividad.

• Diagramas de implementación.

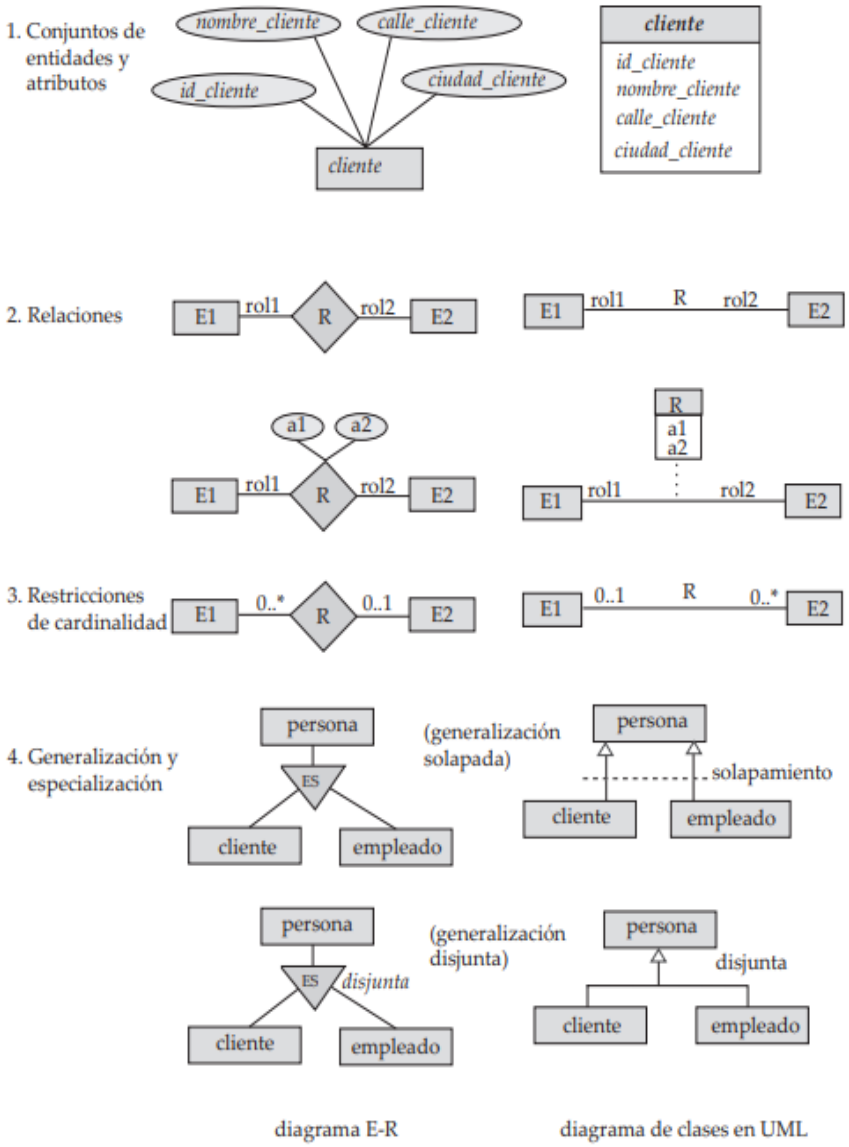


Figura 6.28 Símbolos usados en la notación de diagramas de clases de UML.

La generalización y la especialización se representan en UML conectando conjuntos de entidades mediante una línea con un triángulo al final correspondiente al conjunto de entidades más general. Por ejemplo, el conjunto de entidades persona es una generalización de cliente y de empleado. Los diagramas UML también pueden representar explícitamente las restricciones de la condición de disyunción y de solapamiento de las generalizaciones.

NOTA: Para mayor información revisar el siguiente link

http://openaccess.uoc.edu/webapps/o2/bitstream/10609/9121/1/Intro_UML.pdf

https://docirs.cl/uml.htm

2.7 OTROS ASPECTOS DEL DISEÑO DE BASES DE DATOS.

NOTA: Para mayor información revisar el siguiente link

https://www.lucidchart.com/pages/es/tutorial-de-estructura-y-diseno-de-bases-de-datos

https://www.uoc.edu/pdf/masters/oficiales/img/913.pdf

2.8 RESTRICCIONES

Una restricción es una condición que obliga el cumplimiento de ciertas condiciones en la base de datos. Algunas no son determinadas por los usuarios, sino que son inherentemente definidas por el simple hecho de que la base de datos sea relacional. Algunas otras restricciones las puede definir el usuario, por ejemplo, usar un campo con valores enteros entre 1 y 10.

Las restricciones proveen un método de implementar reglas en la base de datos. Las restricciones restringen los datos que pueden ser almacenados en las tablas. Usualmente se definen usando expresiones que dan como resultado un valor booleano, indicando si los datos satisfacen la restricción o no.

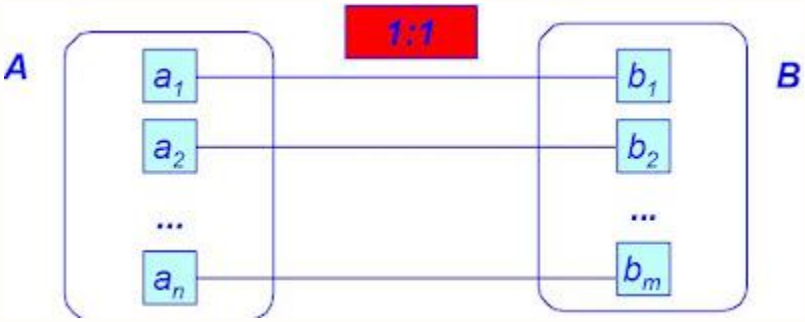
Las restricciones no son parte formal del modelo relacional, pero son incluidas porque juegan el rol de organizar mejor los datos.

Un esquema de desarrollo E-R puede definir ciertas restricciones a las que los contenidos de la base de datos se deben adaptar.

Examinemos la correspondencia de cardinalidades y las restricciones de participación, que son dos de los tipos más importantes de restricciones.

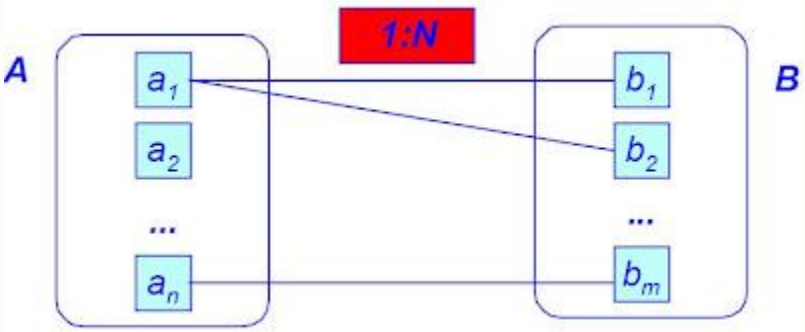
La correspondencia de cardinalidades, o razón de cardinalidad, expresa el número de entidades a las que otra entidad puede estar asociada vía un conjunto de relaciones.

Uno a uno. Una entidad en A se asocia con a lo sumo una entidad en B, y una entidad en B se asocia con a lo sumo una entidad en A.



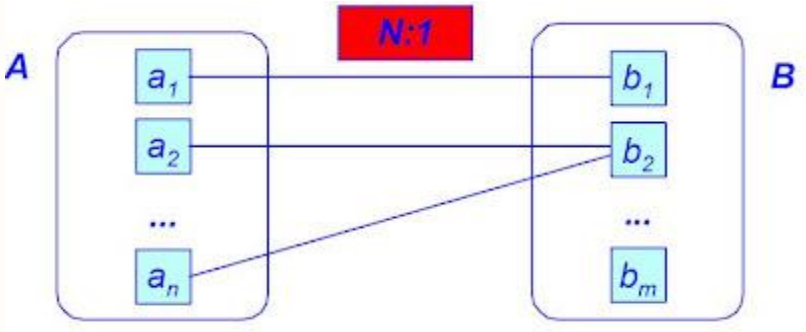
Ej. Una persona tiene un coche y un coche es de una sola persona.

Uno a varios. Una entidad en A se asocia con cualquier número de entidades en B (ninguna o varias). Una entidad en B, sin embargo, se puede asociar con a lo sumo una entidad en A



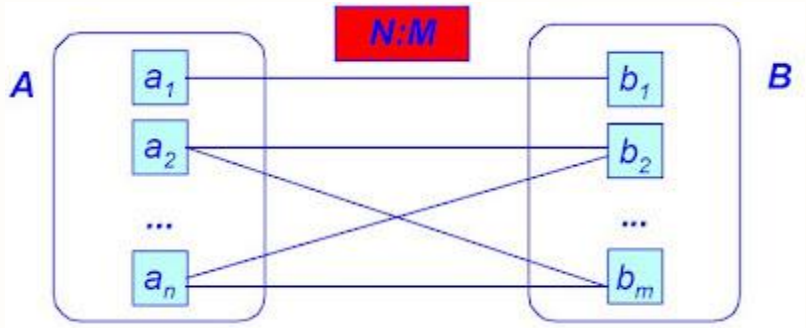
Ej. Una persona tiene varios coches y un coche es de una sola persona.

Varios a uno. Una entidad en A se asocia con a lo sumo una entidad en B. Una entidad en B, sin embargo, se puede asociar con cualquier número de entidades (ninguna o varias) en A



Ej. Una persona tiene un coche y un coche es de varias personas.

Varios a varios. Una entidad en A se asocia con cualquier número de entidades (ninguna o varias) en B, y una entidad en B se asocia con cualquier número de entidades (ninguna o varias) en A.



Ej. Una persona tiene varios coches y un coche es de varias personas.

Como ilustración considérese el conjunto de relaciones prestatario. Si en un banco particular un préstamo puede pertenecer únicamente a un cliente y un cliente puede tener varios préstamos, entonces el conjunto de relaciones de cliente a préstamo es uno a varios. Si un préstamo puede pertenecer a varios clientes (como préstamos que se toman en conjunto por varios socios de un negocio) el conjunto de relaciones es varios a varios.

Restricciones Parciales

La participación de un conjunto de entidades E en un conjunto de relaciones R se dice que es total si cada entidad en E participa al menos en una relación en R. Si sólo algunas entidades en E participan en relaciones en R, la participación del conjunto de entidades E en la relación R se llama parcial. Por ejemplo, se puede esperar que cada entidad préstamo esté relacionada con al primamenos un cliente mediante la relación prestatario. Por lo tanto, la participación de préstamo en el conjunto de relaciones prestatario es total. En cambio, un individuo puede ser cliente de un banco tenga o no tenga un préstamo en el banco. Así, es posible que sólo algunas de las entidades cliente estén relacionadas con el conjunto de entidades préstamo mediante la relación prestatario, y la participación de cliente en el conjunto de relaciones prestatario es por lo tanto parcial.

2.9 DISEÑO CON DIAGRAMAS E-R

La estructura lógica de una Base de Datos se puede representar graficamente a través de un diagrama, el cual llamaremos Diagrama E-R. Estos diagramas se apoyan de diferentes símbolos los cuales tienen un significado particular. Los diagramas se usan para que la información se presente de forma clara y sencilla. Los componentes principales son:

Breve recordatorio:

Entidad: Representa un objeto que tiene vida propia en el sistema que se está modelando, tanto tangible como intangibles. Ejemplo: cliente, producto, estudiante, vacación.

Conjunto de entidades: Grupo (conjunto) de entidades del mismo tipo. Ejemplo: Todos los estudiantes de un curso, representan el conjunto de entidades estudiante.

Relación: Asociación o vinculación entre dos o más entidades. Ejemplo: La relación comprar entre las entidades cliente y producto. Generalmente representa acciones entre las entidades.

Conjunto de relaciones: Son relaciones del mismo tipo.

Atributos: Características o propiedades asociadas al conjunto de entidades o relaciones y que toman valor en una entidad en particular. Ejemplo: nombre, cédula, teléfono. Los posibles valores puede tomar un atributo para un conjunto de entidades se denomina dominio.

Los atributos se pueden clasificar en:
- **Simples o atómicos:** Son aquellos que no contienen otros atributos
- **Compuestos:** Son los que incluyen otros atributos simples.. Ejemplo: dirección (Se puede dividir en calle, número, ciudad).
- **Monovalorados o Univalorados:** Atributo que toma un solo valor, para una entidad en particular.
- **Multivalorados:** Atributo que para una misma entidad puede tomar muchos valores.
- **Derivados o calculados:** Son aquellos atributos cuyos valores se pueden conseguir con operaciones sobre valores de otros atributos.
- **Nulos:** Son aquellos atributos para los cuales en algún momento no existe o no se conoce su valor.

Diagrama Entidad - Relación.

Es la representación gráfica del Modelo Entidad-Relación y permite ilustrar la estructura de la base de datos del negocio modelado.

Escribe Johnson "los diagramas ER constituyen una notación para documentar un diseño tentativo de bases de datos. Los analistas los utilizan para facilitar el proceso de diseño" [Joh00].

Está compuesto por los siguientes elementos.
• Rectángulos: representan conjuntos de entidades.
• Elipses: representan atributos.
• Rombos: representan relaciones.
• Líneas: unen atributos a conjuntos de entidades y conjuntos de entidades a conjuntos de relaciones.
• Elipses dobles: representan atributos multivalorados.
• Elipses discontinuas: que denotan atributos derivados.
• Líneas dobles: indican participación total de una entidad en un conjunto de relaciones.
• Rectángulos dobles: representan conjuntos de entidades débiles.

Tomaremos el ejemplo del Libro, Fundamentos de Bases de Datos

Considérese el diagrama entidad-relación de la siguiente figura, que consta de dos conjuntos de entidades, cliente y préstamo, relacionadas a través de un conjunto de relaciones binarias prestatario. Los atributos asociados con cliente son id-cliente, nombre-cliente, calle-cliente, y ciudad-cliente. Los atributos asociados con préstamo son número-préstamo e importe. Como se muestra en la figura, los atributos de un conjunto de entidades que son miembros de la clave primaria están subrayados. El conjunto de relaciones prestatario puede ser varios a varios, uno a varios, varios a uno o uno a uno. Para distinguir entre estos tipos, se dibuja o una línea dirigida (→) o una línea no dirigida (—) entre el conjunto de relaciones y el conjunto de entidades en cuestión.

• Una línea dirigida desde el conjunto de relaciones prestatario al conjunto de entidades préstamo especifica que prestatario es un conjunto de relaciones uno a uno, o bien varios a uno, desde cliente a préstamo; prestatario no puede ser un conjunto de relaciones varios a varios ni uno a varios, desde cliente a préstamo.

• Una línea no dirigida desde el conjunto de relaciones prestatario al conjunto de relaciones préstamo especifica que prestatario es o bien un conjunto de relaciones varios a varios, o bien uno a
varios, desde cliente a préstamo. Volviendo al diagrama E-R de la figura anterior, se ve que el conjunto de relaciones prestatario es varios a varios.

Si el conjunto de relaciones prestatario fuera uno a varios, desde cliente a préstamo, entonces la línea desde prestatario a cliente sería dirigida, con una flecha apuntando al conjunto de entidades cliente.

Si el conjunto de relaciones prestatario fuera varios a uno desde cliente a préstamo, entonces la línea desde prestatario a préstamo tendría una flecha apuntando al conjunto de entidades préstamo.

Si el conjunto de relaciones prestatario fuera uno a uno, entonces ambas líneas desde prestatario tendrían flechas: una apuntando al conjunto de entidades préstamo y otra apuntando al conjunto de entidades cliente

Diagrama E-R con un atributo unido a un conjunto de relaciones.

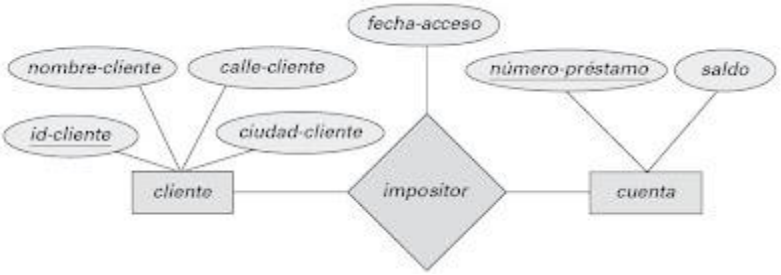
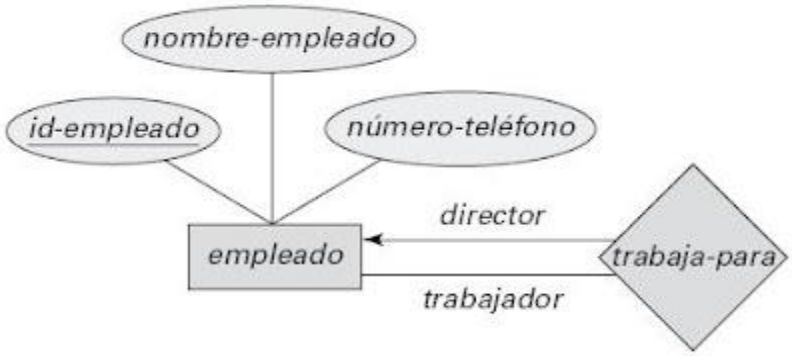


Diagrama E-R con atributos compuestos, multivalorados y derivados.

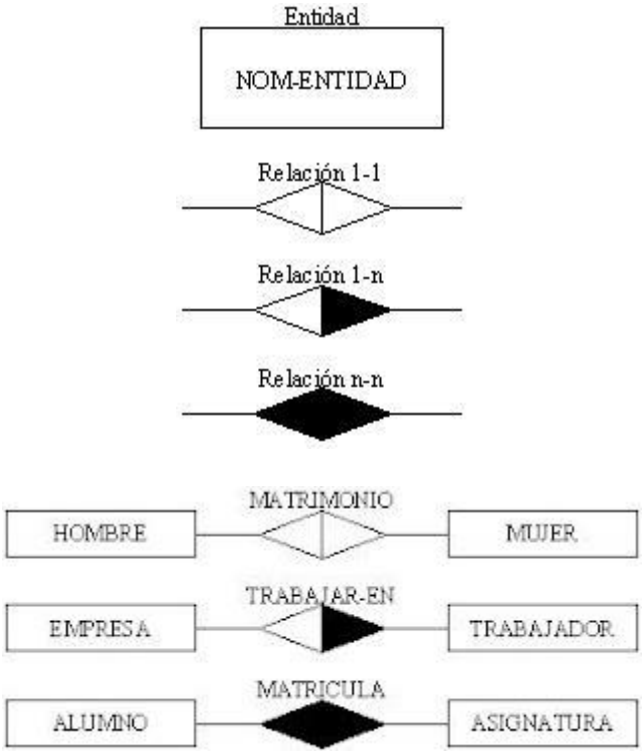
Diagrama E-R con indicadores de papeles.



Los conjuntos de relaciones no binarias se pueden especificar fácilmente en un diagrama E-R. La siguiente figura consta de tres conjuntos de entidades cliente, trabajo y sucursal, relacionados a través del conjunto de relaciones trabaja-en. Se pueden especificar algunos tipos de relaciones varios a uno en el caso de conjuntos de relaciones no binarias. Supóngase un empleado que tenga a lo sumo un trabajo en cada sucursal (por ejemplo, Santos no puede ser director y auditor en la misma sucursal). Esta restricción se puede especificar con una flecha apuntando a trabajo en el borde de trabaja-en.

Límites de cardinalidad en conjuntos de relaciones.

Veamos otra forma de poder representar las relaciones:



2.10 MODELOS DE DATOS SEMANTICOS

modelos de datos semánticos; el aspecto semántico del modelo yace en la representación del significado de los datos. El modelo E-R es extremadamente útil para hacer corresponder los significados e interacciones de las empresas del mundo real con un esquema conceptual. Debido a esta utilidad, muchas herramientas de diseño de bases de datos se basan en los conceptos del modelo E-R. Ejemplo Modelo Entidad - Relacion 2.3 Restricciones Las restricciones de dominio

especifican que el valor de cada atributo A debe ser un valor atómico del dominio dom(a) para ese atributo. Los tipos de datos asociados a los dominios por lo general incluyen los tipos de datos numéricos estándar de los números enteros (como entero- corto, entero, entero-largo) y reales (flotante y flotante de doble precisión). También disponemos de caracteres, cadenas de longitud fija y cadenas de longitud variable, así como tipos de datos de fecha, hora, marca de tiempo y dinero. Otros dominios posibles se pueden

2.11 INTERVALOS DE VALORES

Describir mediante un intervalo de valores de un tipo de datos o como un tipo de datos enumerado en el que se listan explícitamente todos los valores posibles. Restricción de Valores Nulos Para determinado atributos, los valores nulos pueden ser inapropiados. Considérese una tupla en la relación cliente la que nombre-cliente es un valor vació. Una tupla de este tipo da una calle y una ciudad para un cliente anónimo y, por tanto, no contiene información útil. En casos como éste, deseamos prohibir los valores nulos, restringiendo el dominio de ciudad-cliente para que excluya los valores nulos. El SQL estándar permite que la declaración del dominio de un atributo incluya la especificación not null. Esto prohíbe la inserción de un valor nulo para este atributo. Cualquier modificación de la base de datos que causara que se insertase un valor nulo en un dominio not null genera un diagnóstico de error. Hay muchas situaciones en las que la prohibición de valores nulos es deseable. Un caso particular en el que es esencial prohibir los valores nulos es en la clave primaria de un esquema de relación Restricción de clave Es una de las restricciones estándar que con frecuencia aparecen en las aplicaciones de bases de datos. Estas restricciones se manejan de formas ligeramente distintas en los diversos modelos de datos. En el modelo E-R, una clave es un atributo de un tipo de entidades que debe tener un valor único para cada entidad que pertenezca a dicho tipo en cualquier momento específico. Así el valor del atributo clave puede servir para identificar de manera única cada entidad.

UNIDAD III. NORMALIZACION DE BASES DE DATOS Y ALGEBRA RELACIONAL

3.1. NORMALIZACION DE BASE DE DATOS

Normalización de Base de Datos, es el proceso de organizar los datos en una base de datos que incluye la creación de tablas y el establecimiento de relaciones entre ellas. Este proceso es utilizado para ayudar a eliminar los datos redundantes.

Cinco formas de normalización (FN: Forma normal)

1FN: Eliminar grupos repetitivos
2FN: Eliminar datos redundantes
3FN: Eliminar columnas no depende de clave
4FN: Aislar Relaciones Múltiples Independientes
5FN: Aislar relaciones semánticamente relacionadas múltiples

3.2 PRIMERA FORMA NORMAL 1FN

La primera forma normal significa que los datos están en un formato de entidad, lo que significa que se han cumplido las siguientes condiciones:

- Eliminar grupos repetidos en tablas individuales
- Crear una tabla independiente para cada conjunto de datos relacionados
- Identificar cada conjunto de relacionados con la clave principal

No utilice varios campos en una sola tabla para almacenar datos similares. El valor de una columna debe ser una entidad atómica, indivisible, excluyendo así las dificultades que podría conllevar el tratamiento de un dato formado de varias partes.

Supongamos que tienes en una tabla una columna Dirección para almacenar la dirección completa, dato que se compondría del nombre de la calle, el número exterior, el número interior (puerta), el código postal, el estado y la capital.

id	Nombre	Dirección	Teléfono	URL
1	Anaya	J:l: Luca	92199932	Anaya.com
2	Pericles	C/Luna # 20-28018 Tlaxcala	99299492	Pericles.com

Calle	Número	Puerta	CP	Población	Provincia
Luna	20		28018	Tlaxcala	Tlaxcala

Figura 1. Tabla con un atributo divisible en varias partes

Tabla con un atributo divisible en varias partes. Tabla con un atributo divisible en varias partes

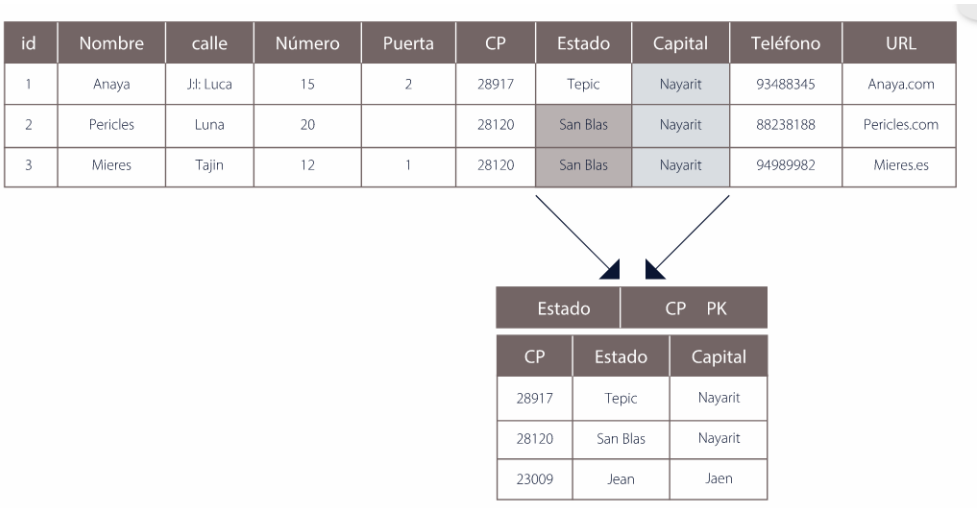
Una tabla con esta estructura plantea problemas a la hora de recuperar información. Imagina que necesitas conocer todas las entradas correspondientes a una determinada población, o que quieres buscar a partir del código postal. Al ser la dirección completa una secuencia de

caracteres de estructura libre no resultaría nada fácil.

Existirán más columnas, pero cada una de ellas contendrá un valor simple e indivisible que facilitará la realización de las operaciones antes mencionadas.

En cuanto a la segunda indicación, se debe evitar la repetición de los datos de la población y provincia en cada una de las filas. Siempre que al muestrear la información de una tabla aparezcan datos repetidos, existe la posibilidad de crear una tabla independiente con ellos.

Si el diseño de nuestra base de datos cumple estas premisas, está preparada para pasar de la primera a la segunda forma normal.



id	Nombre	calle	Número	Puerta	CP	Estado	Capital	Teléfono	URL
1	Anaya	J:l: Luca	15	2	28917	Tepic	Nayarit	93488345	Anaya.com
2	Pericles	Luna	20		28120	San Blas	Nayarit	88238188	Pericles.com
3	Mieres	Tajin	12	1	28120	San Blas	Nayarit	94989982	Mieres.es

Estado	CP PK

CP	Estado	Capital
28917	Tepic	Nayarit
28120	San Blas	Nayarit
23009	Jean	Jaen

Figura 2. Aislamiento de los datos repetitivos de una tabla en otra independiente

Aislamiento de los datos repetitivos de una tabla en otra independiente. Aislamiento de los datos repetitivos de una tabla en otra independiente.

3.3. SEGUNDA FORMA NORMAL (2FN)

Además de cumplir con las dos reglas del punto previo, la segunda forma normal añade la necesidad de *que no existan dependencias funcionales parciales*. Esto significa que todos los valores de las columnas de una fila deben depender de la clave primaria de dicha fila, entendiendo por clave primaria los valores de todas las columnas que la formen, en caso de ser más de una.

Las tablas que están ajustadas a la primera forma normal, y además disponen de una clave primaria formada por una única columna con un valor indivisible, cumplen ya con la segunda forma normal. Ésta afecta exclusivamente a las tablas en las que la clave primaria está formada por los valores de dos o más columnas, debiendo asegurarse, en este caso, que todas las demás columnas son accesibles a través de la clave completa y nunca mediante una parte de esa clave. La segunda forma normal asegura que cada atributo describe la entidad

Crear tablas separadas para el conjunto de valores y los registros múltiples, estas tablas se deben relacionar con una clave externa.

Los registros no deben depender de otra cosa que la clave principal de la tabla, incluida la clave compuesta si es necesario

Segunda Forma Normal: Eliminar Datos Redundantes

Estudiantes:

Estudiante	Tutor	Habitacion
1606	Sarmiento	438
2602	Valle	222

Registro:

Estudiante	Clases
1606	111-01
1606	111-02
1606	111-03
2602	201-01
2602	201-02
2602	201-03

Al pasar a la segunda forma normal vamos a eliminar los datos redundantes, y para lograrlo vamos a crear dos tablas. Una tabla se llamará Estudiantes donde eliminaremos los datos redundantes quedándonos con los datos únicos (Estudiante, Tutor y Habitación) y en una segunda tabla que llamaremos Registro para el numero de estudiante y las clases que llevara en el ejemplo el estudiante 1606 y 2602 llevará cada uno tres clases. El contenido de la (1FN) Primera Forma Normal que estaba en una tabla ha sido divido en dos tablas para eliminar los datos redundantes e introducirlo a la (2FN) Segunda Forma Normal.

3.4. TERCERA FORMA NORMAL. (3FN)

La tercera forma normal comprueba las dependencias transitivas, eliminando campos que no dependen de la clave principal. Los valores que no dependen de la clave principal no pertenecen a la tabla

Los campos que no pertenecen a la clave principal colóquelos en una tabla aparte y relacionen ambas tablas por medio de una clave externa.

Tercera Forma Normal: Eliminar Columnas No Depende De Clave

Estudiantes:

Estudiante	Tutor
1606	Sarmiento
2602	Valle

Facultad:

Nombre	Habitacion	Departamento
Sarmiento	438	42
Vale	222	42

Registro:

Estudiante	Clases
1606	111-01
1606	111-02
1606	111-03
2602	201-01
2602	201-02
2602	201-03

Para pasar a la tercera forma normal tenemos que eliminar los campos de No Dependen de la Clave y para lograrlo dividimos la tabla estudiante en dos tablas y creamos la tabla Facultad donde trasladaremos la columna habitación que No Depende de la Clave que es la columna estudiante, el nombre del tutor será el enlace con la tabla estudiante aunque también podría ser la columna estudiante.

En cuanto a la tercera forma normal, ésta indica que *no deben existir dependencias transitivas entre las columnas de una tabla*, lo cual significa que las columnas que no forman parte de la clave primaria deben depender sólo de la clave, nunca de otra columna no clave.

3.5. FORMA NORMAL BOYCE-CODD. / CUARTA FORMA NORMAL (4FN)

La cuarta forma normal también se llama la forma normal de Boyce Codd (BCNF) y la quinta forma normal existe, pero rara vez se consideran en el diseño práctico. El no tener en cuenta estas dos reglas de normalización adicionales puede resultar en un diseño de base de datos menos perfecto, pero no debería afectar a la funcionalidad La normalización de base de datos es un punto muy importante que deberíamos de tomar muy en serio para establecer cimientos sólidos sobre los cuales podemos construir aplicaciones robustas que en el futuro no presenten problemas de base de datos difíciles de solucionar.

La cuarta forma normal tiene por objetivo eliminar las dependencias multivaluadas.

Definición: Una relación está en 4NF si y sólo si, en cada dependencia multivaluada X ->-> Y no trivial, X es clave candidata.

Una dependencia multivaluada A ->-> B es trivial cuando B es parte de A. Esto sucede cuando A es un conjunto de atributos, y B es un subconjunto de A.

Tomemos por ejemplo la tabla de Agenda, pero dejando sólo los atributos multivaluados:

Agenda (nombre, teléfono, correo)

Lo primero que debemos hacer es buscar las claves y las dependencias. Recordemos que las claves candidatas deben identificar de forma unívoca cada tupla. De modo que estamos obligados a usar los tres atributos para formar la clave candidata. Pero las dependencias que tenemos son:

nombre ->-> teléfono
nombre ->-> correo

Y nombre no es clave candidata de esta relación. Resumiendo, debemos separar esta relación en varias (tantas como atributos multivaluados tenga).

Teléfonos (nombre, teléfono)

Correos (nombre, correo)

Ahora en las dos relaciones se cumple la cuarta forma normal.

DEPENDENCIAS FUNCIONALES

Ya hemos comentado que una relación se compone de atributos y dependencias. Los atributos son fáciles de identificar, ya que forman parte de la estructura de la relación, y además, los elegimos nosotros mismos como diseñadores de la base de datos.

Pero no es tan sencillo localizar las dependencias, ya que requieren un análisis de los atributos, o con más precisión, de las interrelaciones entre atributos, y frecuentemente la intuición no es suficiente a la hora de encontrar y clasificar todas las dependencias.

La teoría nos puede ayudar un poco en ese sentido, clasificando las dependencias en distintos tipos, indicando qué características tiene cada tipo.

Para empezar, debemos tener claro que las dependencias se pueden dar entre atributos o entre subconjuntos de atributos.

Estas dependencias son consecuencia de la estructura de la base de datos y de los objetos del mundo real que describen, y no de los valores actualmente almacenados en cada relación. Por ejemplo, si tenemos una relación de vehículos en la que almacenamos, entre otros atributos, la cilindrada y el color, y en un determinado momento todos los vehículos con 2000 c.c. son de color rojo, no podremos afirmar que existe una dependencia entre el color y la cilindrada. Debemos suponer que esto es sólo algo casual.

Para buscar dependencias, pues, no se deben analizar los datos, sino los entes a los que se refieren esos datos.

Definición: Sean X e Y subconjuntos de atributos de una relación. Diremos que Y tiene una dependencia funcional de X, o que X determina a Y, si cada valor de X tiene asociado siempre un único valor de Y.

El hecho de que X determine a Y no quiere decir que conociendo X se pueda conocer Y, sino que en la relación indicada, cada vez que el atributo X tome un determinado valor, el atributo Y en la misma tupla siempre tendrá el mismo valor.

Por ejemplo, si tenemos una relación con clientes de un hotel, y dos de sus atributos son el número de cliente y su nombre, podemos afirmar que el nombre tiene una dependencia funcional del número de cliente. Es decir, cada vez que en una tupla aparezca determinado valor de número de cliente, es seguro que el nombre de cliente será siempre el mismo. La dependencia funcional se representa como X -> Y.

El símbolo -> se lee como "implica" o "determina", y la dependencia anterior se lee como *X implica Y* o *X determina Y*. Podemos añadir otro símbolo a nuestra álgebra de dependencias: el símbolo | significa negación. Así X -> |Y se lee como *X no determina Y*.

DEPENDENCIA FUNCIONAL COMPLETA: Definición: En una dependencia funcional X -> Y, cuando X es un conjunto de atributos, decimos que la dependencia funcional es completa, si sólo depende de X, y no de ningún subconjunto de X. La dependencia funcional se representa como X => Y.

DEPENDENCIA FUNCIONAL ELEMENTAL: Definición: Si tenemos una dependencia completa X => Y, diremos que es una dependencia funcional elemental si Y es un atributo, y

no un conjunto de ellos. Estas son las dependencias que buscaremos en nuestras relaciones. Las dependencias funcionales elementales son un caso particular de las dependencias completas.

DEPENDECIA FUNCIONAL TRIVIAL: Definición: Una dependencia funcional A -> B es trivial cuando B es parte de A. Esto sucede cuando A es un conjunto de atributos, y B es a su vez un subconjunto de A. Con relación a la segunda forma normal 2FN dice

Definición: Para que una base de datos sea 2FN primero debe ser 1FN, y además todas las columnas que formen parte de una clave candidata deben aportar información sobre la clave completa.

Esta regla significa que en una relación sólo se debe almacenar información sobre un tipo de entidad, y se traduce en que los atributos que no aporten información directa sobre la clave principal deben almacenarse en una relación separada.

Lo primero que necesitamos para aplicar esta forma normal es identificar las claves candidatas.

Además, podemos elegir una clave principal, que abreviaremos como **PK**, las iniciales de Primary Key. Pero esto es optativo, el modelo relacional no obliga a elegir una clave principal para cada relación, sino tan sólo a la existencia de al menos una clave candidata.

La inexistencia de claves candidatas implica que la relación no cumple todas las normas para ser parte de una base de datos relacional, ya que la no existencia de claves implica la repetición de tuplas. En general, si no existe un candidato claro para la clave principal, crearemos una columna específica con ese propósito.

Veamos cómo aplicar esta regla usando un ejemplo. En este caso trataremos de guardar datos relacionados con la administración de un hotel.

Planteemos, por ejemplo, este esquema de relación:

Ocupación(No_cliente, Nombre_cliente, No_habitación, precio_noche, tipo_habitación, fecha_entrada)

Lo primero que tenemos que hacer es buscar las posibles claves candidatas. En este caso sólo existe una posibilidad:

(No_habitación, fecha_entrada)

Recordemos que cualquier clave candidata debe identificar de forma unívoca una clave completa. En este caso, la clave completa es la ocupación de una habitación, que se define por dos parámetros: la habitación y la fecha de la ocupación. Es decir, dos ocupaciones son diferentes si cambian cualquiera de estos parámetros. La misma persona puede ocupar varias habitaciones al mismo tiempo o la misma habitación durante varios días o en diferentes periodos de tiempo. Lo que no es posible es que varias personas ocupen la misma habitación al mismo tiempo (salvo que se trate de un acompañante, pero en ese caso, sólo una de las personas es la titular de la ocupación). El siguiente paso consiste en buscar columnas que no aporten información directa sobre la clave completa: la ocupación. En este caso el precio por

noche de la habitación y el tipo de habitación (es decir, si es doble o sencilla), no aportan información sobre la clave principal.

En realidad, estos datos no son atributos de la ocupación de la habitación, sino de la propia habitación. El número de cliente y su nombre aportan datos sobre la ocupación, aunque no formen parte de la clave completa. Expresado en forma de dependencias se ve muy claro:

(No_habitación, fecha_entrada) -> No_cliente
(No_habitación, fecha_entrada) -> Nombre_cliente
No_habitación -> precio_noche
No_habitación -> tipo_habitación

El último paso consiste en extraer los atributos que no forman parte de la clave a otra relación. En nuestro ejemplo tendremos dos relaciones: una para las ocupaciones y otra para las habitaciones:

Ocupación (No_cliente, Nombre_cliente, No_habitación, fecha_entrada **(PK)**)

Habitación (No_habitación, precio_noche, tipo_habitación)

La segunda tabla tiene una única clave candidata, que es el número de habitación. El resto de los atributos no forman parte de la clave completa (la habitación), pero aportan información sólo y exclusivamente sobre ella. Estas dos relaciones están interrelacionadas por la clave de habitación. Para facilitar el establecimiento de esta interrelación elegiremos la clave candidata de la habitación en clave principal. El mismo atributo en la relación de ocupaciones es, por lo tanto, una clave foránea:

Ocupación (No_cliente, Nombre_cliente, No_habitación, fecha_entrada **(PK)**)

Habitación (No_habitación **(PK)**, precio_noche, tipo_habitación)

Como norma general debemos volver a aplicar la primera y segunda forma normal a estas nuevas tablas. La primera sólo en el caso de que hallamos añadido nuevas columnas, la segunda siempre. La interrelación que hemos establecido es del tipo uno a muchos, podemos elegir una clave de habitación muchas veces, tantas como veces se ocupe esa habitación.

DEPENDENCIA FUNCIONAL TRANSITIVA: Definición: Supongamos que tenemos una relación con tres conjuntos de atributos: X, Y y Z, y las siguientes dependencias X -> Y, Y -> Z, Y -> |X. Es decir X determina Y e Y determina Z, pero Y no determina X. En ese caso, decimos que Z tiene dependencia transitiva con respecto a X, a través de Y. Intentaremos aclarar este concepto tan teórico con un ejemplo. Si tenemos esta relación:

Ciudades (ciudad, población, superficie, renta, país, continente)

Los atributos como población, superficie o renta tienen dependencia funcional de ciudad, así que de momento no nos preocupan. En esta relación podemos encontrar también las siguientes dependencias: ciudad -> país, país -> continente. Además, país -> |ciudad. Es decir, cada ciudad pertenece a un país y cada país a un continente, pero en cada país puede haber muchas ciudades. En este caso continente tiene una dependencia funcional transitiva con respecto a ciudad, a través de país. Es decir, cada ciudad está en un país, pero también en un continente. (¡Ojo! las dependencias transitivas no son siempre tan evidentes :-).

3.6. TERCERA FORMA NORMAL 3FN EN RELACIONES TRASITIVAS

La tercera forma normal consiste en eliminar las dependencias transitivas.

Definición: Una base de datos está en 3FN si está en 2FN y además todas las columnas que no sean claves dependen de la clave completa de forma no transitiva.

Pero esto es una definición demasiado teórica. En la práctica significa que se debe eliminar cualquier relación que permita llegar a un mismo dato de dos o más formas diferentes. Tomemos el ejemplo que usamos para ilustrar las dependencias funcionales transitivas. Tenemos una tabla donde se almacenen datos relativos a ciudades, y una de las columnas sea el país y otra el continente al que pertenecen. Por ejemplo:

Ciudades (ID_ciudad (PK), Nombre, población, superficie, renta, país, continente)

Un conjunto de datos podría ser el siguiente:

Ciudades

ID_ciudad	Nombre	población	superficie	renta	país	continente
1	Paris	6000000	15	1800	Francia	Europa
2	Lion	3500000	9	1600	Francia	Europa
3	Berlín	7500000	16	1900	Alemania	Europa
4	Pekín	19000000	36	550	China	Asia
5	Bonn	6000000	12	1900	Alemania	Europa

Podemos ver que, para cada aparición de un determinado país, el continente siempre es el mismo. Es decir, existe una redundancia de datos, y por lo tanto, un peligro de integridad. Existe una relación entre país y continente, y ninguna de ellas es clave candidata. Por lo tanto, si queremos que esta table sea 3FN debemos separar esa columna:

Ciudades (ID_ciudad (PK), Nombre, población, superficie, renta, nombre_pais)
Países (nombre_pais (PK), nombre continente)
Ciudades

ID_ciudad	Nombre	población	superficie	renta	país

1	Paris	6000000	15	1800	Francia
2	Lion	3500000	9	1600	Francia
3	Berlín	7500000	16	1900	Alemania
4	Pekín	19000000	36	550	China
5	Bonn	6000000	12	1900	Alemania

Países

país	continente
Francia	Europa
Alemania	Europa
China	Asia

Esta separación tendría más sentido si la tabla de países contuviese más información, tal como está no tiene mucho sentido separar estas tablas, aunque efectivamente, se evita redundancia.

Forma Normal de Boycce y Codd (FNBC)

Definición: Una relación está en FNBC si cualquier atributo sólo facilita información sobre claves candidatas, y no sobre atributos que no formen parte de ninguna clave candidata. Esto significa que no deben existir interrelaciones entre atributos fuera de las claves candidatas. Para ilustrar esta forma normal volvamos a uno de nuestros ejemplos anteriores, el de las ocupaciones de habitaciones de un hotel.

Ocupación (No_cliente, Nombre_cliente, No_habitación, fecha_entrada)
Habitación (No_habitación(PK), precio_noche, tipo_habitación)

En la primera relación los atributos No_cliente y Nombre_cliente sólo proporcionan información entre ellos mutuamente, pero ninguno de ellos es una clave candidata.

Intuitivamente ya habremos visto que esta estructura puede producir redundancia, sobre todo en el caso de clientes habituales, donde se repetirá la misma información cada vez que el mismo cliente se aloje en el hotel. La solución, como siempre, es simple, y consiste en separar esta relación en dos diferentes:

Ocupación (No_cliente, No_habitación, fecha_entrada)
Cliente (No_cliente(PK), Nombre_cliente)
Habitación (No_habitación(PK), precio_noche, tipo_habitación)
Atributos multivaluados

Definición: se dice que un atributo es multivaluado cuando para una misma entidad puede tomar varios valores diferentes, con independencia de los valores que puedan tomar el resto de los atributos.

Se representa como X ->-> Y, y se lee como X multidetermina Y.

Un ejemplo claro es una relación donde almacenemos contactos y números de teléfono:

Agenda (nombre, fecha_nacimiento, estado_civil, teléfono)

Para cada nombre de la agenda tendremos, en general, varios números de teléfono, es decir, que nombre multidetermina teléfono: nombre ->-> teléfono. Además, nombre determina funcionalmente otros atributos, como la fecha_nacimiento o estado_civil. Por otra parte, la clave candidata no es el nombre, ya que debe ser unívoca, por lo tanto, debe ser una combinación de nombre y teléfono. En esta relación tenemos las siguientes dependencias:

nombre -> fecha_nacimiento
nombre -> estado_civil

nombre ->-> teléfono, o lo que es lo mismo (nombre,teléfono) -> teléfono

Es decir, la dependencia multivaluada se convierte, de hecho, en una dependencia funcional trivial. Este tipo de atributos implica redundancia ya que el resto de los atributos se repiten tantas veces como valores diferentes tenga el atributo multivaluado:

Agenda

nombre	fecha_nacimiento	estado_civil	teléfono
Mengano	15/12/1985	soltero	12322132
Fulano	13/02/1960	casado	13321232
Fulano	13/02/1960	casado	25565445
Fulano	13/02/1960	casado	36635363
Tulana	24/06/1975	soltera	45665456

Siempre podemos evitar el problema de los atributos multivaluados separándolos en relaciones distintas. En el ejemplo anterior, podemos crear una segunda relación que contenga el nombre y el teléfono:

Agenda (nombre (PK), fecha_nacimiento, estado_civil)
Teléfonos (nombre, teléfono (PK))

Para los datos anteriores, las tablas quedarían así:

Agenda

nombre	fecha_nacimiento	estado_civil
Mengano	15/12/1985	soltero

Fulano	13/02/1960	casado
Tulana	24/06/1975	soltera

Teléfonos

nombre	teléfono
Mengano	12322132
Fulano	13321232
Fulano	25565445
Fulano	36635363
Tulana	45665456

3.8. OPERACIONESFUNDAMENTALES DEL ÁLGEBRA RELACIONAL

Se llama álgebra relacional a un conjunto de operaciones simples sobre tablas relacionales, a partir de las cuales se definen operaciones más complejas mediante composición. Definen, por tanto, un pequeño lenguaje de manipulación de datos. El elemento fundamental del modelo relacional de bases de datos es la tabla relacional. Una tabla relacional es una representación extensional de una relación definida sobre un cierto dominio. Es un método que consiste básicamente en crear o construir nuevas relaciones a partir de relaciones existentes. Existen 2 tipos de operadores algebraicos:

- Operadores básicos o primitivos.
- Operadores no básicos o derivados.
- Operadores básicos o primitivos.

Se clasifican en:

Proyección (π).

Selección (σ). Diferencia (-). Proyección.

Unión (U). Producto cartesiano (X).

Este operador permite extraer columnas de una relación y de esta manera crea un subconjunto de atributos de la relación, además elimina las filas duplicadas.

Selección. Este operador permite seleccionar un subconjunto de filas o registros de una relación y de acuerdo a la condición planteada los registros serán seleccionados para formar parte de un nuevo subconjunto.

Unión. La unión de 2 relaciones R y S es otra relación la cual va a tener los registros de R en S o en ambas, además se eliminan los registros duplicados. En esta relación R y S deben ser compatibles es decir que deben estar definidas sobre el mismo conjunto de atributos.

Diferencia. La diferencia de 2 relaciones R y S es otra relación la cual va a tener los registros que están en R pero no están en S. En esta relación R y S deben ser compatibles.

Producto cartesiano. Es una relación que consiste en la concatenación de cada una de las filas de la relación R con cada una de las filas de la relación S.

Operadores no básicos o derivados. Se clasifican en:

Intersección (∩).

Unión natural ().

División (/).

Intersección. Es una relación que contiene el conjunto de todas las filas que están tanto en la relación R como en S. R y S deben ser compatibles.

Unión natural. El resultado es una relación con los atributos de ambas relaciones y se obtiene combinando vas filas de ambas relaciones que tengan el mismo valor en los atributos comunes. El join se lo usa entre los atributos comunes de las entidades o tablas que poseen la clave primaria de una tabla foránea correspondiente de otra entidad.

División. Define una relación sobre el conjunto de atributos C, incluido en la relación R, y que contiene el conjunto de valores de S, que en las filas de R están combinadas con cada una de las filas de S.

NOTA: Revisar https://www.uoc.edu/pdf/masters/oficiales/img/913.pdf página 83

3.9. VALORES NULOS.

En este apartado se define la forma en que las diferentes operaciones del álgebra relacional tratan los valores nulos y las complicaciones que surgen cuando los valores nulos participan en las operaciones aritméticas o en las comparaciones. Como se verá, a menudo hay varias formas de tratar los valores nulos y, como resultado, las siguientes definiciones pueden ser a veces arbitrarias. Las operaciones y las comparaciones con

valores nulos se deberían evitar

siempre que sea posible. Dado que el valor especial nulo indica «valor desconocido o no existente», cualquier operación aritmética (como +, –, * y /) que incluya valores nulos debe devolver un valor nulo.

De manera similar, cualquier comparación (como <, <=, >, >= y ≠) que incluya un valor nulo se evalúa al valor especial desconocido; no se puede decir si el resultado de la comparación es cierto o falso, así que se dice que el resultado es el nuevo valor lógico desconocido. Las comparaciones que incluyan nulos pueden aparecer dentro de expresiones booleanas que incluyan las operaciones y (conjunción), o (disyunción) y no (negación).

Se debe definir la forma en que estas operaciones tratan el valor lógico desconocido.

NOTA: Consultar https://www.uoc.edu/pdf/masters/oficiales/img/913.pdf páginas 79

Null (SQL). Null (nulo) es un marcador especial usado en el lenguaje de consulta estructurado (SQL) para indicar que no existe un valor dentro de una base de datos. Introducido por el creador del modelo relacional de bases de datos E. F. Codd, su función es la de solventar el requisito de que los sistemas de gestión relacionales de base de datos (en inglés: Database management system, abreviado DBMS) verdaderos puedan representar información "desconocida" o "no aplicable". Asimismo, Codd también introdujo el uso de la letra griega omega (ω) en minúscula para representar el Null en la teoría de la teoría de las bases de datos. NULL es también una palabra reservada en el lenguaje SQL para identificar el marcador especial Null.

Null ha sido un foco de controversia y una fuente de debate debido a su asociación a la lógica ternaria (en inglés: Three-Valued Logic, abreviado 3VL), a sus restricciones de uso en SQL y a la dificultad de su manejo en SQL. Aunque las funciones especiales y predicados sirven para manejar eficazmente el Nulls, la competencia opina que resolver este tipo de cuestiones añade complejidades y contradicciones innecesarias dentro del

modelo relacional de bases de datos. Prueba de valores nulos

Para probar si un campo contiene un valor nulo, utilice el operador IS descrito en Operator=. Efecto de establecer un campo en NULL. Tenga cuidado al asignar el valor Null a un campo. Por ejemplo, el siguiente mandato suprime el campo Name:

SET OutputRoot.XMLNS.Msg.Data.Name = NULL; -- esto suprime el campoCopiar

La forma correcta de asignar un valor Null a un campo es la siguiente:

SET OutputRoot.XMLNS.Msg.Data.Name VALUE = NULL;

-- esto asigna un valor NULL a un campo sin suprimirlo

INFORMACION CON SQL

https://moodle2.unid.edu.mx/dts_cursos_mdl/lic/TI/FB/AM/10/Modificacion_de_bd.pd

3.10. OPERACIONES DE MODIFICACIÓN A LA BASE DE DATOS

Para insertar datos en una relación, se especifica la tupla que se desea insertar o bien se formula una consulta cuyo resultado sea el conjunto de tuplas que se desea insertar. La instrucción para hacer una inserción tiene la siguiente sintaxis:

INSERT INTO R (a1; :::; an) VALUES (v1; :::vn);
Ejemplo:
INSERT INTO cuenta
VALUES (null, 'C-401', 24000);
SELECT num_cuenta
FROM cuenta
WHERE nombreSucursal = 'San Ángel';

Operaciones del álgebra relacional extendida. Las operaciones básicas del álgebra relacional se han ampliado de varias maneras. Una ampliación sencilla es permitir operaciones aritméticas como parte de la proyección. Una ampliación importante es permitir operaciones de agregación, como el cálculo de la suma de los elementos de un conjunto, o su media. Otra ampliación importante es la operación reunión externa, que permite a las expresiones del álgebra relacional trabajar con los valores nulos que modelan la información que falta.

Proyección generalizada. La operación proyección generalizada amplía la operación proyección permitiendo que se utilicen funciones aritméticas en la lista de proyección. La operación proyección generalizada tiene la forma: ΠF1, F2, …, Fn (E) donde E es cualquier expresión del álgebra relacional y F1, F2, …, Fn son expresiones aritméticas que incluyen constantes y atributos en el esquema de E. Como caso especial la expresión aritmética puede ser simplemente un atributo o una constante. El atributo resultante de la expresión límite – saldo-crédito no tiene un nombre. Se puede aplicar la operación renombramiento al resultado de la proyección generalizada para darle un nombre. Como conveniencia notacional, el renombramiento de atributos se puede combinar con la proyección generalizada como se ilustra a continuación: nombre-cliente, (límite – saldo-crédito) as crédito-disponible (información-crédito).

Al segundo atributo de esta proyección generalizada se le ha dado el nombre crédito-disponible.

Las funciones de agregación son funciones que toman una colección de valores y devuelven como resultado un único valor. Por ejemplo, la función de agregación sum toma un conjunto de valores y devuelve la suma de los mismos. Por tanto, la función sum aplicada a la colección {1, 1, 3, 4, 4, 11} devuelve el valor 24. La función de agregación avg devuelve la media de los valores. Cuando se aplica al conjunto anterior devuelve el valor 4. La función de agregación count devuelve el número de elementos del conjunto, y

devolvería 6 en el caso anterior. Otras funciones de agregación habituales son min y max, que devuelven el valor mínimo y el máximo de la colección; en el ejemplo anterior devuelven 1 y 11, respectivamente. Las colecciones en las que operan las funciones de agregación pueden tener valores repetidos; el orden en el que aparezcan los valores no tiene importancia. Estas colecciones se denominan multiconjuntos. Los conjuntos son un caso especial de los multiconjuntos, en los que sólo hay una copia de cada elemento. Para ilustrar el concepto de agregación se utilizará la relación trabajo-por-horas, que muestra los empleados a tiempo parcial. Supóngase que se desea averiguar la suma total de los sueldos de los empleados del banco a tiempo parcial. La expresión del álgebra relacional para esta consulta es: Gsum(sueldo) (trabajo-por-horas)

La operación del álgebra relacional G significa que se debe aplicar agregación, y el subíndice indica la operación de agregación a aplicar. El resultado de la expresión anterior es una relación con un único atributo, que contiene una sola fila con un valor correspondiente a la suma de todos los sueldos

3.11.- TIPOS DE ALMACENAMIENTO DE DATOS.

Para este tema consultar los siguientes links:

https://techlandia.com/cuales-son-dispositivos-almacenamiento-datos-sobre_349592/

https://www.informaticaparatunegocio.com/blog/tipos-almacenamiento-la-nube-mas-comunes/

https://blog.simyo.es/los-10-mejores-servicios-de-almacenamiento-en-la-nube/

3.12. SEGURIDAD, INTEGRIDAD Y CONFIDENCIALIDAD DE DATOS.

Para este tema consultar los siguientes links:

https://blog.powerdata.es/el-valor-de-la-gestion-de-datos/la-importancia-de-la-seguridad-e-integridad-en-base-de-datos

https://www.powerdata.es/seguridad-de-datos

https://es.slideshare.net/Drakonis11/integridad-y-seguridad-en-las-bases-de-datos-presentation

3.13 VALORES NULOS

Se utiliza un valor NULL en una base de datos relacional cuando el valor de una columna es desconocido o falta. Un valor NULL no es una cadena vacía (para tipos de datos de caracteres o de fecha y hora) ni un valor cero (para tipos de datos numéricos). La especificación ANSI SQL-92 indica que un valor NULL debe ser el mismo para todos los tipos de datos, de modo que todos los valores NULL se traten de manera uniforme. El espacio de nombres System.Data.SqlTypes proporciona la semántica de NULL implementando la interfaz INullable. Cada uno de los tipos de datos de System.Data.SqlTypes tiene su propia propiedad IsNull y un valor Null que se puede asignar a una instancia de ese tipo de datos.

3.14 TABLAS DE VERDAD

Toda proposición puede ser verdadera (V) o falsa (F). Las tablas de verdad son un método para saber si una fórmula molecular (es decir, formada por varias proposiciones) es siempre V, a veces V o nunca V (es decir, siempre F). Si los valores son siempre V tenemos una Tautología, si siempre son F estamos ante una contradicción. Como hemos dicho, cualquier proposición puede tener valor V (verdadero) o F (falso). Cuando hacemos una tabla de verdad asignamos todas las combinaciones posibles de valores para esas variables proposicionales. Si la formulación consta de una variable "p" tenemos 2 valores de verdad (V y F). Si la formulación consta de 2 variables "p" y "q" tenemos 2 elevado a dos, es decir, 4 posibilidades. Si la formulación consta de 3 variables “p”, “q” y “r”, tenemos 2 elevado a 3, es decir, 8 posibilidades.

La construcción de la tabla de verdad que previamente hemos realizado nos indica todas las

posibilidades que hay para combinar proposiciones dependiendo de si estas son verdaderas o falsas. Pero para tener la tabla de verdad completa debemos incluir las relaciones lógicas existentes entre esas proposiciones. Una vez hemos construido la parte de la tabla de verdad que nos indica las distintas combinaciones de los valores de verdad debemos añadir nuevas columnas a la derecha, en ellas iremos escribiendo las relaciones entre proposiciones, atendiendo siempre a su estructura lógica, y debajo de la fórmula escrita iremos poniendo el valor de verdad correspondiente en caso de ser V-V o V-F o F-V… Veamos el siguiente ejemplo.

3.15 INTEGRIDAD DE ENTIDADES

La integridad de entidad define una fila como entidad única para una tabla determinada. La integridad de entidad exige la integridad de las columnas de los identificadores o la clave principal de una tabla, mediante índices y restricciones UNIQUE, o restricciones PRIMARY KEY.

La integridad de dominio viene dada por la validez de las entradas para una columna determinada. Puede exigir la integridad de dominio para restringir el tipo mediante tipos de datos, el formato mediante reglas y restricciones CHECK, o el intervalo de valores posibles mediante restricciones FOREIGN KEY, restricciones CHECK, definiciones DEFAULT, definiciones NOT NULL y reglas.

Ejemplos:

Ø Si en la relación EMPLEADOS (*DNI, nombre, apellido, edademp*) hemos declarado que dominio (*DNI*) es el dominio predefinido de los enteros, entonces no podremos insertar, por ejemplo, ningún empleado que tenga por *DNI* el valor "Luis", que no es un entero.

Ø Si en la relación EMPLEADOS (DNI, nombre, apellido, edademp) se ha declarado que dominio (DNI) es el dominio predefinido de los enteros, entonces no se permitirá consultar todos aquellos empleados cuyo DNI sea igual a 'Elena' (DNI = 'Elena'). El motivo es que no tiene sentido que el operador de comparación = se aplique entre un DNI que tiene por dominio los enteros, y el valor 'Elena', que es una serie de caracteres.

UNIDAD IV. LENGUAJE SQL

4.1. INTRODUCCION

Oracle vs MySQL vs SQL Server: una comparación entre los Sistemas Gestores de Bases de Datos Relacionales más Populares. En una entrada anterior estuvimos analizando a grandes rasgos algunos de los sistemas gestores de base de datos existentes, tratamos de darte una idea a grandes rasgos de lo que era cada uno y los beneficios que ofrecen para que al momento de necesitar de estas herramientas pudieras elegir la que mejor se acomodara a tus necesidades, en esta ocasión me interesa apoyarte un poco más en esta tarea, pero concentrando la investigación en los 3 principales gestores de la actualidad, me refiero a Oracle, MySQL y SQL Server, espero y en esta ocasión te quede un conocimiento más claro al respecto. Desde su introducción en la década de 1980, los sistemas de gestión de bases de datos relacionales (RDBMS) se han convertido en el tipo de base de datos estándar para una gran cantidad e industrias. Como su nombre lo indica, estos sistemas se basan en el modelo relacional que organiza los datos en grupos de tablas que se relación por el tipo de datos que contienen. Este artículo explora la historia y las características de tres RDBMS populares: Oracle, MySQL y SQL Server. La comparación le ayudará a entender las diferencias entre los 3 sistemas, y si está considerando la implementación de un RDBMS, intentaremos proporcionarle información que le ayude a tomar una decisión. Si usted está interesado en aprender más sobre cómo funcionan las RDBMS, hay muchos cursos online disponibles. Por ejemplo, Introducción a Oracle SQL es un curso que lo puede ayudar a introducirse a esta plataforma, y le enseñaré información detallada acerca de cómo funciona.

Sumario Comparativo de Características

La siguiente tabla muestra información acerca de Oracle, MySQL y Bases de Datos de Servidores SQL, y como ellas se comparan.

Feature	Oracle	MySQL	SQL Server
Interfaz	GUI, SQL	SQL	GUI, SQL, Various

Lenguaje Soportado	Many, including C, C#, C++, Java, Ruby, and Objective C	Many, including C, C#, C++, D, Java, Ruby, and Objective C	Java, Ruby, Python, VB, .Net, and PHP
Sistema Operativo	Windows, Linux, Solaris, HP-UX, OS X, z/OS, AIX	Windows, Linux, OS X, FreeBSD, Solaris	Windows
Licencia	Propietario	Código Libre	Propietario

Oracle

IBM fue la primera empresa en desarrollar un sistema de gestión de bases de datos relacionales, sin embargo, Oracle Corporation hizo historia en 1980 por la liberación para uso comercial de su RDBMS, Oracle. Solo unos pocos años después, la compañía lanzaría una versión de su sistema de computadoras de IBM. Desde su exposición al mercado de RDBMS, Oracle ha llevado el camino constantemente. De acuerdo con Gartner, Oracle poseía casi el 50 % del mercado de RDBMS en 2011. Además de la apertura del mercado comercial para RDBMS, Oracle Corporation también fue la primera empresa en desarrollar una versión de nivel comercial de SQL, que fue diseñado para manipular datos en un RDBMS utilizando (en ese momento) consultas y conexiones.

Características

La primera versión "real" del sistema de gestión de bases de datos relacionales Oracle fue Oracle 2. Este sistema admitía sólo las características básicas de SQL, y estaba escrito en un lenguaje ensamblador. Al año siguiente, y durante los próximos 10 años más o menos, Oracle Corporation lanzó actualizaciones a su base de datos buque insignia. Probablemente una de las razones por las que el sistema de gestión de bases de datos relacionales de Oracle ha logrado permanecer en la cima sea gracias a sus actualizaciones de productos que están estrechamente vinculados a los cambios en el mercado. Palabras

de moda de bases de datos tales como “escalable”, “programable”, “distribuida”, y “portátil” también están vinculadas a la liberación de Oracle. Por ejemplo, en 1985 se añadió soporte para un modelo cliente-servidor a la espera de una aceptación cada vez mayor de la comunicación por red. A medida que Internet allanó el camino para la era digital, el sistema de gestión de bases de datos relacionales de Oracle se ha actualizado para incluir una máquina virtual Java nativa (JVM) .

Oracle Database 12c es la más reciente liberación, e incluye las siguientes características:

- Nueva redacción de datos para mejorar la seguridad de datos sensibles
- La introducción de la plataforma de Oracle Advanced Analytics
- Nuevo manejo de base de datos para los archivos Flash Data Archive (FDA)
- El apoyo a la integración con los grupos de procesadores de sistema operativo
- Apoyo al bombeo de los datos para la consolidación de las base de datos
- Varias mejoras en Oracle Application Express, una herramienta de desarrollo rápido que permite a los usuarios desarrollar aplicaciones web utilizando SQL y / o PL /SQL.
- Compresión avanzada de la red para mejorar el rendimiento

SQL Server

Microsoft SQL Server entró en el mercado de los RDBMS como un competidor serio a mediados de 1990, cuando Microsoft compró a Sybase, y luego lanzó la versión 7.0. Las empresas originalmente trabajaron juntas para desarrollar la plataforma y hacerla funcionar en el sistema operativo de IBM OS/2. No obstante, Microsoft finalmente desarrolló su propio sistema operativo (Windows NT), y quería trabajar solo para crear una gestión de base de datos única para su nuevo sistema operativo. Se necesitarían varios años para que Microsoft y Sybase cortaran completamente sus lazosg. Sybase finalmente cambió el nombre de su producto de modo que fuera totalmente diferente al producto vendido a Microsoft. Microsoft SQL Server versión 4.2 fue la versión inicial.

Características

En 2000, Microsoft lanzó SQL Server 2000. El lanzamiento fue un hito importante para la empresa, ya que fue la primera versión del producto, donde se reemplazó el código original de Sybase. Trabajando en la misma línea que Oracle Corporation, Microsoft ha tratado de mejorar SQL Server para seguir el ritmo de los cambios tecnológicos. SQL Server 2005 es un ejemplo. El lenguaje de marcado extensible (XML) recibió el sello de aprobación del W3C y comenzó a ganar terreno a finales de 1990. Una de las principales novedades de SQL Server 2005 fue el apoyo a los datos XML. Otras características notables del producto insignia incluyen la introducción de SQL Server Always On (tecnología de gestión de datos para disminuir el tiempo de inactividad del usuario a raíz de fallos en el sistema), soporte para datos estructurados y semi-estructurados, una mayor compresión, y varios complementos para apoyar a otros productos en el mercado. SQL Server 2012 se proclamó como la última versión que incluye soporte nativo para OLE. Un curso esencial de SQL Server 2012, Crea, Gestiona y realiza peticiones puede ofrecer más información sobre esta plataforma y cómo usarla.

SQL Server 2014 es la última versión de SQL Server e incluye las siguientes características:

- Introducción de In-Memory Online Traction Processing (OLTP), una característica incorporada que permite una sofisticada gestión de base de datos para mejorar el rendimiento.
- Nuevas soluciones para manejar la recuperación de desastres
- Versión actualizada de la Herramientas de Datos de SQL Server para Inteligencia de Negocios (BI SSDT)

MySQL

Hay dos aspectos únicos de MySQL en comparación con Oracle y SQL Server: no se desarrolló originalmente para uso comercial y es una base de datos de código abierto. El surgimiento de esta plataforma de base de datos fue una casualidad sucedida a las personas que comenzaron a desarrollarla mientras trataban de usar mSQL para conectar sus tablas en la base de datos, y decidieron que necesitaban una interfaz mucho más

potente. La fase inicial de MySQL utilizó una API heredada de mSQL, mejoras que aumentan considerablemente la velocidad, y otras características que incluían el motor de almacenamiento InnoDB, búsqueda de texto, la portabilidad y la internacionalización. Otra diferencia de la plataforma de MySQL en comparación con las otros dos es que es de código abierto. La era digital dio lugar a un movimiento de colaboración para el desarrollo de software que se ha convertido en un mercado competitivo para las bases de datos y otros softwares. De acuerdo con informes de mercado, hay alrededor de 10 millones de instalaciones de MySQL, lo que indica que la plataforma se está moviendo rápidamente en el espacio empresarial. La propiedad de MySQL ha hecho la transición desde los humildes inicios del producto. Las dos adquisiciones más notables son: (1) en 2008, cuando Sun Microsystems adquirió MySQL AB, la compañía que creó MySQL, y (2) en 2010, cuando Oracle compró Sun Microsystems.

Características

Oracle y SQL Server se consideran herramientas que favorecen a los usuarios con los sistemas empresariales de gran tamaño, mientras que MySQL se considera una herramienta que apela más a menudo a las personas interesadas en la gestión de las bases de datos asociadas a sus sitios web. Al igual que con Oracle y SQL Server, MySQL ha liberado actualizaciones para su software casi todos los años. La versión original fue desarrollada a mediados de la década de 1990. Los cambios más notables a MySQL fueron en 2010, el momento de la última adquisición en 2010. Las mejoras en esta versión (GA release 5.5) incluyen replicación semisincrónica, el particionamiento personalizado, soporte mejorado para SMP y las actualizaciones del subsistema InnoDB E/S. A lo mejor le parezca interesante conocer más detalles de MySQL. Esta vez se te han presentado más detalles, los cuales podrías considerar importantes ahora que estás adquiriendo un poco más de conocimiento sobre lo que son las bases de datos y los sistemas gestores de base de datos. Es importante que recuerdes tal y como te lo dije anteriormente, que no se trata de ayudarte a decidir cuál de estos gestores es el mejor de todos, sino de ayudarte a determinar cuál de ellos es el mejor para lo que tú necesitas, es por eso que no debes solamente fijarte en las ventajas o desventajas de cada uno, sino en tus necesidades,

podría ser que el que a primera vista tiene menos ventajas, sea el que resuelve mejor aquellos problemas o satisface aquellas necesidades de tu proyecto o sistema.

¿Qué Diferencia Hay Entre Sqlite Y Sql? Sé SQLite Base de datos se utiliza en dispositivos móviles (Android, iPhone) y es ligero, sólo tiene espacio Kb. ¿Existe alguna limitación en SQLite? Quiero saber cómo son diferentes. Cada base de datos SQL utiliza su propia implementación del lenguaje que varía ligeramente. Si bien las consultas básicas son casi universales, destacan los matices entre MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database, etc.

Lo que es particularmente notable acerca de SQLite es que a diferencia de todos los otros mencionados anteriormente, este software de base de datos no viene con un daemon que las consultas se pasan a través. Esto significa que si varios procesos están utilizando la base de datos a la vez, estarán alterando directamente los datos a través de la biblioteca SQLite y haciendo las llamadas de datos de lectura / escritura al mismo sistema operativo. También significa que los mecanismos de bloqueo no se ocupan muy bien de contención. Esto no es un problema para la mayoría de las aplicaciones donde uno podría pensar en usar SQLite – los pequeños beneficios generales y la recuperación de datos fácil valen la pena. Sin embargo, si va a tener acceso a su base de datos con más de un proceso o no considerar la asignación de todas sus solicitudes a través de un hilo, podría ser un poco molesto.

Sqlite es una versión muy ligera de SQL que soporta muchas características de SQL. Básicamente se ha desarrollado para dispositivos pequeños como teléfonos móviles, tabletas, etc. SQLite es un motor de base de datos de terceros, de código abierto y en proceso. SQL Server Compact es de Microsoft, y es una versión simplificada de SQL Server. Son dos motores de base de datos de competencia. SQL es el lenguaje de consulta. Sqlite es un sistema de gestión de base de datos relacional incorporable. Sqlite también no requiere un servidor de base de datos especial ni nada. Es sólo un motor de sistema de archivos directo que utiliza la sintaxis de SQL. (Por: Adam Plocher)

Techinicamente, SQLite no es un software de código abierto, sino más bien de dominio público. No hay licencia. (Por: Larry Lustig) SQL es el lenguaje de consulta. Sqlite es un sistema de gestión de base de datos relacional incorporable. A diferencia de otras bases de datos (como SQL Server y MySQL), SQLite no admite procedimientos almacenados.

SQLite está basado en archivos, a diferencia de otras bases de datos, como SQL Server y MySQL que están basadas en servidor. SQL es un lenguaje de consulta estructurado utilizado para consultar una base de datos generalmente sistemas de base de datos relacional. SQL es un estándar que especifica cómo se crea un esquema relacional, se insertan o se actualizan los datos en las relaciones, se inician y se detienen las transacciones, etc.

El Lenguaje de Definición de Datos (DDL) y el Lenguaje de Manipulación de Datos (DML), Embedded SQL y Dynamic SQL son Componentes de SQL. Algunas de las bases de datos SQL son MySQL, Oracle, Microsoft SQL Server, IBM DB2, etc. SQLite es Embeddable Relational Database Management Systems escrito en ANSI-C. SQLite está basado en archivos, mientras que SQL Server y MySQL están basados en servidor, SQLite soporta muchas características de SQL y tiene alto rendimiento y no admite procedimientos almacenados. SQLite se utiliza en Android Development para implementar el concepto de base de datos. SQL es una base de datos que consulta el lenguaje y SQLite es una base de datos (RDBMS) que utiliza las especificaciones de SQL. SQLite se puede decir como competidor de SQL Server de Microsoft. El propio nombre sugiere que es la versión ligera de SQL RDBMS. Se utiliza en la mayoría de los dispositivos pequeños y portátiles como Android y dispositivos iOS.

4.2. LENGUAJE DE MANIPULACION DE DATOS

Lenguaje de Manipulación de Datos (Data Manipulation Language, DML) es un idioma proporcionado por los sistemas gestores de bases de datos que permite a los usuarios de

la misma llevar a cabo las tareas de consulta o modificación de los datos contenidos en las Bases de Datos del Sistema Gestor de Bases de Datos. El lenguaje de manipulación de datos más popular hoy día es SQL, usado para recuperar y manipular datos en una base de datos relacional. Otros ejemplos de DML son los usados por bases de datos IMS/DL1, CODASYL u otras. Elementos del lenguaje de manipulación de datos Select, Insert, Delete y Update

Clasificación de los DML. Se clasifican en dos grandes grupos:

Lenguajes procedimentales. En este tipo de lenguaje el usuario da instrucciones al sistema para que realice una serie de procedimientos u operaciones en la base de datos para calcular un resultado final.

En los lenguajes no procedimentales el usuario describe la información deseada sin un procedimiento específico para obtener esa información.

4.3. LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

Un lenguaje de manipulación de datos (Data Manipulation Language, o DML en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado. El lenguaje de manipulación de datos más popular hoy día es SQL, usado para recuperar y manipular datos en una base de datos relacional. Otros ejemplos de DML son los usados por bases de datos IMS/DL1, CODASYL u otras.

1- INSERT: Una sentencia INSERT de SQL agrega uno o más registros a una (y sólo una) tabla en una base de datos relacional.

Ejemplo 1 (inserto valores alumno pepe en la materia spd2 a la tabla cursada):

INSERT INTO "cursada" ("alumno", "materia") VALUES ("pepe", "spd2")

2- UPDATE: Una sentencia UPDATE de SQL es utilizada para modificar los valores de un conjunto de registros existentes en una tabla.

Ejemplo 1 (modifico la materia donde el alumno sea pepe):

UPDATE "cursada" SET "materia"= "spd3" WHERE "alumno"= "pepe"

3- DELETE: Una sentencia DELETE de SQL borra uno o más registros existentes en una tabla.

Ejemplo 1 (borro todos los valores de las columnas alumno y materia donde la materia sea spd2):

DELETE FROM "cursada" WHERE "materia"= "spd2"

Tutorial SQlite3 Tipos DML DDL: Este es el primer tutorial para usar Sqlite en su versión 3 que es la versión que uso, yo tengo instalada la versión 3.7.17, no la instale directamente pero se debe haber instalado como dependencia con algún conjunto de herramientas que instale en sistema, por lo que lo más seguro es que no necesiten instalar en su sistema o lo que es lo mismo ya la deben tener instalada.

Antes que nada este tutorial o serie de tutoriales va dedicado solamente a sistemas Linux de escritorio, no BSD,no Windows,no iOS, no Android, etc.Ya que estos tutoriales los vamos a realizar desde la misma shell de SQlite o sea desde la interfaz que nos ofrece la terminal de nuestro sistema Linux. Bien el título de la entrada parece un poco raro ya que no acostumbro a hacer referencia a las introducciones de temas que hago, pero vamos a explicarlo un poco. Porqué en este caso es necesario:

SQL es un lenguaje de consulta más en específico un **lenguaje de consulta estructurados** ahora en inglés:

Structured Query Language

Los SQL se usan para hacer consultas a bases de datos, en este caso SQlite3 es nuestro consultor y aunque vamos a usar un nivel intermedio para comunicarnos con las bases de datos y no me refiero al nivel de dificultad, bueno un poco ya que por ejemplo hasta un usuario de una Pc que no sepa nada de programación o de nada, solo que entre a un navegador web y busque algo en Google, por ejemplo ya está manipulando una base de datos, pero a un nivel muy alto, de usuario final, en este caso nosotros usaremos un nivel intermedio gracias a SQL,en este caso SQLite.

El DML es llamado así por la abreviación de Lenguaje de Manipulación de Datos, ahora en inglés:

Data Manipulation Language

Los comandos (en rojo) se usan para lo siguiente(morado):

SELECT Consulta registros

INSERT Inserta datos en una sola operación.

DELETE Borra (modifica) los datos especificados de un registro.

UPDATE Actualiza (modifica) los datos especificados de un registro**.**
Esto se usa para manipular datos en una base de datos de modelo relacional, lo que es SQLite.

DML es llamado así por la abreviación de Lenguaje de Definicion de Datos,en inglés:

Data definition language

Los comandos (en rojo) se usan para lo siguiente (morado):

CREATE Crea nuevas tablas e índices
DROP Elimina tablas e índices
ALTER Modifica las tablas agregando columnas o modificando su valor.

Bien no me gusta mucho meterme en esa parte pero bueno, primero que nada vamos a abrir una terminal y vamos a crear un archivo db llamado hola.db:

sqlite3 hola.db

Si no encuentras sqlite3 deben instalarlo según su distribución, pero lo más seguro es que

ya lo tengan.

En este caso crea un base de datos hola.db si ya existe la abre si no existe la crea.

Los comentarios en SQLite son similares a los de Lua para una sola linea, deben comenzar con dos guiones medios:

-- Un comentario

Y para los de varias líneas se parecen a los de PHP:

```
/*
 Un comentario de varias lineas
 De varias lineas :D
*/;
```

Los comandos básicos que podemos usar son:

.help – *Muestra ayuda*

.version – *Muestra la versión*

.databases – *Muestra las bases de datos*

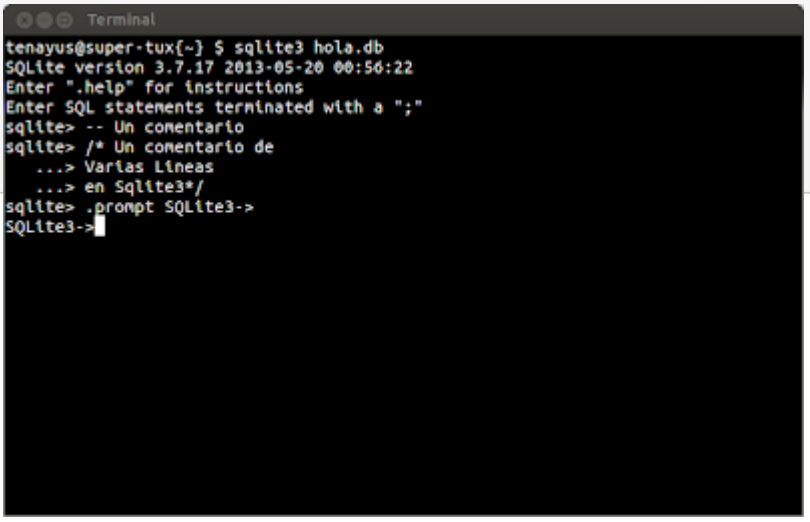
..exit – *Sale*

.quit – *Sale*

.tables – *Muestra las tablas*

.read script**(**lineas de ordenes **SQL)** – *Ejecuta scripts en .sqlite*

.prompt estilo**->**PS1 estilo**->**PS2 – *Cambia el estilo de la prompt*



Estos son los tipos de datos que podemos implementar en SQLite3:

NULL Valor nulo.

INTEGER Numero entero almacenado en 1, 2, 3, 4, 6 bytes.

REAL Valor de punto flotante, guardado en 8-byte como numero de coma flotante.

TEXT Texto.

BLOB Dato no especificado (Binario) que se guarda como entra.

De momento eso es todo en las siguientes partes veremos ejemplos del uso de DLL y DML.

CONSULTAS SQL

Estructura básica de consultas SQL

En un lenguajes relacional siempre vamos a tener una estructura básica de consulta, esta consta de tres cláusulas **select , from , where.** Cláusula select: se usa para obtener una relación de los atributos deseados de una consulta, en claro español muy costarricense es seleccionar

Cláusula from: se corresponde con la operación productos cartesianos del algebra relacional. Genera una lista de las relaciones que debe ser analizada en la evaluación de la expresión, en un claro léxico muy porteño es "DE".

Además, cabe mencionar que por lo general cuando se hace una sentencia de consulta se emplea un comodín que hacer referencia a todas las columnas este comodín es el asterisco "*"

Uniendo todo lo anteriores se puede interpretarse algo así como selección todas las columnas DE y el nombre de la tabla. La sentencia completa serial así:

Select * from empleados_nuevos

	numero_identifiacion	nombre_empleado	primer_apellido	segundo_apellido	genero
1	55555555	Allan	Mathison	Turing	M

Y muestra todos los valores que tiene la tabla que le indicamos. Muy bien ahora ya es hora de que vallamos recordando lo que dijimos al principio de esta serie de artículos la utilización de la base de datos de ejemplo que nos brinda Microsoft para SQL Server si requiere ver esta entrada aquí les dejo el link

http://codigoabiertopuntarenas.blogspot.com/2016/05/preparandonos-para-sql.html

Aquí es donde iniciamos a usar AdventureWork la versión que desee en mi caso estoy usando 2014, vamos hacer la siguiente consulta select NAME

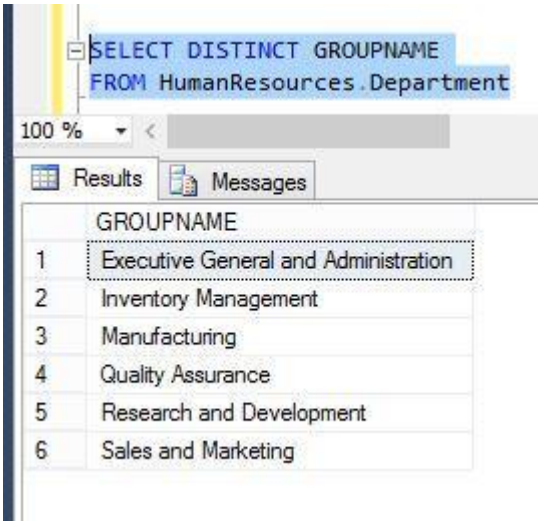
FROM HumanResources.Department

Podemos hacer un select con el nombre de la columna y le indicamos el nombre de la tabla

Podemos hacer otro select pero que muestre solo los diferentes o distintos,

SELECT DISTINCT GROUPNAME

FROM HumanResources.Department



Podemos hacer un select que muestre todos los valores todos los valores incluso los duplicados

SELECT ALL GROUPNAME

FROM HumanResources.Department

Podemos hacer un select especificado el nombre de las columnas que deseamos que muestre

SELECT DepartmentID,Name,ModifiedDate

FROM HumanResources.Department

```
SELECT DepartmentID,Name,ModifiedDate
FROM HumanResources.Department
```

100 %

Results | Messages

	DepartmentID	Name	ModifiedDate
1	1	Engineering	2008-04-30 00:00:00.000
2	2	Tool Design	2008-04-30 00:00:00.000
3	3	Sales	2008-04-30 00:00:00.000
4	4	Marketing	2008-04-30 00:00:00.000
5	5	Purchasing	2008-04-30 00:00:00.000
6	6	Research and Development	2008-04-30 00:00:00.000
7	7	Production	2008-04-30 00:00:00.000
8	8	Production Control	2008-04-30 00:00:00.000
9	9	Human Resources	2008-04-30 00:00:00.000
10	10	Finance	2008-04-30 00:00:00.000
11	11	Information Services	2008-04-30 00:00:00.000
12	12	Document Control	2008-04-30 00:00:00.000
13	13	Quality Assurance	2008-04-30 00:00:00.000
14	14	Facilities and Maintenance	2008-04-30 00:00:00.000
15	15	Shipping and Receiving	2008-04-30 00:00:00.000
16	16	Executive	2008-04-30 00:00:00.000

La siguiente clausula es where: se corresponde con el predicado selección del algebra relacional. Es un predicado que engloba a los atributos a de las relaciones que aparecen en la cláusula from. En claro español muy tico esto significa donde, ¿pero nos preguntamos donde qué ?, eso es lo que determinamos y le indicamos lo que buscamos

Select * from HumanResources.Department

Y nos mostrara lo siguiente:

	DepartmentID	Name	GroupName	ModifiedDate
1	1	Engineering	Research and Development	2008-04-30 00:00:00.000
2	2	Tool Design	Research and Development	2008-04-30 00:00:00.000
3	3	Sales	Sales and Marketing	2008-04-30 00:00:00.000
4	4	Marketing	Sales and Marketing	2008-04-30 00:00:00.000
5	5	Purchasing	Inventory Management	2008-04-30 00:00:00.000
6	6	Research and Development	Research and Development	2008-04-30 00:00:00.000
7	7	Production	Manufacturing	2008-04-30 00:00:00.000
8	8	Production Control	Manufacturing	2008-04-30 00:00:00.000
9	9	Human Resources	Executive General and Administration	2008-04-30 00:00:00.000
10	10	Finance	Executive General and Administration	2008-04-30 00:00:00.000
11	11	Information Services	Executive General and Administration	2008-04-30 00:00:00.000
12	12	Document Control	Quality Assurance	2008-04-30 00:00:00.000
13	13	Quality Assurance	Quality Assurance	2008-04-30 00:00:00.000
14	14	Facilities and Maintenance	Executive General and Administration	2008-04-30 00:00:00.000
15	15	Shipping and Receiving	Inventory Management	2008-04-30 00:00:00.000
16	16	Executive	Executive General and Administration	2008-04-30 00:00:00.000

Como vemos muestra todo el contenido de la tabla pero ahora aplicamos el where y lo que deseamos ver es solo el departamento con el id 1. Como seria esta sentencia

```
select *
from HumanResources.Department
where DepartmentID= 1
```

100 %

Results | Messages

	DepartmentID	Name	GroupName	ModifiedDate
1	1	Engineering	Research and Development	2008-04-30 00:00:00.000

Pues bien cómo podemos observar el where es un gran aliado para buscar elementos en una tabla pero con el where podemos usar conectivas lógicas AND, OR y NOT, pero

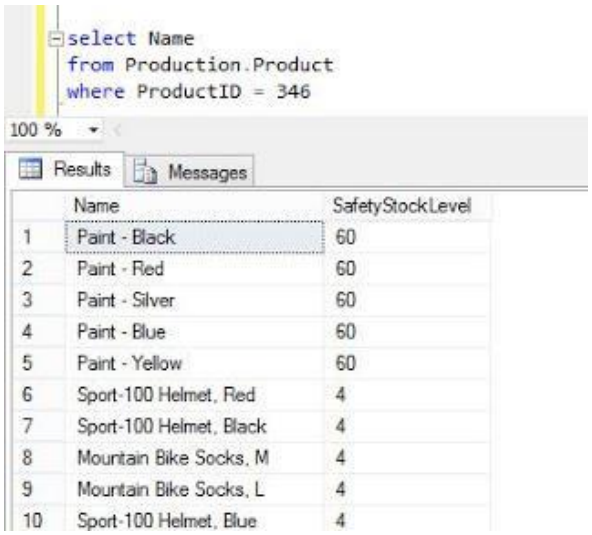
además podemos utilizar operadores de comparación <, <=,>, <=,= y <> que son menor, menor igual que, mayor, mayor igual que, igual y distinto, adicional a todo lo anterior se incluye en SQL un operador de comparación que simplifica la cláusula where este es el between

Vamos a cambiar de tabla para hacer esto más interesante utilizaremos la tabla production.production

```
select Name

from Production.Product

where ProductID = 346
```



	Name	SafetyStockLevel
1	Paint - Black	60
2	Paint - Red	60
3	Paint - Silver	60
4	Paint - Blue	60
5	Paint - Yellow	60
6	Sport-100 Helmet, Red	4
7	Sport-100 Helmet, Black	4
8	Mountain Bike Socks, M	4
9	Mountain Bike Socks, L	4
10	Sport-100 Helmet, Blue	4

Esta sentencia lo que dice es muéstrame el producto que posee el id numero 346

Where se puede emplear para realizar búsquedas por monto o valores como los indicamos anteriormente aquí está la forma

```
select Name,SafetyStockLevel
from Production.Product
where SafetyStockLevel between 800 and 1000
```

100 %

Results | Messages

	Name	SafetyStockLevel
1	Adjustable Race	1000
2	Bearing Ball	1000
3	BB Ball Bearing	800
4	Headset Ball Bearings	800
5	Blade	800
6	Chainring Bolts	1000
7	Chainring Nut	1000
8	Chainring	1000
9	Crown Race	1000
10	Chain Stays	1000
11	Decal 1	1000
12	Decal 2	1000
13	Down Tube	800

Betwee se emplea para buscar valores entre un rango menor y uno mayor

```
select Name,SafetyStockLevel
from Production.Product
where SafetyStockLevel < 800 and SafetyStockLevel <=100
```

100 %

Results | Messages

	Name	SafetyStockLevel
1	Paint - Black	60
2	Paint - Red	60
3	Paint - Silver	60
4	Paint - Blue	60
5	Paint - Yellow	60
6	Sport-100 Helmet, Red	4
7	Sport-100 Helmet, Black	4
8	Mountain Bike Socks, M	4
9	Mountain Bike Socks, L	4
10	Sport-100 Helmet, Blue	4
11	AWC Logo Cap	4

Esta sentencia lo que hace es buscar entre los rangos de 800 a 100 hace lo mimos que betwee

```
select Name, ReorderPoint
from Production.Product
where ReorderPoint <> 500
```

100 %

Results | Messages

	Name	ReorderPoint
1	Adjustable Race	750
2	Bearing Ball	750
3	BB Ball Bearing	600
4	Headset Ball Bearings	600
5	Blade	600
6	LL Crankarm	375
7	ML Crankarm	375
8	HL Crankarm	375
9	Chainring Bolts	750
10	Chainring Nut	750
11	Chainring	750
12	Crown Race	750

Aquí se hace una búsqueda en las columnas name y reorderpoint diferentes a 500

```
select Name, ReorderPoint
from Production.Product
where ReorderPoint = 45
```

100 %

Results | Messages

	Name	ReorderPoint
1	Paint - Black	45
2	Paint - Red	45
3	Paint - Silver	45
4	Paint - Blue	45
5	Paint - Yellow	45

La búsqueda lo que hace es mostrar todos productos que en su columna reorderpoint sea igual a 45

```
select Name, ReorderPoint
from Production.Product
where ReorderPoint > 500
```

100 %

Results | Messages

	Name	ReorderPoint
1	Adjustable Race	750
2	Bearing Ball	750
3	BB Ball Bearing	600
4	Headset Ball Bearings	600
5	Blade	600
6	Chainring Bolts	750
7	Chainring Nut	750
8	Chainring	750

La búsqueda lo que hace es mostrar todos productos que en su columna reorderpoint sea mayor a 500

```
select Name, ReorderPoint
from Production.Product
where ReorderPoint < 600
```

100 %

Results | Messages

	Name	ReorderPoint
1	Adjustable Race	750
2	Bearing Ball	750
3	BB Ball Bearing	600
4	Headset Ball Bearings	600
5	Blade	600
6	Chainring Bolts	750
7	Chainring Nut	750
8	Chainring	750
9	Crown Race	750
10	Chain Stays	750
11	Decal 1	750

la búsqueda lo que hace es mostrar todos productos que en su columna reorderpoint sea menor a 600

Además podemos hacer combinaciones entre tablas usando el from y el where veamos como, utilizaremos las tablas person .person y person.address

```
select FirstName, MiddleName,LastName,AddressLine1,City
from Person.Person, Person.Address
where FirstName = 'ken' and City = 'Bothell'
```

100 %

Results | Messages

	FirstName	MiddleName	LastName	AddressLine1	City
1	Ken	NULL	Kwok	1226 Shoe St.	Bothell
2	Ken	NULL	Kwok	1318 Lasalle Street	Bothell
3	Ken	NULL	Kwok	1399 Firestone Drive	Bothell
4	Ken	NULL	Kwok	1873 Lion Circle	Bothell
5	Ken	NULL	Kwok	1902 Santa Cruz	Bothell
6	Ken	NULL	Kwok	1970 Napa Ct.	Bothell
7	Ken	NULL	Kwok	250 Race Court	Bothell
8	Ken	NULL	Kwok	25111 228th St Sw	Bothell
9	Ken	NULL	Kwok	3148 Rose Street	Bothell
10	Ken	NULL	Kwok	40 Ellis St.	Bothell
11	Ken	NULL	Kwok	4912 La Vuelta	Bothell
12	Ken	NULL	Kwok	5415 San Gabriel Dr.	Bothell
13	Ken	NULL	Kwok	5509 Newcastle Road	Bothell
14	Ken	NULL	Kwok	5672 Hale Dr.	Bothell
15	Ken	NULL	Kwok	5747 Shirley Drive	Bothell
16	Ken	NULL	Kwok	6387 Scenic Avenue	Bothell
17	Ken	NULL	Kwok	6696 Anchor Drive	Bothell
18	Ken	NULL	Kwok	6872 Thornwood Dr.	Bothell
19	Ken	NULL	Kwok	7057 Striped Maple Court	Bothell
20	Ken	NULL	Kwok	7484 Roundtree Drive	Bothell
21	Ken	NULL	Kwok	8157 W. Book	Bothell
22	Ken	NULL	Kwok	8713 Yosemite Ct.	Bothell
23	Ken	NULL	Kwok	9265 La Paz	Bothell
24	Ken	NULL	Kwok	9539 Glenside Dr	Bothell
25	Ken	NULL	Kwok	9833 Mt. Dias Blv.	Bothell
26	Ken	NULL	Kwok	99300 223rd Southeast	Bothell
27	Ken	NULL	Meyer	1226 Shoe St.	Bothell
28	Ken	NULL	Meyer	1318 Lasalle Street	Bothell
29	Ken	NULL	Meyer	1399 Firestone Drive	Bothell
30	Ken	NULL	Meyer	1873 Lion Circle	Bothell

Lo que se hace en esta consulta es realizar una búsqueda en la tabla person.person y person.address y en donde encuentre los valores ken y bothell los mostrar

```
select FirstName,LastName,AddressLine1,City
from Person.Person, Person.Address
where FirstName = 'ken' and LastName ='Sánchez' and City = 'Bothell'
```

100 %

Results | Messages

	FirstName	LastName	AddressLine1	City
1	Ken	Sánchez	1226 Shoe St.	Bothell
2	Ken	Sánchez	1318 Lasalle Street	Bothell
3	Ken	Sánchez	1399 Firestone Drive	Bothell
4	Ken	Sánchez	1873 Lion Circle	Bothell
5	Ken	Sánchez	1902 Santa Cruz	Bothell
6	Ken	Sánchez	1970 Napa Ct.	Bothell
7	Ken	Sánchez	250 Race Court	Bothell
8	Ken	Sánchez	25111 228th St Sw	Bothell
9	Ken	Sánchez	3148 Rose Street	Bothell

Querido lector creo que hasta aquí es suficiente por el día de hoy, nos veremos en una próxima entrega y le recomiendo que practique, practiquen y practiquen es la única manera de aprender.

NOTA: Para realizar consultas con SQLITE puede consultar el siguiente link

http://www.sgoliver.net/blog/bases-de-datos-en-android-iii-consultarrecuperar-registros/

4.4 OPERACIONES CON CONJUNTOS

SQL proporciona varias operaciones de conjuntos muy eficaces. Por ejemplo, incluye operadores de conjuntos del estilo de los de SQL como UNION, INTERSECT, EXCEPT y EXISTS. Entity SQL también es compatible con operadores para la eliminación de duplicados (SET), la prueba de pertenencia a un grupo (IN) y las combinaciones (JOIN). En los temas siguientes se describen los operadores de conjuntos de SQL.

- UNION, disponible en todas las versiones de SQL Server.

· EXCEPT, nuevo en SQL Server 2005.
· INTERSECT, nuevo en SQL Server 2005.

Para utilizar operaciones de conjuntos debemos cumplir una serie de normas.

·Las consultas a unir deben tener el mismo número campos, y además los campos deben ser del mismo tipo.
Sólo puede haber una única clausula ORDER BY al final de la sentencia SELECT.

https://www.campusmvp.es/recursos/post/Fundamentos-de-SQL-Operaciones-con-conjuntos.aspx

FUNCIONES DE AGREGACION

Las funciones de agregación en SQL nos permiten efectuar operaciones sobre un conjunto de resultados, pero devolviendo un único valor agregado para todos ellos. Es decir, nos permiten obtener medias, máximos, etc... Sobre un conjunto de valores.
Las funciones de agregación básicas que soportan todos los gestores de datos son las siguientes:

COUNT: devuelve el número total de filas seleccionadas por la consulta.

MIN: devuelve el valor mínimo del campo que especifiquemos.

MAX: devuelve el valor máximo del campo que especifiquemos.

SUM: suma los valores del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.

AVG: devuelve el valor promedio del campo que especifiquemos. Sólo se puede utilizar en columnas numéricas.

Las funciones anteriores son las básicas en SQL, pero cada sistema gestor de bases de datos relacionales ofrece su propio conjunto, más amplio, con otras funciones de agregación particulares. Puedes consultar las que ofrecen SQL Server, Oracle o MySQL.
Todas estas funciones se aplican a una sola columna, que especificaremos entre paréntesis, excepto la función COUNT, que se puede aplicar a una columna o indicar un "*". La diferencia entre poner el nombre de una columna o un "*", es que en el primer caso no cuenta los valores nulos para dicha columna, y en el segundo si.
Nota: Si esta serie de artículos te está pareciendo interesante, entonces ni te imaginas lo que puedes aprender con este curso de fundamentos de SQL.

Así, por ejemplo, si queremos obtener algunos datos agregados de la tabla de pedidos de la base de datos de ejemplo Northwind, podemos escribir una consulta simple como la siguiente:

SELECT COUNT(*) AS TotalFilas, COUNT(ShipRegion) AS FilasNoNulas,

MIN(ShippedDate) AS FechaMin, MAX(ShippedDate) AS FechaMax,

SUM(Freight) AS PesoTotal, AVG(Freight) PesoPromedio

FROM Orders

y obtendríamos el siguiente resultado en el entorno de pruebas:

Valores-Agregados-SQL-Ej1

De esta manera sabremos que existen en total 830 pedidos en la base de datos, 323 registros que tienen asignada una zona de entrega, la fecha del pedido más antiguo (el 10 de julio de 1996), la fecha del pedido más reciente (el 6 de mayo de 1998 ¡los datos de ejemplo son muy antiguos!), el total de peso enviado entre todos los pedidos (64.942,69 Kg o sea, más de 64 toneladas) y el peso promedio de los envíos (78,2442Kg). No está mal para una consulta tan simple. Como podemos observar del resultado de la consulta anterior, las funciones de agregación devuelven una sola fila, salvo que vayan unidas a la cláusula GROUP BY, que veremos a continuación.

https://campusvirtual.univalle.edu.co/moodle/pluginfile.php/670528/mod_resource/content/1/ClaBD05.pdf

4.5 VALORES NULOS

Si queremos cambiar un valor NULL por otro valor cualquiera, utilizaremos las siguientes funciones (ISNULL, IFNULL, NVL, COLACESCE) según el sistema de base de datos.

Para nuestros ejemplos, queremos que si el valor es NULL se cambie por el valor 0

Ejemplo para SQL SERVER se utiliza ISNULL:

SELECT producto,

preciounidad * (unidadesstock + ISNULL(unidadespedido, 0)

FROM productos

Ejemplo para ORACLE se utiliza NVL:

SELECT producto,
preciounidad * (unidadesstock + NVL(unidadespedido, 0)
FROM productos

Ejemplo para MySQL, hay 2 funciones equivalentes (IFNULL, COALESCE):

SELECT producto,
preciounidad * (unidadesstock + IFNULL(unidadespedido, 0)
FROM productos
SELECT producto,
preciounidad * (unidadesstock + COALESCE(unidadespedido, 0)
FROM productos
SQL DATOS TEXTO MYSQL >>

NOTA: Información de este tema para SQLITE, consultar el siguiente link. https://iessanvicente.com/colaboraciones/sqlite.pdf

4.6. CONSULTAS ANIDADAS

https://www.cs.us.es/blogs/bd2013/files/2013/09/Consultas-SQL.pdf

4.7. VISTAS

1. Tipos de vistas
2. Tareas de vista comunes
3. Consulte también

SE APLICA A: SQL Server Azure SQL Database Azure SQL Data Warehouse Almacenamiento de datos paralelos

Una vista es una tabla virtual cuyo contenido está definido por una consulta. Al igual que una tabla, una vista consta de un conjunto de columnas y filas de datos con un nombre.Sin embargo, a menos que esté indizada, una vista no existe como conjunto de valores de datos almacenados en una base de datos. Las filas y las columnas de datos proceden de tablas a las que se hace referencia en la consulta que define la vista y se producen de forma dinámica cuando se hace referencia a la vista.

Una vista actúa como filtro de las tablas subyacentes a las que se hace referencia en ella.La consulta que define la vista puede provenir de una o de varias tablas, o bien de otras vistas de la base de datos actual u otras bases de datos. Asimismo, es posible utilizar las consultas distribuidas para definir vistas que utilicen datos de orígenes heterogéneos.Esto puede resultar de utilidad, por ejemplo, si desea combinar datos de estructura similar que proceden de distintos servidores, cada uno de los cuales almacena los datos para una región distinta de la organización.

Las vistas suelen usarse para centrar, simplificar y personalizar la percepción de la base de datos para cada usuario. Las vistas pueden emplearse como mecanismos de seguridad, que permiten a los usuarios obtener acceso a los datos por medio de la vista, pero no les conceden el permiso de obtener acceso directo a las tablas base subyacentes de la vista. Las vistas pueden utilizarse para proporcionar una interfaz compatible con versiones anteriores con el fin de emular una tabla que existía pero cuyo esquema ha cambiado. También pueden usarse para copiar datos entre SQL Server a fin de mejorar el rendimiento y crear particiones de los datos.

Tipos de vistas

Además del rol estándar de las vistas básicas definidas por el usuario, SQL Serverproporciona los siguientes tipos de vistas que permiten llevar a cabo objetivos especiales en una base de datos.

Una vista indizada es una vista que se ha materializado. Esto significa que se ha calculado la definición de la vista y que los datos resultantes se han almacenado como una tabla.Se puede indizar una vista creando un índice clúster único en ella. Las vistas indizadas pueden mejorar de forma considerable el rendimiento de algunos tipos de consultas. Las vistas indizadas funcionan mejor para consultas que agregan muchas filas. No son adecuadas para conjuntos de datos subyacentes que se actualizan frecuentemente.

Una vista con particiones combina datos horizontales con particiones de un conjunto de tablas miembro en uno o más servidores. Esto hace que los datos aparezcan como si fueran de una tabla. Una vista que combina tablas miembros en la misma instancia de SQL Server es una vista con particiones local.

Las vistas de sistema exponen metadatos de catálogo. Puede usar las vistas del sistema para devolver información acerca de la instancia de SQL Server u objetos definidos en la instancia. Por ejemplo, puede consultar la vista de catálogo sys.databases para devolver información sobre las bases de datos definidas por el usuario disponibles en la instancia.Para obtener más información, vea Vistas del sistema (Transact-SQL).

En la tabla siguiente se proporcionan vínculos a las tareas comunes asociadas con la creación o modificación de una vista.

Tareas de vista	**Tema**
Describe cómo crear una vista.	Crear vistas
Describe cómo crear una vista indizada.	Crear vistas indizadas
Describe cómo modificar la definición de la vista.	Modificar vistas
Describe cómo modificar datos mediante una vista.	Modificar datos mediante una vista

Describe cómo eliminar una vista.	Eliminar vistas
Describe cómo devolver información acerca de una vista, como la definición de la vista.	Obtener información sobre una vista
Describe cómo cambiar el nombre de una vista.	Cambiar el nombre de las vistas

4.8 COMPONENTES DEL SQL

El lenguaje SQL está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.Existen dos tipos de comandos SQL:DDL(Data Definition Langiage) que permiten crear y definir nuevas bases de datos, campos e índices.DML(Data Manipulation Language) que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.

SQL Serveres un sistema de gestión de base de datos relacionales (SGDB) basada en el lenguaje SQL, capaz de poner a disposición de muchos usuarios grandes cantidades de datos de manera simultánea.Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, Sybase ASE o MySQL.Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en su versión 2005 pasa a ser el SQL Express Edition.

4.9 VENTAJAS DE SQL SERVER SOPORTE DE TRANSACCIONES.

Escalabilidad, estabilidad y seguridad. Soporta procedimientos almacenados. Incluye también un potente entorno gráfico de administración, que permite el uso de comandos DDL y DML gráficamente. Permite trabajar en modo cliente-servidor donde la información y datos se alojan en el servidor y las terminales o clientes de la red sólo acceden a la información. Además permite administrar información de otros servidores de datos.

4.10 COMPONENTES DE SQL SERVER

Administrador de serviciosEs la herramienta que se utiliza para ejecutar el servicio de SQL Server y tener disponibles las BD. Cuenta también con los siguientes servicios:1.- Coordinador de Transacciones Distribuidas.2.- SQL Server Agent.3.- SQL Server.

4.11 DDL(DATA DEFINITION LANGUAGE)

1.- Creación de base de datos.2.- Creación de tablas.3.- Integridad referencial:- Llave primaria (Primary Key).- Llave externa (Foreign Key).- Llave única (Unique Constraint).- Restricción de comprobación (Check Constraint).- Restricción de valor predefinido (Default Constraint).

BIBLIOGRAFÍA

ABRAHAM SILBERSCHATZ, H. F. (2010). *Fundamentos de bases de datos.* España: Mc Graw Hill.

Rafael Camps Paré, L. A. (2010). *SOFTWARE LIBRE.* ESPAÑA: UOC FORMACION DE POSGRADO.

http://codigoabiertopuntarenas.blogspot.com/2016/06/estructura-basica-de-consultas-sql.html

http://itimetux.blogspot.com/2013/12/tutorial-sqlite3-tipos-dml-dll.html

http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro14/53__lenguaje_de_manipulacin_de_datos_dml.html

http://www.w3bai.com/es/sql/sql_null_values.html

https://docs.microsoft.com/es-es/sql/relational-databases/views/views?view=sql-server-2017

http://itpn.mx/recursositics/5semestre/fundamentosderedes/Unidad%20V.pdf

http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro14/33_algebra_relacional.html

http://mysql.conclase.net/curso/?cap=004a

http://www.marcossarmiento.com/2017/06/28/normalizacion-de-base-de-datos/

https://blog.udemy.com/es/oracle-vs-mysql-vs-sql-server-una-comparacion-entre-los-sistemas-gestores-de-bases-de-datos-relacionales-mas-populares/

http://openaccess.uoc.edu/webapps/o2/bitstream/10609/9121/1/Intro_UML.pdf