

MODELAMIENTO Y DISEÑO DE BASE DE DATOS

Indice

✓ Presentación	3
✓ Semana 1: Fundamentos de Base de Datos.....	4
Semana 2: Modelamiento Conceptual.....	22
✓ Semana 3: Taller MER - Erwin	47
✓ Semana 4: Modelo Relacional - Erwin	63
✓ Semana 5: Taller de Formas Normales.....	86
✓ Semana 6: Modelo Lógico Global – Creación de tablas DDL.....	111
✓ Semana 7: Algebra Relacional – Integridad Referencial.....	131
✓ Semana 8: Diccionario de datos – Manipulación de datos DML.....	151
✓ Semana 9: Revisión de proyectos finales.....	171
➤ Bibliografía:	176

PRESENTACIÓN

Esta guía didáctica es un material de ayuda institucional, perteneciente a las especialidades de computación, **Ingeniería de Software e Ingeniería de Redes y Comunicaciones** tiene por finalidad proporcionar los conocimientos las técnicas de Modelamiento de Base de Datos a los estudiantes del segundo ciclo de estudios.

La **Organización SISE**, líder en la enseñanza tecnológica a nivel superior, promueve la elaboración de materiales educativos, en concordancia a las exigencias de las tecnologías de estos tiempos, que permiten la creación de nuevas herramientas de aprendizaje con el objetivo de facilitar el acceso de los estudiantes a la educación en el marco del desarrollo tecnológico de la informática u de las telecomunicaciones.

Esta guía se divide en 9 temas principales, las cuales se irán desarrollando por medio de contenidos especialmente preparados para un mejor aprendizaje del educando. Permite conocer las herramientas indispensables para la elaboración de diagramas de diseño de datos con el uso del Modelo Entidad relación. Se inicia con la descripción de conceptos básicos en las cuales se tiene por objetivo que el alumno se introduzca en lo concerniente a Base de datos.

En el proceso de desarrollo de sistemas informáticos, orientados a producir software que apoye a las actividades empresariales, así como a sus procesos, se tienen que respetar ciertas fases propias de las metodologías del análisis de información, de la metodología de procesos de negocios (IDEF) hoy en día se emplea la metodología orientada a objetos, sin embargo, para el desarrollo de software es primordial el manejo del análisis y diseño de sistemas, para el análisis tenemos herramientas de recopilación de información, mientras que para la fase de diseño de sistemas, tenemos las llamadas herramientas 'CASE', que son el apoyo informático de todo diseñador de sistemas para plasmar todo el análisis de requerimientos previos en diagramas, conocidos como 'MODELOS', la herramientas de diseño de sistemas más empleado es el Platinum Erwin, que es la que emplearemos para el curso.

Este material en su primera edición, servirá para ayudar a nuestros estudiantes SISESINOS a tener una formación solida para resolver casos y problemáticas presentados en una organización empresarial.

SEMANA 1

Contenido:

- Introducción a un DBMS.
- Definiciones básicas, características, importancia, alcances.
 - o Nivel Físico y Lógico de los datos.
- Administración de Base de Datos, Usuarios.
 - o Representación de la información
 - o Niveles.
 - o Relaciones de correspondencia
 - o Conociendo los SABDR.
 - o Ejercicios

FUNDAMENTOS DE BASE DE DATOS

En el proceso de desarrollo de sistemas informáticos, orientados a producir software que apoye a las actividades empresariales, así como a sus procesos, se tienen que respetar ciertas fases propias de las metodologías del análisis de información, de la metodología de procesos de negocios (IDEF) hoy en día se emplea la metodología orientada a objetos, sin embargo, para el desarrollo de software es primordial el manejo del análisis y diseño de sistemas, para el análisis tenemos herramientas de recopilación de información, mientras que para la fase de diseño de sistemas, tenemos las llamadas herramientas 'CASE', que son el apoyo informático de todo diseñador de sistemas para plasmar todo el análisis de requerimientos previos en diagramas, conocidos como 'MODELOS', la herramientas de diseño de sistemas más empleado es el Platinum Erwin, que es la que emplearemos para el curso.

➤ INTRODUCCION A DBMS

✓ DEFINICIONES BASICAS

Una Base de Datos es un contenedor de objetos como tablas, vistas, procedimientos, etc., almacenados en una plataforma denominada Sistema Gestor de Base de Datos, viene a ser un servidor como SQL Server, Oracle, DB2, MySQL, etc. Dicho de otra forma, una base de datos contiene las tablas y componentes sobre las que se almacena información de todos los procesos del negocio. Y nos permite generar información y ser compartida por distintos usuarios.

El DBMS es el software que almacenará nuestra base de datos, por la cual es la que nos va a permitir acceder a la información por medio de lenguajes como SQL. Tienen la capacidad de responder a múltiples usuarios en forma concurrente a los datos, lo que se llama 'Arquitectura Cliente – Servidor'.

DBMS: Database Management System (Sistema Administrador de Base de datos)

✓ **CARACTERÍSTICAS**

- Escalabilidad: Se refiere de la organización, mejorando su performance.
 - Horizontal: Crecimiento de los usuarios.
 - Vertical: Se refiere al crecimiento del servidor de datos.
- Rendimiento: Característica de brindar respuestas a los múltiples requerimientos de los usuarios como consultas, actualización, recuperación de datos, etc.
- Portabilidad: Característica de transportar con facilidad el producto de una plataforma a otra incluyendo toda la data contenida.
- Universalidad: Característica de manejar múltiples tipos de datos como caracteres, numéricos, de fecha, multimedia, etc.
- Disponibilidad: Debe ser permanente e ininterrumpida, factor crucial del servicio de la base de datos que da apoyo a las aplicaciones (programas) de los negocios.

✓ **IMPORTANCIA DE LOS DBMS**

Se tiene que elegir el que tenga la mejor interfaz a las necesidades de la empresa y sobre todo que este apta para la base de datos que vayamos a utilizar.

El objetivo es crear un ambiente en el que sea posible almacenar, manipular y recuperar la información en forma oportuna y eficiente.

El administrador de base de datos o servidor de base de datos conocido como sistema de administración de base de datos (DBMS) maneja todas las solicitudes de acceso a la base de datos ya sea para agregar y eliminar archivos, recuperar y almacenar datos desde y en dichos archivos. Por lo tanto, una función general que ofrece el DBMS consiste en ocultar a los usuarios de la base de datos los detalles al nivel de hardware. Es decir, que el DBMS ofrece a los usuarios una percepción de la base de datos que está en cierto modo, por encima del nivel del hardware y que maneja las operaciones del usuario expresadas en términos de ese nivel más alto de percepción.

El DBMS es el componente de software más importante del sistema en general, aunque no es el único.

✓ **ALCANCES**

El alcance de una Base de Datos abarca varios aspectos como:

- Los usuarios que podrán tener acceso a los datos almacenados, por medio de permisos otorgados por el Administrador de BD (DBA).
- Desde dónde (terminal de usuario) y cómo accederán a la BD, para ello se impondrán restricciones a los accesos de usuarios.
- Hasta donde y sobre todo a qué tipo de información tendrán acceso los usuarios registrados.

➤ **ADMINISTRACION DE BASE DE DATOS**

El administrador de datos (DA) es la persona identificable que tendrá la responsabilidad central sobre los datos dentro de la empresa. Ya que los datos son uno de los activos más valiosos de la empresa, es imperativo que exista una persona que los entienda junto con las necesidades de la empresa con respecto a esos datos, a un nivel de administración superior. Por lo tanto, es labor del administrador decidir en primer lugar qué datos deben ser almacenados en la base de datos y establecer políticas para mantener y manejar esos datos una vez almacenados. El administrador de base de datos (DBA) es el técnico responsable de implementar las decisiones del administrador de datos. Por lo tanto, debe ser un profesional en IT. El trabajo del DBA consiste en crear la base de datos real e implementar los controles técnicos necesarios para hacer cumplir las diversas decisiones de las políticas hechas por el DA. El DBA también es responsable de asegurar que el sistema opere con el rendimiento adecuado y de proporcionar una variedad de otros servicios técnicos.

✓ **USUARIOS**

Existen tres grandes clases de usuarios:

- Programadores de aplicaciones, que son los responsables de escribir los programas de aplicación de base de datos en algún lenguaje de programación. Estos programas acceden a la base de datos emitiendo la solicitud apropiada al DBMS. Los programas en sí pueden ser aplicaciones convencionales por lotes o pueden ser aplicaciones en línea, cuyo propósito es permitir al usuario final el acceso a la base de datos desde una estación de trabajo o terminal en línea.
- Los usuarios finales, quienes interactúan con el sistema desde estaciones de trabajo o terminales en línea. Un usuario final puede acceder a la base de datos a través de las aplicaciones en línea, o bien puede usar una interfaz proporcionada como parte integral del software del sistema de base de datos. Las interfaces proporcionadas por el fabricante están apoyadas también por aplicaciones en línea, aunque esas aplicaciones están integradas, es decir, no son escritas por el usuario. La mayoría de

los sistemas de base de datos incluyen por lo menos una de estas aplicaciones integradas.

- La mayoría de los sistemas proporcionan además interfaces integradas adicionales en las que los usuarios no emiten en absoluto solicitudes explícitas a la base de datos, sino que en vez de ello operan mediante la selección de elementos en un menú o llenando casillas de un formulario. Estas interfaces controladas por menús o por formularios tienden a facilitar el uso a personas que no cuentan con una capacitación formal en tecnología de la información (IT). En contraste, las interfaces controladas por comandos tienden a requerir cierta experiencia profesional en IT, aunque tal vez no demasiada. Por otra parte, es probable que una interfaz controlada por comandos sea más flexible que una controlada por menús o por formularios, dado que los lenguajes de consulta por lo regular incluyen ciertas características que no manejan esas otras interfaces.
- El administrador de base de datos o DBA.

Algunos usuarios son:

- Jefes de proyecto.
- Analistas de sistemas.
- Analistas programadores.
- Programadores.
- Diseñadores de sistemas.

✓ **ADMINISTRADOR DE BASE DE DATOS**

Es el profesional informático responsable de diseñar la estructura de la base de datos, así como del mantenimiento y seguridad tanto de la información como del servidor de datos. Su denominación es DBA (Database Administrator), entre sus funciones principales tenemos:

- Definición de la estructura de tablas, y componentes.
- Asignación y administración de permisos de acceso a los usuarios.
- Responsable de la seguridad de toda la información, por medio de copias de seguridad de datos (backups).
- Administrar la estructura de la Base de Datos
- Administrar la actividad de los datos
- Administrar el Sistema Manejador de Base de Datos
- Establecer el Diccionario de Datos
- Asegurar la confiabilidad de la Base de Datos
- Confirmar la seguridad de la Base de Datos
- Asegurar una óptima performance de la organización de los datos.

Detallemos algunos de ellos:

Administración de la estructura de la Base de Datos

La administración de la estructura de la Base de Datos incluye participar en el diseño inicial de la misma y su puesta en práctica así como controlar, y administrar sus requerimientos,

ayudando a evaluar alternativas, incluyendo los DBMS a utilizar y ayudando en el diseño general de BD. En los casos de grandes aplicaciones de tipo organizacional, el DBA es un gerente que supervisa el trabajo del personal de diseño de la BD.

Una vez diseñada la BD, es puesta en práctica utilizando productos del DBMS, procediéndose entonces a la creación de los datos (captura inicial). El DBA participa en el desarrollo de procedimientos y controles para asegurar la calidad y la alta integridad de la BD

Administración de la actividad de datos

Aunque el DBA protege los datos, no los procesa. El DBA no es usuario del sistema, en consecuencia, no administra valores de datos; el DBA administra actividad de datos. Dado que la BD es un recurso compartido, el DBA debe proporcionar estándares, guías de acción, procedimientos de control y la documentación necesaria para garantizar que los usuarios trabajan en forma cooperativa y complementaria al procesar datos en la BD.

Entre las alternativas más utilizadas por el DBA para tratar de resolver o minimizar este problema se encuentran las siguientes:

- a) Restringir el acceso a los procedimientos para ciertos usuarios.
- b) Restringir al acceso a los datos para ciertos usuarios procedimientos y/o datos.
- c) Evitar la coincidencia de horarios para usuarios que comparten.

Administración del DBMS

A demás de administrar la actividad de datos y la estructura de la BD, el DBA debe administrar el DBMS mismo. Deberá compilar y analizar estadísticas relativas al rendimiento del sistema e identificar áreas potenciales del problema. Dado que la BD está sirviendo a muchos grupos de usuarios, el DBA requiere investigar todas las quejas sobre el tiempo de respuesta del sistema, la precisión de los datos y la facilidad de uso. Si se requieren cambios el DBA deberá planearlos y ponerlos en práctica.

El DBA deberá vigilar periódica y continuamente las actividades de los usuarios en la BD. Los productos DBMS incluyen tecnologías que reúnen y publican estadísticas. Estos informes pudieran indicar cuáles fueron los usuarios activos, que archivos y que elementos de datos han sido utilizados, e incluso el método de acceso que se ha aplicado. Pueden capturarse y reportarse las tasas de error y los tipos de errores. El DBA analizará estos datos para determinar si se necesita una modificación en el diseño de la BD para manejar su rendimiento o para facilitar las tareas de los usuarios; de ser así, el DBA la llevará a cabo.

Establecer el Diccionario de Datos.

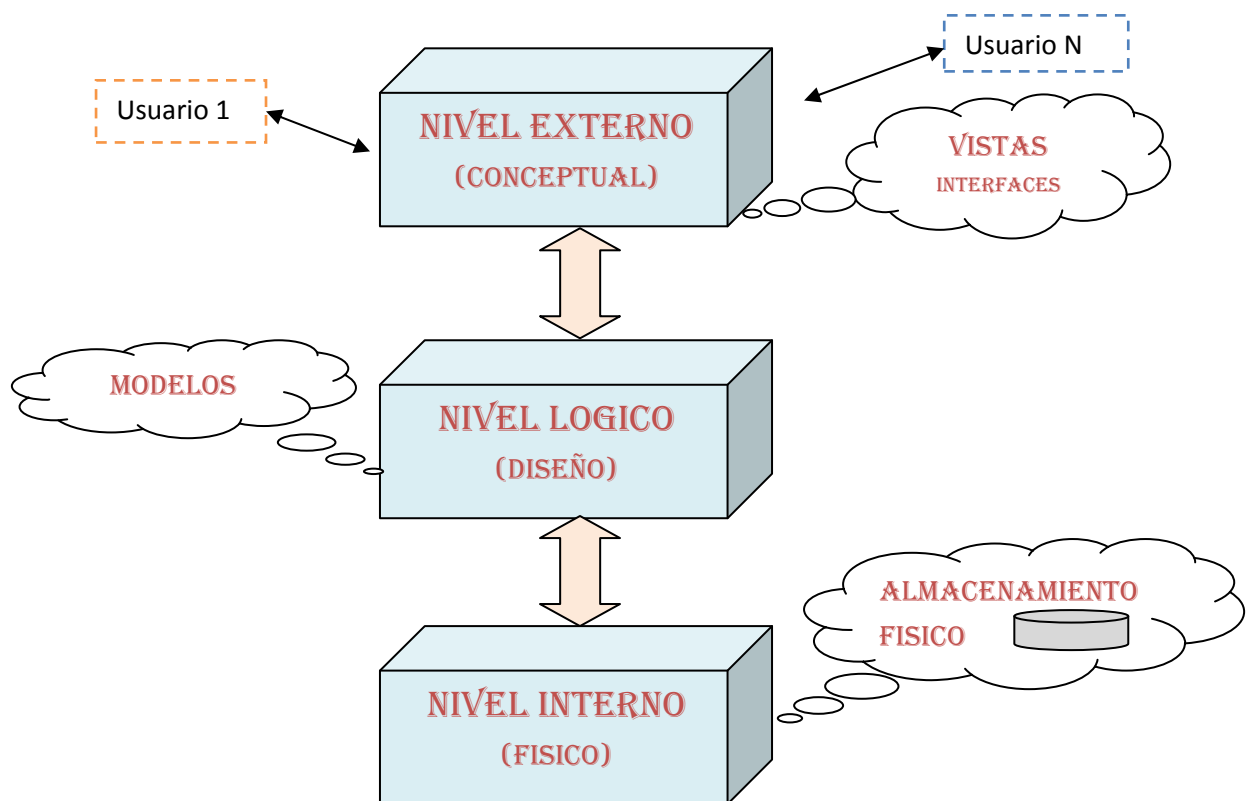
Cuando se definen estándares sobre la estructura de la base de datos, se deben de registrarse en una sección del diccionario de datos a la que todos aquellos usuarios relacionados con ese tipo de proceso pueden acceder. Este metadato debe precisar información que nos indique con claridad el tipo de datos que serán utilizados, sus ámbitos de influencia y sus limitantes de seguridad.

Mantener la Disponibilidad de los Datos.

La posibilidad de fallos de hardware o de software requiere procedimientos de recuperación de la base de datos. Tiene que proporcionar medios para el restablecimiento de las bases de datos que se hayan corrompido por desperfectos del sistema, a un estado uniforme.

➤ REPRESENTACION DE LA INFORMACION**✓ NIVELES DE LA INFORMACION**

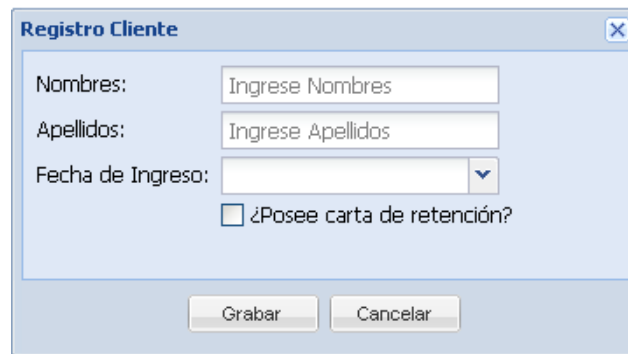
Está diseñado en base a la Arquitectura ANSI (American National Standards Institute), comprende de tres niveles de datos, cuyo objetivo es la de separar los programas de aplicación de la base de datos física, éstos niveles en realidad son descripciones de los mismos datos pero con distintos niveles de abstracción (acceso), los únicos datos que realmente existen están en el nivel físico, pero es importante especificar qué tipos de usuarios y en qué nivel de acceso se encuentran disponibles para el manejo de la información.

ESQUEMA DE LA ARQUITECTURA ANSI**✓ NIVEL EXTERNO (Conceptual)**

Describe una parte de la base de datos que interesa a un grupo de usuarios y ocultándola a otro de grupo de usuarios, aquí se encuentran las vistas (interfaces), que será el único medio

de acceso de estos usuarios hacia la información almacenada en el servidor de datos (el DBMS).

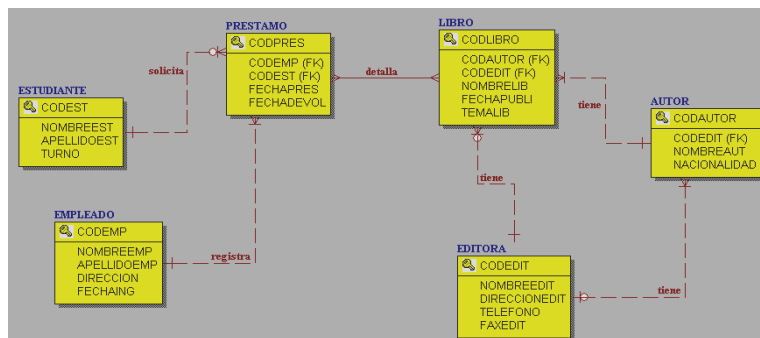
Ejemplo: A través de un formulario de acceso, el usuario podrá acceder al sistema.



A screenshot of a software window titled "Registro Cliente". It contains three input fields: "Nombres:" with the placeholder text "Ingrese Nombres", "Apellidos:" with the placeholder text "Ingrese Apellidos", and "Fecha de Ingreso:" with a dropdown arrow. Below these fields is a checkbox labeled "¿Posee carta de retención?". At the bottom of the window are two buttons: "Grabar" and "Cancelar".

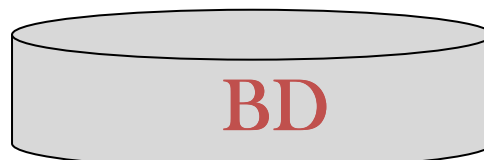
✓ NIVEL LOGICO (Diseño)

Es aquel nivel que describe la estructura de la base de datos, realizada en la fase de diseño del sistema, satisface los requerimientos de los usuarios, se representa mediante un modelo de datos, ocultando los detalles de almacenamiento físico.



✓ NIVEL INTERNO (Físico)

Este nivel describe la estructura física de almacenamiento de la base de datos, aquí se encuentra realmente los únicos datos existentes, es decir, la BD.



Explicaremos a mayor profundidad...

Arquitectura ANSI

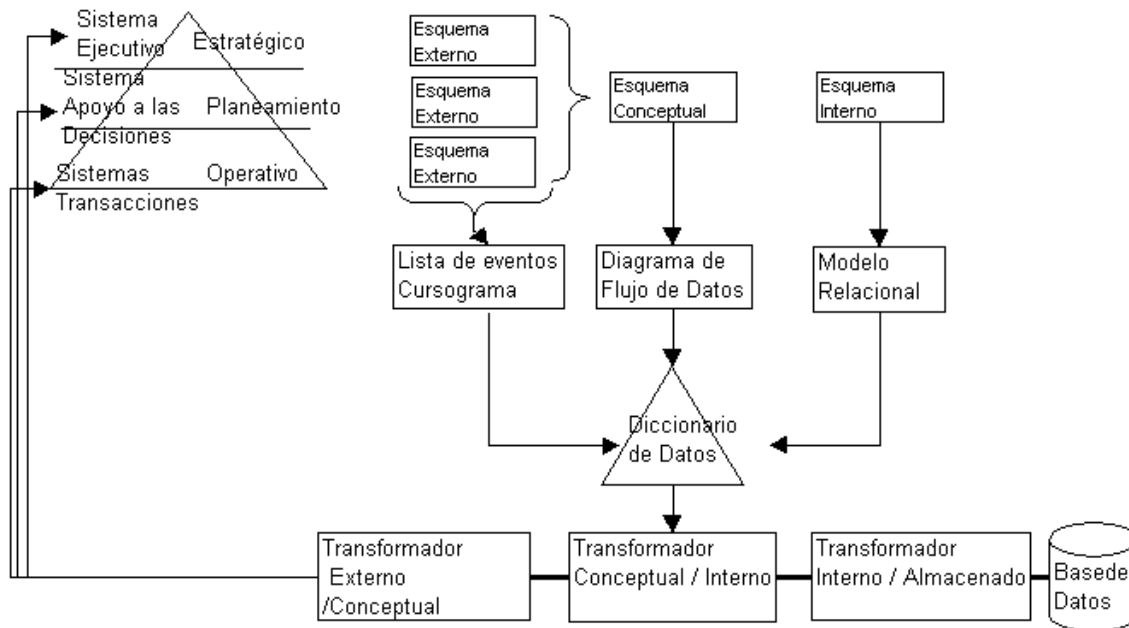
La arquitectura de sistemas de bases de datos de tres esquemas fue aprobado por la ANSI-SPARC (American National Standard Institute - Standards Planning and Requirements Committee) en 1975 como ayuda para conseguir la separación entre los programas de aplicación y los datos, el manejo de múltiples vistas por parte de los usuarios y el uso de un catálogo para almacenar el esquema de la base de datos.

- Nivel interno: Tiene un esquema interno que describe la estructura física de almacenamiento de base de datos. Emplea un modelo físico de datos y los únicos datos que existen están realmente en este nivel.
- Nivel conceptual: tiene esquema conceptual. Describe la estructura de toda la base de datos para una comunidad de usuarios. Oculta los detalles físicos de almacenamiento y trabaja con elementos lógicos como entidades, atributos y relaciones.
- Nivel externo o de vistas: tiene varios esquemas externos o vistas de usuario. Cada esquema describe la visión que tiene de la base de datos a un grupo de usuarios, ocultando el resto.

El objetivo de la arquitectura de tres niveles es el de separar los programas de aplicación de la base de datos física. La mayoría de los SGBD no distinguen del todo los tres niveles. Algunos incluyen detalles del nivel físico en el esquema conceptual. En casi todos los SGBD que se manejan vistas de usuario, los esquemas externos se especifican con el mismo modelo de datos que describe la información a nivel conceptual, aunque en algunos se pueden utilizar diferentes modelos de datos en los niveles conceptuales y externo.

Hay que destacar que los tres esquemas no son más que descripciones de los mismos datos pero con distintos niveles de abstracción. Los únicos datos que existen realmente están a nivel físico, almacenados en un dispositivo como puede ser un disco. En un SGBD basado en la arquitectura de tres niveles, cada grupo de usuarios hace referencia exclusivamente a su propio esquema externo. Por lo tanto, el SGBD debe transformar cualquier petición expresada en términos de un esquema externo a una petición expresada en términos del esquema conceptual, y luego, a una petición en el esquema interno, que se procesará sobre la base de datos almacenada. Si la petición es de una obtención (consulta) de datos, será preciso modificar el formato de la información extraída de la base de datos almacenada, para que coincida con la vista externa del usuario. El proceso de transformar peticiones y resultados de un nivel a otro se denomina correspondencia o transformación. Estas correspondencias pueden requerir bastante tiempo, por lo que algunos SGBD no cuentan con vistas externas.

La arquitectura de tres niveles es útil para explicar el concepto de independencia de datos que podemos definir como la capacidad para modificar el esquema en un nivel del sistema sin tener que modificar el esquema del nivel inmediato superior.



Otra vista de los niveles de datos según la Arquitectura Ansi.

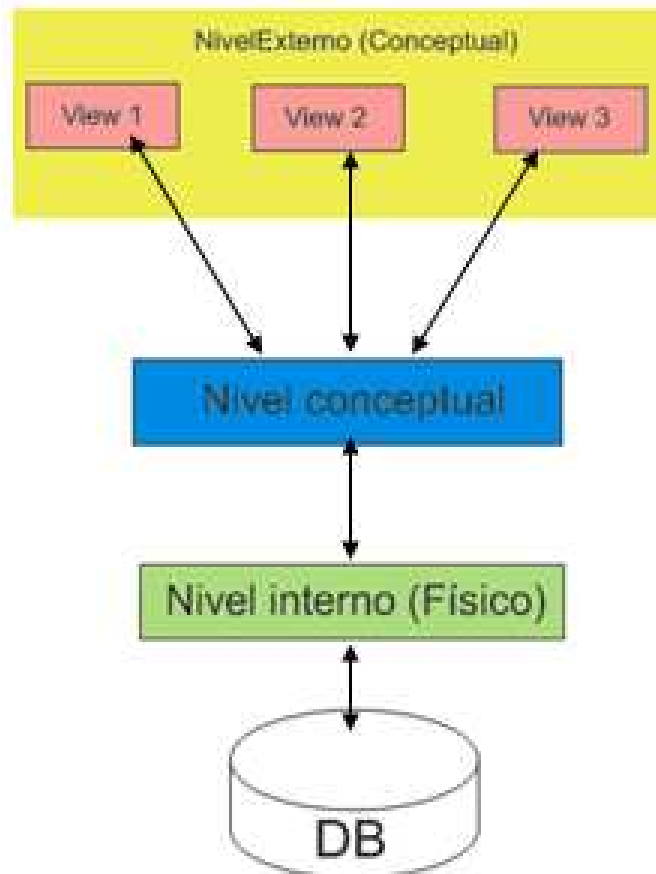
➤ NIVELES DE CORRESPONDENCIA

Hay 2 niveles de correspondencias, uno entre los niveles externo y conceptual del sistema, y otro entre los niveles conceptual e interno.

La correspondencia conceptual / interna es la que existe entre la vista conceptual y la BD almacenada; especifica cómo se representan los registros y campos conceptuales en el nivel interno. Si se modifica la estructura de la BD almacenada deberá modificarse para que no varíe (DBA). Los efectos de las alteraciones deberán aislarse por debajo del nivel conceptual, a fin de conservar la independencia de los datos.

La correspondencia externo / conceptual es la que existe entre una determinada vista externa y la vista conceptual. Las diferencias que pueden existir entre éstos 2 niveles son similares a las que pueden existir entre la vista conceptual y la BD almacenada. Puede existir cualquier cantidad de vistas externas; cualquier número de usuarios puede compartir una determinada vista externa; puede haber traslapes entre vistas externas distintas.

Algunos sistemas permiten expresar la definición de una vista externa en términos de otras a través de una correspondencia externa / externa en vez de requerir siempre una definición explícita de la correspondencia respecto al nivel conceptual, cosa que resulta útil si existe una relación íntima entre varias vistas externas. Los sistemas relacionales en particular casi siempre permiten hacer esto.



LABORATORIO # 1

➤ CONOCIENDO LOS SGBDR

✓ **SISTEMA GESTOR DE BASE DE DATOS RELACIONAL (SGDBR).**

Software que gestiona el uso de las bases de datos relacionales, y optimiza y controla el acceso al contenido de las mismas.

- El almacenamiento físico de los datos se gestiona únicamente a través del gestor. El usuario sólo debe preocuparse de la estructura lógica de los mismos.
- La manipulación de la estructura y contenido de una base de datos relacional se realiza mediante el lenguaje SQL (Structured Query Language).
- SGBDRs existentes son: SQL Server, PostgreSQL, MySQL, Oracle, Sybase, DB2, Access, Informix, etc...

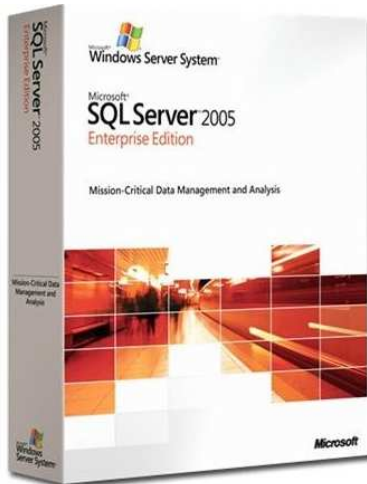
➤ **USUARIOS DE UNA BASE DE DATOS**

- Los usuarios de una base de datos no están relacionados con los usuarios del sistema.
- Al igual que en un sistema informático, existe la figura del administrador. En casi todos los SGBDRs el administrador
- de una base de datos no tiene por qué ser el administrador del sistema.

- Un administrador crea los usuarios, y les otorga o deniega privilegios (operaciones que pueden realizar).
- Un privilegio es: crear, modificar o borrar una tabla; consultar, insertar, borrar o modificar los datos de una tabla; consultar o crear una vista; crear usuarios o grupos; otorgar privilegios; etc...

VEAMOS ALGUNOS DE ELLOS BREVEMENTE...

MICROSOFT SQL SERVER



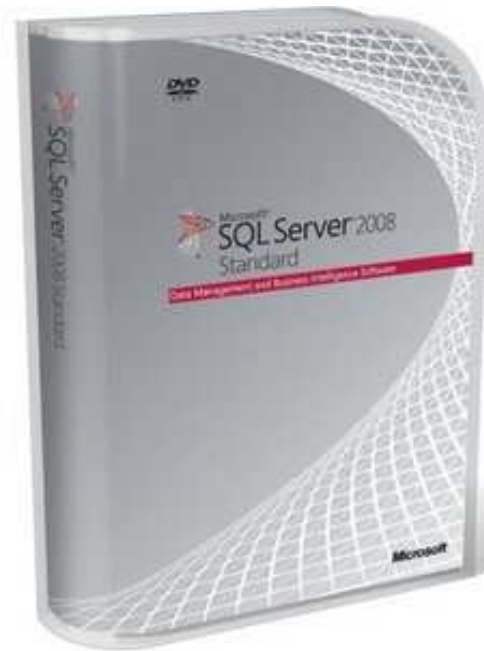
Microsoft SQL Server constituye la alternativa de Microsoft a otros potentes sistemas gestores de bases de datos como son Oracle, Sybase ASE, PostgreSQL, Interbase, Firebird o MySQL.

CARACTERÍSTICAS DE MICROSOFT SQL SERVER

- 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.
- Este sistema incluye una versión reducida, llamada MSDE con el mismo motor de base de datos pero orientado a proyectos más pequeños, que en sus versiones 2005 y 2008 pasa a ser el SQL Express Edition, que se distribuye en forma gratuita.
- Es común desarrollar completos proyectos complementando Microsoft SQL Server y Microsoft Access a través de los llamados ADP (Access Data Project). De esta forma se completa la base de datos (Microsoft SQL Server), con el entorno de desarrollo (VBA Access), a través de la implementación de aplicaciones de dos capas mediante el uso de formularios Windows.
- En el manejo de SQL mediante líneas de comando se utiliza el SQLCMD
- Para el desarrollo de aplicaciones más complejas (tres o más capas), Microsoft SQL Server incluye interfaces de acceso para varias plataformas de desarrollo, entre ellas .NET, pero el servidor sólo está disponible para Sistemas Operativos Windows.

LO NUEVO DE SQL SERVER 2008

La nueva base de datos contiene mayor seguridad, integración con PowerShell, remueve La consola configuración del área expuesta (consola para configurar seguridad), tiene correctores de sintaxis del lenguaje Transact-SQL e intellisense (una característica del visual studio que permite a la base de datos sugerir objetos existentes mientras uno escribe la mitad de la palabra). Así mismo incluye nuevos tipos de datos y funciones....



ORACLE



Oracle es un sistema de gestión de base de datos relacional (o RDBMS por el acrónimo en inglés de Relational Data Base Management System), desarrollado por Oracle Corporation.

Se considera a Oracle como uno de los sistemas de bases de datos más completos,[cita requerida] destacando:

- soporte de transacciones.
- estabilidad.
- escalabilidad.
- Soporte multiplataforma.

Ha sido criticada por algunos especialistas la seguridad de la plataforma, y las políticas de suministro de parches de seguridad, modificadas a comienzos de 2005 y que incrementan el nivel de exposición de los usuarios. En los parches de actualización provistos durante el primer semestre de 2005 fueron corregidas 22 vulnerabilidades públicamente conocidas, algunas de ellas con una antigüedad de más de 2 años.

Aunque su dominio en el mercado de servidores empresariales ha sido casi total hasta hace poco, recientemente sufre la competencia del Microsoft SQL Server de Microsoft y de la oferta de otros RDBMS con licencia libre como PostgreSQL, MySQL o Firebird. Las últimas versiones de Oracle han sido certificadas para poder trabajar bajo GNU/Linux.

MySQL

MySQL es un sistema de gestión de base de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones.[1] MySQL AB —desde enero de 2008 una subsidiaria de Sun Microsystems y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libre en un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y el copyright del código está en poder del autor individual, MySQL es propietario y está patrocinado por una empresa privada, que posee el copyright de la mayor parte del código.

Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQL AB fue fundado por David Axmark, Allan Larsson y Michael Widenius.

MySQL es muy utilizado en aplicaciones web, como Drupal o phpBB, en plataformas (Linux/Windows-Apache-MySQL-PHP/Perl/Python), y por herramientas de seguimiento de errores como Bugzilla. Su popularidad como aplicación web está muy ligada a PHP, que a menudo aparece en combinación con MySQL. MySQL es una base de datos muy rápida en la lectura cuando utiliza el motor no transaccional MyISAM, pero puede provocar problemas de integridad en entornos de alta concurrencia en la modificación. En aplicaciones web hay baja concurrencia en la modificación de datos y en cambio el entorno es



intensivo en lectura de datos, lo que hace a MySQL ideal para este tipo de aplicaciones. Sea cual sea el entorno en el que va a utilizar MySQL, es importante adelantar monitoreos sobre el desempeño para detectar y corregir errores tanto de SQL como de programación.

DB2



DB2 es una marca comercial, propiedad de IBM, bajo la cual se comercializa un sistema de gestión de base de datos.

DB2 versión 9 es un motor de base de datos relacional que integra XML de manera nativa, lo que IBM ha llamado pureXML, que permite almacenar documentos completos dentro del tipo de datos xml para realizar operaciones y búsquedas de manera jerárquica dentro de éste, e integrarlo con búsquedas relacionales.

DB2 Express-C es la versión gratuita soportada por la comunidad de DB2 que permite desarrollar, implementar y distribuir aplicaciones que no usen las características avanzadas de las versiones comerciales de DB2. Esta versión de DB2 puede ser concebida como el núcleo de DB2, las diferentes ediciones incluyen las características de Express-C más funcionalidades específicas.

CARACTERISTICAS

- Permite el manejo de objetos grandes (hasta 2 GB), la definición de datos y funciones por parte del usuario, el chequeo de integridad referencial, SQL recursivo, soporte multimedia: texto, imágenes, video, audio; queries paralelos, commit de dos fases, backup/recuperación on-line y offline.
- Además cuenta con un monitor gráfico de performance el cual posibilita observar el tiempo de ejecución de una sentencia SQL y corregir detalles para aumentar el rendimiento.
- Mediante los extensores se realiza el manejo de los datos no tradicionales, por ejemplo si tengo un donde tengo almacenados los curriculums de varias personas, mediante este puedo realizar búsquedas documentos con los datos que me interesen sin tener que ver los CV uno por uno.
- Esta capacidad se utiliza en sistemas de búsqueda de personas por huellas digitales, en sistemas información geográfica, etc.
- Internet es siempre la gran estrella, con DB2 es posible acceder a los datos usando JDBC (tan potente como escribir directamente C contra la base de datos), Java y SQL (tanto el SQL estático, como complementa el SQL dinámico).

PLATAFORMAS HOST:

- OS/390(MVS), VM & VSE, OS/400

PLATAFORMAS DE SERVIDOR:

- OS/2 Warp Server, Sinix, SCO Openserver, Windows NT, Aix, HP Ux, Solaris.

PLATAFORMAS CLIENTE:

- OS/2, DOS, Sinix, SCO OpenServer, Windows 3.1/95/NT, Macintosh System 7, Aix, HP Ux, Solaris.

VENTAJAS

- Permite agilizar el tiempo de respuestas de esta consulta
- Recuperación utilizando accesos de sólo índices.
- Predicados correlacionados.
- Tablas de resumen
- Tablas replicadas
- Uniones hash
- DB2 utiliza una combinación de seguridad externa y control interno de acceso a proteger datos.
- DB2 proporciona un juego de datos de acceso de las interfaces para los diferentes tipos de usuarios y aplicaciones.
- DB2 guarda sus datos contra la pérdida, acceso desautorizado, o entradas inválidas.
- Usted puede realizar la administración de la DB2 desde cualquier puesto de trabajo.
- La tecnología de replicación heterogénea (heterogeneous replication) en SQL Server permite la publicación automática de los datos en otros sistemas que no sean SQL Server, entre los que se incluyen DB2.
- La mayoría de los que utilizan equipos IBM utilizan DB2 porque es confiable y tiene un muy buen soporte técnico".
- El DB2 se basa en dos ejes que lo hacen fuerte en su rendimiento: utiliza un sistema multiprocesador (SMP) simétrico y un sistema de procesador paralelo masivo.
- el DB2 distribuye y recuerda la ubicación de cada pista donde se encuentra la información. En el contexto de una larga base de datos, este sistema de partición hace que la administración sea mucho más fácil de manejar que una base de datos de la misma medida no particionada.
- La base de datos se puede programar para tener una exacta cantidad de particiones que contienen la información del usuario, índice, clave de transacción y archivos de configuración. De esta forma, los administradores definen grupos de nodos, que son una serie de particiones de la base, lo que posteriormente facilita cualquier búsqueda.

DESVENTAJAS

- El DB2 - IBM es la tercera base de datos que más se vende, de acuerdo con los VARs recientemente encuestados en el número de junio de 1996 de la revista VAR Business Magazine. El Microsoft SQL Server se anotó un 38%, Oracle, 21%, IBM, 10%, Informix, 9%, y Sybase un 8%.
- En sistemas grandes la base más usada es DB2 ya que corre en diferentes plataformas operativas, pero en realidad, en la mayoría de los casos la decisión para optar por un software de estas características es corporativa.
- Se tiene que ver con las aplicaciones que se tienen desarrolladas y las que se van a implementar.
- Influye en la elección el hardware utilizado.
- Una serie de error del sistema operativo, que cae DB2.

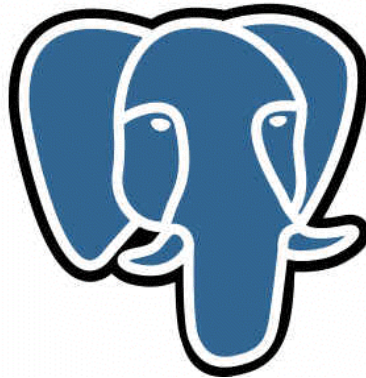
PostgreSQL

PostgreSQL es un sistema de gestión de base de datos relacional orientada a objetos y libre, publicado bajo la licencia BSD.

Como muchos otros proyectos de código abierto, el desarrollo de PostgreSQL no es manejado por una sola empresa sino que es dirigido por una comunidad de desarrolladores y

organizaciones comerciales las cuales trabajan en su desarrollo. Dicha comunidad es denominada el PGDG (PostgreSQL Global Development Group).

PostgreSQL



✓ HERRAMIENTAS DE DISEÑO DE BD

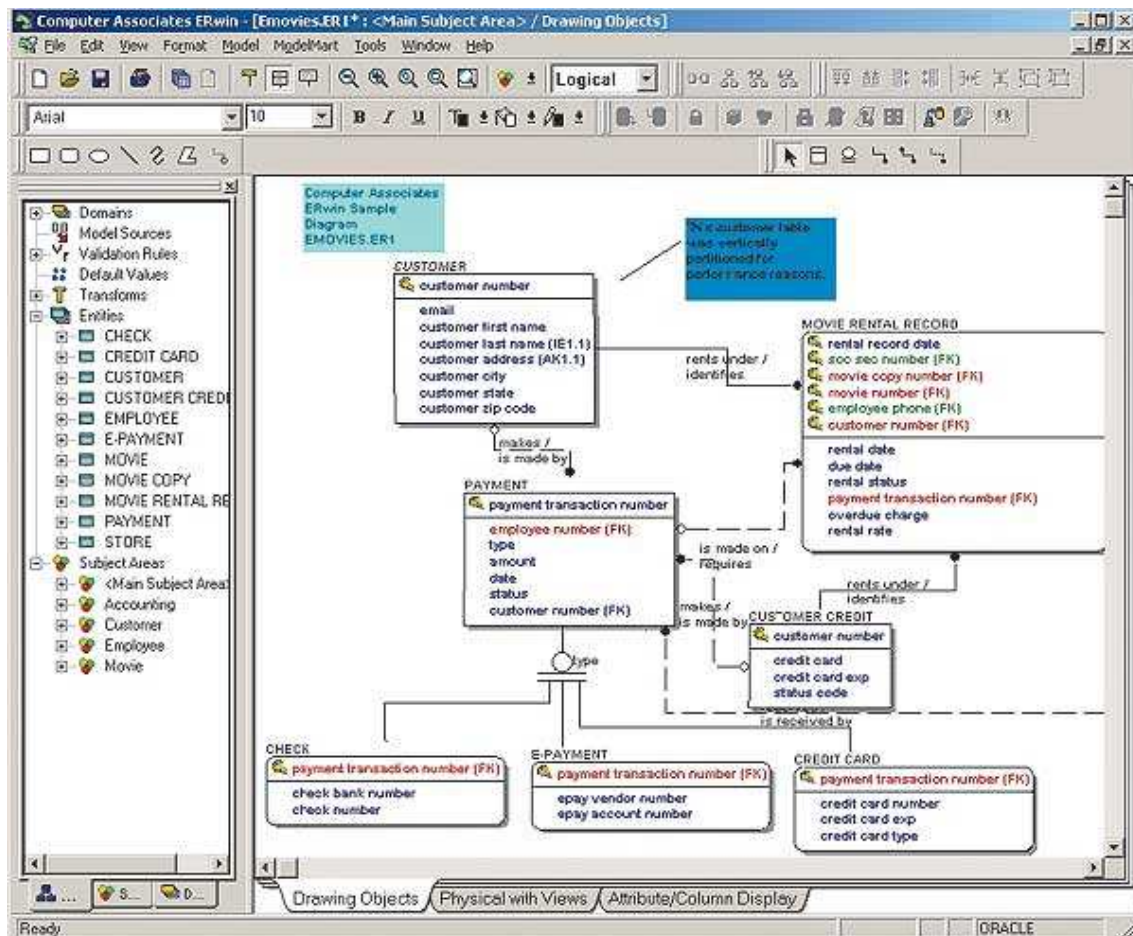
La mayoría de las empresas se han extendido a la adquisición de herramientas CASE (Computer Aided Software Engineering, Ingeniería Asistida por Computadora) con el fin de automatizar los aspectos clave de todo lo que implica el proceso de desarrollo de un sistema e incrementar su posición en el mercado competitivo. Sin embargo, en algunos se obtienen elevados costos tanto en la adquisición de herramientas y costos de entrenamiento de personal, como a la falta de adaptación de tal herramienta a la arquitectura de la información y a metodologías de desarrollo utilizadas por la organización.

Por otra parte, algunas herramientas CASE no ofrecen o evalúan soluciones potenciales para los problemas relacionados con sistemas o virtualmente no llevan a cabo ningún análisis de los requerimientos de la aplicación. Sin embargo, CASE proporciona un conjunto de herramientas semiautomatizadas y automatizadas que están desarrollando una cultura de ingeniería nueva para muchas empresas. Uno de los objetivos más importante del CASE (a largo plazo) es conseguir la generación automática de programas desde una especificación al nivel de diseño.

CA ERWIN DATA MODELER

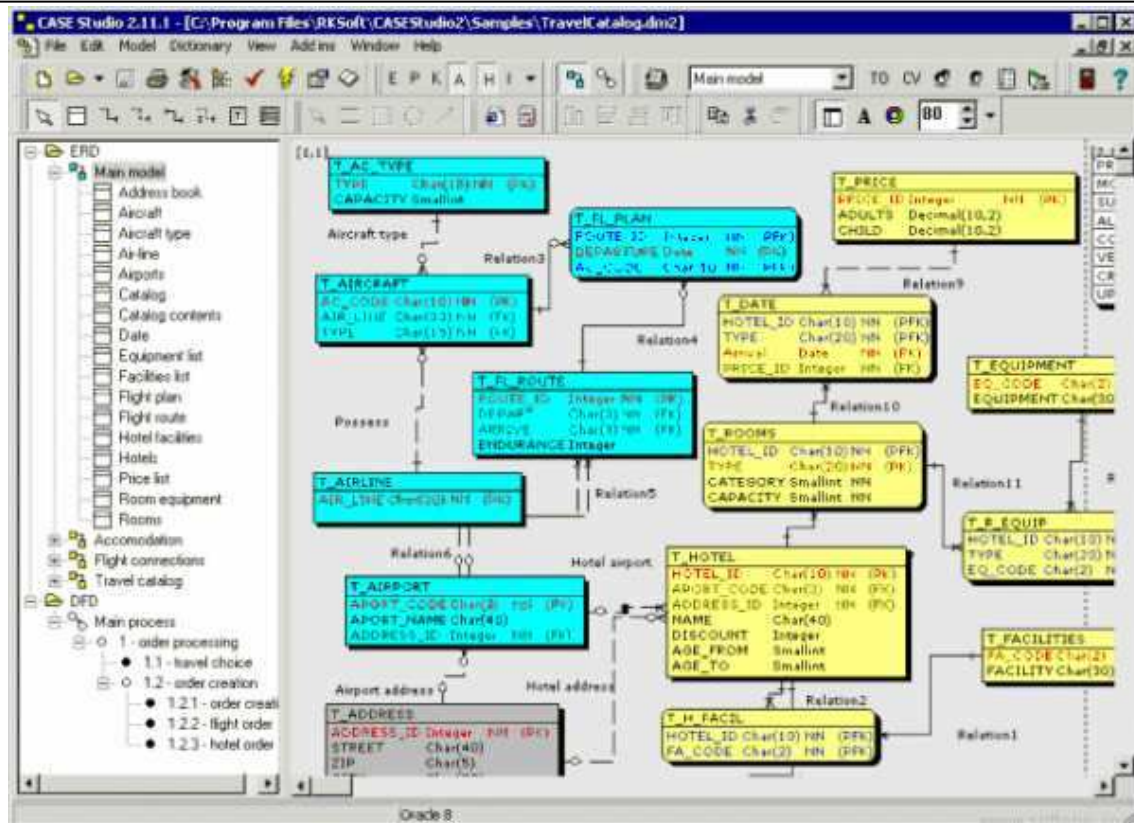
CA ERwin Modeling Suite proporciona una automatización de las tareas de diseño, así como funcionalidades de colaboración y optimización que permiten a los usuarios ofrecer soluciones alineadas con el negocio que se ajusten al tiempo planificado, al presupuesto y lo más importante, a los objetivos.

AllFusion ERwin Data Modeler automáticamente genera tablas y miles de líneas de procedimientos almacenados y códigos disparadores para las base da datos líderes. Su tecnología de “comparación completa” permite el desarrollo iterativo, de forma tal que los modelos están siempre sincronizados con la base de datos del usuario. Al integrarse con entornos de desarrollo líderes, AllFusion ERwin Data Modeler también acelera la creación de aplicaciones centralizadas en datos.



STUDIO CASE

Herramienta para el diseño de bases de datos que todo profesional de sistemas debe tener en cuenta. Esta aplicación permite realizar Diagramas Entidad-Relación (DER) y Diagramas de Flujos de Datos (DFD) para distintos motores de base de datos. Algunos de éstos pueden ser: Oracle, DB2, InterBase, MS SQL, MySQL y PostgreSQL entre otros. Otra de las características importantes es que permite realizar ingeniería inversa, o sea, a partir del modelo de tablas llegar al modelo lógico.



SYBASE POWER DESIGNER

Es un ambiente integrado de Ingeniería de Software para el análisis y diseño de entornos empresariales, con capacidades para el modelamiento de negocios, aplicaciones, datos y objetos, que incluyen administración de requerimientos y generación de documentación. Sincroniza y encadena las capas y perspectivas de la Arquitectura Empresarial, permitiendo documentar el estado actual de la organización y el impacto que genera aplicar un cambio de manera predictiva. Sus alternativas de licenciamiento contemplan:

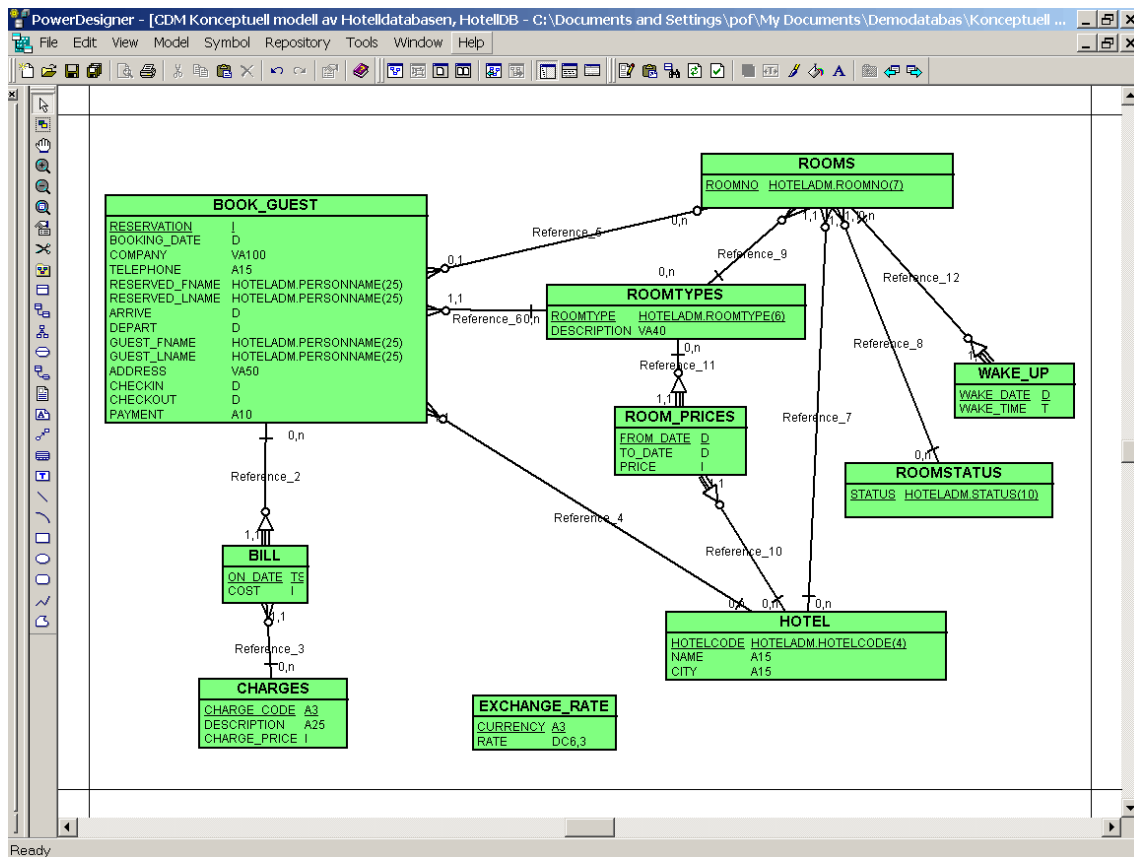
MODELAMIENTO DE PROCESOS DE NEGOCIO: Brinda el poder a los usuarios no técnicos para diseñar y modelar procesos en términos reales del negocio, usando un modelo simple, fácil de usar, altamente gráfico y no técnico. Soporta a la interpretación y generación de documentos XML.

MODELAMIENTO DE DATOS: Facilita el diseño y generación del esquema de la base de datos a través de un modelamiento de Bases de Datos Relacionales y/o Bodega de Datos de dos niveles (conceptual y físico). También permite la Ingeniería reversa de bases de datos vía ODBC o desde archivos SQL.

MODELAMIENTO DE OBJETOS: Complementa el análisis y el diseño usando métodos y diagramas basados en el estándar UML. A partir de un diagrama de clases, automáticamente genera y realiza ingeniería reversa a través de un generador personalizable.

MODELAMIENTO XML: Este modelo contiene una interfaz gráfica para representar la estructura de documentos XML, facilitando la visualización de los mismos a través de diagramas.

REPOSITORIO EMPRESARIAL: Permite visualizar y compartir modelos y otra información en la organización. Soporta seguridad basada en roles, control de versiones, búsqueda y generación de reportes.



Semana

2

Contenido:

- Modelo Conceptual.
 - Características
 - El Modelo Entidad Relación (MER)
 - Operaciones de Generalización y Agregación
 - Relaciones Recursivas
 - Diagrama Entidad Relación (DER)
 - Tipos y Representación de entidades
 - Atributos y tipos
 - Tipos y Representación de relaciones
 - Introducción a Erwin (MER)
 - Creación del Diagrama Entidad Relación (MER)
 - Entidades. Tipos
 - Relaciones. Tipos
 - Ejercicios
-

MODELAMIENTO CONCEPTUAL

➤ MODELO CONCEPTUAL

Concluida la fase de análisis de requerimientos, y definidos los procedimientos del negocio por intermedio de herramientas de análisis, lo que corresponde a continuación es capturar formalmente toda la información relevante en un medio físico, una representación que refleje las soluciones a los requerimientos establecidos, para ello utilizamos la técnica de Modelamiento de datos, que será la base para estructurar nuestra BD. La información que se almacenará debe cumplir con todas las exigencias propias del sistema.

Luego de haber recopilado la información necesaria para definir los requerimientos de los usuarios, las problemáticas y necesidades del negocio que se está modelando, el primer paso es crear un modelo conceptual que refleje la realidad del negocio, entonces esta es la primera fase del diseño de datos, aunque generalmente se suele obviar este paso para ir de frente al diseño de entidades y atributos. Fue creado por Peter Chen a finales de los 70's.

✓ CARACTERISTICAS

Un modelo de datos es una serie de conceptos que puede utilizarse para describir un conjunto de datos y las operaciones para manipularlos. Hay dos tipos de modelos de datos: los modelos conceptuales y los modelos lógicos. Los modelos conceptuales se utilizan para representar la realidad a un alto nivel de abstracción. Mediante los modelos conceptuales se puede construir

una descripción de la realidad fácil de entender. En los modelos lógicos, las descripciones de los datos tienen una correspondencia sencilla con la estructura física de la base de datos.

En el diseño de bases de datos se usan primero los modelos conceptuales para lograr una descripción de alto nivel de la realidad, y luego se transforma el esquema conceptual en un esquema lógico. El motivo de realizar estas dos etapas es la dificultad de abstraer la estructura de una base de datos que presente cierta complejidad. Un esquema es un conjunto de representaciones lingüísticas o gráficas que describen la estructura de los datos de interés.

Los modelos conceptuales deben ser buenas herramientas para representar la realidad, por lo que deben poseer las siguientes cualidades:

- Expresividad: deben tener suficientes conceptos para expresar perfectamente la realidad.
- Simplicidad: deben ser simples para que los esquemas sean fáciles de entender.
- Unicidad: cada concepto debe tener un significado distinto.
- Formalidad: todos los conceptos deben tener una interpretación única, precisa y bien definida.

En general, un modelo no es capaz de expresar todas las propiedades de una realidad determinada, por lo que hay que añadir aseveraciones que complementen el esquema.

➤ **EL MODELO ENTIDAD RELACION (MER)**

Peter Chen – Edward Codd (1976)

Se basa en la percepción del mundo real y consiste en una colección de entidades y relaciones. Se emplea para interpretar, especificar y documentar los requerimientos del usuario. Se utiliza para describir la realidad mediante un conjunto de representaciones gráficas y lingüísticas.

✓ **ELEMENTOS DE UN MER**

✓ **ENTIDAD**

Una entidad es un objeto, instancia, persona, o cosa puramente conceptual o real sobre el cual se desea guardar información por ser de relevancia para una organización. Una entidad es aquello que más adelante se convertirá en un elemento de nuestra base de datos, por lo mismo contendrá información propia que será manipulado por los usuarios del sistema. Se recomienda que los nombres de las entidades estén en singular.

¿Cómo identifico una entidad?

Una entidad puede ser identificada como un sustantivo dentro de la narración de un proceso de negocio dentro de la organización.

Ejm: Los **clientes** compran **productos** por medio de un comprobante que es la **factura**.

Otro ejemplo:

Los alumnos tienen cursos nuevos en este semestre, los profesores dictarán las clases desde el día de hoy.

Entidades: alumnos, cursos, profesores. Son entidades porque son los sustantivos que me representan algo de la cual puedo guardar información.

✓ **RELACION**

Las entidades se relacionan de acuerdo a algo que tengan en común, entonces diremos que una relación o interrelación es la asociación o correspondencia entre dos o más entidades.

Cada relación tiene un nombre que describe su función. Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior.

Las entidades que están involucradas en una determinada relación se denominan entidades participantes. El número de participantes en una relación es lo que se denomina grado de la relación. Por lo tanto, una relación en la que participan dos entidades es una relación binaria; si son tres las entidades participantes, la relación es ternaria; etc.

➤ **DIAGRAMA ENTIDAD RELACION (DER)**

Denominado por sus siglas como: E-R; Este modelo representa a la realidad a través de un esquema gráfico empleando la terminología de entidades, que son objetos que existen y son los elementos principales que se identifican en el problema a resolver con el diagramado y se distinguen de otros por sus características particulares denominadas atributos, el enlace que rige la unión de las entidades está representada por la relación del modelo.

Una entidad en el modelo conceptual se representa por un rectángulo:



La relación entre entidades se representa mediante un rombo, dentro del cual se coloca el verbo o frase verbal de la relación.



✓

CICLOS

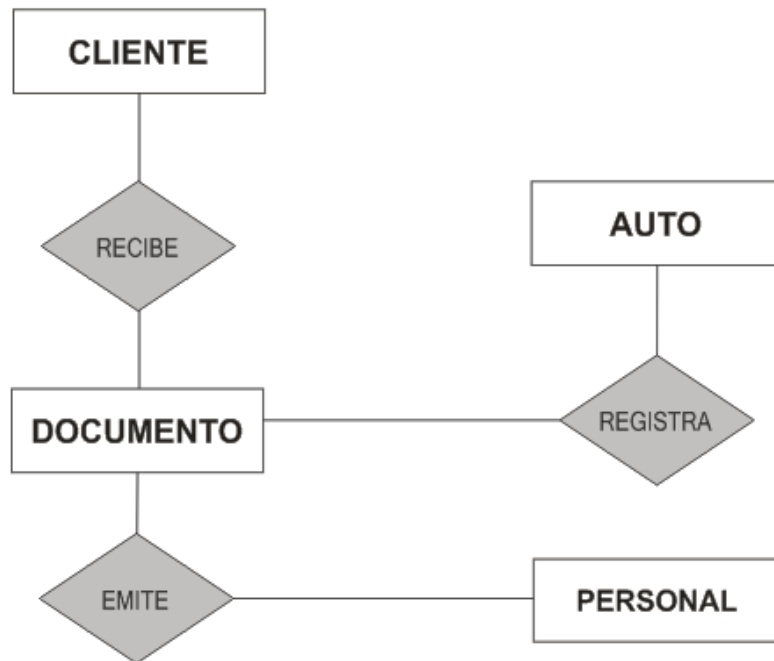
EJER

1) Identificar las entidades y sus relaciones en el siguiente caso:

- ✓ En una concesionaria de automóviles, se venden autos de marcas Honda, Volvo y Mercedes Benz, los clientes son atendidos por personal de ventas especializados en el tema de automóviles, si la venta se lleva a cabo, se le entrega los documentos correspondientes al cliente.

Resolviendo:

Entidades: auto, cliente, personal de venta, documentos, porque son los sustantivos del cual se requiere información, no Honda ni Volvo ni Mercedes Benz, éstas son elementos de una entidad auto, no confundir entidad y elementos, mas adelante detallaré este punto. Graficando en el modelo conceptual:



2) Relacione las siguientes entidades:

- Personas – Distrito



- Cliente – Pedido



- País - Ciudad



✓ TIPOS DE ENTIDADES

ENTIDAD FUERTE: También conocida como entidad Padre, es aquella entidad cuya existencia no depende de la existencia de alguna otra entidad, es identificada fácilmente dentro de un proceso y da lugar a la posibilidad de otras entidades.

ENTIDAD DEBIL: También conocida como entidad Hijo, es aquella cuya existencia depende de la existencia de otra entidad, en este caso de una entidad fuerte, tiene sus propias características aunque está ligada a la entidad fuerte.

En el ejemplo anterior, diremos que las entidades Cliente, Auto y Personal son fuertes, mientras que Documentos es débil ya que no se generará elemento alguno de esa entidad sin la existencia de las otras tres.

✓ EJERCICIO

Identificar las entidades fuertes y débiles:

1.	PlanCuentas	F	Boucher	D
2.	Asistencia	D	Personal	F
3.	Alumno	F	Nota	D
4.	Factura	D	Cliente	F
5.	Pedido	D	Proveedor	F
6.	Cotización	F	Producto	F
7.	Alumno	F	Curso	F
8.	Producto	F	OrdenCompra	F
9.	Orden Compra	D	Proveedor	F
10.	Personal	D	Cargo	F

➤ ATRIBUTOS

Los atributos son aquellos que caracterizan a una entidad, son las propiedades que posee cada entidad y que la hacen distinta ante las otras entidades, no existe entidad que no tenga al menos un atributo.

Por ejemplo, los atributos propios para una persona serían su nombre, dirección, edad, salario, peso, talla, color, religión, etc. Para un objeto sería su textura, tamaño, material, utilidad, tipo, etc.

✓ **TIPOS DE ATRIBUTOS**

- 1) **ATRIBUTOS SIMPLES:** Son aquellos atributos que son fáciles de identificar, inherentes a la entidad y no pueden seguir descomponiéndose.



- 2) **ATRIBUTOS COMPUESTOS:** Son aquellos atributos que tienen 2 o más atributos simples que lo componen, llamados también atributos concatenados, ya que se forman por la unión de dos atributos simples. es un atributo con varios componentes, cada uno con un significado por sí mismo. Un grupo de atributos se representa mediante un atributo compuesto cuando tienen afinidad en cuanto a su significado, o en cuanto a su uso. Un atributo compuesto se representa gráficamente mediante un óvalo.



Digamos que el nuevo atributo Apellido se compone así: AP + AM.

DONDE: AP es apellido paterno y AM es apellido materno

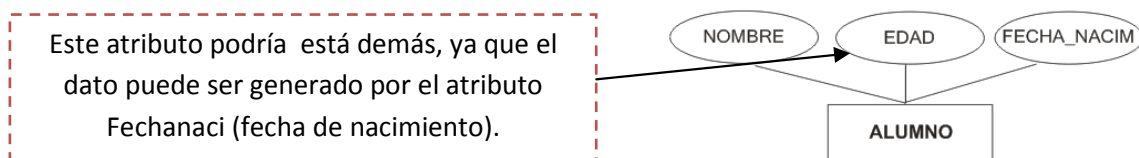
- 3) **ATRIBUTOS MULTIVALORADOS:** Son aquellos atributos que tienen un conjunto de valores para un solo registro de datos, éstos atributos son capaces de generar inclusive una nueva entidad. También se le conoce como atributos Polivalentes. pueden tener un número máximo y un número mínimo de valores. La cardinalidad de un atributo indica el número mínimo y el número máximo de valores que puede tomar para cada ocurrencia de la entidad o relación a la que pertenece.



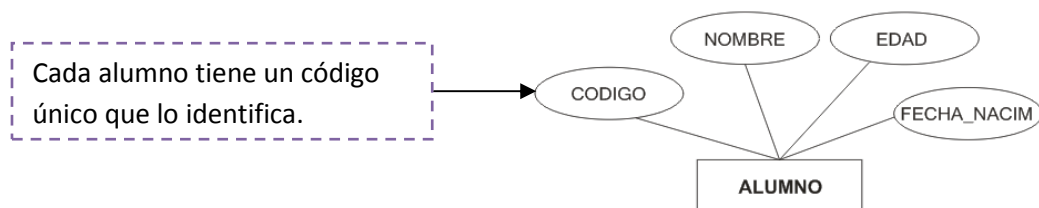
Digamos que el cliente de nuestra empresa, en realidad tenga varios números telefónicos donde ubicarlo, entonces estamos ante un atributo 'Teléfono' con varios valores posibles, entonces podría generar una nueva entidad.



- 4) **ATRIBUTO DERIVADO:** Es aquel que puede ser generado por otros atributos de la misma entidad e inclusive por atributos de otra entidad. Un atributo derivado es aquel que representa un valor que se puede obtener a partir del valor de uno o varios atributos, que no necesariamente deben pertenecer a la misma entidad o relación.



- 5) **ATRIBUTO CLAVE:** Es aquel atributo que identifica unívocamente a la entidad, es decir, son aquellos atributos cuyos valores no se repetirán jamás para otra entidad, inclusive en la misma entidad, no existirán dos registros con el mismo valor para ese atributo, a propósito de este tipo de atributo hablaremos de los índices o claves de las entidades.



➤ CLAVES O LLAVES

Una clave es un atributo especial que identifica de manera única cada ocurrencia de la entidad, es decir, es el que identifica a la entidad como única e irrepetible en el universo que es el proceso de negocio. No puede existir dos valores iguales para clave de la entidad, ya que se estaría infringiendo la integridad referencial de los datos (más adelante detallaré este punto).

✓ **TIPOS DE CLAVES:**

CLAVE PRIMARIA (PRIMARY KEY – PK)

Una clave primaria es aquella clave candidata que el diseñador eligió como principal arbitrariamente, es decir, esta será la que identifique los elementos de cada entidad de manera única e irrepetible.

Un elemento, conocido también como instancia de entidad, es una existencia u ocurrencia de la entidad, por ejemplo, para una entidad Alumno, tenemos matriculados 20 alumnos, entonces en ese caso tenemos 20 instancias de la entidad Alumno.

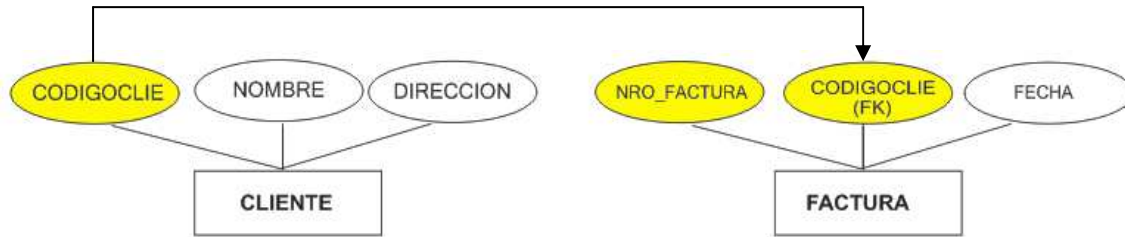
Recomendaciones del profesor al momento de asignar claves primarias:

- ✓ Para entidades que representen personas u objetos, generalmente se utilizan códigos o ID's para identificarlos en el modelo.
- ✓ Si la entidad representa productos manufacturados o artefactos, también puede utilizarse el código de fabricación o # de serie para identificarlos.
- ✓ Para documentos, lo recomendable es identificarlos por el número del documento que fue generado por sistema.
- ✓ Para entidades que representen cosas abstractas, puede elegirse cualquiera de las opciones anteriores.



CLAVE FORANEA (FOREIGN KEY – FK)

Es la llave primaria de la entidad padre que fue agregada a la entidad hijo por medio de la relación entre ambas, vale decir que en la entidad fuerte sigue siendo clave primaria, pero al pasar a la entidad débil, pasa como clave secundaria o foránea, pero sólo en la entidad débil será foránea.



CLAVE CANDIDATA

Las claves candidatas son simplemente aquellas claves las cuales no fueron seleccionadas como clave primaria, exactamente alguna de esas claves es seleccionada como PK y las restantes si existe alguna son llamadas claves alternas, pero pueden servir para identificar en una consulta de datos a la entidad en un momento dado.

Ejemplo: para un cliente de nuestra empresa le asignamos como PK un código que será único para cada cliente registrado en el sistema, sin embargo también es cierto que como personas civiles que son, tiene DNI que es único por cada persona, entonces éste puede ser otro identificador del cliente cuando se requiera consultar a los clientes del negocio, es entonces una clave candidata.

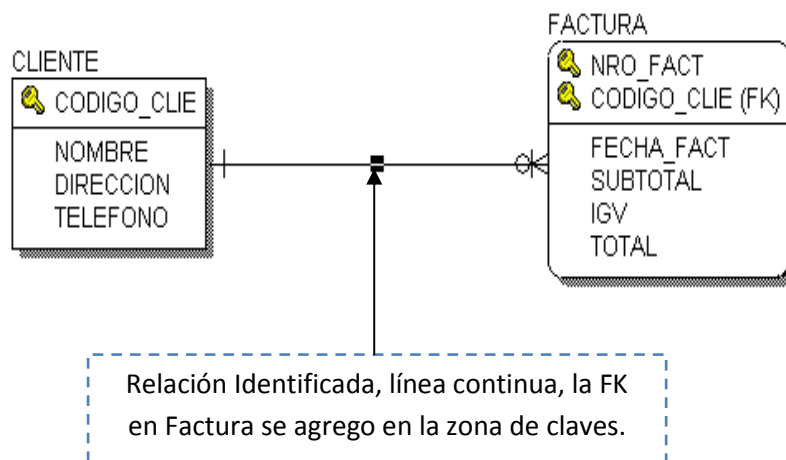
Otras claves candidatas pueden ser: teléfono, lote, número de serie, etc.

➤ TIPOS DE RELACIONES

✓ RELACION IDENTIFICADA (Obligatoriedad)

Una relación identificada representa la obligatoriedad de la existencia de una entidad (fuerte) para que se produzca ocurrencia de elementos en otra entidad (débil), acá se trabajan con las claves para relaciones los registros de ambas entidades, cada registro representa una ocurrencia de la entidad. Las relaciones identificadas crean claves primarias compuestas, ya que agrega la FK dentro de la zona de claves en la entidad Hijo.

La PK de la entidad padre se agrega a la entidad hijo como FK, dentro de la zona de claves y con línea de relación continua.

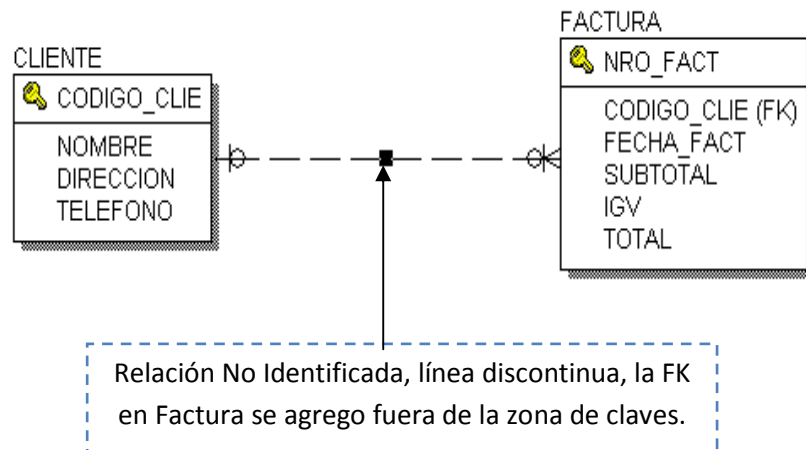


Este ejemplo indica que podemos conocer los detalles de una factura conociendo el código del cliente (y los datos de esta entidad también). Esto no siempre debe ser así.

✓ RELACION NO IDENTIFICADA (No obligatoriedad)

En una relación no identificada no es obligatoria identificar la PK de la entidad padre para identificar algún elemento de la entidad hijo, se representa con una línea discontinua.

La PK de la entidad padre se agrega a la entidad hijo como FK fuera de la zona de claves.

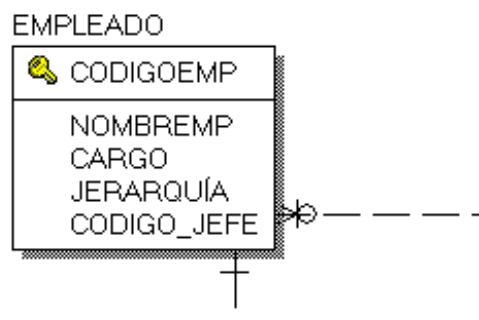


En este ejemplo se indica que para identificar una factura en el sistema, no es necesario conocer los datos del cliente, algo que es más aceptable.

GRADO DE LA RELACION: Se refiere al número de entidades participantes en una relación, digamos que si tenemos dos entidades participantes en la relación se trata de una **relación binaria**, si tenemos tres entidades participantes en la relación, se trata entonces de una **relación ternaria**, y así sucesivamente.

➤ RELACION RECURSIVA

En un tipo de relación muy especial, en donde los elementos de la misma entidad participan más de una vez en la relación con distintos papeles, esto se da por medio de la PK de la entidad, se denomina **Recursividad de datos**. Este tipo de relación se utiliza para representar la dependencia de los elementos de la entidad con otros elementos de la misma entidad.

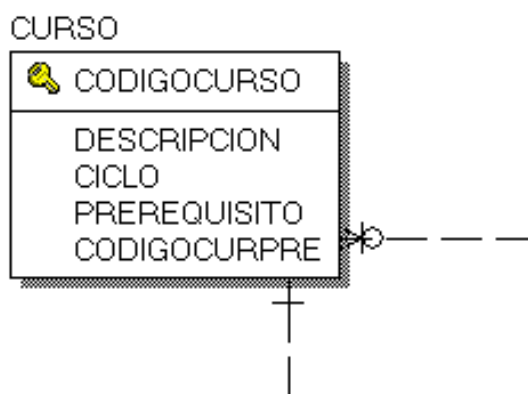


CODIGO_EMP	NOMBRE	CARGO	JERARQUIA	CODIGO_JEFE
E0001	PEDRO	VENDEDOR	SUBORDINADO	E0002
E0002	JOSE	GERENTE_VENTA	JEFE DE AREA	NULL
E0003	MARIA	ASISTENTE	SUBORDINADO	E0002

Tenemos que cada empleado tiene una jerarquía asociado a su cargo y área, si definimos que cada jefe tiene un subordinado, diremos que existen elementos de la entidad que están relacionados con otros elementos de la misma entidad, estamos ante una relación recursiva.

Una relación recursiva es una relación donde la misma entidad participa más de una vez en la relación con distintos papeles. El nombre de estos papeles es importante para determinar la función de cada participación.

Otro ejemplo puede ser en el caso de aquellos cursos que tienen como prerequisites el haber aprobado un curso anterior o el módulo anterior del curso en cuestión, entonces tendríamos algo como esto...



Para poder llevar un curso en el ciclo actual de estudios, se sabe que previamente debemos aprobar el curso que le antecede en el ciclo anterior, lo que se conoce como cursos prerequisites, entonces tendremos la siguiente tabla como resultado.

CODIGO_CURSO	DESCRIPCION	CICLO	CODIGOCURPRE
C0002010011	MATEMATICA I	I	NULL
C0002010012	MATEMATICA II	II	C0002010011
C0002010013	ANALISIS MATEMATICO I	III	C0002010012
C0002010014	ANALISIS MATEMATICO II	IV	C0002010013

Vemos entonces cómo se cumple la recursividad de la relación entre los cursos, en la cual existe dependencia entre uno y otro.

✓ CARDINALIDAD DE UNA RELACION

La cardinalidad con la que una entidad participa en una relación especifica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad. La participación de una entidad en una relación es obligatoria (total) si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante. Si no, la participación es opcional (parcial). Las reglas que definen la cardinalidad de las relaciones son las reglas de negocio.

A veces, surgen problemas cuando se está diseñado un esquema conceptual. Estos problemas, denominados trampas, suelen producirse a causa de una mala interpretación en el significado de alguna relación, por lo que es importante comprobar que el esquema conceptual carece de dichas trampas. En general, para encontrar las trampas, hay que asegurarse de que se entiende completamente el significado de cada relación. Si no se entienden las relaciones, se puede crear un esquema que no represente fielmente la realidad.

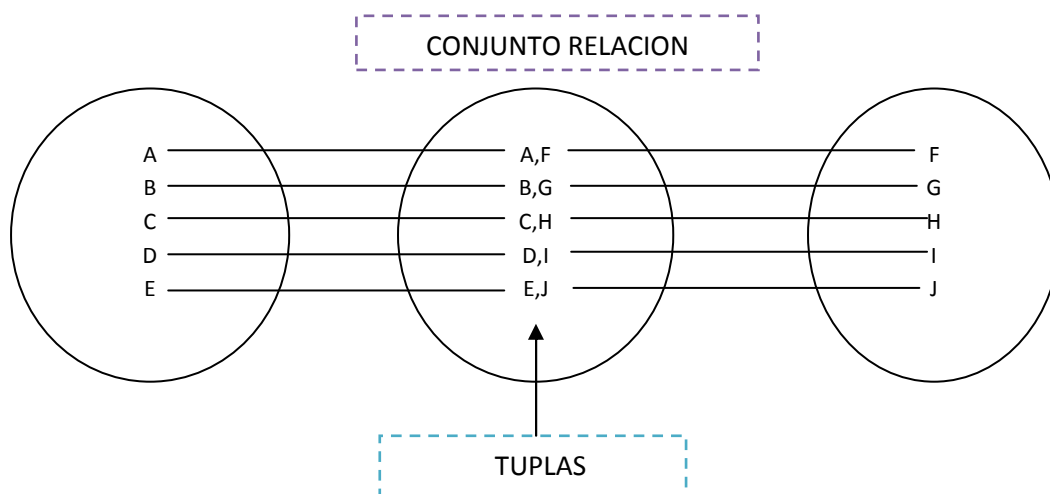
Una de las trampas que pueden encontrarse ocurre cuando el esquema representa una relación entre entidades, pero el camino entre algunas de sus ocurrencias es ambiguo. El modo de resolverla es reestructurando el esquema para representar la asociación entre las entidades correctamente.

Otra de las trampas sucede cuando un esquema sugiere la existencia de una relación entre entidades, pero el camino entre una y otra no existe para algunas de sus ocurrencias. En este caso, se produce una pérdida de información que se puede subsanar introduciendo la relación que sugería el esquema y que no estaba representada.

✓ TUPLAS

La unión de dos entidades da como producto un conjunto relación cuyos elementos son denominados como 'TUPLAS', cada tupla representa la relación entre los elementos de las entidades participantes, estas tuplas reflejan la Cardinalidad de la relación, de acuerdo a ello diremos que existe 2 clases de cardinalidades:

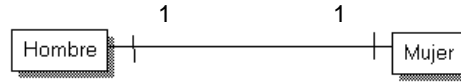
- ▶ **Cardinalidad mínima:** Es la mínima cantidad de tuplas que cada elemento de la entidad A puede tener con elementos de la entidad B.
- ▶ **Cardinalidad máxima:** Es la máxima cantidad de tuplas que cada elemento de la entidad A puede tener con elementos de la entidad B.



✓ **TIPOS DE CARDINALIDADES**

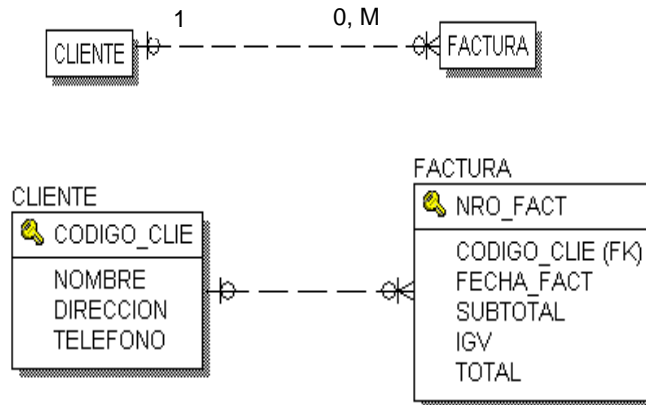
RELACION DE UNO A UNO

En este tipo de relaciones, cada instancia o elemento de la entidad A está asociado solamente a un elemento de la entidad B. Se recomienda que cuando se identifique una relación de este tipo, se una ambas entidades formando una sola, salvo casos especiales.



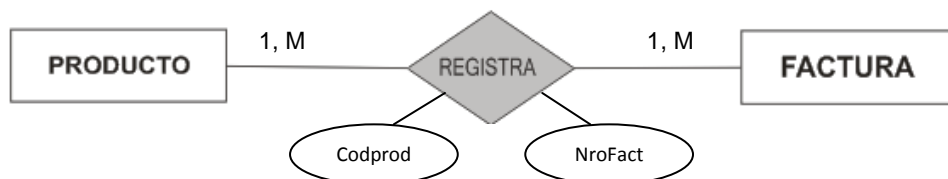
RELACION DE UNO A MUCHOS

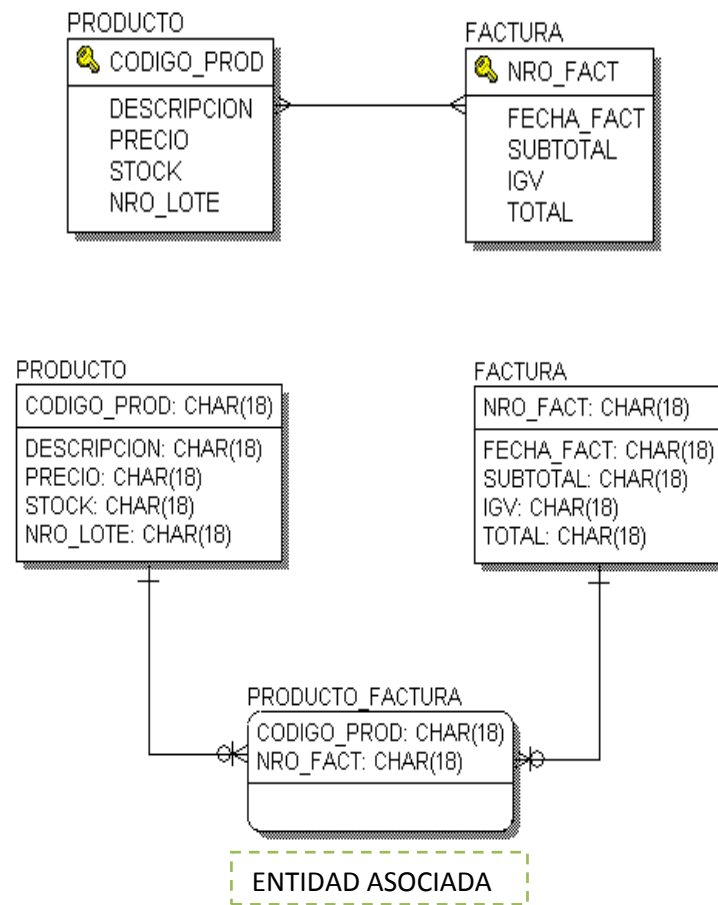
En este tipo de relaciones, cada instancia o elemento de la entidad A está asociado a varios elementos de la entidad B, entonces la clave que forma el vínculo entre ambas entidades, pasa hacia la entidad que tiene el mayor grado de Cardinalidad, es decir el que posee la denominación 'muchos'.



RELACION DE MUCHOS A MUCHOS

En este tipo de relación, los elementos de la entidad A están asociados a varios elementos de la entidad B, y los elementos de la entidad B están asociados a varios elementos de la entidad A, cuando sucede esto, se genera una nueva entidad denominada 'Entidad Asociada', generalmente toma el nombre de ambas entidades participantes o la denominación del verbo de la relación. La entidad asociada se grafica sólo en el modelo físico de datos, en el nivel lógico se representa la relación muchos a muchos.

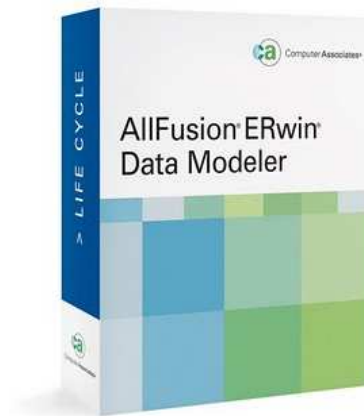




A TENER EN CUENTA:

- ✓ Las entidades asociadas heredan las PK's de las entidades padres que la generaron, ambas forman la clave principal de la entidad asociada, lo que se conoce como 'CLAVES COMPUESTAS', es otro tipo de claves que se dan en este tipo de casos.
- ✓ La Cardinalidad mínima se representa con el número 0 o 1, mientras que la Cardinalidad máxima se representa con el 1 o M.
- ✓ La variante '0' se da cuando exista la posibilidad de que algún elemento de la entidad A no esté necesariamente asociado a un elemento de la entidad B.
- ✓ En el modelo conceptual las claves se representan como pequeñas circunferencias que sobresalen de los rectángulos que representan a la entidad.

LABORATORIO # 2: INTRODUCCION A ERWIN 7.1



¿QUE ES ALLFUSION ERWIN DATA MODELER?

AllFusion ERwin Data Modeler es una herramienta de diseño de base de datos que ayuda a los usuarios a diseñar, generar y mantener alta calidad de las aplicaciones de base de datos de alta performance. AllFusion ERwin Data Modeler permite al usuario visualizar la estructura correcta, elementos claves y el diseño optimizado de su base de datos, desde los requerimientos de un modelo lógico de información y reglas de negocio que definen la base de datos, a un modelo físico optimizado para las características específicas de la base de datos seleccionada.

AllFusion ERwin Data Modeler automáticamente genera tablas y miles de líneas de procedimientos almacenados y códigos disparadores para las bases de datos líderes. Su tecnología de “comparación completa” permite el desarrollo iterativo, de forma tal que los modelos están siempre sincronizados con la base de datos del usuario. Al integrarse con entornos de desarrollo líderes, AllFusion ERwin Data Modeler también acelera la creación de aplicaciones centralizadas en datos.

La mejor gestión de la información empieza por un diseño óptimo de las bases de datos.

Mediante AllFusion™ ERwin® Data Modeler, las empresas pueden visualizar estructuras complejas de datos y activos de información corporativa, así como establecer estándares de gestión de datos para toda la empresa. Permite automatizar de forma inteligente procesos de diseño y sincronizar el modelo con el diseño

o de bases de datos. Los modeladores pueden utilizar este producto para diseñar sistemas transaccionales, data warehouses y data marts en un entorno integrado.

AllFusion™ ERwin® Data Modeler también permite:

- Incrementar la productividad proporcionando un entorno gráfico fácil de utilizar que simplifica el diseño de las bases de datos y automatiza muchas tareas tediosas. Agiliza la creación de bases de datos transaccionales y data warehouses de alta calidad y rendimiento.
- Comunicarse de forma más eficaz permitiendo que los DBA y desarrolladores compartan y reutilicen modelos, además de poder representar innumerables y complejos activos de datos mediante un formato fácil de comprender y mantener.

- Proporcionar respuestas más rápidas a las necesidades empresariales en evolución permitiendo a las empresas comprender el impacto del cambio en los activos de información y facilitando la rápida implementación de cambios.

✓ **CARACTERÍSTICAS GENERALES**

- Aumenta la productividad.
- Comunica en forma más efectiva.
- Responde más rápidamente a las necesidades de la evolución de los negocios.
- Diseña arquitecturas en capas.
- Tecnología transformable.
- Administra grandes modelos.
- Comparación completa.
- Genera diseños de base de datos.
- Diseña almacenes de datos y mercados de datos.

✓ **ENTORNOS SOPORTADOS**

- Plataformas:

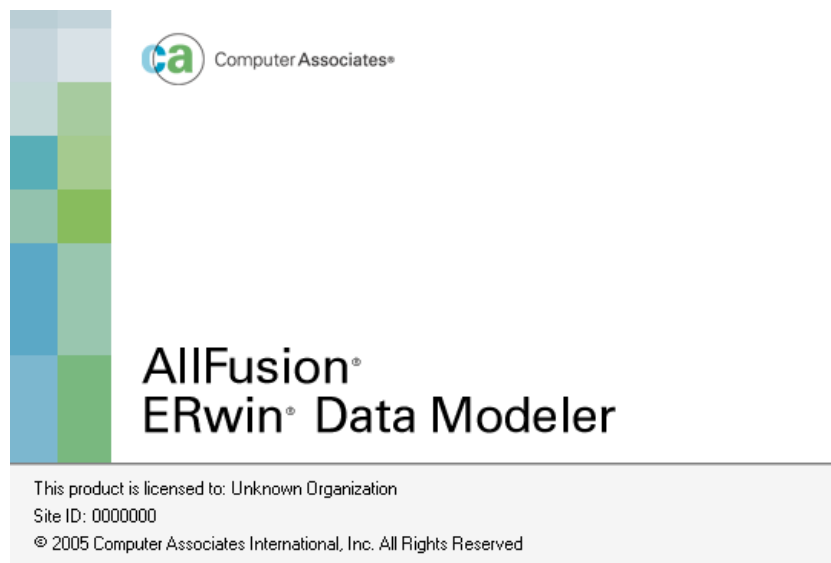
Windows 95, 98, 2000 SP3, NT 4.0, XP y 2003 server

- Bases de datos:

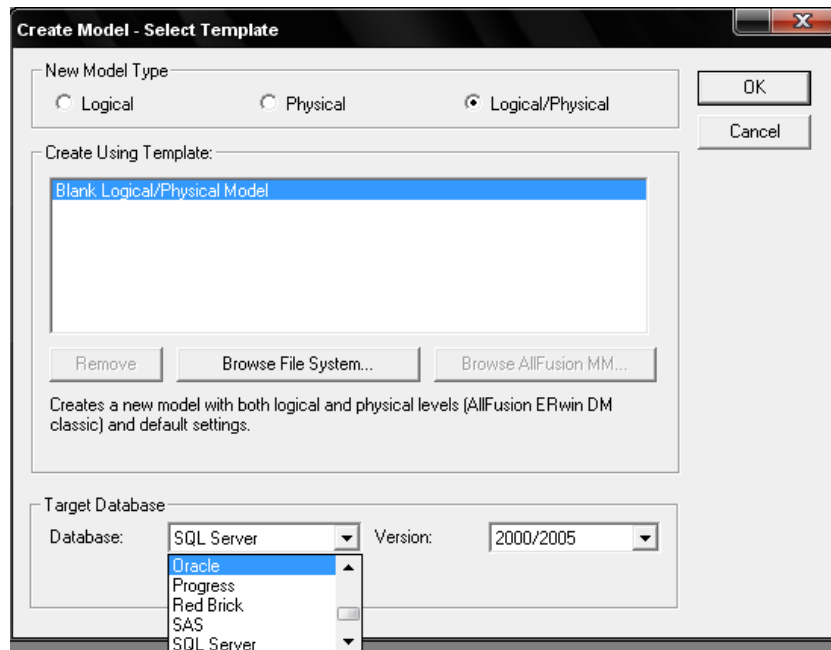
Advantage™ Ingres® Enterprise Relational Database, Advantage™ CA-Clipper®, DB2, dBASE, FoxPro, HiRDB, Informix, InterBase, Microsoft Access, Teradata, Microsoft SQL Server, ODBC 2.0, 3.0, Oracle, Paradox, Rdb, Red Brick Warehouse, SAS, SQL Anywhere, SQL Base y Sybase.

INGRESANDO A ERWIN...

Menú Inicio – Programas –Computer Associates – All Fusion – Erwin Data Modeler r7 – Erwin Data Modeler r7.

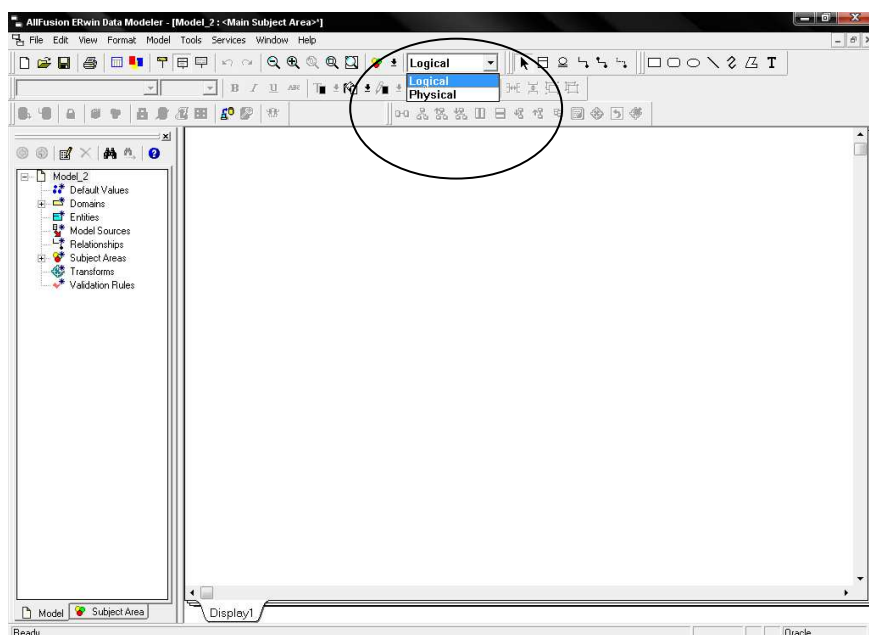


CREANDO UN NUEVO MODELO DE DATOS: MENÚ ARCHIVO – NUEVO (en caso no aparezca desde el inicio la ventana de selección de modelos).



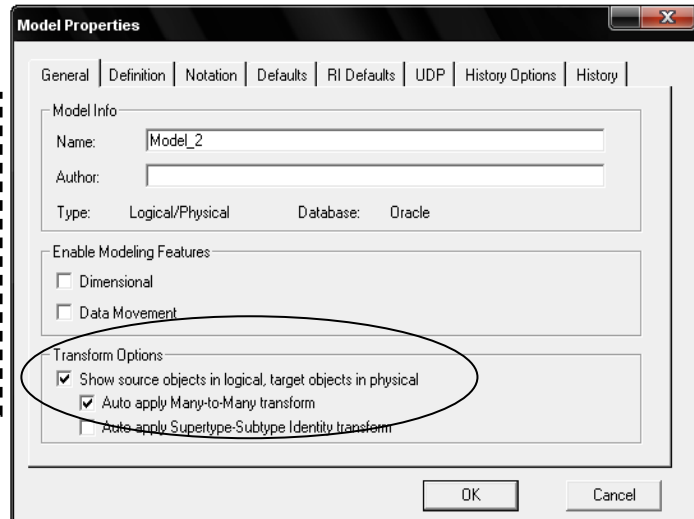
En este caso debemos seleccionar el tipo de modelo Logical/Physical, además podemos seleccionar el motor de base de datos con la que queremos trabajar, como Oracle, SQL Server DB2, Access, etc. Al lado derecho podemos seleccionar la versión del DBMS seleccionado.

Como vemos el entorno de trabajo presenta una serie de menús y cuadros de herramientas para el trabajo de diseño. Para poder crear los modelos no vamos a utilizar todos generalmente, veamos los más importantes. Primero debemos conocer los dos niveles con los que trabaja Erwin: El lógico y el nivel físico, para seleccionarlos debemos desplegar el combo que se presenta:

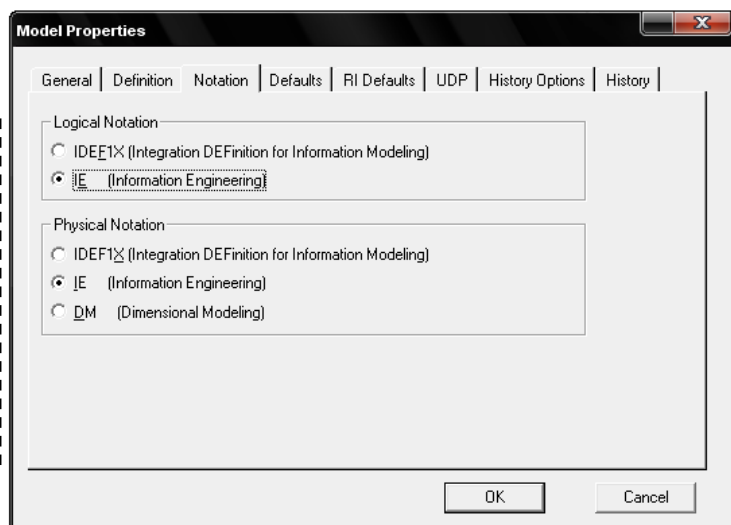


Lo siguiente sería activar la opción Entidad asociada para las relaciones Muchos a Muchos, generalmente vienen desactivadas por lo que al pasar al nivel físico no aparece la entidad asociada, tenemos que activarla. Además debemos cambiar la metodología de información para que presente las cardinalidades tal cual las conocemos. Esto se hace desde el menú Model – Model Properties. Veamos:

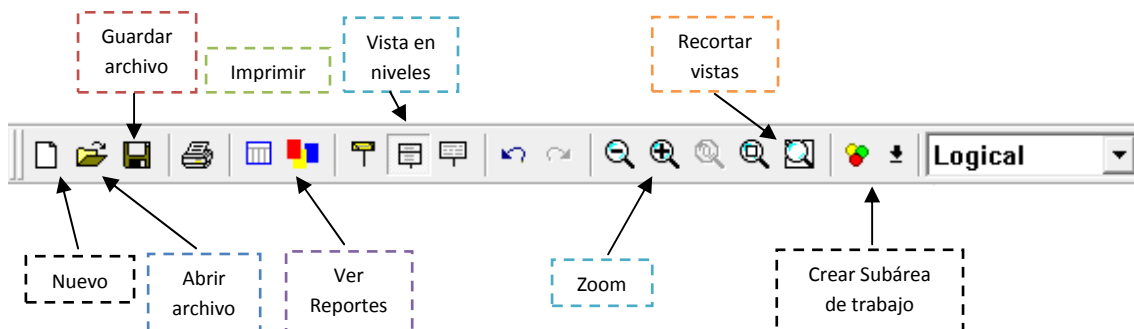
Debemos activar la opción Show source objectspara luego seleccionar la opción Auto play Many – to many transform. De otro modo Erwin no mostrará las entidades asociadas en el modelo físico al realizar las relaciones Muchos a Muchos.



Igualmente cambiaremos la notación del modelo Lógico, cardinalidad a IE (Information Engineering). Igualmente haremos para el modelo Físico, más adelante veremos la diferencia con IDEF1X.



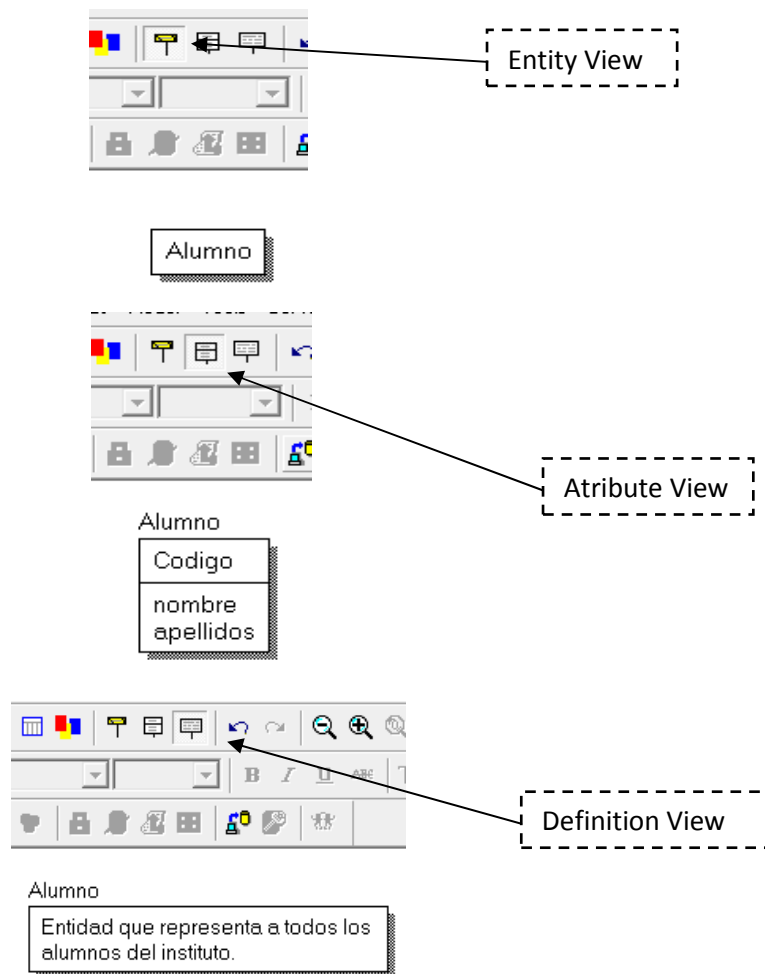
BARRA DE HERRAMIENTAS



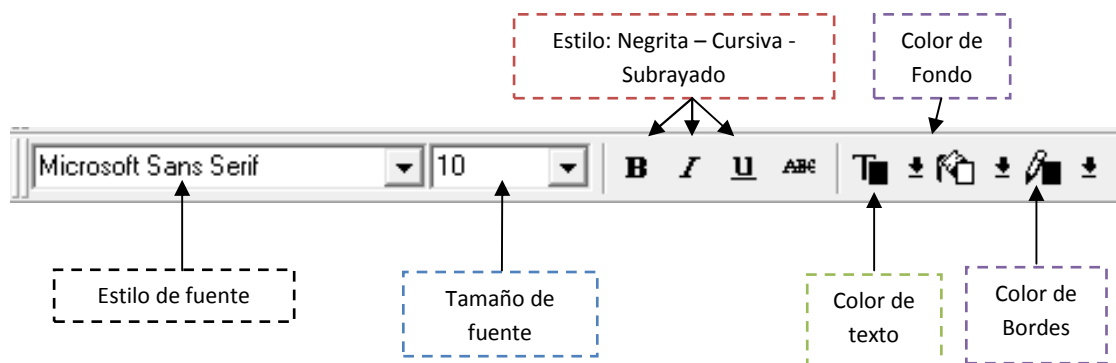
VISTAS EN NIVELES

Erwin muestra 3 niveles para la presentación de los modelos, estos representan a los 3 tipos de modelos: Conceptual, lógico y físico.

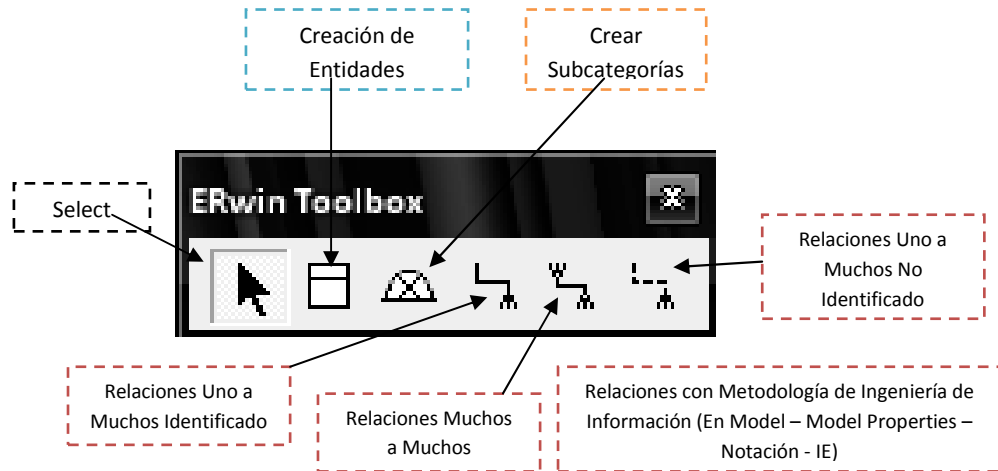
- Entity View: Muestra el modelo en forma de entidades, como un modelo conceptual.
- Attribute View: Aparece activado por defecto, muestra el modelo con todos los atributos ingresados, es la vista de un modelo lógico y físico.
- Definition View: Muestra las entidades con sus respectivas definiciones.



BARRA DE FORMATO



EL TOOLBOX

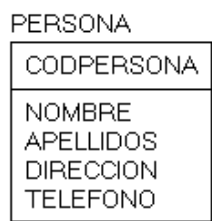
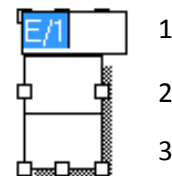
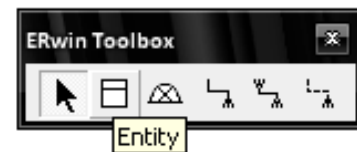


✓ CREACION DEL DIAGRAMA ENTIDAD RELACION (MER)

Para crear una entidad utilizamos la herramienta Entity la cual tiene 3 zonas bien marcadas:

1. Zona de Nombre de entidad.
2. Cabecera o zona de claves principales.
3. Zona de atributos.

Para pasar de una zona a otra lo hacemos con la tecla TAB, para agregar más elementos, como por ejemplo atributos, le damos Enter.



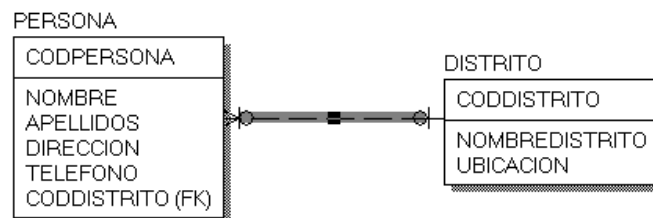
Creamos una nueva entidad con la herramienta Entity:



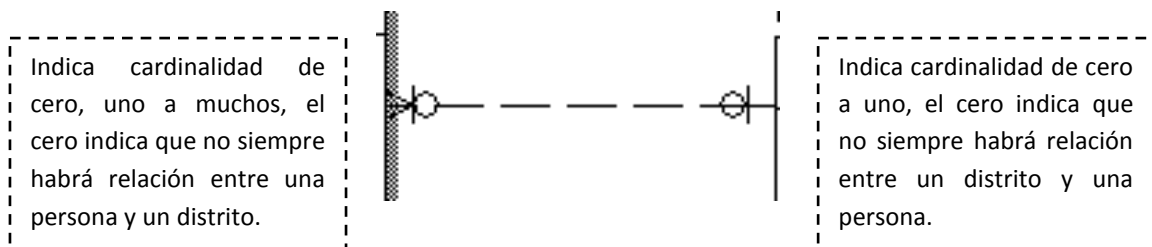
Ahora debemos relacionar ambas entidades mediante las herramientas de relaciones, debemos tener en cuenta además el tipo de relación que utilizaremos, las identificadas, no identificadas o las de muchos a muchos...

Analicemos, una persona reside en un distrito, y en un distrito... ¿Cuántas personas residen?, la respuesta es varias persona residen en un distrito, entonces es una relación de Uno a Muchos, asimismo, ¿es obligatorio conocer el nombre o dirección de una de las personas residentes para, por ejemplo, conocer la ubicación de alguno de los distritos? Entonces nos encontramos ante una relación de tipo no identificada.

Otra pregunta que debemos hacernos es: ¿Qué entidad es la que depende de la otra?, podemos decir que sin distritos las personas no tendrían donde residir, entonces la entidad Distrito es la entidad Fuerte mientras que Persona es la entidad Débil.

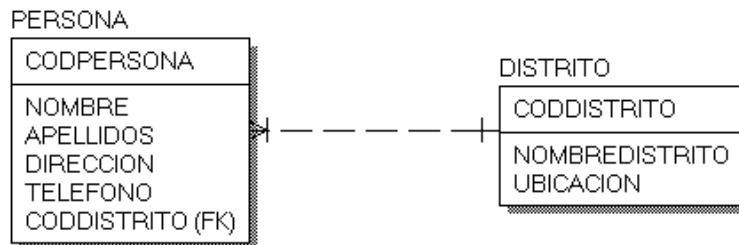


La clave principal de la entidad Distrito es traspasada a la entidad Persona como clave foránea (FK), esto indica el tipo de relación de 1 a M. Analicemos la cardinalidad de la relación:

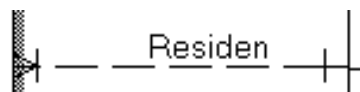


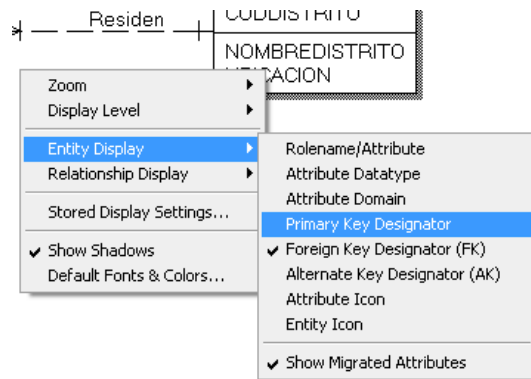
Pero.... ¿Es posible que una persona no resida en ningún distrito?, no verdad, por ello debemos modificar la cardinalidad, para ello le damos clic derecho sobre la línea de relación y seleccionamos la opción Relationship Properties..., o simplemente hacemos doble clic.

El área Cardinality indica la cantidad de elementos que pueden tener relacionados los elementos de la entidad fuerte, mientras que los valores nulos (Nulls) indica la cantidad de elementos relacionados de la entidad débil hacia la entidad fuerte. Le decimos One or More y No nulls.

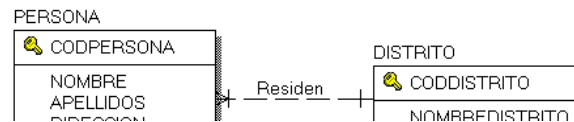


Para colocar la frase verbal, en la ventana de propiedades de relación debemos colocar la frase en el área Verb Phrase, de Padre a Hijo (Parent to child). En el otro recuadro no colocamos nada (A elección del diseñador). Para poder ver la frase verbal damos clic derecho sobre un espacio vacío del modelo, seleccionamos Relationship display y elegimos Verb Phrase.

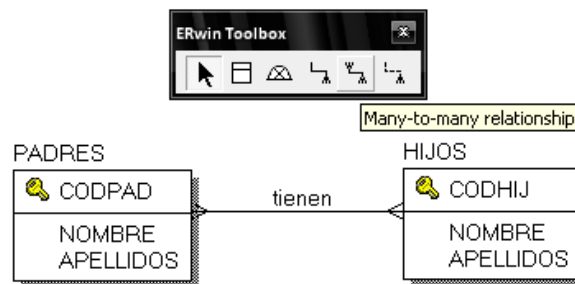




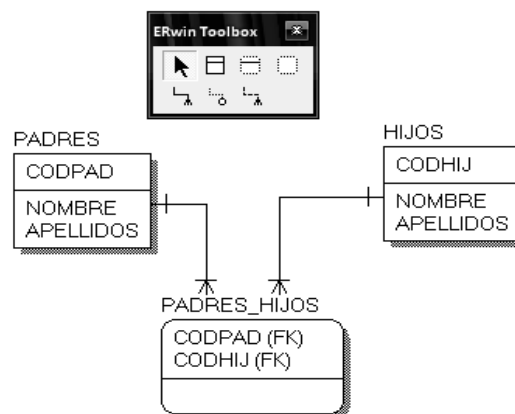
Para colocar un símbolo de llave como identificador de clave primaria a las entidades, hacemos clic derecho sobre un espacio vacío del modelo y seleccionamos Entity display, ahí seleccionamos la opción Primary Key Designator. Con esto se verá...



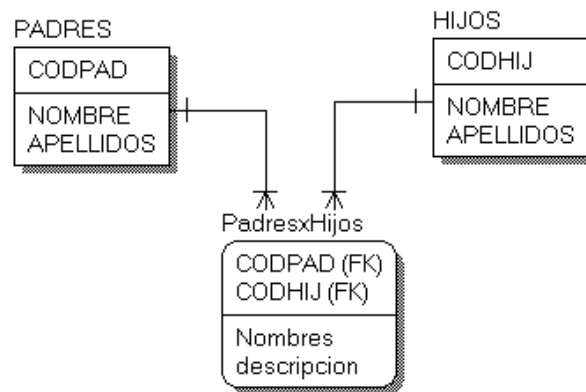
Veamos otro caso, ¿Cuál es la cardinalidad en la relación entre una entidad Padres y una entidad hijos? Una persona tiene máximo dos padres, mientras que los padres pueden tener muchos hijos, entonces nos encontramos ante una relación de Muchos a Muchos. Utilizamos la herramienta Many to many relationship...



En este caso no existe traspaso de claves primarias, ya que esto indica que ambas entidades son Fuertes, debiendo crearse una entidad asociada, esto en el modelo Físico. Erwin lo agregará automáticamente siempre y cuando la hayamos previamente activado en la ventana Model Properties del menú Model (explicado anteriormente).



Con esto sólo nos queda agregar algunos atributos que creamos convenientes, así también podemos modificar el nombre de la entidad...



En este caso todas las relaciones son identificadas, la entidad Asociada hereda las claves primarias de las tablas que la generan, conformando una clave compuesta por las dos claves foráneas, entre ambas claves forman la clave principal de la entidad resultante.

✓ **EJERCICIOS**

Relacionar las siguientes entidades indicando Claves, Atributos, Tipo de Relación y cardinalidad.

1. Cliente – Boleta
2. Autobús – Paradero
3. Vendedor – Artículo
4. Empleado – Área
5. Persona – Estado Civil
6. Cliente – Pedido
7. Alumno – Curso
8. Producto – Factura

Semana

3

Contenido:

- ✓ Generalización
- ✓ Agregación
- ✓ Construcción diagramas Entidad Relación a partir de casos.
- ✓ Tipos y Representación de entidades
- ✓ Tipos y Representación de relaciones.
- ✓ Relación recursiva
- ✓ Introducción a ERWIN 7.1
- ✓ Creación de Modelos Lógicos en Erwin
- ✓ ToolBox ERWIN
- ✓ Entidades - tipos
- ✓ Relaciones - tipos
- ✓ Práctica Calificada 01 - Teoría
- ✓ Conceptos de BD
- ✓ Construcción de un MER a partir de un caso.

TALLER MER EN ERWIN

➤ **ABSTRACCION DE DATOS (Generalización – Agregación)**

✓ **CONCEPTO DE ABSTRACCIÓN DE DATOS**

La abstracción de algo posee dos cualidades: suprime los detalles irrelevantes y busca para aislar la esencia de ese algo. Por ejemplo, para el tipo de dato *integer* sólo usamos su definición y sus operaciones, pero no sabemos cómo está implementado; por lo tanto es un tipo de dato abstracto.

Los datos representan una abstracción de la realidad ya que algunas características y propiedades de los objetos reales son ignoradas, porque son irrelevantes a un problema en particular.

Abstracción: simplificación de los hechos.

Ejemplo: Archivo de empleados (sueldos).

Datos útiles: nombre, nómina, sueldo, etc.

Datos no útiles: color pelo, peso, etc.

Depende de la naturaleza del problema para seleccionar el conjunto de datos necesarios para resolverlo. Al manejar un lenguaje de alto nivel no es necesario conocer la estructura interna de un arreglo, registro, sólo el cómo utilizarlo. La abstracción de datos proporciona un poderoso mecanismo para escribir programa bien estructurado y fácil de modificar. Los detalles internos de la representación y manipulación de datos puede modificarse a voluntad, suponiendo que las interfaces de los procedimientos de manipulación permanezcan iguales, los otros componentes del programa no serán afectados por la modificación, excepto por las modificaciones en las características del funcionamiento y límites de capacidad.

✓ TIPOS DE ABSTRACCION

CLASIFICACIÓN:

Define un concepto como una clase de objetos de la realidad con propiedades comunes (ES_MIEMBRO_DE)

- Árbol de un nivel que tiene como raíz la clase y como hoja los elementos de la clase
- Cada elemento hoja es miembro de cada elemento de la raíz
- Cada elemento puede ser miembro de varias clases

Una forma de abstracción en la que una colección de objetos se considera como una clase de objetos de más alto

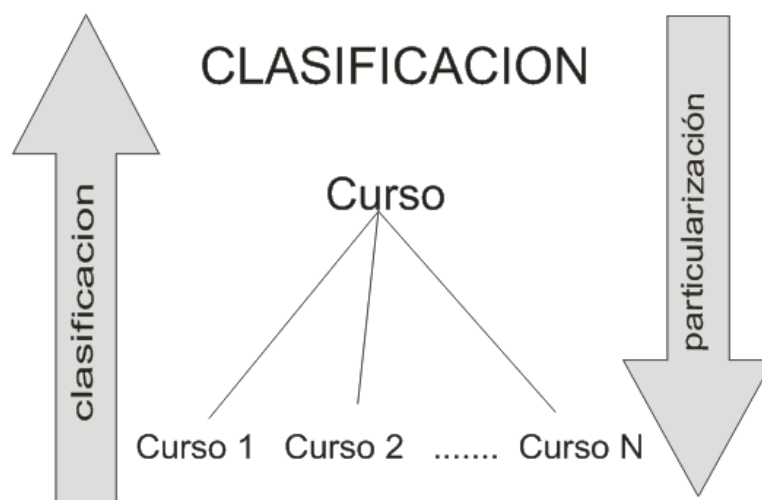
Una clase de objetos es una caracterización precisa de todas las propiedades compartidas por todos los objetos en la colección. Un objeto es un ejemplar de una clase de objetos si tiene las propiedades definidas en la clase Brodie (1984)

– Enero, Febrero, etc. à MES

– Tienen distinta cantidad de días. Características similares (correspondencia a la clase) con valores concretos (en los ejemplares)

Clasificaciones diferenciadas a mismos objetos

– Ejemplo: Materias: Teóricas, Aplicadas Anuales, Cuatrimestrales, Mensuales, etc.



AGREGACIÓN:

Define una clase nueva a partir de otras que representan sus partes componentes
(**ES_PARTE_DE**)

Construir un nuevo elemento del modelo como **compuesto** de otros elementos (**componentes**)

- Inverso: Desagregación
- Se establece una relación **ES_PARTE_DE** entre los elementos componentes y el elemento compuesto

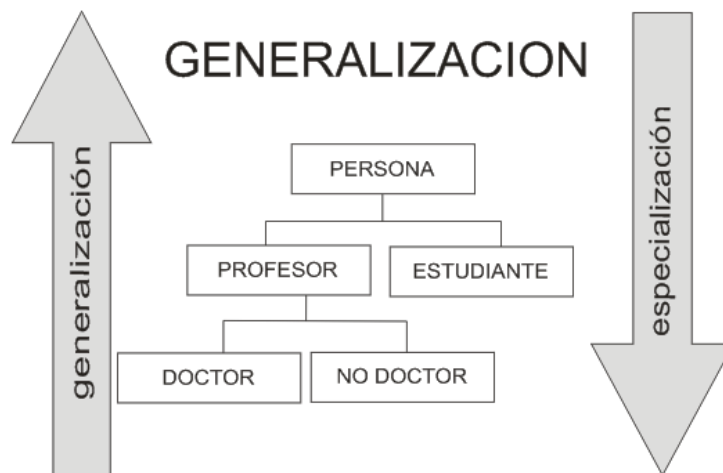
GENERALIZACIÓN:

Define una relación de subconjunto entre los elementos de 2 o + clases (**ES_UN**)

- Se representa con un árbol de un nivel, en el que todos los nodos son clases, con la clase genérica como raíz y las clases subconjuntos como hojas.
- Todas las abstracciones definidas para la clase genérica, son heredadas por la clase subconjunto

Acción de abstraer las características comunes a varias clases (**subclases**) para constituir una clase más general (**superclase**) que las comprenda.

- El conjunto de ejemplares de una subclase "es un" subconjunto de los ejemplares de la correspondiente superclase.
- Entre los elementos subclase y el elemento superclase se establece una relación del tipo **ES_UN**.
- Inverso: Especialización



ASOCIACION

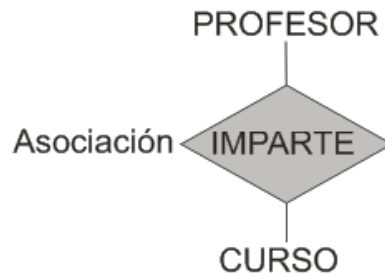
Se utiliza para relacionar dos o más clases (y, por tanto sus ejemplares), creándose un elemento de un tipo distinto.

- Inverso: Disociación
- En algunos MD no aparece esta abstracción como tal, no existiendo ningún concepto especial para representarla (p.e. Relacional).

No considerado o incluido en agregación pero tiene características especiales, cuando se asocian dos o más categorías, el nuevo elemento que aparece tiene determinadas características que lo distinguen de las categorías normales, por lo que, en general, los modelos de datos crean un nuevo concepto para representarlo.

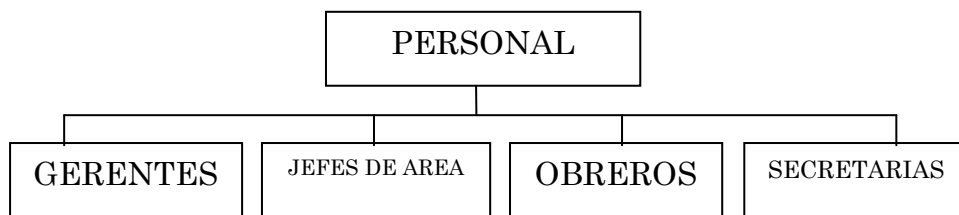
- El nuevo elemento no *está compuesto*, como en el caso de la agregación, por los elementos que asocia.
- En la agregación puede existir herencia, y no así en la asociación.

ASOCIACION



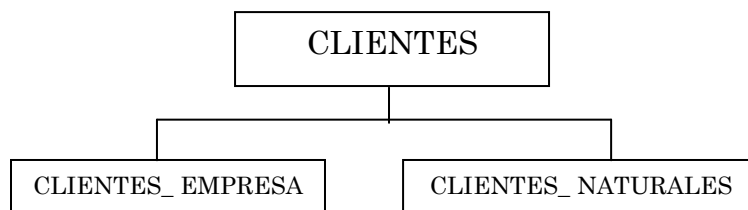
EJEMPLOS DE ABSTRACCION DE DATOS

Generalización



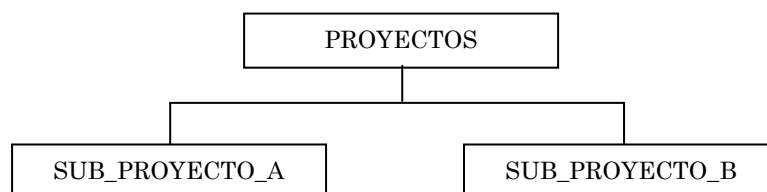
Si tenemos en nuestro caso el requerimiento de mantener información de todo el personal de la empresa, entonces en vez de crear entidades para cada tipo de empleado, lo recomendable es generalizarlos a todos en una sola entidad, e identificados según un código que diferencie el tipo de cargo que poseen, además se ahorra espacio y se evita redundancia de datos. Se cumple la relación: Es_Un... Los gerentes son un personal de la empresa.

Clasificación



Para tener un mejor control de nuestros clientes, podemos clasificarlos según el tipo de cliente, entonces se cumple la definición ES_Miembro_de....

Agregación



Digamos que un proyecto puede estar compuesto de varios subproyectos que definen el cumplimiento del proyecto principal, entonces se cumple la definición Es_Parte_De...

Asociación



En este tipo de Abstracción, podemos decir que entre dos entidades no siempre existe relación directa, sino que están asociados indirectamente mediante una relación que define su interconexión, en este caso se crea una entidad asociada que defina la relación. En este caso una entidad Matricúlas.

➤ CONSTRUIR UN DER A PARTIR DE UN CASO DE ESTUDIO

Vamos a trabajar en un sencillo caso de estudio, en la cual iremos creando el Diagrama Entidad Relación hasta culminar con un Modelo Entidad Relación representado en la herramienta de diseño Case Erwin.

CASO DE ESTUDIO:

En una concesionaria de automóviles, se desea automatizar los procesos de ventas, en sí lo que se desea conocer es a quienes se vende, qué vehículos son vendidos y quiénes son los vendedores que realizan las ventas.

- ✓ Cada cliente puede solicitar la adquisición de un vehículo, por cada vez, se le entrega un comprobante de pago por medio del representante de área de ventas, indicando los datos más importantes como nombre, DNI, dirección, situación legal, etc.
- ✓ Cada automóvil pertenece a un modelo en especial, debemos clasificar los modelos de autos disponibles como deportivos, elegantes, etc.
- ✓ Trabajamos con distintas marcas de automóviles (Entre las que tenemos Hyundai, Toyota y Nissan). Cada marca tiene distintos modelos.
- ✓ Los autos pueden ser deportivos, carreteras y camionetas.
- ✓ Los vendedores son los que realizan todo el proceso de venta.

ENTIDADES:

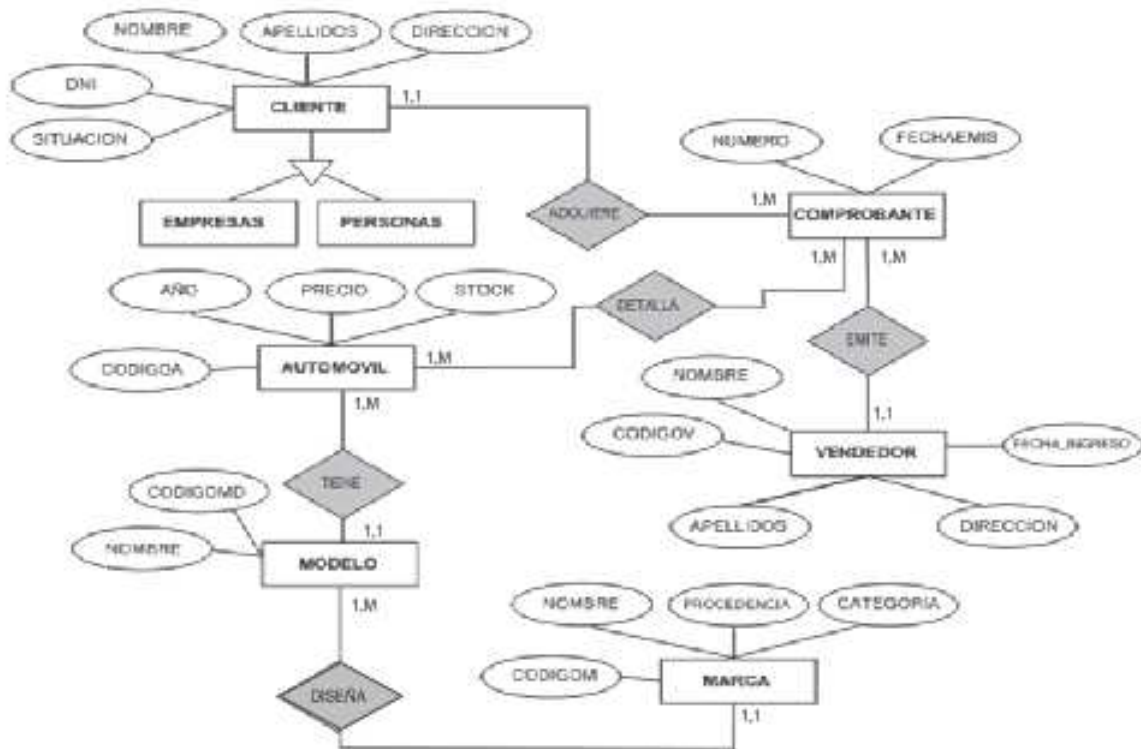
Debemos comenzar con identificar las entidades de nuestro caso.

- ✓ Cliente: La persona que hace la compra de automóviles Pueden ser pernas naturales o empresas.
- ✓ Vendedor: La persona que atiende y realiza la venta al cliente.
- ✓ Automóvil: El objeto de la transacción.
- ✓ Marca: Los distintos fabricantes de automóviles con los que trabajamos.
- ✓ Modelo: Los distintos modelos que ofrecemos al público.
- ✓ Comprobante de pago: El documento que se otorga al cliente luego de la cancelación.

RELACIONES:

- ✓ El cliente adquiere un vehículo, por medio del comprobante de pago podemos conocer al cliente que pagó por el automóvil.
- ✓ El vendedor atiende el proceso de la venta, él será quien emita el comprobante de pago final.
- ✓ Una marca fabricante tiene distintos modelos disponibles en catálogo.
- ✓ Un modelo pertenece a una marca en especial.

REALIZAMOS EL DIAGRAMA ENTIDAD RELACION



Esto es el Modelo Conceptual representado en un Diagrama Entidad Relación, lo que sigue ahora es realizar el Modelo Lógico de datos representado por un Modelo Entidad Relación.

El MER a un nivel lógico, comprende otros elementos asociados a las entidades y sus interrelaciones:

- ✓ Atributos
- ✓ Claves o Llaves
- ✓ Tipos de relaciones
- ✓ Cardinalidad

➤ MODELO LOGICO

✓ MODELO ENTIDAD RELACION (MER)

Los modelos lógicos basados en objetos se usan para describir datos en el nivel conceptual y el externo. Se caracterizan porque proporcionan capacidad de estructuración bastante flexible y permiten especificar restricciones de datos. Los modelos más conocidos son el modelo entidad-relación y el orientado a objetos.

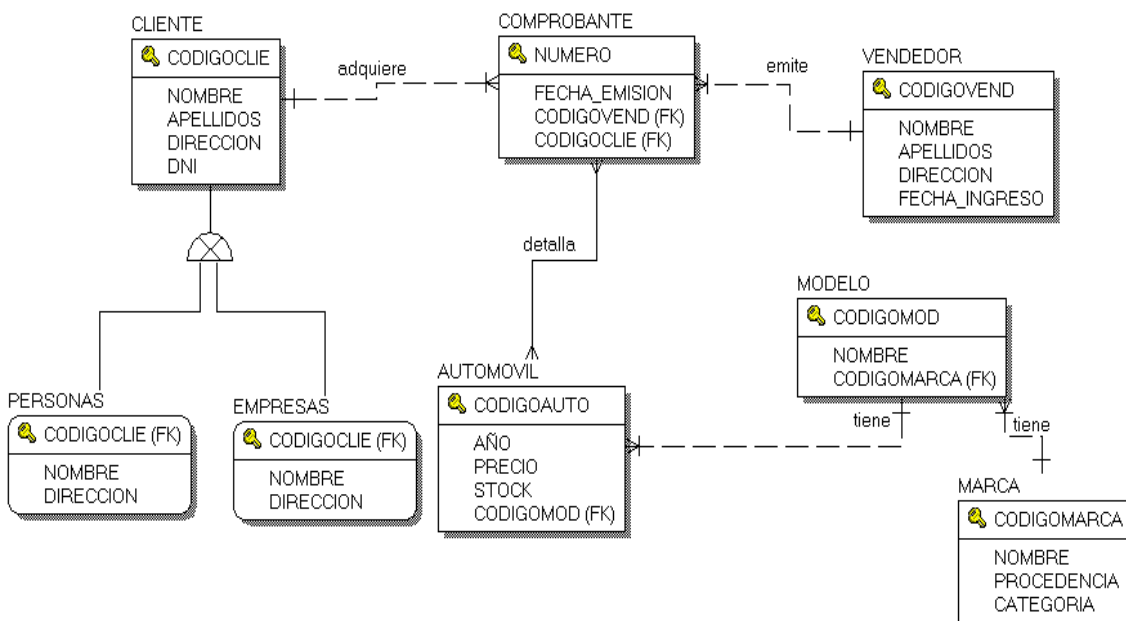
Actualmente, el más utilizado es el modelo entidad-relación, aunque el modelo orientado a objetos incluye muchos conceptos del anterior, y poco a poco está ganando mercado. La mayoría de las BBDD relacionales añaden extensiones para poder ser relacionales-orientadas a objetos.

CREACIÓN DEL MODELO LÓGICO DEL CASO ANTERIOR

Lo primero que debemos hacer es identificar los atributos de las entidades, colocándoles los atributos que serán las claves primarias.

- ✓ Cliente: Códigoclie, nombre apellidos, dirección, DNI, situación.
- ✓ Vendedor: Codigovend, nombre, apellidos, dirección, fecha_ingreso.
- ✓ Automóvil: Códigoauto, año, precio, stock.
- ✓ Comprobante: numero, fecha_emisión.
- ✓ Modelo: Codigomod, descripción.
- ✓ Marca: Codigomarca, nombre, procedencia, categoría.

El Modelo Entidad Relación...



En este modelo todos los datos son almacenados en relaciones, y como cada relación es un conjunto de datos, el orden en el que estos se almacenen no tiene relevancia (a diferencia de otros modelos como el jerárquico y el de red). Esto tiene la considerable ventaja de que es más fácil de entender y de utilizar por un usuario no experto. La información puede ser recuperada o almacenada por medio de consultas que ofrecen una amplia flexibilidad y poder para administrar la información.

Este modelo considera la base de datos como una colección de relaciones. De manera simple, una relación representa una tabla que no es más que un conjunto de filas, cada fila es un conjunto de campos y cada campo representa un valor que interpretado describe el mundo real. Cada fila también se puede denominar tupla o registro y a cada columna también se le puede llamar campo o atributo.

Para manipular la información utilizamos un lenguaje relacional, actualmente se cuenta con dos lenguajes formales el Álgebra relacional y el Cálculo relacional. El Álgebra relacional permite describir la forma de realizar una consulta, en cambio, el Cálculo relacional sólo indica lo que se desea devolver.

El lenguaje más común para construir las consultas a bases de datos relacionales es SQL, Structured Query Language o Lenguaje Estructurado de Consultas, un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

El Modelo Lógico está muy orientado a registros, proporciona una vista más cercana a la estructura de la base de datos, que vendría a ser el modelo Físico de datos.

✓ **METODOLOGÍA DE DISEÑO DE BASES DE DATOS**

El diseño de una base de datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos subproblemas independientemente, utilizando técnicas específicas. Así, el diseño de una base de datos se descompone en diseño conceptual, diseño lógico y diseño físico.

El diseño conceptual parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la base de datos. Un esquema conceptual es una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD que se vaya a utilizar para manipularla. Un modelo conceptual es un lenguaje que se utiliza para describir esquemas conceptuales. El objetivo del diseño conceptual es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar esta información.

El diseño lógico parte del esquema conceptual y da como resultado un esquema lógico. Un esquema lógico es una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. Un modelo lógico es un lenguaje usado para especificar esquemas lógicos (modelo relacional, modelo de red, etc.). El diseño lógico depende del tipo de SGBD que se vaya a utilizar, no depende del producto concreto.

➤ **MODELO FÍSICO**

El diseño físico parte del esquema lógico y da como resultado un esquema físico. Un esquema físico es una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Por ello, el diseño físico depende del SGBD concreto y el esquema físico se expresa mediante su lenguaje de definición de datos.

En el modelo físico de datos, las entidades son consideradas tablas, los atributos vienen a ser los campos de las tablas, los elementos como registros o datos de las tablas, además agregamos el término de dominio de datos, es aquí donde se resuelve la relación

✓ **DOMINIOS**

Se refiere al conjunto de valores posibles que puede tener un atributo o campo o grupo de atributos o campos de una entidad o tabla. Cada atributo está asignado a uno de cuatro dominios básicos o primitivos, que son:

- Caracter o texto - String.
- Date o Fecha - Datetime
- Numero o Integer - Number
- Lógico - Blob

Los dominios primitivos son la base para formar otros dominios más complejos definidos por el usuario, dependerá también del gestor de base de datos (plataforma) sobre la cual se apoyará la data.

✓ **EXTENSION**

Indica el número máximo de caracteres o dígitos para cada uno de sus valores, podemos considerar que esto va a ser un subconjunto del dominio de un atributo, dado que el número de caracteres o dígitos restringe el conjunto posible de valores para el atributo o campo.

✓ **VALORES PERMITIDOS**

El conjunto de valores permitidos para un atributo describe exhaustivamente los valores potenciales de un atributo, es decir, un conjunto de valores posibles que el atributo soportará como data. Por ejemplo:

Unidad_venta= TM (tonelada métrica), RO (rollo), BO (bolsa), PQ (paquete)

✓ **VALOR A ALGORITMO POR OMISION**

Para cada atributo que pueda contener valores permitidos se puede especificar un algoritmo por omisión o bien un valor por omisión (pero no ambos). Por ejemplo:

Codigo_cliente = Codigo_cliente + 1

✓ **ALGORITMO POR DERIVACION**

Solamente podemos especificar algoritmos de derivación para atributos derivados. En la práctica el diseñador debe tomar la decisión sobre si un atributo derivado debe ser calculado o almacenado por memoria. Por ejemplo:

Total_ventaitem = valor_venta + igv

Total_venta = total_ventaitem

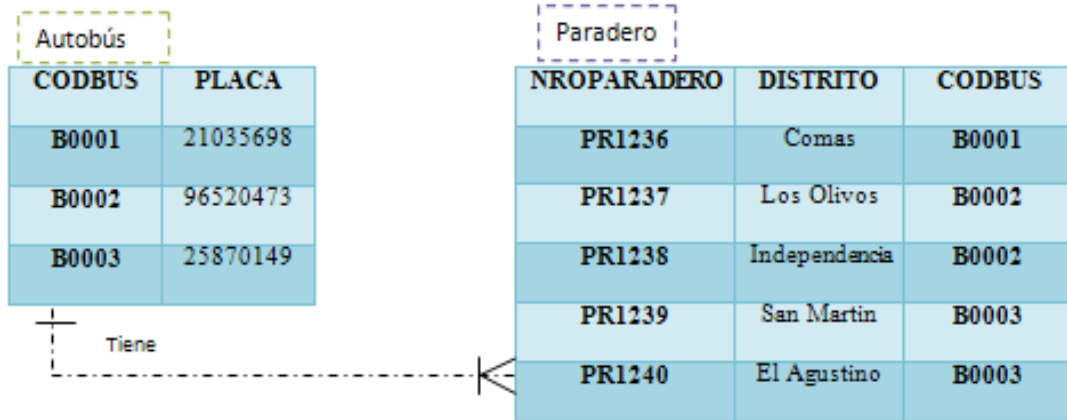
Una entidad en el modelo físico tiene esta forma:

CODIGO_CLIE	NOMBRE	DIRECCION	TELEFONO
C0001	AUGUSTO	SAN JUAN	2567348
C0002	JORGE LUIS	LINCE	9453889

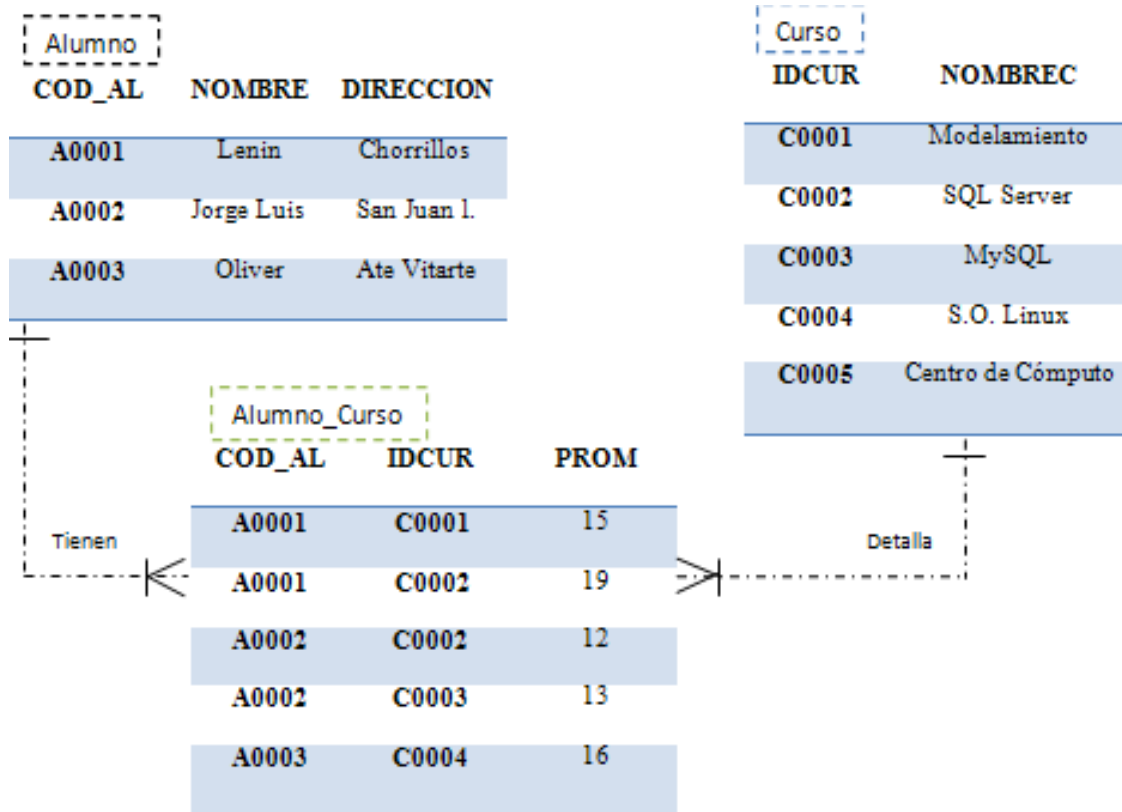
Esta entidad nos representa una tabla de nuestra base de datos, con los dominios para cada atributo o columna (campo), por defecto aparecen todos como Char (caracter), para asignar un tipo de dato (dominio) a cada uno se utiliza las opciones de la herramienta case Erwin, más adelante veremos cómo hacerlo. La tabla con los datos (valores) correspondientes sería de la siguiente manera:

EJERCICIOS

1. Autobús – Paradero



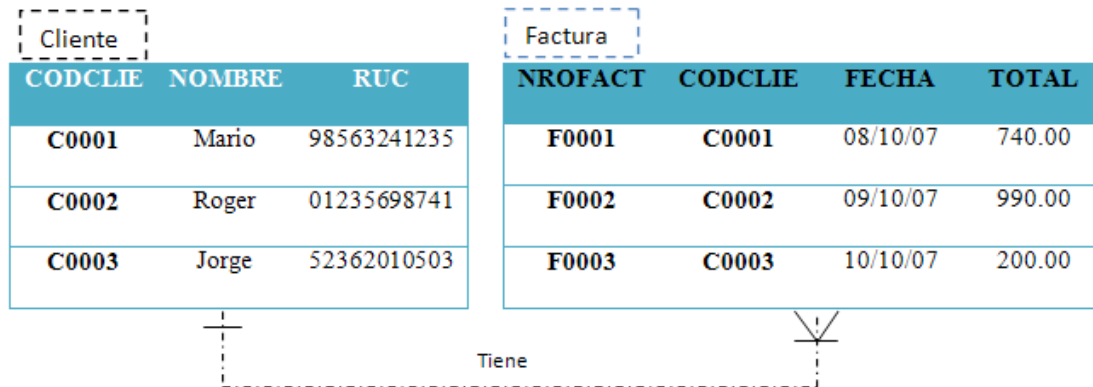
2. Alumno - Curso



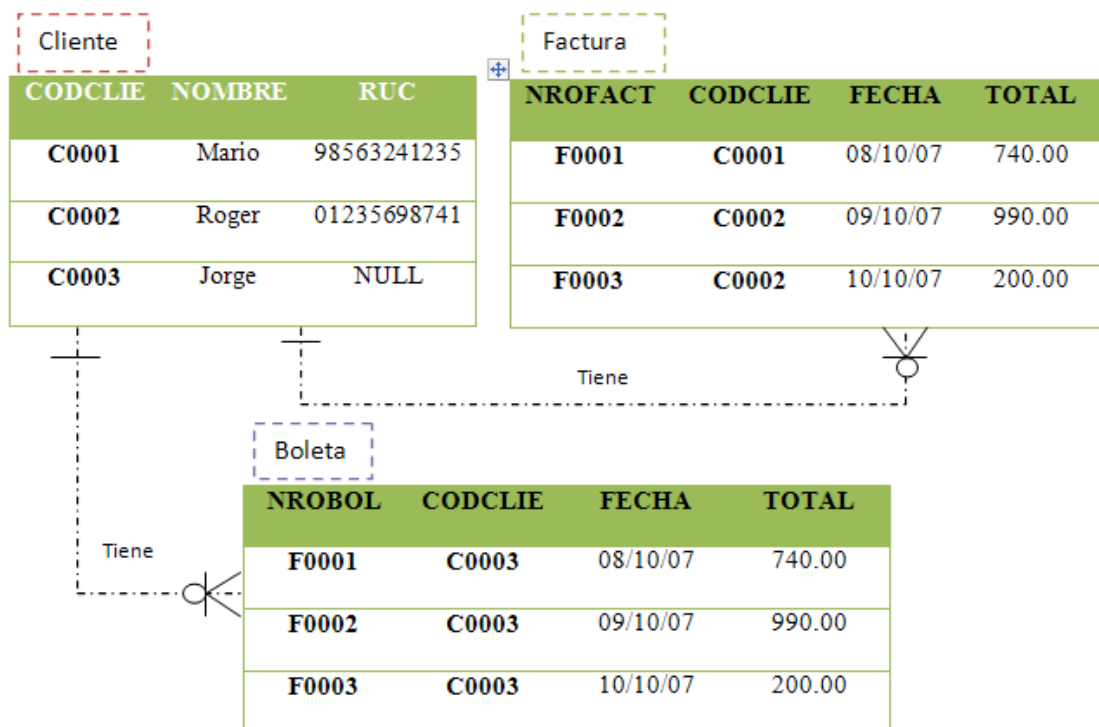
EJERCICIOS

Interpretar los siguientes casos.

Se tiene como política de ventas de nuestra tienda, vender sólo a clientes jurídicos, es decir a aquellos que cuentan con número de ruc, se les facturará por cada compra que los clientes nos realicen.



El caso anterior muestra una relación Uno a Muchos entre Cliente y Factura ya que la regla del negocio así lo dispone, ahora veamos el caso de una tienda que venda no solo a clientes con ruc sino también a personas naturales (no jurídicos), a estos se les factura con boletas de venta.



➤ **PRACTICA CALIFICADA 01**

Temas:

1. Defina los siguientes conceptos de Modelamiento:

- a) Entidad
- b) Relaciones
- c) DER
- d) MER
- e) Generalización

2. Concepto y descripción de los tipos de atributos existente. Colocar ejemplos por cada uno de los tipos de atributos.

3. Niveles de Datos (Arquitectura ANSI). Explicar cada uno.

4. Relaciones colocando cardinalidad las siguientes entidades. Por ejemplo:

a) Cliente – Pedido

b) Personal – Área

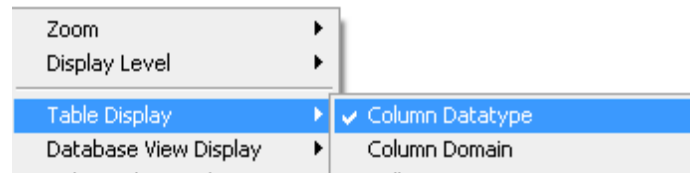
5. Crear el modelo Entidad – Relación para un caso de estudio propuesto por el profesor, indicar:

- Entidades.
- Atributos.
- Relaciones.
- Cardinalidad.
- Claves.

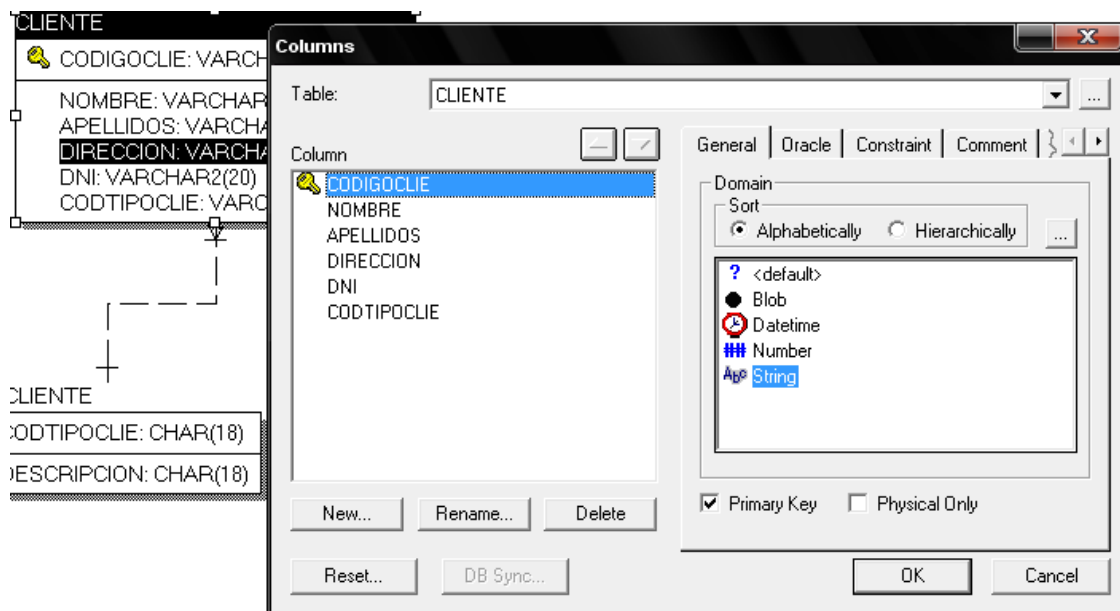
LABORATORIO # 3

✓ CREANDO EL MODELO FÍSICO EN ERWIN

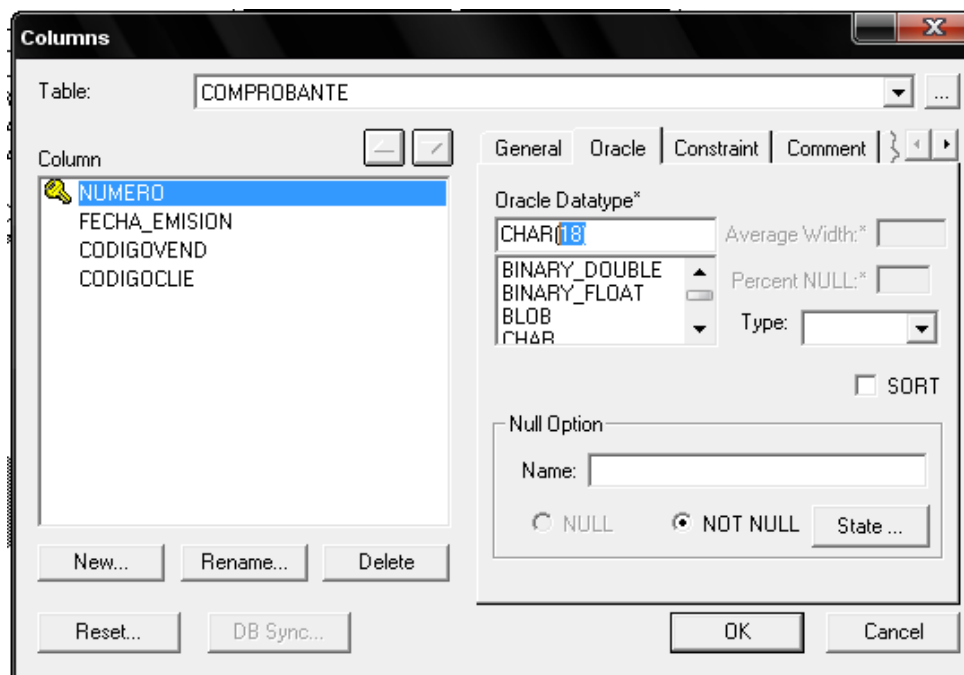
Para ello debemos editar los tipos de datos soportados para cada columna de cada tabla, para ello hacemos clic derecho sobre un espacio vacío y seleccionamos la opción Table Display, ahí elegimos Column Datatype.



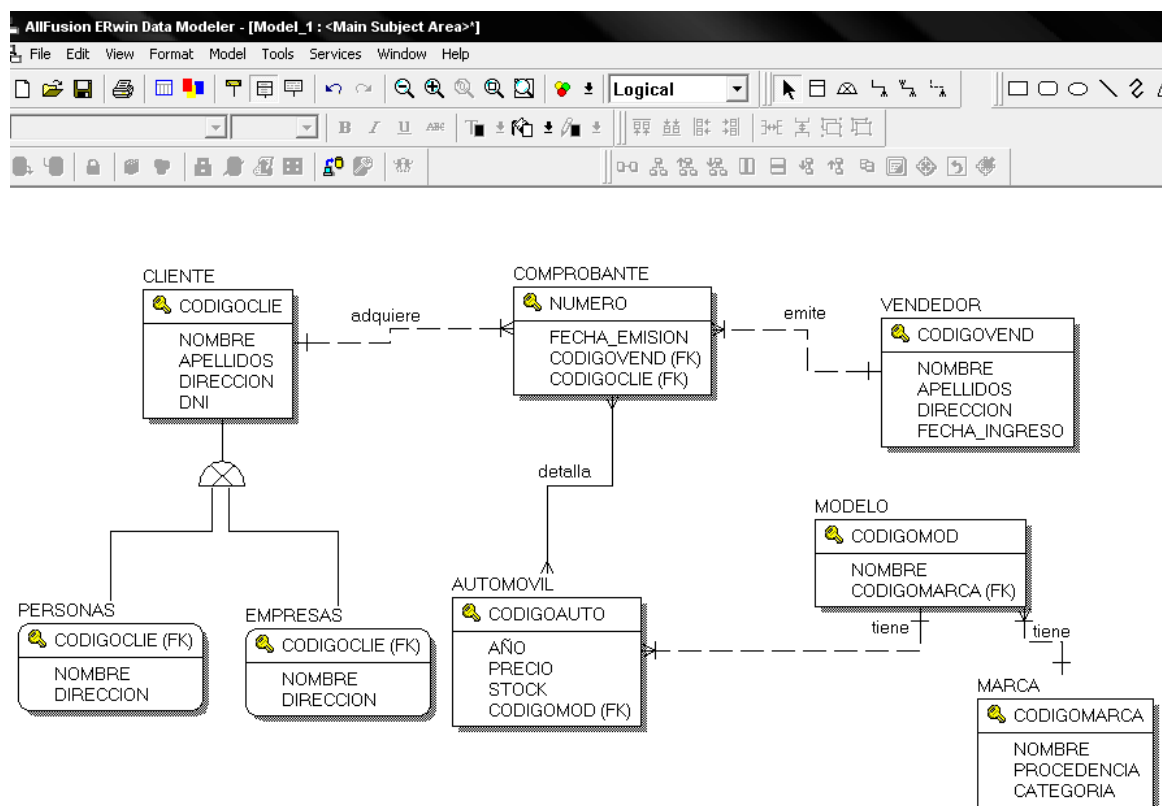
Luego seleccionamos una tabla y hacemos clic derecho, ahí seleccionamos la opción Column con la cual aparecerá la ventana para editar los tipos de datos...



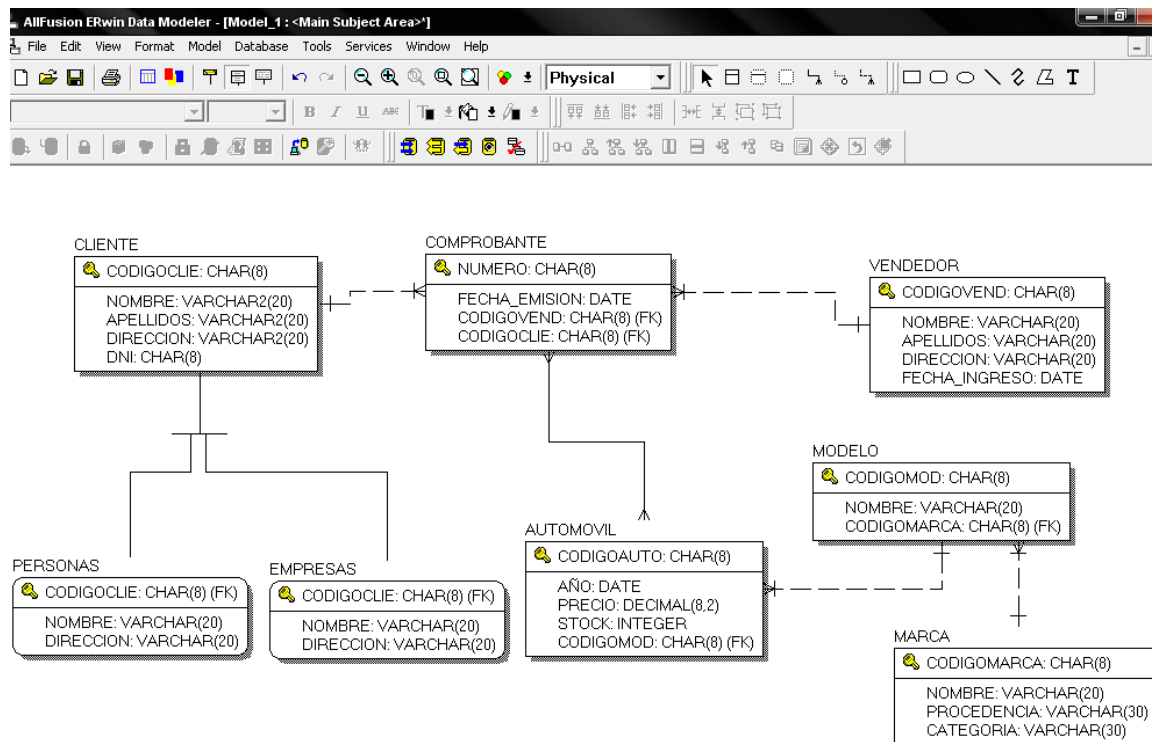
Podemos seleccionar algunos de los tipos de datos estándar (primitivos), o tomar el motor de Base de datos para asignar dominios y extensiones, en la imagen se muestra la pestaña Oracle.



En la siguiente imagen se muestra el modelo Lógico del caso anterior (La concesionaria de automóviles).

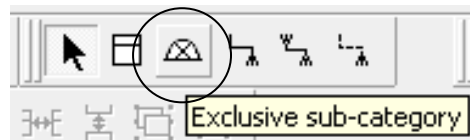


En la siguiente imagen se muestra el modelo Físico del caso anterior (La concesionaria de automóviles), con todos los tipos de datos indicados.

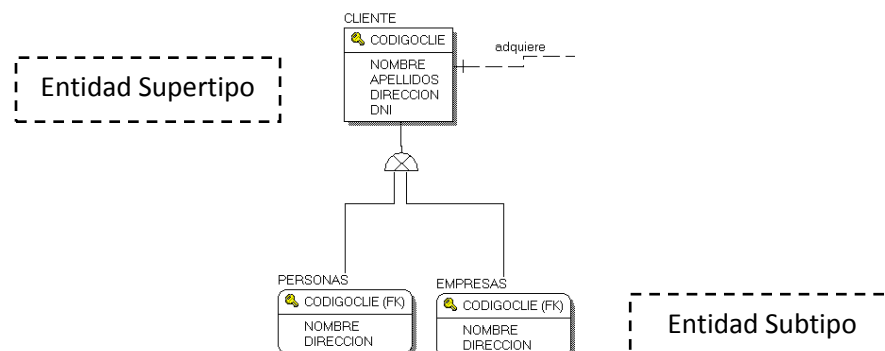


✓ CREAR UNA GENERALIZACION JERARQUICA (EN EL MODELO LOGICO)

1. Seleccione el icono de categoría desde el Toolbox.



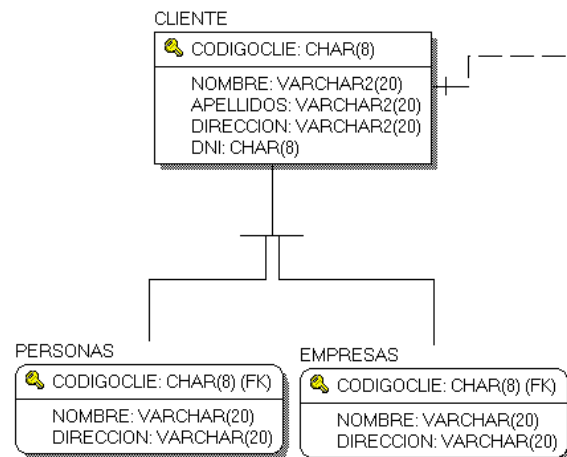
2. Seleccione la entidad Supertipo.
3. Seleccione la entidad Subtipo.
4. Para agregar más entidades hacemos clic en el símbolo de categoría y luego clic en la entidad Subtipo.



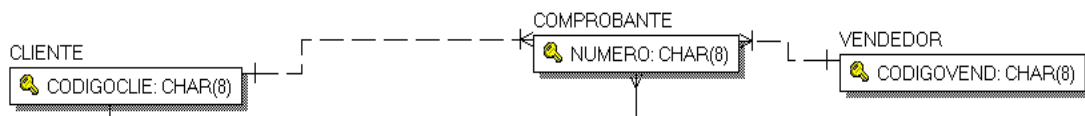
Esto indica que las entidades Subtipos son parte de la entidad Supertipo, en este caso, un cliente puede ser personas naturales o empresas, esto puede indicar distinto trato en las ventas o la aplicación de promociones y/o beneficios dependiendo del tipo de cliente.

EN EL MODELO FÍSICO

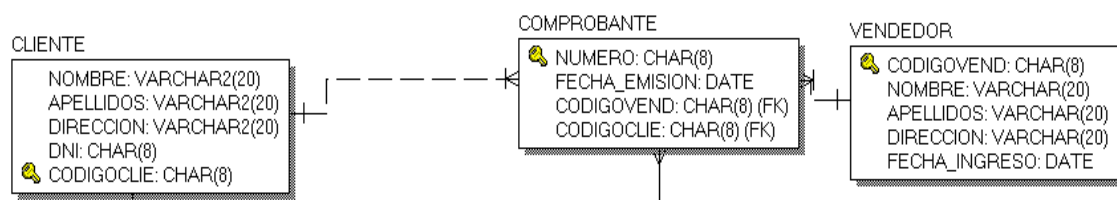
La Generalización se muestra así:



Podemos ver el modelo físico en vista de claves, haciendo clic derecho y seleccionando la opción Table Display, y ahí elegimos Primary key.



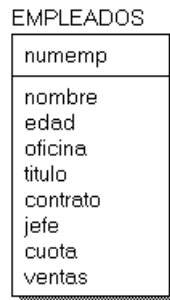
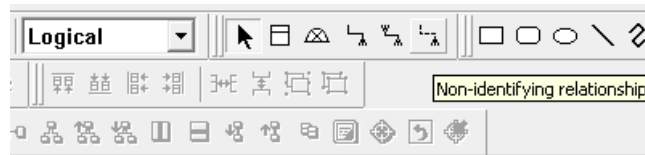
Podemos verlo también en vista de columnas, para ello seleccionamos la opción Physical order también desde Table Display.



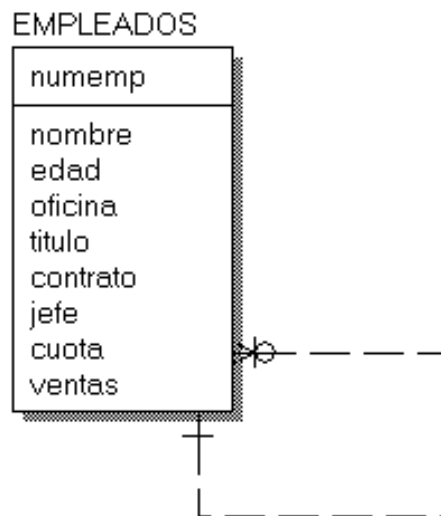
Tanto el Modelo Lógico como el Modelo Físico tienen sus propias opciones para la optimización de los modelos.

✓ CREANDO UNA RELACION RECURSIVA

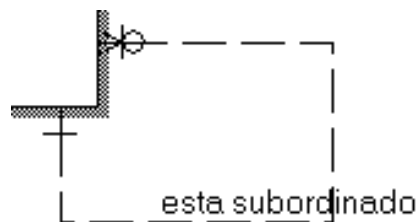
Creamos una entidad en el Modelo Lógico.



Hacemos clic en el icono de la relación Uno a Muchos no identificada, luego hacemos dos veces clic sobre la tabla (no doble clic), con esto tendremos la relación recursiva.



Con la opción Relationship Properties podemos editar la cardinalidad de la relación, así como la frase verbal.



Semana 4

CONTENIDO:

- ✓ Conceptos.
- ✓ Datos Atómicos
- ✓ Dependencia Funcional.
- ✓ Tipos
- ✓ Normalización.
- ✓ Forma Normal 1
- ✓ Forma Normal 2
- ✓ Forma Normal 3
- ✓ Herramienta Case ERWIN 7.1
- Modelo Lógico Físico
- Optimización de tipos de datos
- Representación de un DER en un modelo Relacional (ERWIN)
- Ingeniería Directa / Reversa
- ✓ Ejercicios.

MODELO RELACIONAL

El modelo relacional para la gestión de una base de datos es un modelo de datos basado en la lógica de predicado y en la teoría de conjuntos. Es el modelo más utilizado en la actualidad para modelar problemas reales y administrar datos dinámicamente. Tras ser postuladas sus bases en 1970 por Edgar Frank Codd, de los laboratorios IBM en San José (California), no tardó en consolidarse como un nuevo paradigma en los modelos de base de datos.

Su idea fundamental es el uso de «relaciones». Estas relaciones podrían considerarse en forma lógica como conjuntos de datos llamados «tuplas». Pese a que ésta es la teoría de las bases de datos relacionales creadas por Edgar Frank Codd, la mayoría de las veces se conceptualiza de una manera más fácil de imaginar, esto es, pensando en cada relación como si fuese una tabla que está compuesta por registros (cada fila de la tabla sería un registro o tupla), y columnas (también llamadas campos).

➤ CONCEPTOS

✓ DATOS ATOMICOS

Las Bases de Datos relacionales tienen en la estructura de sus tablas en realidad, datos atómicos (es así como debe de ser). Un dato atómico es aquel que no puede descomponerse en dos o más datos simples, es decir, son indivisibles en sus valores. Los datos atómicos son opuestos a los multivaluados, que pueden ser descompuestos en otros tipos de datos no atómicos. Un atributo multivaluado tiene valores de dominio con características propias (atributos propios).

Veamos un ejemplo, tenemos la siguiente tabla:

Personas (nombre, apellido, fecha_nacimiento, sexo, estado_civil)

Las tuplas en una relación son un conjunto en el sentido matemático del término, es decir una colección no ordenada de elementos diferentes. Para distinguir una tupla de otra, se recurre al concepto de "llave primaria", o sea a un conjunto de atributos que permiten identificar unívocamente una tupla en una relación. Naturalmente, en una relación puede haber más combinaciones de atributos que permitan identificar unívocamente una tupla ("llaves candidatas"), pero entre éstas se elegirá una sola para utilizar como llave primaria. Los atributos de la llave primaria no pueden asumir el valor nulo (que significa un valor no determinado), en tanto que ya no permitirían identificar una tupla concreta en una relación. Esta propiedad de las relaciones y de sus llaves primarias está bajo el nombre de integridad de las entidades (entity integrity).

A menudo, para obtener una llave primaria "económica", es decir compuesta de pocos atributos fácilmente manipulables, se introducen uno o más atributos ficticios, con códigos identificativos unívocos para cada tupla de la relación.

Cada atributo de una relación se caracteriza por un nombre y por un dominio. El dominio indica qué valores pueden ser asumidos por una columna de la relación. A menudo un dominio se define a través de la declaración de un tipo para el atributo (por ejemplo diciendo que es una cadena de diez caracteres), pero también es posible definir dominios más complejos y precisos. Por ejemplo, para el atributo "sexo" de nuestra relación "Personas" podemos definir un dominio por el cual los únicos valores válidos son 'M' y 'F'; o bien por el atributo "fecha_nacimiento" podremos definir un dominio por el que se consideren válidas sólo las fechas de nacimiento después del uno de enero de 1960, si en nuestra base de datos no está previsto que haya personas con fecha de nacimiento anterior a esa. El DBMS se ocupará de controlar que en los atributos de las relaciones se incluyan sólo los valores permitidos por sus dominios. Característica fundamental de los dominios de una base de datos relacional es que sean "atómicos", es decir que los valores contenidos en las columnas no se puedan separar en valores de dominios más simples. Más formalmente se dice que no es posible tener atributos multivalor (multivalued). Por ejemplo, si una característica de las personas en nuestra base de datos fuese la de tener uno o más hijos, no sería posible escribir la relación Personas de la siguiente manera:

Personas (nombre, apellido, fecha_nacimiento, sexo, estado_civil, hijos)

En efecto, el atributo hijos es un atributo no-atómico, bien porque una persona puede tener más de un hijo o porque cada hijo tendrá diferentes características que lo describen. Para representar estas entidades en una base de datos relacional hay que definir dos relaciones:

Personas (*número_persona, nombrepers, apellidopers, fecha_nacimiento, sexo, estado_civil)

Hijos(*número_hijo,número_persona, *nombrehijo, apellidohijo, edad, sexo)

En las relaciones precedentes, los asteriscos (*) indican los atributos que componen sus llaves primarias. Nótese la introducción en la relación Personas del atributo número_persona, a través del cual se asigna a cada persona un identificador numérico unívoco que se usa como llave primaria. Estas relaciones contienen sólo atributos atómicos. Si una persona tiene más de un hijo, éstos se representarán en tuplas diferentes de la relación Hijos. Las diferentes características de los hijos las representan los atributos de la relación Hijos. La unión entre las dos relaciones está constituida por los atributos número_persona que aparecen en ambas relaciones y que permiten que se asigne cada tupla de la relación hijos a una tupla concreta de la relación Personas. Más formalmente se dice que el atributo número_persona de la relación Hijos es una llave externa (foreign key) hacia la relación Personas. Una llave externa es una combinación de atributos de una relación que son, a su vez, una llave primaria para otra relación. Una característica fundamental de los valores presentes en una llave externa es que, a no ser que no sean null, tienen que corresponder a valores existentes en la llave primaria de la relación a la que se refieren. En nuestro ejemplo, esto significa que no puede existir en la relación Hijos una tupla con un valor del atributo número_persona sin que también en la relación Personas exista una tupla con el mismo valor para su llave primaria. Esta propiedad va bajo el nombre de integridad referencial

En realidad una persona puede tener uno o varios hijos, y un por lo tanto un hijo tendrá uno o dos padres, esto sería así...

NRO_PERSONA	NOMBREPERS	APELLIDOPERS	FECHA_NACI	SEXO	EST_CIVIL
PE0001	ARTURO	FLORIAN	12/10/1970	M	C
PE0002	VERONICA	DIAZ	02/05/1976	F	C
PE0003	LUIS	GUEVARA	20/10/1980	M	S

NRO_HIJO	NOMBREHIJO	APELLIDOHIJO	EDAD	SEXO
HJ0001	CESAR	FLORIAN DIAZ	15	M
HJ0002	LUCERO	FLORIAN DIAZ	20	F
HJ0003	CARLOS	GUEVARA CHAVEZ	16	M

NRO_PERSONA	NRO_HIJO	APELLIDOPERS	APELLIDOPERS
PE0001	HJ0001	FLORIAN	FLORIAN DIAZ
PE0001	HJ0002	FLORIAN	FLORIAN DIAZ
PE0002	HJ0001	DIAZ	FLORIAN DIAZ
PE0002	HJ0002	DIAZ	FLORIAN DIAZ
PE0003	HJ0003	GUEVARA	GUEVARA CHAVEZ

✓ **DEPENDENCIA FUNCIONAL (DF)**

Hay veces en que los atributos están relacionados entre sí de manera más específica que la de pertenecer a una misma relación. Hay veces en que es posible determinar que un atributo depende de otro funcionalmente, como si existiera una función f en el "mundo", tal que $t[A] = f(t[B])$.

La función se anotaría como $f : A \rightarrow B$, pero como f es desconocida (o sino B sería un atributo derivado), sólo nos quedamos con $A \rightarrow B$, la dependencia funcional, que se lee: "A determina B".

Formalmente, $X \rightarrow Y$ en R se cumple si y sólo si $\forall s, t \in R, s[X] = t[X] \Rightarrow s[Y] = t[Y]$. Esto es análogo a las funciones: $\forall x_1, x_2 \in X, x_1 = x_2 \Rightarrow f(x_1) = f(x_2)$, con $f : X \rightarrow Y$...

UTILIDAD EN EL DISEÑO DE BASES DE DATOS

Las dependencias funcionales son restricciones de integridad sobre los datos. Conocer las dependencias funcionales en el momento del diseño de la base de datos permite crear mecanismos para evitar la redundancia (y los potenciales problemas de integridad que eso conlleva) y mejorar la eficiencia.

¿COMO OBTENER LAS DEPENDENCIAS FUNCIONALES?

La mejor manera de obtenerlas es a través del conocimiento del problema. Es lo más disponible en la fase de diseño de una base de datos. Sin embargo, esto puede tornarse bastante difícil, como en el caso del vehículo (honestamente, esto puede ocurrir cuando la base de datos modela conocimiento técnico, que escapa al sentido común).

Otra manera, relacionada con el ejemplo anterior, es comprobar dependencias funcionales sobre una gran población de datos usando la definición.

EJEMPLO

Una dependencia funcional es una relación de dependencia entre uno o más atributos. Por ejemplo si conocemos el valor FechaDeNacimiento podemos conocer el valor de Edad.

Las dependencias funcionales se escriben utilizando una flecha, de la siguiente manera:

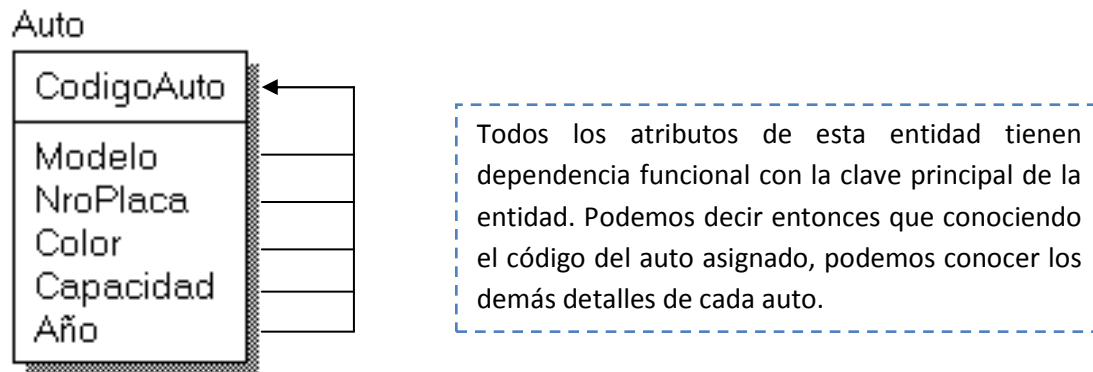
FechaDeNacimiento \rightarrow Edad

Aquí a FechaDeNacimiento se le conoce como un determinante. Se puede leer de dos formas FechaDeNacimiento determina a Edad o Edad es funcionalmente dependiente de FechaDeNacimiento. De la normalización (lógica) a la implementación (física o real) puede ser sugerible tener estas dependencias funcionales para lograr mayor eficiencia en las tablas.

VEAMOS OTRO EJEMPLO:

Tenemos la entidad

Entidad Auto (CodigoAuto, Modelo, NroPlaca, Color, Capacidad, Año)



✓ DEPENDENCIA FUNCIONAL TRANSITIVA

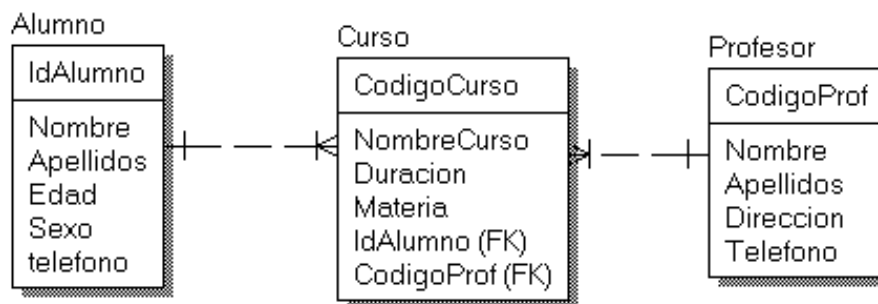
Supongamos que en una relación en la que los estudiantes solo pueden estar matriculados en un solo curso y supongamos que los profesores solo pueden dar un curso.

ID_Estudiante -> Curso_Tomando

Curso_Tomando -> Profesor_Asignado

ID_Estudiante -> Curso_Tomando -> Profesor_Asignado

Entonces tenemos que ID_Estudiante determina a Curso_Tomando y el Curso_Tomando determina a Profesor_Asignado, indirectamente podemos saber a través del ID_estudiante el Profesor_Asignado. Entonces en la relación tenemos una dependencia transitiva entre alumno y profesor.



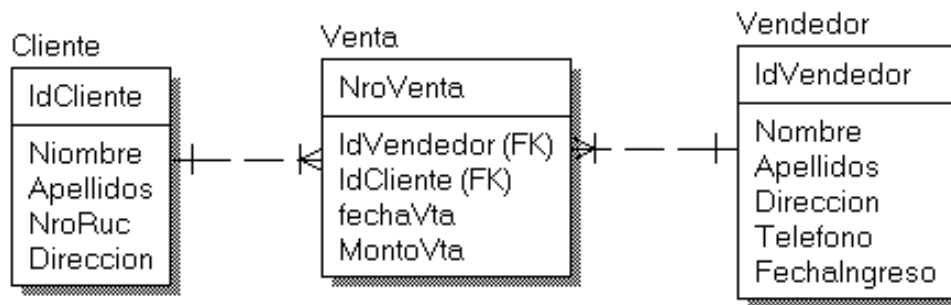
VEAMOS OTRO EJEMPLO:

IdCliente -> Venta realizada

Venta realizada -> Vendedor encargado

IdCliente -> Venta realizada -> Vendedor encargado

Entonces tenemos que el IdCliente determina a quién se le hizo la venta, y la venta realizada determina qué vendedor llevó a cabo la venta. Entonces en la relación tenemos una dependencia transitiva entre el cliente y el vendedor.



➤ NORMALIZACION DE DATOS

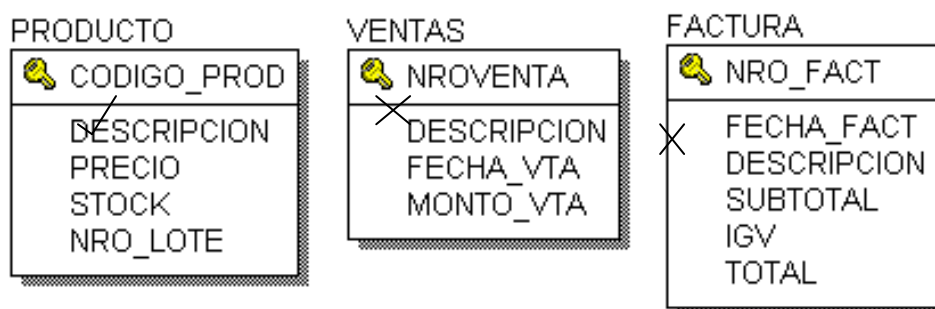
La normalización de datos es el proceso de transformación de las entidades complejas en entidades simples, siempre que se normaliza se crean por lo menos dos entidades nuevas. Esta es otra forma de encontrar las entidades del proceso de negocio, por medio de los documentos que son los que se puede normalizar, podemos diseñar los modelos de datos.

¿CUÁL ES EL OBJETIVO DE LA NORMALIZACIÓN?

- El objetivo principal es el de evitar la redundancia de los datos en las tablas, mejorar u optimizar el diseño del sistema para brindar una mejor performance de los procesos. Solo un diseño normalizado puede garantizar que nuestro sistema cumple con los requisitos de los usuarios.
- Además Evitar problemas de actualización de los datos en las tablas.
- Proteger la integridad de los datos.

¡EVITAR LA REDUNDANCIA!

Ejemplo:



En el proceso de normalizar datos, nos vamos a encontrar con que existen procedimientos para lograr la optimización de nuestro diseño de datos, estos procedimientos son conocidos como formas normales, las cuales a su vez tienen sus propias características, veamos cada uno de ellos.

Existen 5 formas normales, de las cuales podemos decir que cumplidas las 3 primeras formas normales tendremos un diseño adecuado de datos.

✓ 1ª FORMA NORMAL (1FN)

Una relación se encuentra en primera forma normal si y sólo si sus atributos son atómicos, es decir son no descomponibles. El objetivo de la 1FN es hallar aquellos los atributos que tienen dependencia funcional directamente con la PK.

► **DEPENDENCIA FUNCIONAL (DF)**

Es la relación que existe entre los atributos no primos (no claves) y la clave primaria de la entidad.

Ejemplo:

Alumno (código, nombre, apellido, nota1, nota2, promedio)

CODIGO	←	
NOMBRE	—	✓
APELLIDO	—	✓
NOTA 1	—	✗
NOTA 2	—	✗
PROMEDIO	—	✗

Diremos entonces: El campo Nombre y Apellido tienen DF con la clave Código.

Nota1, Nota2 y Promedio no tienen DF con la clave Código.

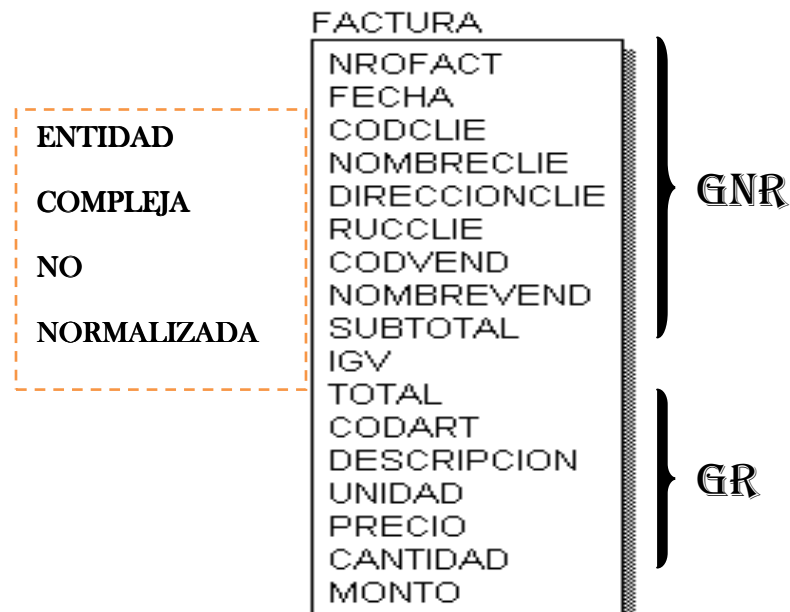
Sólo aquellos atributos que pertenezcan a las características propias de la entidad, tienen dependencia funcional con la PK, sin no dependen funcionalmente de la clave principal, entonces no pertenecen a la entidad.

PASOS DE LA 1FN:

1. Identificar los grupos repetitivos y no repetitivos (GR, GNR).
2. Remover los GR y crear una nueva entidad con ellos.
3. Llevar la clave a la nueva entidad.

Para explicar las formas normales, utilizaremos una factura de venta la cual iremos descomponiendo paso a paso.

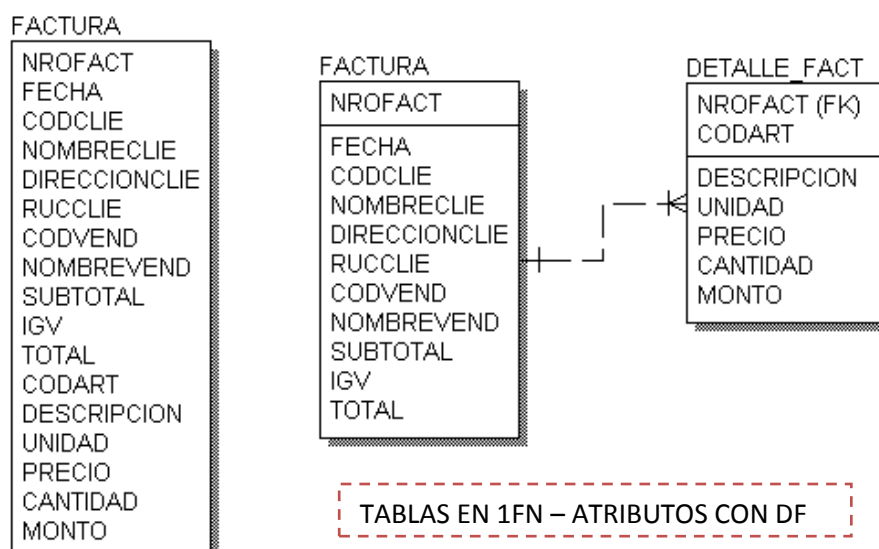
Tenemos una factura cuyo modelo es simple, una típica factura de una bodega o una farmacia por ejemplo, debemos ubicar todos aquellos datos que representan información importante para el negocio, las listamos para luego proceder a normalizarlo. Aquí la lista de atributos encontrados...



Veamos la factura en forma de tabla:

NROFACT	FECHA	CODCLIE...	CODVEND....	CODART	DESCRIPCION...
F0001	17/10/07	C0001	V0001	A0001	Televisor
F0001	17/10/07	C0001	V0001	A0002	Plancha
F0001	17/10/07	C0001	V0001	A0003	Cocina

GNR **GR**



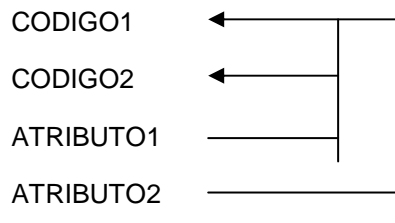
✓ **2ª FORMA NORMAL (2FN)**

Una relación estará en 2FN si y sólo si está en 1FN y además se cumple que los atributos no primos tienen dependencia funcional completa con respecto a la clave concatenada o compuesta.

► **DEPENDENCIA FUNCIONAL COMPUESTA (DFC)**

Es la relación que existe entre los atributos no primos (no claves) y la clave concatenada, una clave concatenada es aquella que está compuesta por dos o más atributos claves, la tienen las entidades asociadas y las entidades con relación identificada.

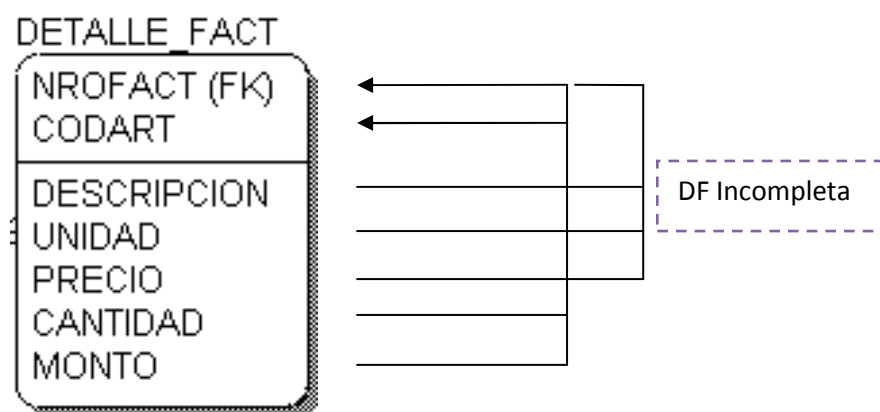
Ejemplo: Una entidad que tiene una clave compuesta.



Diremos: Atributo 1 tiene DFC con ambas claves, Atributo 2 no tiene DFC con ambas claves, entonces remover Atributo 2.

PASOS DE LA 2FN

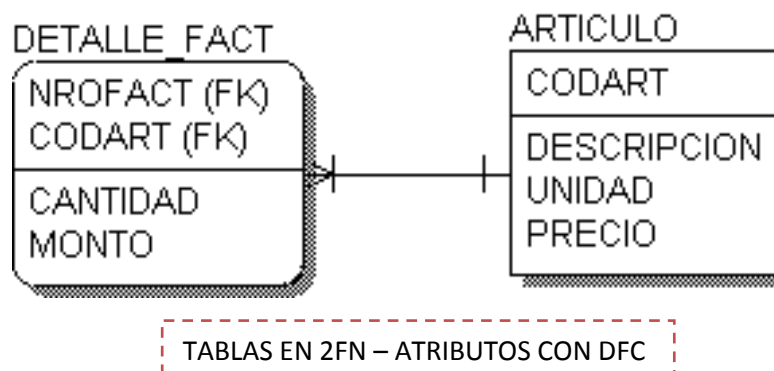
1. Identificar los atributos con dependencia funcional incompleta.
2. Remover los atributos con DF incompleta y crear una nueva entidad.
3. Llevar la clave a la nueva entidad.



Veamos esto en forma de tabla:

NROFACT	CODART	CANTIDAD	MONTO
F0001	A0001	2	400.00
F0001	A0002	3	390.00
F0001	A0003	1	540.00

CODART	DESCRIPCION	UNIDAD	PRECIO
A0001	Televisor	Unid	200.00
A0002	Plancha	Unid	130.00
A0003	Cocina	unid	540.00

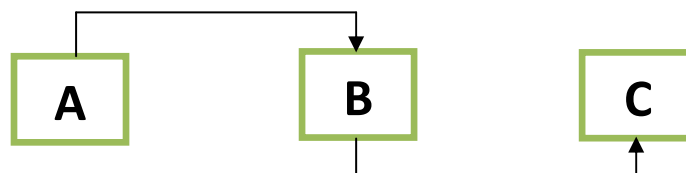


3ª FORMA NORMAL (3FN)

Una relación estará en 3FN si y sólo si está en 2FN y además existen atributos no claves que dependen de otros atributos no claves de la entidad compleja. Estos atributos no claves tienen relación transitiva con la entidad principal.

► DEPENDENCIA TRANSITIVA

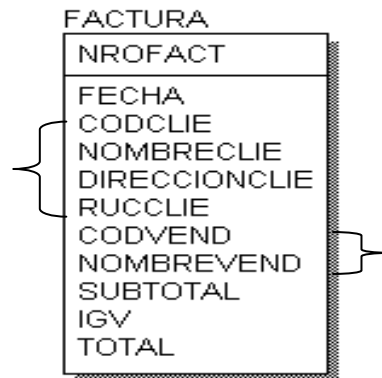
Se refiere a la relación indirecta entre dos o más entidades, esta relación indirecta se da por medio de otra entidad que funge de puente entre ambas.



Diremos entonces que: La entidad A es transitiva a la entidad C, relación indirecta por medio de la entidad B.

PASOS DE LA 3FN

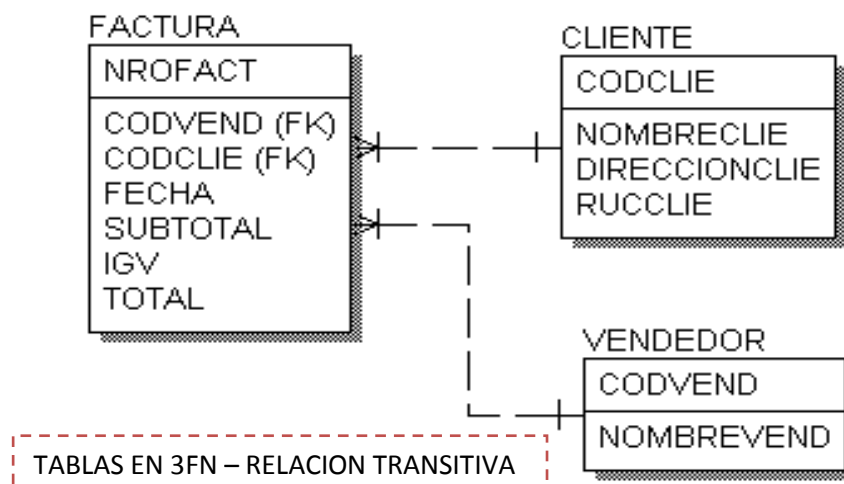
1. Identificar los atributos no claves con DF con otros atributos no claves.
2. Remover los atributos transitivos y crear una nueva entidad.
3. Llevar la clave a la nueva entidad.
- 4.



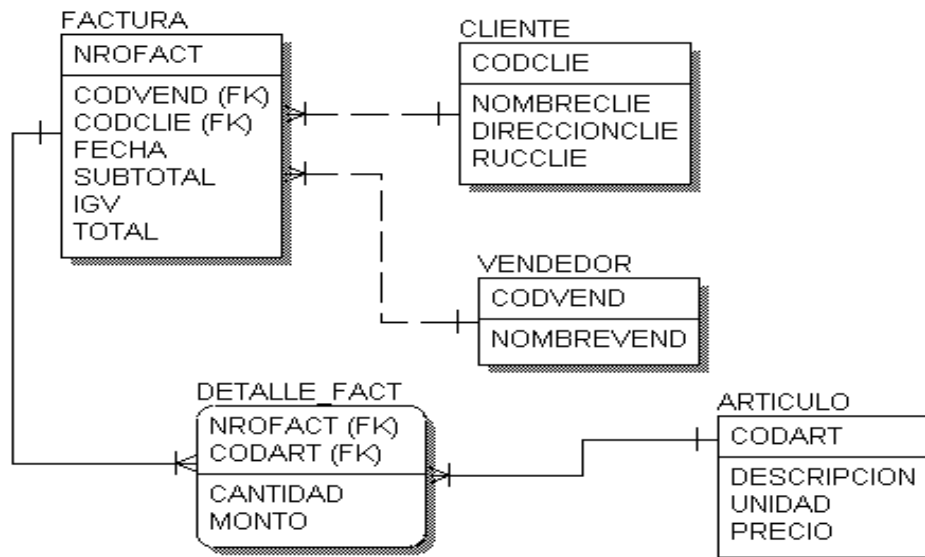
NROFACT	CODCLIE	NOMBREC	DIRECCIONC	RUCC	CODVEND	NOMBVEND
F0001	C0001	Augusto	San Isidro	1235698	V0001	Nelly
F0001	C0001	Pedro	Surco	5698745	V0002	Eduardo
F0001	C0001	María	Miraflores	5630214	V0001	Teresa

Diagrama de dependencias:

- Clave Entidad Principal (NROFACT)
- Atributos con DF con atributo no clave (CODCLIE, NOMBREC, DIRECCIONC, RUCC)
- Atributos con DF con atributo no clave (CODVEND, NOMBVEND)



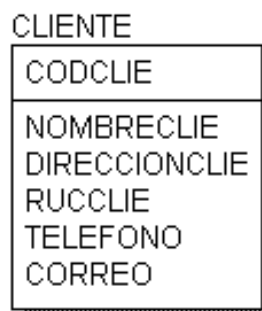
Para que un diseño de datos tenga credibilidad y de suficiente soporte al cumplimiento de requerimiento de los usuarios, se acepta hasta la 3FN, es decir, si el diseño se encuentra normalizado hasta la 3FN entonces cumple con los requisitos del sistema, este ejemplo quedaría así:



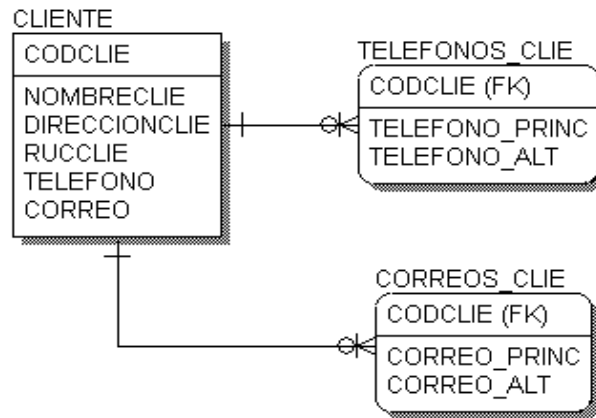
✓ 4ª FORMA NORMAL (4FN)

Una relación está en 4FN si y solo si se encuentra en 3FN y se cumple que no existan dependencias multivaluadas en alguno de los atributos no claves. Un atributo multivaluado es aquel que tiene varios posibles valores para una sola instancia de la entidad.

Por ejemplo: tenemos una entidad Cliente, cada uno de los clientes registrados puede tener varios teléfonos y correos alternativos, entonces estamos ante una dependencia multivaluada entre ambos atributos y el atributo no clave Nombreclie.



Debemos resolver creando dos nuevas entidades con los atributos multivaluados para evitar así la redundancia de datos.



¿QUÉ TAN LEJOS SE DEBE LLEVAR LA NORMALIZACIÓN?

La siguiente decisión es ¿qué tan lejos debe llevar la normalización? La normalización es una ciencia subjetiva. Determinar las necesidades de simplificación depende de nosotros. Si nuestra base de datos va a proveer información a un solo usuario para un propósito simple y existen pocas posibilidades de expansión, normalizar los datos hasta la 3FN quizá sea algo exagerado. Las reglas de normalización existen como guías para crear tablas que sean fáciles de manejar, así como flexibles y eficientes. A veces puede ocurrir que normalizar los datos hasta el nivel más alto no tenga sentido.

¿Se están dividiendo tablas sólo para seguir las reglas o estas divisiones son en verdad prácticas?. Éstas son el tipo de cosas que nosotros como diseñadores de la base de datos, necesitamos decidir, y la experiencia y el sentido común nos pueden auxiliar para tomar la decisión correcta. La normalización no es una ciencia exacta, más bien subjetiva.

Existen seis niveles más de normalización que no se han discutido aquí. Ellos son Forma Normal Boyce-Codd, Cuarta Forma Normal (4NF), Quinta Forma Normal (5NF) o Forma Normal de Proyección-Unión, Forma Normal de Proyección-Unión Fuerte, Forma Normal de Proyección-Unión Extra Fuerte y Forma Normal de Clave de Dominio. Estas formas de normalización pueden llevar las cosas más allá de lo que necesitamos. Éstas existen para hacer una base de datos realmente relacional. Tienen que ver principalmente con dependencias múltiples y claves relacionales.

EN RESUMEN

La normalización es una técnica que se utiliza para crear relaciones lógicas apropiadas entre tablas de una base de datos. Ayuda a prevenir errores lógicos en la manipulación de datos. La normalización facilita también agregar nuevas columnas sin romper el esquema actual ni las relaciones.

Existen varios niveles de normalización: Primera Forma Normal, Segunda Forma Normal, Tercera Forma Normal, Forma Normal Boyce-Codd, Cuarta Forma Normal, Quinta Forma Normal o Forma Normal de Proyección-Unión, Forma Normal de Proyección-Unión Fuerte, Forma Normal de Proyección-Unión Extra Fuerte y Forma Normal de Clave de Dominio. Cada nuevo nivel o forma nos acerca más a hacer una base de datos verdaderamente relacional.

Se discutieron las primeras tres formas. Éstas proveen suficiente nivel de normalización para cumplir con las necesidades de la mayoría de las bases de datos. Normalizar demasiado puede conducir a tener una base de datos ineficiente y hacer a su esquema demasiado complejo para trabajar. Un balance apropiado de sentido común y práctico puede ayudarnos a decidir cuándo normalizar.

✓ DESNORMALIZACIÓN

Se puede definir como el proceso de poner la misma información en varios lugares. Una normalización reduce problemas de integridad y optimiza las actualizaciones, quizás con el costo del tiempo de recuperación. Cuando se pretende evitar esta demora resultado de la combinación de muchas tablas entonces se puede utilizar la desnormalización.

Antes de denormalizar es importante considerar:

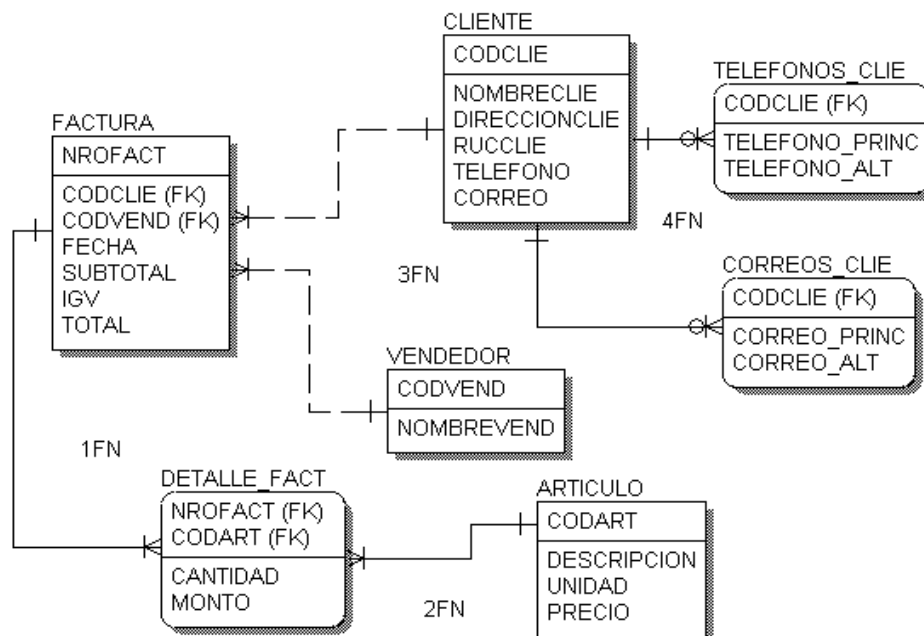
- ¿El sistema puede tener un desempeño aceptable sin la desnormalización?
- ¿Aún con la denormalización el desempeño será siendo malo?
- ¿El sistema será menos confiable debido a la desnormalización?

Candidatos a desnormalización:

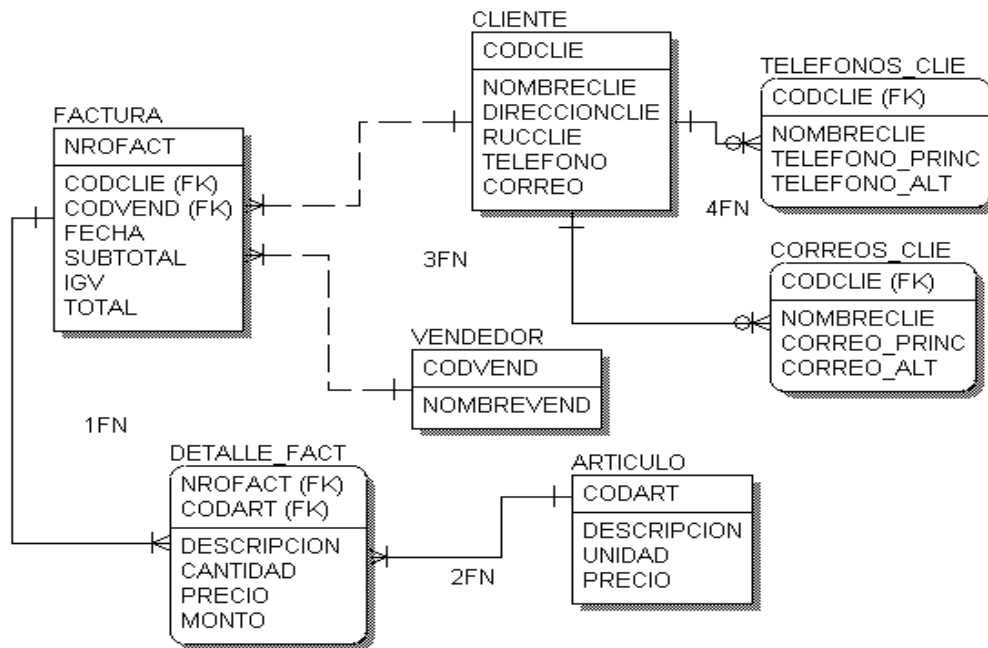
- Numerosas consultas críticas o reportes incluyen datos que incluyen más de una tabla.
- Grupos repetidos de elementos necesitan ser procesados en un grupo en lugar de individualmente.
- Muchos cálculos necesitan realizarse a una o más columnas antes de procesar las consultas.
- Las tablas necesitan ser accedidas de diferentes maneras por diferentes usuarios durante el mismo lapso de tiempo.
- Llaves primarias mal diseñadas que requieren tiempo al usarlas en relaciones.
- Algunas columnas son interrogadas un gran porcentaje del tiempo.

Importante: nunca se realiza un desnormalización en un modelo lógico.

Comparación: Veamos como quedó nuestro ejemplo de Normalización hasta la 4FN:



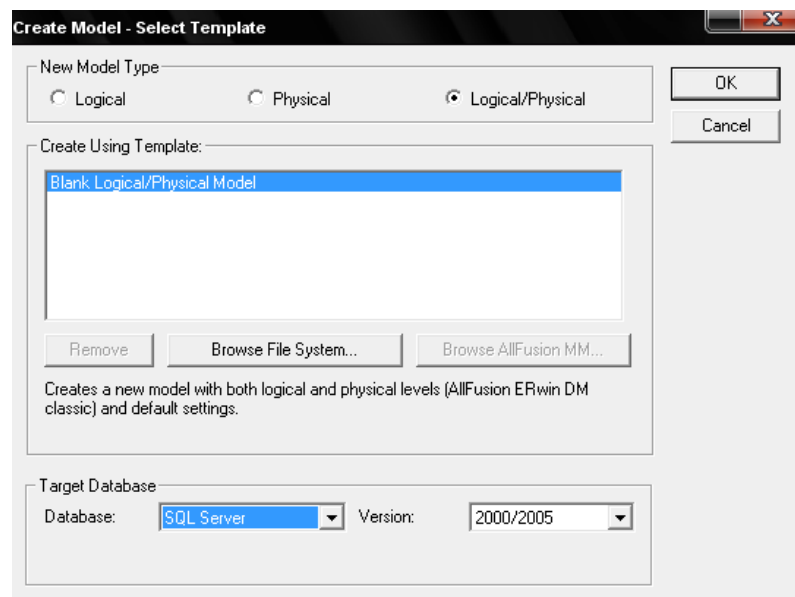
Veamos ahora como debe quedar el mismo diseño con la técnica de Desnormalización:



LABORATORIO # 4

MODELO LOGICO- FISICO

Al iniciar un nuevo archivo tenemos la opción de seleccionar el tipo de modelo a trabajar, bien podemos trabajar solo a nivel de modelo lógico, o sólo a nivel de modelo físico, pero lo recomendable es trabajar ambos modelos, para poder seleccionar un motor de base de datos.



➤ REPRESENTACION DE UN DER EN UN MODELO RELACIONAL

A continuación veremos un caso de estudio en la cual se muestra el modelo lógico y el modelo físico resultante...

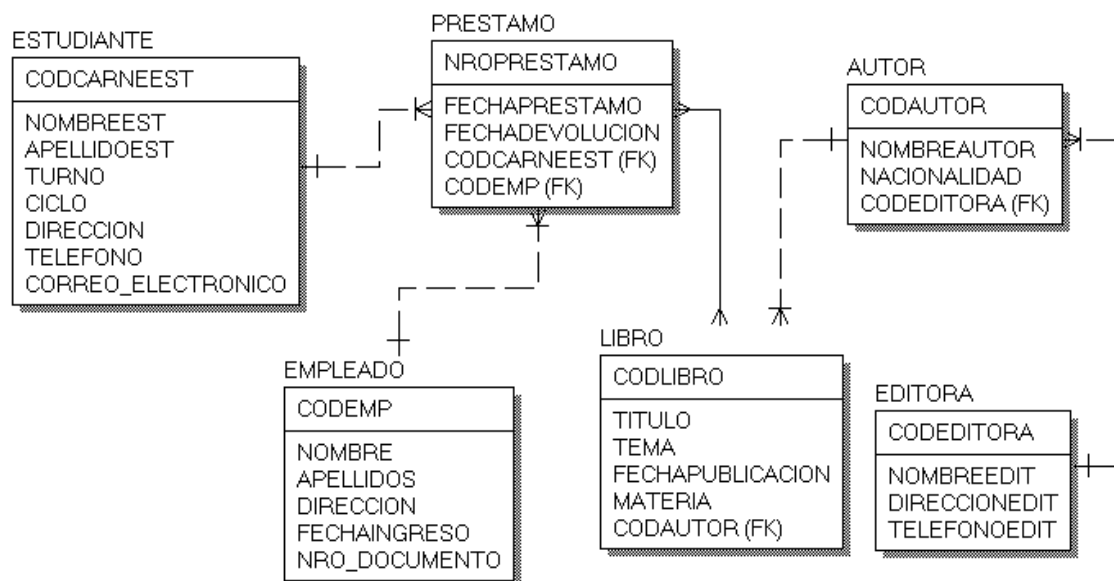
PROCESO DE NEGOCIO BIBLIOTECA

La biblioteca del instituto tecnológico desea implementar un sistema de control de préstamos de libros a los estudiantes, para lo cual se nos brindó la información necesaria.

El estudiante debe solicitar el préstamo al empleado encargado de la biblioteca quién entregará el libro en cuestión. El estudiante debe dejar un documento personal para que se le entregue el libro.

Por cada préstamo el estudiante puede solicitar hasta un máximo de tres libros, de distintas especialidades, se desea registrar cada libro clasificándolos según el autor y la editora que la distribuye.

EL MODELO LOGICO



En el modelo físico resolvemos la relación de Muchos a Muchos entre las entidades Libro y Préstamo, para ello no olvidar que debemos activar la opción Auto apply Many-to-Many transform, desde el menú Model – Model Properties, tal como se muestra en la siguiente imagen...

Model Properties

General | Definition | Notation | Defaults | RI Defaults | UDP | History Options | History

Model Info

Name:

Author:

Type: Logical/Physical Database: SQL Server

Enable Modeling Features

☐ Dimensional

☐ Data Movement

Transform Options

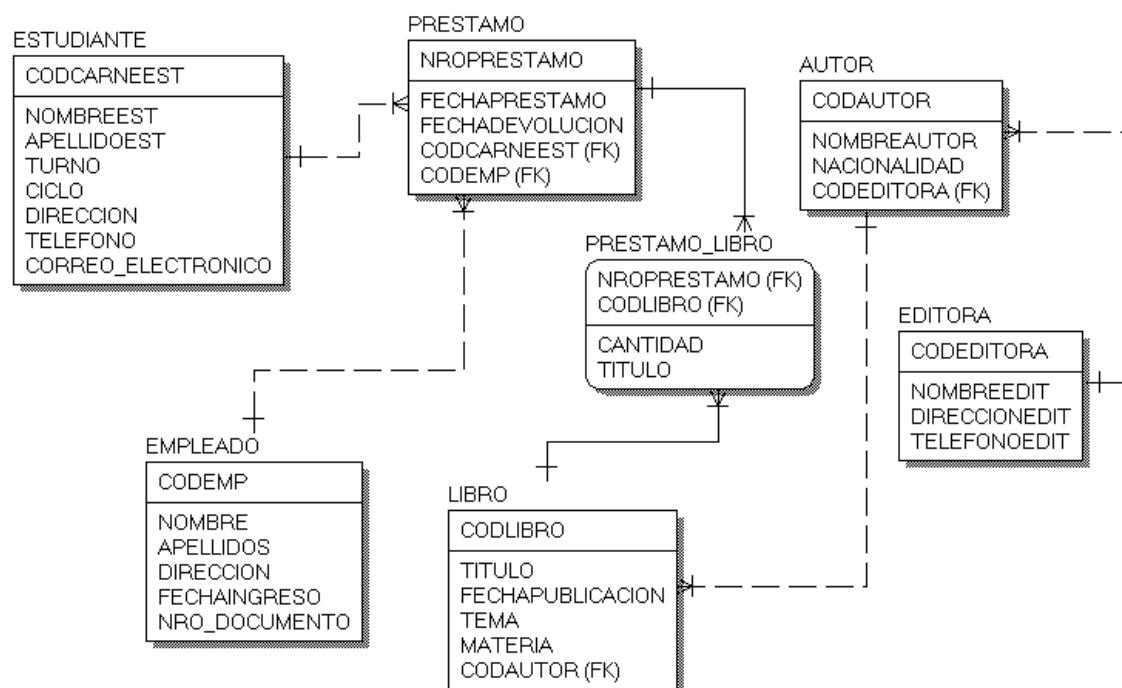
☒ Show source objects in logical, target objects in physical

☒ Auto apply Many-to-Many transform

☒ Auto apply Supertype-Subtype Identity transform

OK Cancel

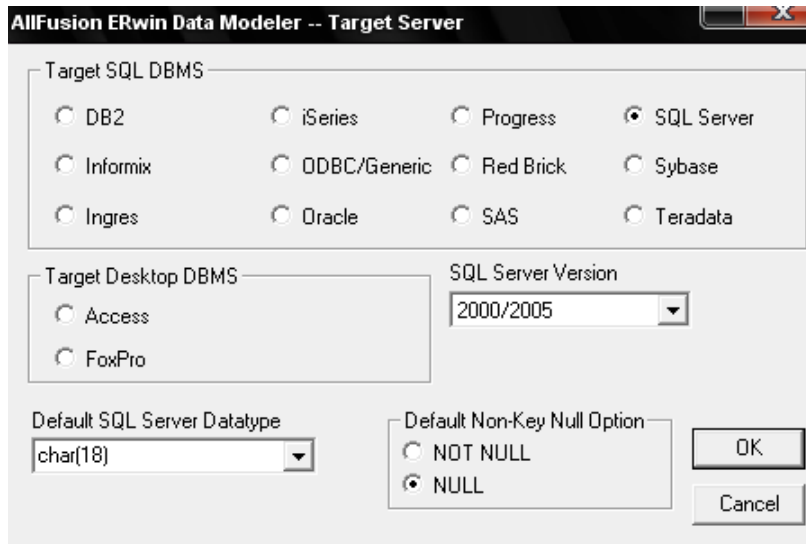
EL MODELO FISICO



➤ OPTIMIZACION DE TIPO DE DATOS

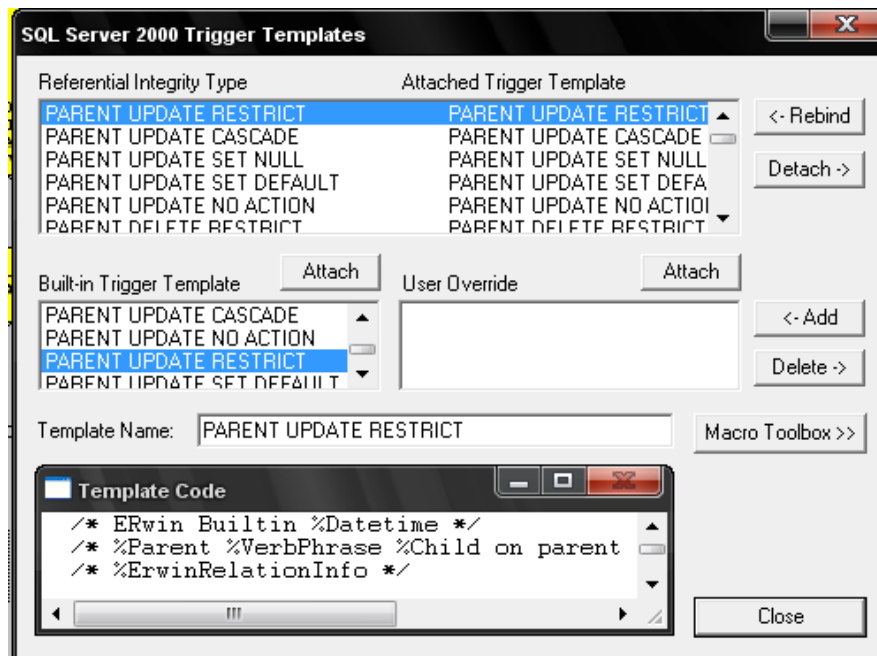
✓ **SELECCIONAR EL SERVIDOR DE DATOS**

- ✓ Hacemos clic en el menú Databases – Choose Database.
- ✓ Aparece la siguiente ventana:



Aquí debemos seleccionar el DBMS con la que nos conectaremos. Luego la versión del servidor disponible. El tipo de dato por defecto y los valores no llaves por defecto, en este caso No Null. Luego de ello le damos clic en Ok.

Ahora debemos chequear las opciones por defecto de Integridad Referencial, para ello nos vamos al menú Database y elegimos RI Trigger Templates..



Podemos trabajarlo también desde el menú Model – Model Properties en la pestaña RI Defaults...

Action	Relationship Type			
	Identifying	Non-Identifying Nulls Allowed	Non-Identifying No Nulls	Subtype
Child Delete	NO ACTION	NO ACTION	NO ACTION	NO ACTION
Child Insert	NONE	NONE	NONE	NONE
Child Update	NO ACTION	NO ACTION	NO ACTION	NO ACTION
Parent Delete	NO ACTION	NO ACTION	NO ACTION	CASCADE
Parent Insert	NONE	NONE	NONE	NONE
Parent Update	NO ACTION	NO ACTION	NO ACTION	CASCADE

Rebind Reset

OK Cancel

✓ ASIGNANDO PROPIEDADES DE COLUMNAS

- ✓ Hacemos clic derecho sobre las tablas, elegimos la opción Column.
- ✓ Aparece la siguiente ventana.

Table: MERCADERIA

Column

- NROLOTE
- DESCRIPCION
- MARCA
- TIPO
- PRECIO
- STOCK_MINIMO
- STOCK_MAXIMO
- STOCK_ACTUAL
- FECHA_VENCIMIENTO

New... Rename... Delete

Reset... DB Sync...

General SQL Server Constraint Co... }

SQL Server Datatype*

char(18) Average Width: 18

char()
character varying()
character()
datetime

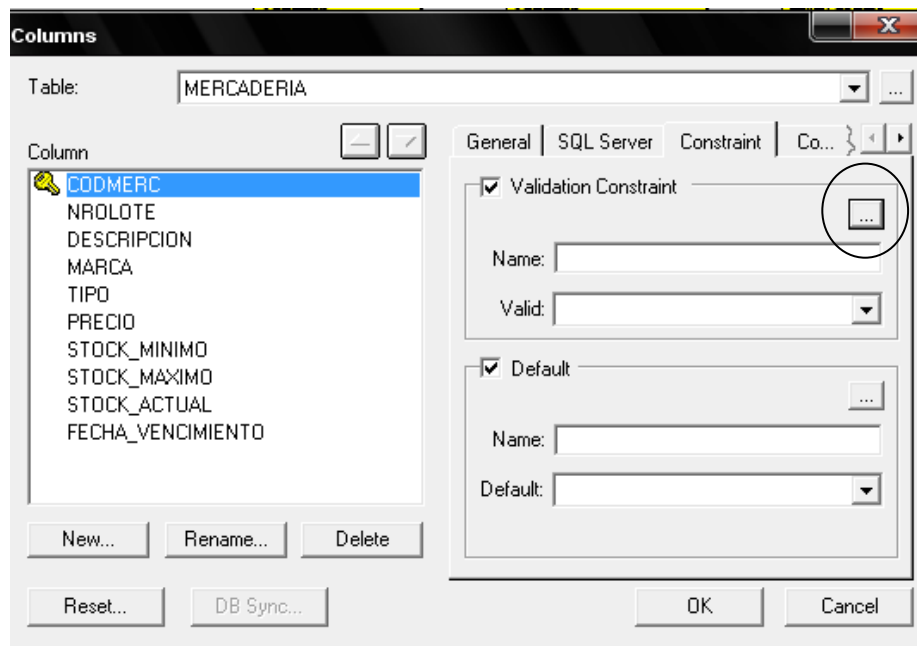
Percent NULL:*

Null Option

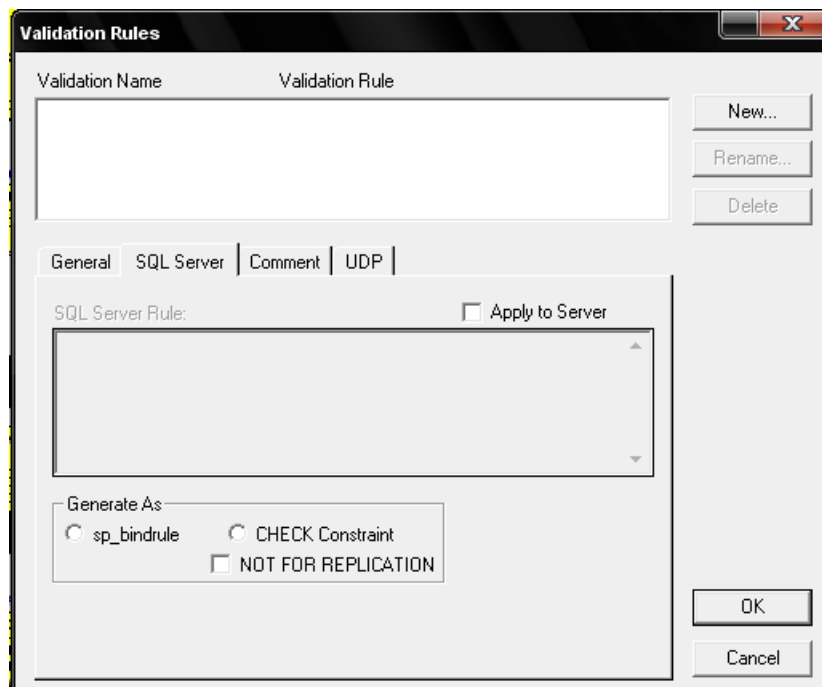
☒ NOT NULL
☐ NULL
☐ IDENTITY

OK Cancel

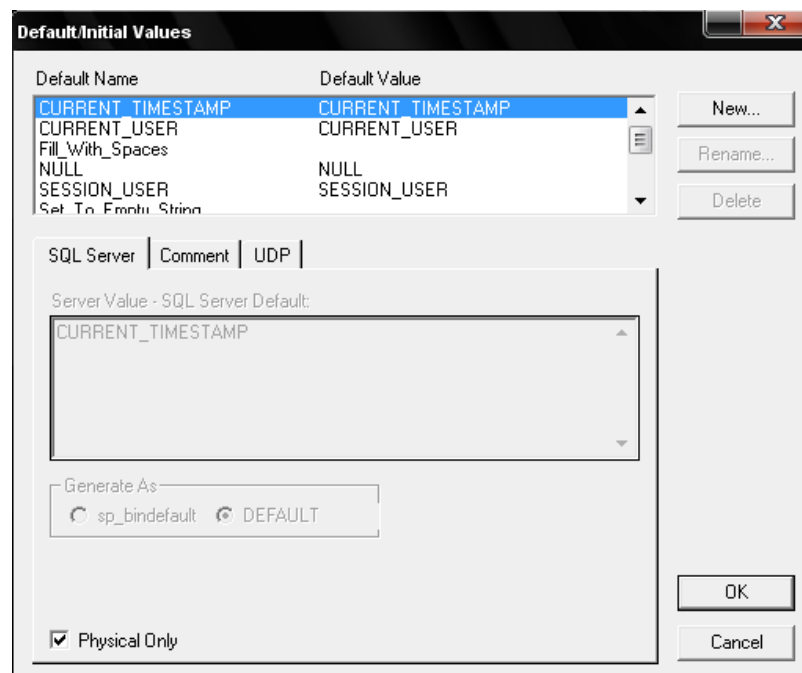
Aquí seleccionamos cada columna y el tipo de dato compatible con el servidor de datos seleccionado, además asignamos la opción de colocar valores nulos a los campos. Podemos asignar una regla de validación a cada campo, desde la pestaña Constraint...



En Validation Constraint hacemos clic en el botón Agregar para poder asignar una regla de validación de datos, aparecerá la siguiente ventana...



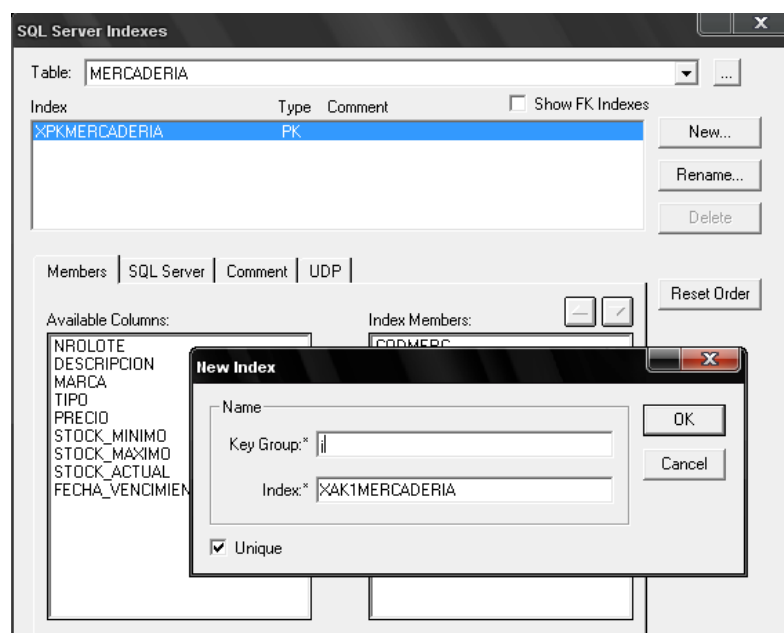
Si lo que deseamos es asignar un valor por defecto, hacemos clic en el botón Agregar del área Default, aparecerá la siguiente ventana en donde tenemos un listado de funciones disponibles...



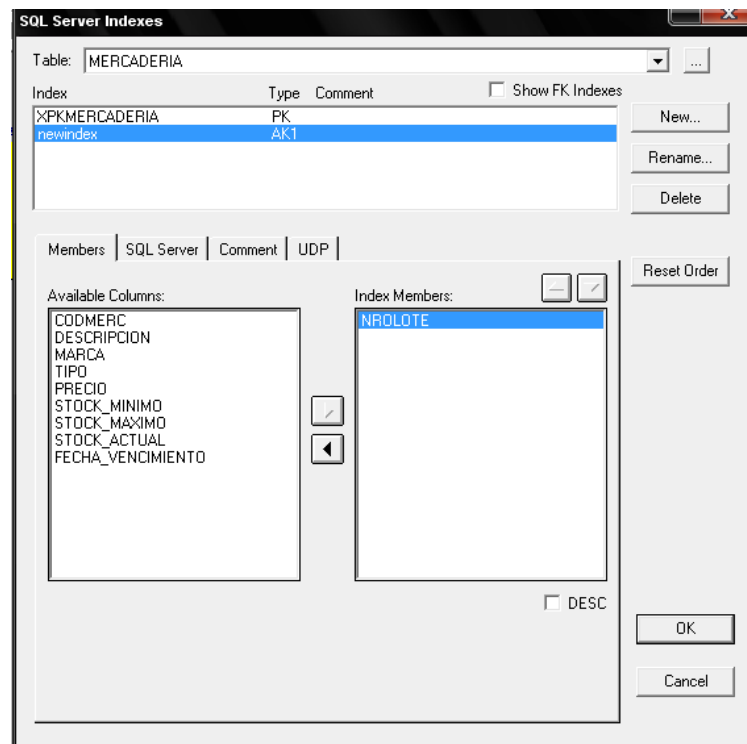
Por ejemplo tenemos la opción de colocar el usuario actual del sistema (CURRENT_USER), o podemos colocar un valor nuevo creándolo desde el botón New...

✓ CREAR INDICES

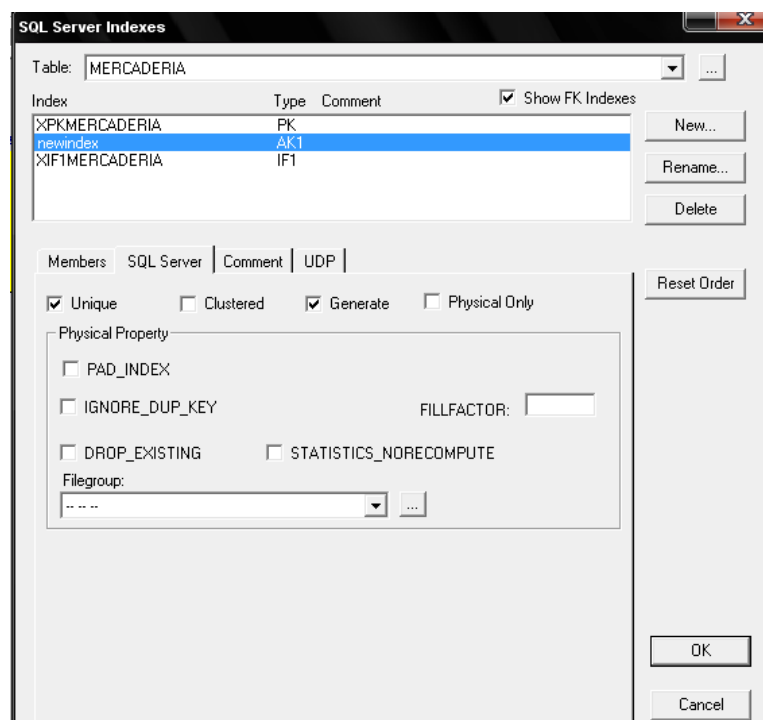
Hacemos clic derecho sobre la tabla, luego elegimos la opción Indexes, en la ventana veremos el único índice creado que es la Llave primaria (Primary Key). Para agregar un nuevo índice damos clic en el botón New...



Le damos un nombre al índice, luego en la ventana principal seleccionamos el o los campos que pertenecerán al índice.



Los índices nos ayudan a crear un patrón de búsquedas para los registros de datos de las tablas, podemos seleccionar los campos asignados como claves alternas como el índice, para ello debemos tener seleccionada la opción Unique en la pestaña SQL Server...



➤ **INGENIERIA DIRECTA / REVERSIVA**

(FORWARD ENGINEER / REVERSE ENGINEER)

Ingeniería Directa: Es el proceso de producción del código de una aplicación a partir de sus especificaciones. La ingeniería directa reduce el tiempo de creación de una base de datos ya que será la herramienta de diseño quien genere los códigos DDL dentro del DBMS.

Ingeniería Inversa: Conjunto de tareas destinadas a obtener las especificaciones de un sistema de información partiendo del propio sistema. Es una actividad típica del mantenimiento de equipos lógicos, cuando no existen las especificaciones del diseño del equipo a mantener.

La ingeniería inversa tiene la misión de interpretar las fuentes de los sistemas informáticos. Consiste básicamente en recuperar el diseño de una aplicación a partir del código fuente generado. Esto se realiza mediante herramientas que extraen información de los datos, procedimientos y arquitectura del sistema existente.

La Ingeniería inversa es aplicable a aquellos sistemas cuya documentación es inexistente, demanda grandes cambios en su estructura frecuentemente y contiene bloques de código muy grandes y sin reestructurar.

ERWIN – SQL SERVER

Ca Erwin Data Modeler facilita en gran medida la gestión de modelos para el análisis del negocio y la recogida de requisitos, así como el diseño y la implementación de bases de datos y aplicaciones data warehouse de calidad. Además, permite a los profesionales de las bases de datos que utilicen SQL Server diseñar visualmente y generar objetos gestionados para utilizarlos en el entorno de desarrollo de Microsoft Visual Studio.

Los profesionales de las bases de datos también pueden aprovechar CA ERwin DM para generar sus esquemas de datos en SQL Server y así acelerar la implantación de proyectos Visual Studio.

EJERCICIOS

Realizar el modelo Lógico y Físico en Erwin del siguiente caso de estudio:

PROCESO DE NEGOCIO

FARMACIA

Una farmacia 'X' desea implementar un pequeño sistema para el control de su almacén, Cada salida de medicamento hacia ventas debe estar registrado en una orden de venta, sin este documento no podrá salir. Cada medicamento tiene un tipo de lo clasifica, como jarabe, comprimidos, etc. Cada adquisición de nuevos lotes de medicamentos debe estar registrada en una orden de compra, se debe registrar el laboratorio proveedor de los medicamentos así como la orden de compra que se le envía, cada medicamento se clasifica según la especialidad, como pediatría, urología, tipo de enfermedad, etc.

Semana

5

Contenido:

- ✓ Taller de Formas Normales
 - ✓ Forma Normal 1
 - ✓ Forma Normal 2
 - ✓ Forma Normal 3
 - ✓ Creación de bases de datos físicas en SQL Server.
 - ✓ Archivo de datos y registro de transacciones
 - ✓ Creación de un MER a partir de un DER.
 - ✓ Creación de BD en SQL Server 2005.
 - ✓ Forward Engineer (Ingeniería Directa)
-

NORMALIZACIÓN – CREACIÓN DE BASE DE DATOS (DDL)

La normalización de bases de datos relacionales toma un esquema relacional y le aplica un conjunto de técnicas para producir un nuevo esquema que representa la misma información pero contiene menos redundancias y evita posibles anomalías en las inserciones, actualizaciones y borrados.

✓ Breve recordatorio del modelo (formal) relacional

El modelo relacional de bases de datos se basa en un modelo formal especificado de acuerdo a la teoría de conjuntos. Una base de datos relacional puede considerarse como un conjunto de relaciones o tablas de la forma $R(A_1, \dots, A_n)$, donde R es el nombre de la relación, que se define por una serie de atributos A_i .

Sobre las tablas relacionales se pueden definir diferentes restricciones. La integridad de entidad es una restricción que nos indica que cada entidad representada por una tupla tiene que ser diferente de las demás en su relación, es decir, debe haber algunos atributos cuyos valores identifiquen unívocamente las tuplas. La integridad referencial indica que una clave ajena solo debe contener valores que o bien sean nulos, o bien existan en la relación referenciada por la clave ajena.

✓ EL PROCESO DE NORMALIZACIÓN

El proceso de normalización consiste en comprobar en secuencia si el esquema original está en 1FN, 2FN y 3FN, analizando las dependencias funcionales en cada paso.

Un ejemplo completo

Tenemos una empresa pública donde los puestos de trabajo están regulados por el Estado, de modo que las condiciones salariales están determinadas por el puesto. Se ha creado el siguiente esquema relacional

EMPLEADOS(nss, nombre, puesto, salario, emails) con nss como clave primaria.

nss	nombre	puesto	salario	emails
111	Juan Pérez	Jefe de Área	3000	juanp@ecn.es; jefe2@ecn.es
222	José Sánchez	Administrativo	1500	jsanchez@ecn.es
333	Ana Díaz	Administrativo	1500	adiaz@ecn.es; ana32@gmail.com
...

✓ Primera forma normal (1FN)

Una tabla está en 1FN si sus atributos contienen valores atómicos. En el ejemplo, podemos ver que el atributo emails puede contener más de un valor, por lo que viola 1FN.

En general, tenemos una relación R con clave primaria K. Si un atributo M viola la condición de 1FN, tenemos dos opciones.

Solución 1: duplicar los registros con valores repetidos

En general, esta solución pasa por sustituir R por una nueva relación modificada R', en la cual:

- El atributo M que violaba 1FN se elimina.
- Se incluye un nuevo atributo M' que solo puede contener valores simples, de modo que si R'[M'] es uno de los valores que teníamos en R[M], entonces R'[K] = R[K]. En otras palabras, para una tupla con n valores duplicados en M, en la nueva relación habrá n tuplas, que sólo varían en que cada una de ellas guarda uno de los valores que había en M.
- La clave primaria de R' es (K, M'), dado que podrá haber valores de K repetidos, para los valores multivaluados en M.

Siguiendo el ejemplo, tendríamos el siguiente esquema para la nueva tabla EMPLEADOS'(a) con clave primaria (nss, email):

nss	nombre	puesto	salario	email
111	Juan Pérez	Jefe de Área	3000	juanp@ecn.es
111	Juan Pérez	Jefe de Área	3000	jefe2@ecn.es
222	José Sánchez	Administrativo	1500	jsanchez@ecn.es
333	Ana Díaz	Administrativo	1500	adiaz@ecn.es
333	Ana Díaz	Administrativo	1500	ana32@gmail.com
...

Solución 2: separar el atributo que viola 1FN en una tabla

En general, esta solución pasa por:

Sustituir R por una nueva relación modificada R' que no contiene el atributo M.

Crear una nueva relación N(K, M'), es decir, una relación con una clave ajena K referenciando R', junto al atributo M', que es la variante mono-valuada del atributo M.

La nueva relación N tiene como clave (K, M').

Siguiendo el ejemplo, tendríamos el siguiente esquema para la nueva tabla EMPLEADOS'(b).

nss	nombre	puesto	salario
111	Juan Pérez	Jefe de Área	3000
222	José Sánchez	Administrativo	1500
333	Ana Díaz	Administrativo	1500
...

Y además tendríamos una nueva tabla EMAILS con clave primaria (nss, email):

nss	email
111	juanp@ecn.es
111	jefe2@ecn.es
222	jsanchez@ecn.es
333	adiaz@ecn.es
333	ana32@gmail.com
...	...

Segunda Forma Normal (2fn)

Un esquema está en 2FN si:

Está en 1FN. Todos sus atributos que no son de la clave principal tienen dependencia funcional completa respecto de todas las claves existentes en el esquema. En otras palabras, para determinar cada atributo no clave se necesita la clave primaria completa, no vale con una subclave.

La 2FN se aplica a las relaciones que tienen claves primarias compuestas por dos o más atributos. Si una relación está en 1FN y su clave primaria es simple (tiene un solo atributo), entonces también está en 2FN. Por tanto, de las soluciones anteriores, la tabla EMPLEADOS'(b) está en 1FN (y la tabla EMAILS no tiene atributos no clave), por lo que el esquema está en 2FN. Sin embargo, tenemos que examinar las dependencias funcionales de los atributos no clave de EMPLEADOS'(a). Las dependencias funcionales que tenemos son las siguientes:

nss->nombre, salario, email

puesto->salario

Como la clave es (nss, email), las dependencias de nombre, salario y email son *incompletas*, por lo que la relación no está en 2FN.

En general, tendremos que observar los atributos no clave que dependan de parte de la clave.

Para solucionar este problema, tenemos que hacer lo siguiente para los grupos de atributos con dependencia incompleta M:

Eliminar de R el atributo M.

Crear una nueva relación N con el atributo M y la parte de la clave primaria K de la que depende, que llamaremos K'.

La clave primaria de la nueva relación será K'.

Siguiendo el ejemplo anterior, crearíamos una nueva relación con los atributos que tienen dependencia incompleta:

nss	nombre	puesto	salario
111	Juan Pérez	Jefe de Área	3000
222	José Sánchez	Administrativo	1500
333	Ana Díaz	Administrativo	1500
...

Y al eliminar de la tabla original estos atributos nos quedarían:

nss	email
111	juanp@ecn.es
111	jefe2@ecn.es
222	jsanchez@ecn.es
333	adiaz@ecn.es
333	ana32@gmail.com
...	...

Como vemos, la solución a la que llegamos es la misma que en la otra opción de solución para el problema de 1FN.

Tercera forma normal (3FN)

Una relación está en tercera forma normal si, y sólo si:

Está en 2FN y, además, cada atributo que no está incluido en la clave primaria no depende transitivamente de la clave primaria.

Por lo tanto, a partir de un esquema en 2FN, tenemos que buscar dependencias funcionales entre atributos que no estén en la clave.

En general, tenemos que buscar dependencias transitivas de la clave, es decir, secuencias de dependencias como la siguiente: $K \rightarrow A$ y $A \rightarrow B$, donde A y B no pertenecen a la clave. La solución a este tipo de dependencias está en separar en una tabla adicional N el/los atributos B, y poner como clave primaria de N el atributo que define la transitividad A.

Siguiendo el ejemplo anterior, podemos detectar la siguiente transitividad:

nss \rightarrow puesto

puesto \rightarrow salario

Por lo tanto la descomposición sería la siguiente:

nss	nombre	puesto
111	Juan Pérez	Jefe de Área
222	José Sánchez	Administrativo
333	Ana Díaz	Administrativo
...

En la nueva tabla PUESTOS, la clave sería el puesto, que también queda como clave ajena referenciando la tabla EMPLEADOS. El resto de las tablas quedan como estaban.

A través del siguiente ejercicio se intenta afirmar los conocimientos de normalización con un ejemplo simplificado de una base de datos para una pequeña biblioteca.

CodLibro	Título	Autor	Editorial	NombreLector	FechaDev
1001	Variable compleja	Murray Spiegel	McGraw Hill	Pérez Gómez, Juan	15/04/2005
1004	Visual Basic 5	E. Petroustsos	Anaya	Ríos Terán, Ana	17/04/2005
1005	Estadística	Murray Spiegel	McGraw Hill	Roca, René	16/04/2005
1006	Oracle University	Nancy Greenberg y Priya Nathan	Oracle Corp.	García Roque, Luis	20/04/2005
1007	Clipper 5.01	Ramalho	McGraw Hill	Pérez Gómez, Juan	18/04/2005

Esta tabla no cumple el requisito de la Primera Forma Normal (1NF) de sólo tener campos atómicos, pues el nombre del lector es un campo que puede (y conviene) descomponerse en apellido paterno, apellido materno y nombres. Tal como se muestra en la siguiente tabla.

1NF

CodLibro	Título	Autor	Editorial	Paterno	Materno	Nombres	FechaDev
1001	Variable compleja	Murray Spiegel	McGraw Hill	Pérez	Gómez	Juan	15/04/2005
1004	Visual Basic 5	E. Petroustsos	Anaya	Ríos	Terán	Ana	17/04/2005
1005	Estadística	Murray Spiegel	McGraw Hill	Roca		René	16/04/2005
1006	Oracle University	Nancy Greenberg	Oracle Corp.	García	Roque	Luis	20/04/2005
1006	Oracle University	Priya Nathan	Oracle Corp.	García	Roque	Luis	20/04/2005
1007	Clipper 5.01	Ramalho	McGraw Hill	Pérez	Gómez	Juan	18/04/2005

Como se puede ver, hay cierta redundancia característica de 1NF.

La Segunda Forma Normal (2NF) pide que no existan dependencias parciales o dicho de otra manera, todos los atributos no clave deben depender por completo de la clave primaria. Actualmente en nuestra tabla tenemos varias dependencias parciales si consideramos como atributo clave el código del libro.

Por ejemplo, el título es completamente identificado por el código del libro, pero el nombre del lector en realidad no tiene dependencia de este código, por tanto estos datos deben ser trasladados a otra tabla.

2NF

CodLibro	Título	Autor	Editorial
1001	Variable compleja	Murray Spiegel	McGraw Hill
1004	Visual Basic 5	E. Petroustos	Anaya
1005	Estadística	Murray Spiegel	McGraw Hill
1006	Oracle University	Nancy Greenberg	Oracle Corp.
1006	Oracle University	Priya Nathan	Oracle Corp.
1007	Clipper 5.01	Ramalho	McGraw Hill

La nueva tabla sólo contendrá datos del lector.

CodLector	Paterno	Materno	Nombres
501	Pérez	Gómez	Juan
502	Ríos	Terán	Ana
503	Roca		René
504	García	Roque	Luis

Hemos creado una tabla para contener los datos del lector y también tuvimos que crear la columna CodLector para identificar unívocamente a cada uno. Sin embargo, esta nueva disposición de la base de datos necesita que exista otra tabla para mantener la información de qué libros están prestados a qué lectores. Esta tabla se muestra a continuación:

CodLibro	CodLector	FechaDev
1001	501	15/04/2005
1004	502	17/04/2005
1005	503	16/04/2005
1006	504	20/04/2005
1007	501	18/04/2005

Para la Tercera Forma Normal (3NF) la relación debe estar en 2NF y además los atributos no clave deben ser mutuamente independientes y dependientes por completo de la clave primaria. También recordemos que dijimos que esto significa que las columnas en la tabla deben contener solamente información sobre la entidad definida por la clave primaria y, por tanto, las columnas en la tabla deben contener datos acerca de una sola cosa.

En nuestro ejemplo en 2NF, la primera tabla conserva información acerca del libro, los autores y editoriales, por lo que debemos crear nuevas tablas para satisfacer los requisitos de 3NF.

3NF

CodLibro	Título
1001	Variable compleja
1004	Visual Basic 5
1005	Estadística
1006	Oracle University
1007	Clipper 5.01
CodAutor	Autor
801	Murray Spiegel
802	E. Petroustsos
803	Nancy Greenberg
804	Priya Nathan
806	Ramalho

CodEditorial	Editorial
901	McGraw Hill
902	Anaya
903	Oracle Corp.

Aunque hemos creado nuevas tablas para que cada una tenga sólo información acerca de una entidad, también hemos perdido la información acerca de qué autor ha escrito qué libro y las editoriales correspondientes, por lo que debemos crear otras tablas que relacionen cada libro con sus autores y editoriales.

CodLibro	codAutor
1001	801
1004	802
1005	801
1006	803
1006	804
1007	806

CodLibro	codEditorial
1001	901
1004	902
1005	901
1006	903
1007	901

Y el resto de las tablas no necesitan modificación.

SQL SERVER 2008

SQL Server es el Gestor de Base de datos relacional más popular (junto a Oracle) en el mercado, permite crear espacios de almacenamiento físico de datos para aplicaciones en entorno Cliente – Servidor. Ya en la primera semana se detalló las características de este gestor, pasemos a explicar algunos conceptos básicos.

✓ BASE DE DATOS

Una base de datos o banco de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

✓ TIPOS DE BASES DE DATOS

Las bases de datos pueden clasificarse de varias maneras, de acuerdo al criterio elegido para su clasificación:

SEGÚN LA VARIABILIDAD DE LOS DATOS ALMACENADOS

Bases de datos estáticas

Éstas son bases de datos de sólo lectura, utilizadas primordialmente para almacenar datos históricos que posteriormente se pueden utilizar para estudiar el comportamiento de un conjunto de datos a través del tiempo, realizar proyecciones y tomar decisiones.

Bases de datos dinámicas

Éstas son bases de datos donde la información almacenada se modifica con el tiempo, permitiendo operaciones como actualización y adición de datos, además de las operaciones fundamentales de consulta. Un ejemplo de esto puede ser la base de datos utilizada en un sistema de información de una tienda de abarrotes, una farmacia, un videoclub, etc.

SEGÚN EL CONTENIDO

Bases de datos bibliográficas

Solo contienen un representante de la fuente primaria, que permite localizarla. Un registro típico de una base de datos bibliográfica contiene información sobre el autor, fecha de publicación, editorial, título, edición, de una determinada publicación, etc. Puede contener un resumen o extracto de la publicación original, pero nunca el texto completo, porque de lo contrario estaríamos en presencia de una base de datos a texto completo (o de fuentes primarias—ver más abajo). Como su nombre lo indica, el contenido son cifras o números. Por ejemplo, una colección de resultados de análisis de laboratorio, entre otras.

Bases de datos de texto completo

Almacenan las fuentes primarias, como por ejemplo, todo el contenido de todas las ediciones de una colección de revistas científica.

Directorios

Un ejemplo son las guías telefónicas en formato electrónico.

Bases de datos o "bibliotecas" de información Biológica

Son bases de datos que almacenan diferentes tipos de información proveniente de las ciencias de la vida o médicas. Se pueden considerar en varios subtipos:

- ✓ Aquellas que almacenan secuencias de nucleótidos o proteínas.
- ✓ Las bases de datos de rutas metabólicas
- ✓ Bases de datos de estructura, comprende los registros de datos experimentales sobre estructuras 3D de biomoléculas
- ✓ Bases de datos clínicas
- ✓ Bases de datos bibliográficas (biológicas).

➤ ARCHIVOS DE BASE DE DATOS

SQL Server maneja tres tipos de archivos básicos para dividir lo que es la data física y los registros de transacciones, llamados archivos lógicos. Veamos.

✓ Archivos de datos principales

El archivo de datos principal es el punto de partida de la base de datos y apunta a los otros archivos de la base de datos. Cada base de datos tiene un archivo de datos principal. La extensión recomendada para los nombres de archivos de datos principales es .mdf.

✓ Archivos de datos secundarios

Los archivos de datos secundarios son todos los archivos de datos menos el archivo de datos principal. Puede que algunas bases de datos no tengan archivos de datos secundarios, mientras que otras pueden tener varios archivos de datos secundarios. La extensión de nombre de archivo recomendada para los archivos de datos secundarios es .ndf.

✓ Archivos de registro

Los archivos de registro almacenan toda la información de registro que se utiliza para recuperar la base de datos. Como mínimo, tiene que haber un archivo de registro por cada base de datos, aunque puede haber varios. La extensión de nombre de archivo recomendada para los archivos de registro es .ldf.

SQL Server no exige las extensiones de nombre de archivo .mdf, .ndf y .ldf, pero estas extensiones ayudan a identificar las distintas clases de archivos y su uso.

En SQL Server, las ubicaciones de todos los archivos de una base de datos se guardan tanto en el archivo principal de la base de datos como en la base de datos maestra. SQL Server Database Engine (Motor de base de datos de SQL Server) utiliza casi siempre la información de ubicación del archivo de la base de datos maestra. Sin embargo, Database Engine (Motor de base de datos) utiliza la información de ubicación del archivo principal para inicializar las entradas de ubicación de archivos de la base de datos maestra en las siguientes situaciones:

- Al adjuntar una base de datos mediante la instrucción CREATE DATABASE con la opción FOR ATTACH o la opción FOR ATTACH_REBUILD_LOG.
- Al actualizar desde SQL Server versión 2000 o versión 7.0
- Al restaurar la base de datos maestra.

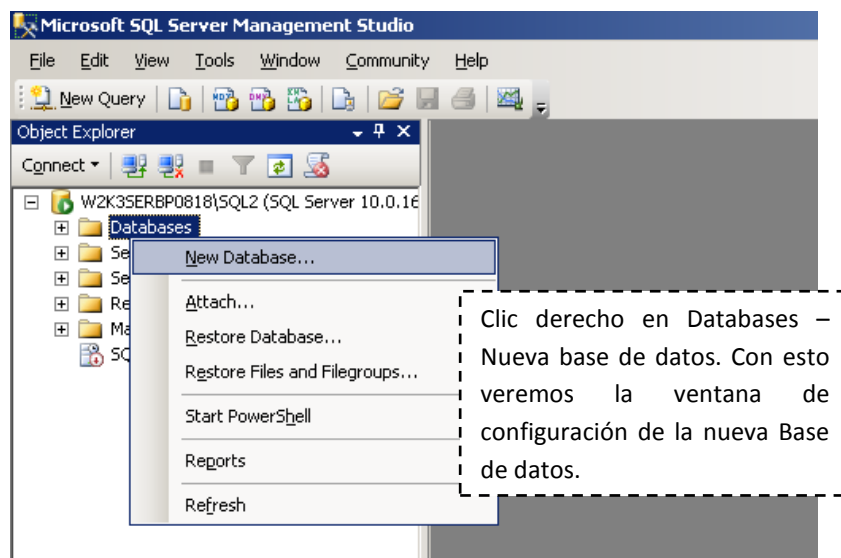
El usuario que crea una base de datos en SQL Server se hace propietario de la BD automáticamente, asimismo los nombres de las bases de datos deben respetar las reglas de todo identificador, como el no utilizar caracteres especiales.

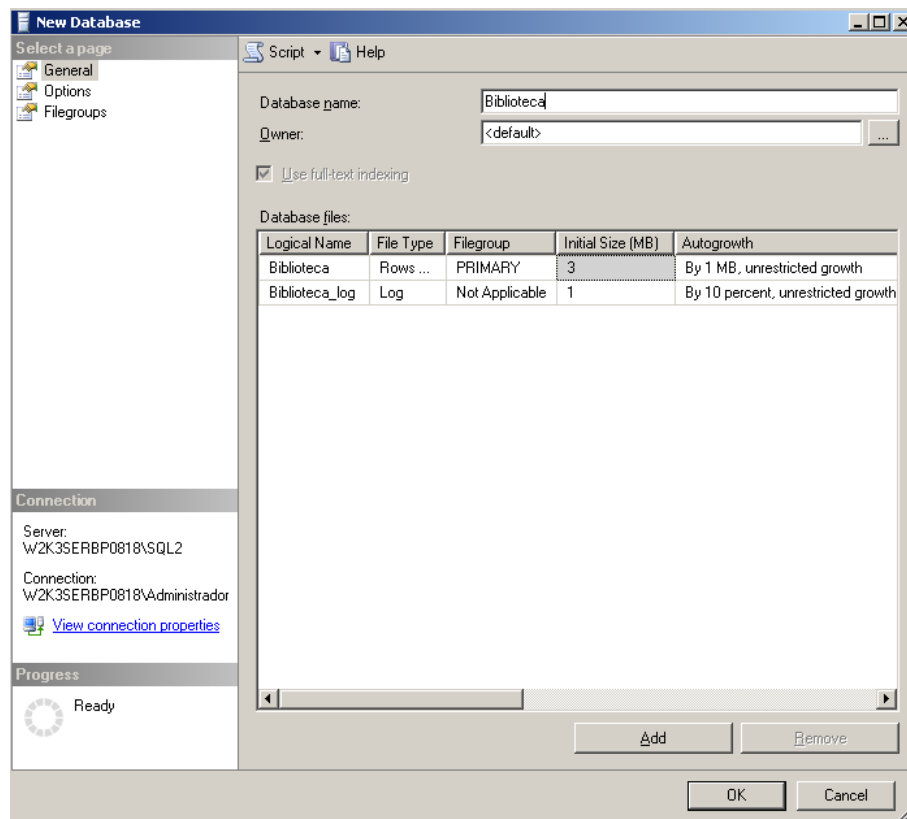
➤ CREACION DE UNA BD EN SQL SERVER 2008

Para ingresar a SQL Server nos vamos a Inicio – Programas – Microsoft SQL Server 2008 – SQL Server Management Studio...

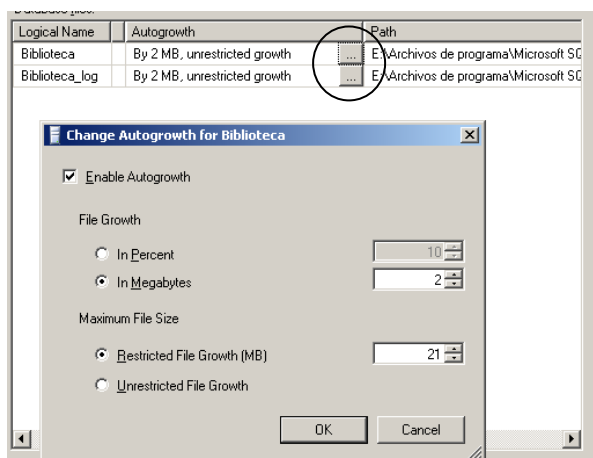


Cuando hayamos accedido el entorno, veremos que encontramos un Explorador de objetos, muy parecido al Explorador de Windows.



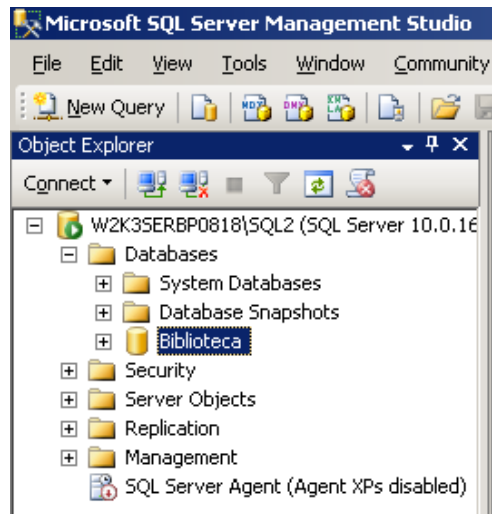


- Colocamos el nombre de la BD en la opción Database name.
- Owner, se refiere al propietario que por defecto es el usuario actual.
- Vemos los dos tipos de archivos: básico, el archivo de datos y el archivo lógico. Por defecto el nombre del archivo de datos es el mismo de la base de datos, mientras que el nombre del archivo de transacciones tiene la extensión '_log'.
- El tamaño inicial por defecto del archivo de datos es de 3 MB, mientras que el tamaño por defecto del archivo lógico de transacciones es de 1 MB.
- SQL Server almacena la base de datos dentro de la carpeta Data que crea al momento de la instalación del servidor.
- Autogrowth, se refiere al incremento que tendrá la base de datos cuando se llegue al tamaño máximo asignado, podemos seleccionar dos tipos de incremento:



El crecimiento puede ser en porcentaje o en Megabytes, mientras que el tamaño máximo puede tener un número predeterminado de bytes, lo determina el usuario, o en todo caso puede tener un crecimiento irrestricto, es decir sin límites. El archivo de registros de transacciones debe tener un tamaño menor o igual que el archivo de datos.

Una vez configurado las opciones de la nueva base de datos, veremos la carpeta respectiva de la base de datos creada en el Explorador de objetos.



Cuando Ud. crea una base de datos, deberá especificar el tamaño máximo que un archivo tiene autorizado a alcanzar. Esto previene que el archivo crezca, cuando los datos son ingresados, hasta que el espacio en disco se termine.

SQLServer implementa una nueva base de datos en dos pasos:

- SQLServer usa una copia de la base de datos “Model” para inicializar la nueva base de datos y sus metadatos.
- SQLServer luego llena el resto de la base de datos con páginas vacías (excepto aquellas páginas que tienen grabados datos internos como el espacio usado)

Cualquier objeto definido por el usuario en la base de datos “Model” es copiado a todas las bases de datos que sean creadas. Se pueden agregar objetos a la base de datos “Model”, tales como tablas, vistas, procedimientos almacenados, tipos de datos, etc. que serán incluidos en las nuevas bases de datos. Además, cada nueva base de datos hereda la configuración de las opciones de la base de datos “Model”.

✓ **Bases de datos del Sistema**

A continuación describimos las bases de datos del sistema de SQL Server.

Base de datos maestra

Registra toda la información del sistema para una instancia de SQL Server.

Msdb

La utiliza el Agente SQL Server para programar alertas y trabajos.

Model

Se utiliza como plantilla para todas las bases de datos creadas en la instancia de SQL Server. Las modificaciones hechas a la base de datos model, como el tamaño de la base de datos, la intercalación, el modelo de recuperación y otras opciones de base de datos, se aplicarán a las bases de datos que se creen con posterioridad.

Resource

Base de datos de sólo lectura que contiene objetos del sistema que se incluyen con SQL Server. Los objetos del sistema persisten físicamente en la base de datos Resource, pero aparecen lógicamente en el esquema sys de cada base de datos.

Tempdb

Área de trabajo que contiene objetos temporales o conjuntos de resultados intermedios.

➤ CREACION DE BASE DE DATOS (DDL)

✓ EL LENGUAJE SQL

El Lenguaje de consulta estructurado (SQL - Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre las mismas. Una de sus características es el manejo del álgebra y el cálculo relacional permitiendo lanzar consultas con el fin de recuperar -de una forma sencilla- información de interés de una base de datos, así como también hacer cambios sobre la misma. Es un lenguaje de cuarta generación (4GL).

El SQL es un lenguaje de acceso a bases de datos que explota la flexibilidad y potencia de los sistemas relacionales permitiendo gran variedad de operaciones sobre los mismos.

Es un lenguaje declarativo de "alto nivel" o "de no procedimiento", que gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros, y no a registros individuales, permite una alta productividad en codificación y la orientación a objetos. De esta forma una sola sentencia puede equivaler a uno o más programas que utilizas en un lenguaje de bajo nivel orientado a registro.

✓ LENGUAJE DE DEFINICIÓN DE DATOS (DDL)

El lenguaje de definición de datos (en inglés Data Definition Language, o DDL), es el que se encarga de la modificación de la estructura de los objetos de la base de datos. Existen cuatro operaciones básicas: CREATE, ALTER, DROP y TRUNCATE.

CREATE

Este comando crea un objeto dentro de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte. Permite crear además la misma base de datos.

ALTER

Este comando permite modificar la estructura de un objeto. Se pueden agregar/quitar campos a una tabla, modificar el tipo de un campo, agregar/quitar índices a una tabla, modificar un trigger, etc.

DROP

Este comando elimina un objeto de la base de datos. Puede ser una tabla, vista, índice, trigger, función, procedimiento o cualquier otro objeto que el motor de la base de datos soporte. Se puede combinar con la sentencia ALTER.

TRUNCATE

Este comando trunca todo el contenido de una tabla. La ventaja sobre el comando DELETE, es que si se quiere borrar todo el contenido de la tabla, es mucho más rápido, especialmente si la tabla es muy grande, la desventaja es que TRUNCATE solo sirve cuando se quiere eliminar absolutamente todos los registros, ya que no se permite la cláusula WHERE. Si bien, en un principio, esta sentencia parecería ser DML (Lenguaje de Manipulación de Datos), es en realidad una DDL, ya que internamente, el comando truncate borra la tabla y la vuelve a crear y no ejecuta ninguna transacción.

✓ CREAR UNA BASE DE DATOS CON CÓDIGO SQL

Para crear una base de datos lo haremos de la siguiente forma:

CREATE DATABASE **NOMBRE_BD**

Ejemplo: Create database Prueba

Si creamos la base de datos sólo con esta instrucción (Seleccionamos la línea de instrucción y presionamos el botón Execute ó F5), la base de datos se creará con los valores por defecto en sus propiedades:

- Tamaño inicial del archivo de datos: 3 MB.
- Tamaño inicial del archivo de Transacciones o de registros: 1 MB.
- Nombre del archivo de datos idéntico al de la BD, mientras que al archivo lógico se le agrega el prefijo '_log'.
- Crecimiento ilimitado en Megabytes con un tamaño máximo indefinido.

Para asignar un valor personalizado a cada uno de los elementos de la base de datos, debemos utilizar las siguientes instrucciones:

- Name: Establece el nombre del archivo de datos primario, secundario y de registros.
- Filename: Establece la ruta de almacenamiento de la base de datos, indicando la extensión para cada archivo (mdf, ndf, ldf).
- Size: Establece el tamaño inicial de la base de datos. La suma de los tamaños iniciales de los archivos creados, indicará el tamaño inicial total de la base de datos.
- Maxsize: Establece el tamaño máximo inicial que tendrá la base de datos. La suma de los tamaños máximos de los archivos creados, indicará el tamaño máximo total de la base de datos.
- Filegrowth: Establece una razón de crecimiento a la base de datos cuando ésta llegue a su tamaño máximo asignado, puede ser establecido en MB (incremento fijo) o en porcentajes (incremento según el volumen de datos).

Ejemplo 1:

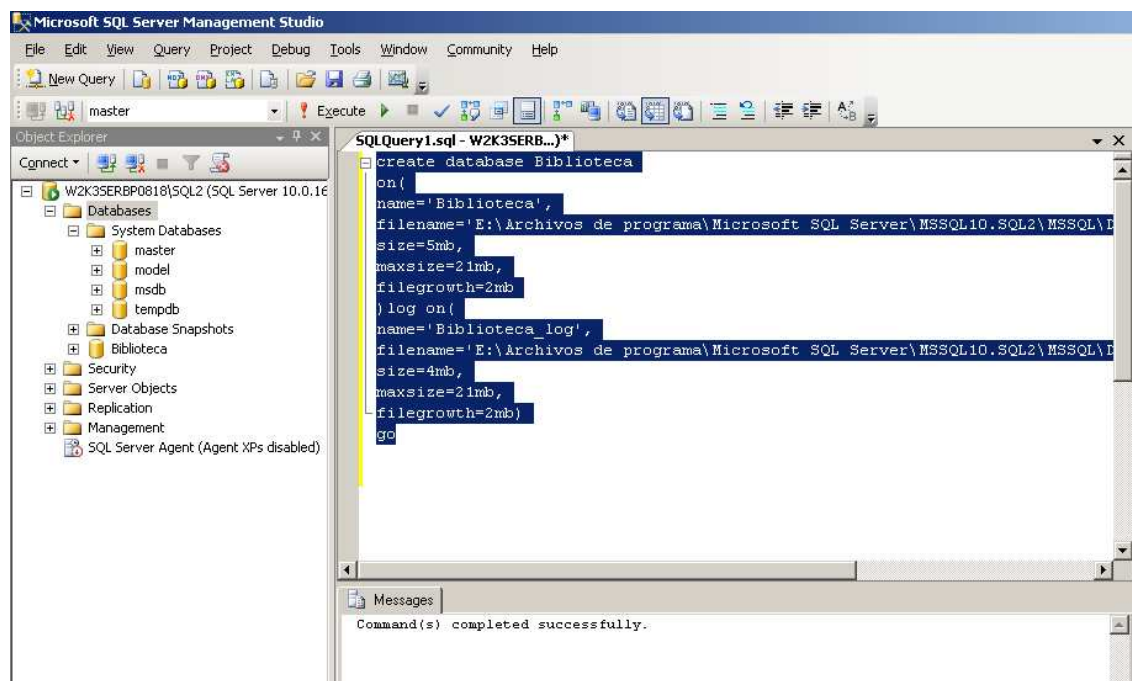
Crear una base de datos llamada Biblioteca, con un tamaño inicial de 9 Mb, un tamaño máximo de 42 Mb y un crecimiento de 4 Mb.

Para escribir el código SQL, debemos conectarnos al servidor)SQL Server Management Studio), una vez conectado hacemos clic en el botón Nueva consulta, con esto creamos un nuevo script cuya base de datos por defecto será Master, ahí escribiremos el siguiente código que se muestra...

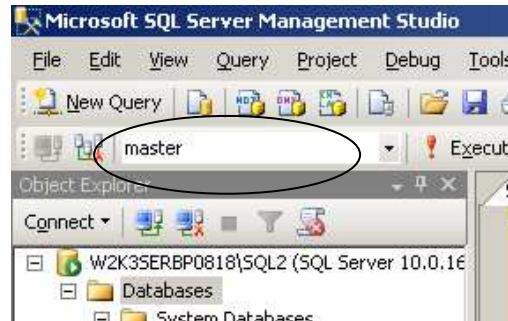
Hacemos click en el botón New Query (Nueva consulta)

```
create database Biblioteca
on(
name='Biblioteca',
filename='E:\Archivos de programa\Microsoft SQL Server\MSSQL10.SQL2\MSSQL\DATA\Biblioteca.mdf',
size=5mb,
maxsize=21mb,
filegrowth=2mb
) log on(
name='Biblioteca_log',
filename='E:\Archivos de programa\Microsoft SQL Server\MSSQL10.SQL2\MSSQL\DATA\biblioteca_log.ldf',
size=4mb,
maxsize=21mb,
filegrowth=2mb)
go
```

Al seleccionar todo el código escrito, presionar F5 o clic en el botón Execute (Ejecutar), se verá en pantalla el siguiente mensaje...



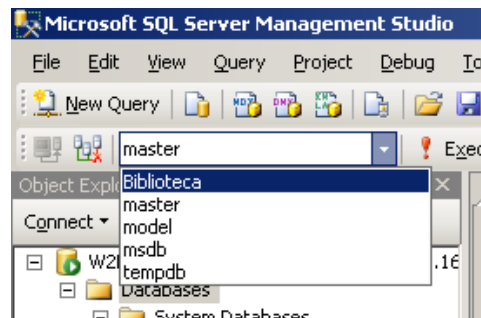
Esto nos indica que la base de datos ya está creada, el servidor asignó el espacio en disco para cada uno de los archivos creado, en este caso tenemos dos archivos asignados: el archivo de datos y el archivo de registros de transacciones. Sin embargo todavía estamos conectados a la base de datos Master...



Para poder utilizar la base de datos Biblioteca, debemos seleccionarla del combo de opciones o escribir y ejecutar la siguiente instrucción SQL...

Use Biblioteca

Esto indica que vamos a activar la base de datos para trabajar en ella (use nombre_BD).



Ejemplo 2:

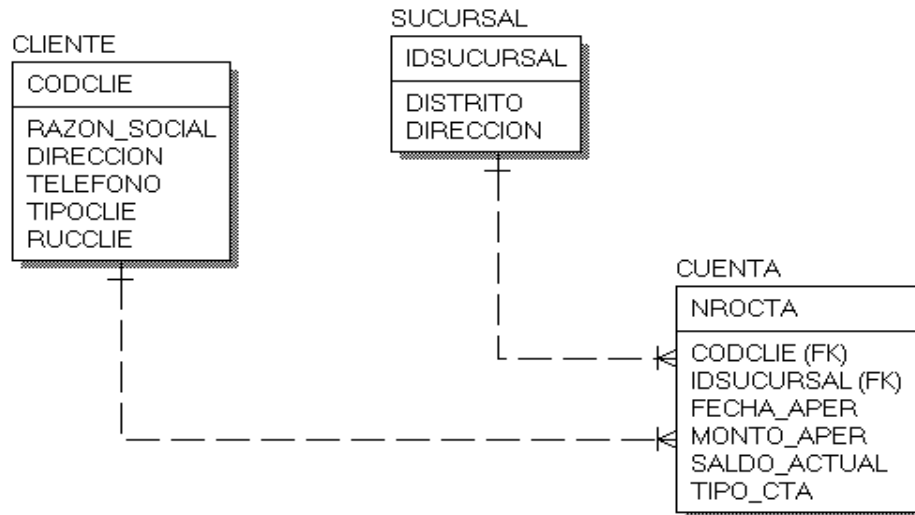
Crear una base de datos llamada Colegio2010 con un tamaño inicial de 6 Mb, un tamaño máximo de 9 Mb y un crecimiento porcentual de 10%.

```
create Database Colegio2010
on(
name=Colegio2010_dat,
filename='D:\Archivos de Programa\Microsoft SQL Server\MSSQL.3\MSSQL\Data\
Colegio2010_dat.mdf',
size= 4MB,
maxsize= 5MB,
filegrowth = 10%
)
log on (
name = Apafa6_log,
filename = 'D:\Archivos de Programa\Microsoft SQL Server\MSSQL.3\MSSQL\Data\
Colegio2010_dat.ldf',
size = 2MB,
maxsize = 4MB,
filegrowth = 10%
)
```

➤ INGENIERIA DIRECTA (FORWARD ENGINEER)

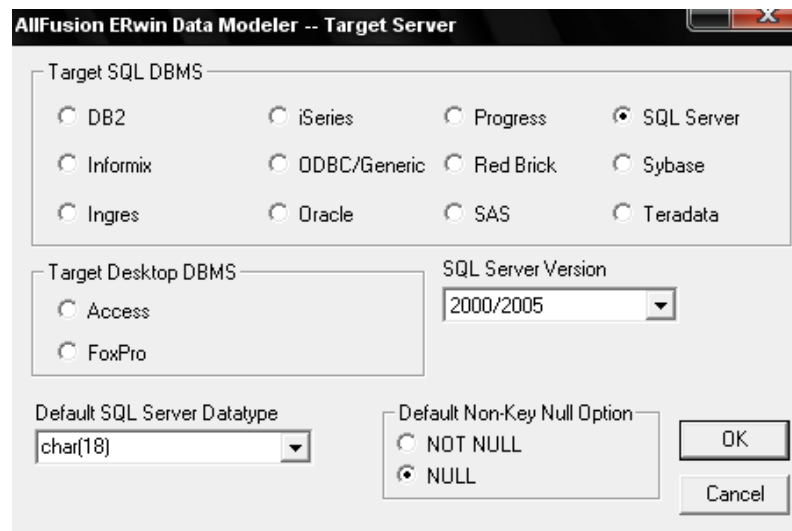
Vamos a realizar la exportación de un diseño físico en Erwin hacia SQL Server, para ello debemos tener preparado nuestro modelo físico tal como se muestra a continuación.

1. Primero diseñamos nuestro modelo entidad relación en Erwin...



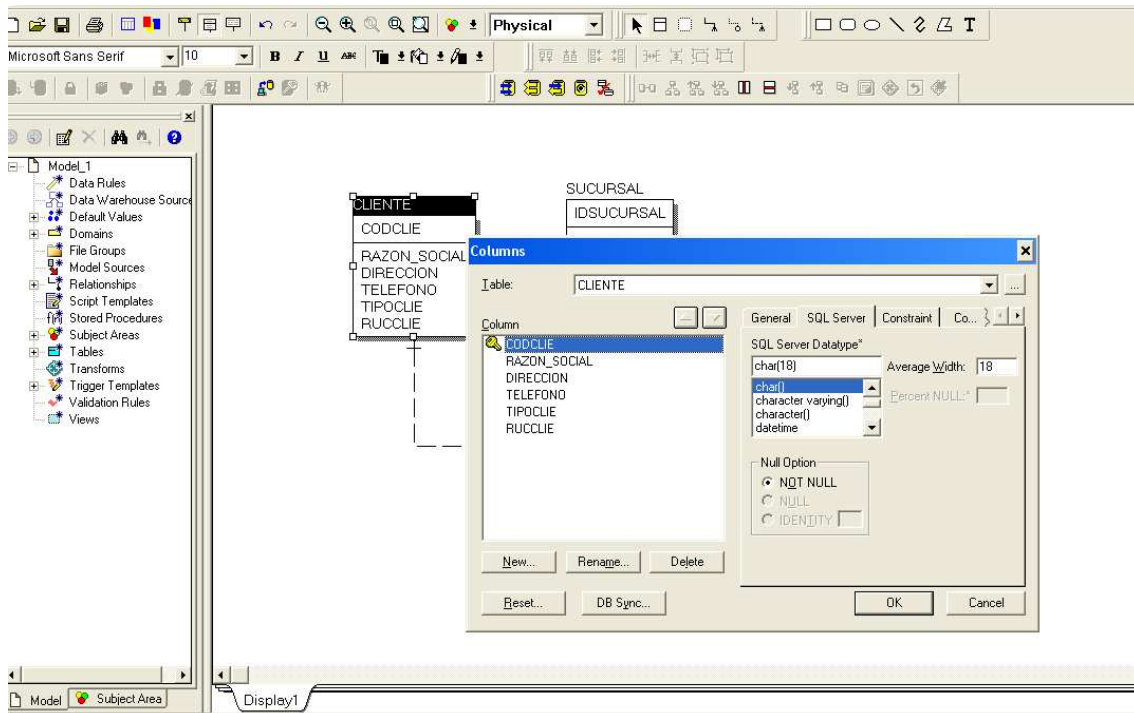
Para nuestro ejemplo coloqué un pequeño modelo de 3 tablas para realizar la exportación.

2. Seleccionamos el servidor de base de datos con el que iremos a trabajar, para ello nos vamos al menú Database – Choose Database. Tras lo cual se nos aparecerá la siguiente ventana ya antes mostrada...

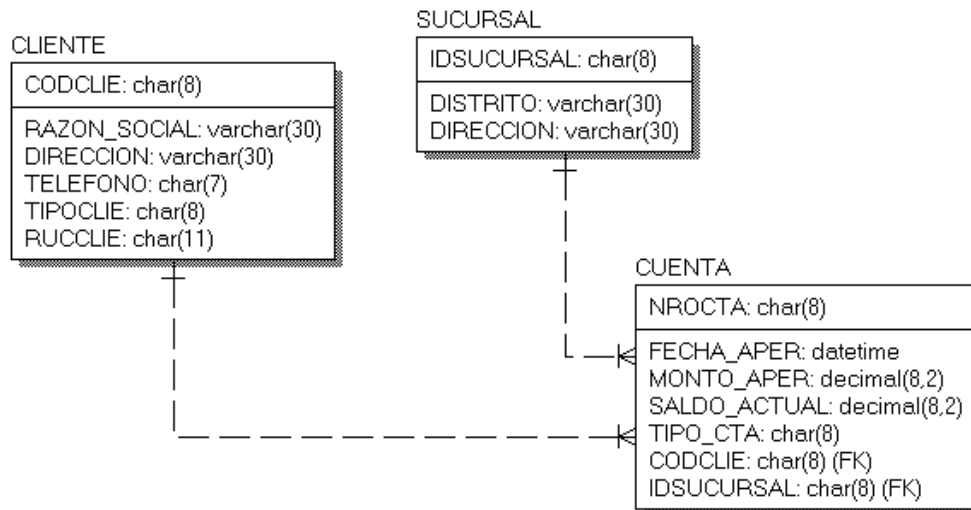


En este caso seleccionamos SQL Server que es el gestor de datos con el que estamos trabajando, si en caso el gestor que deseamos no existe en la lista o no encontramos la versión actual, entonces debemos utilizar la opción del ODBC, que es una conexión abierta de base de datos.

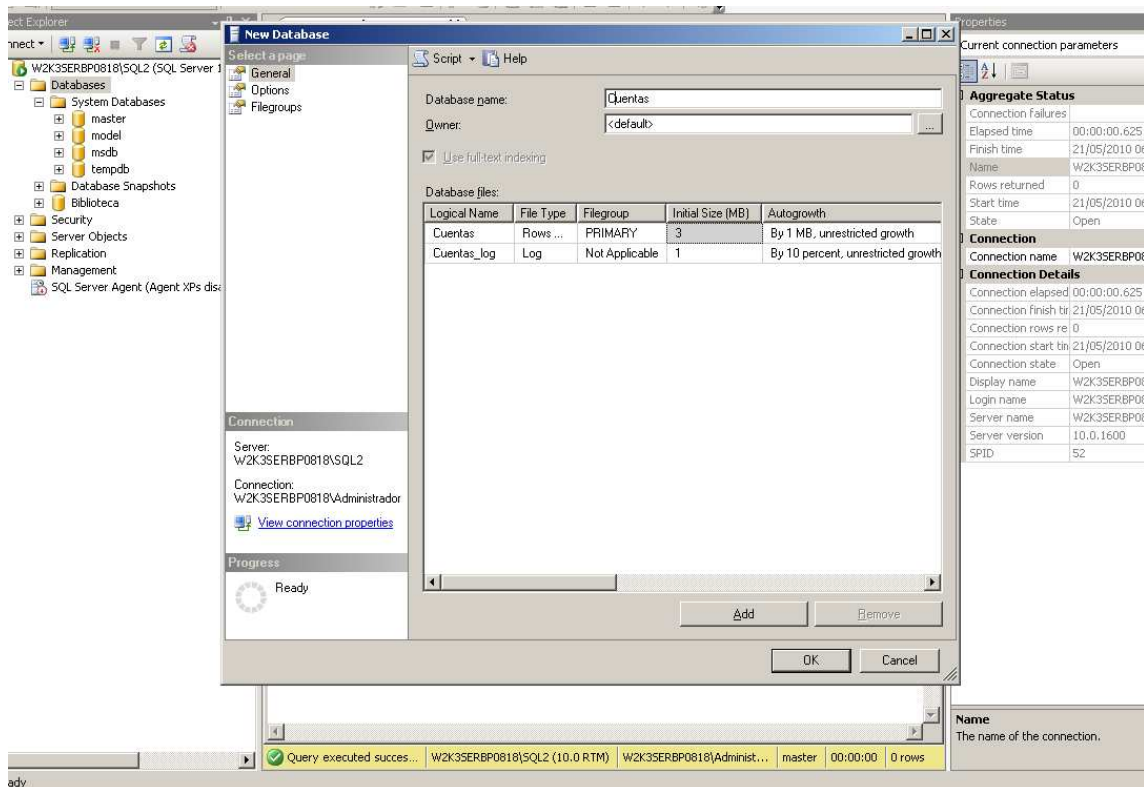
3. Editamos los tipos de datos de todas las columnas de las tablas, para ello nos vamos al Editor de columna haciendo clic derecho sobre las tablas y seleccionando la opción Column...



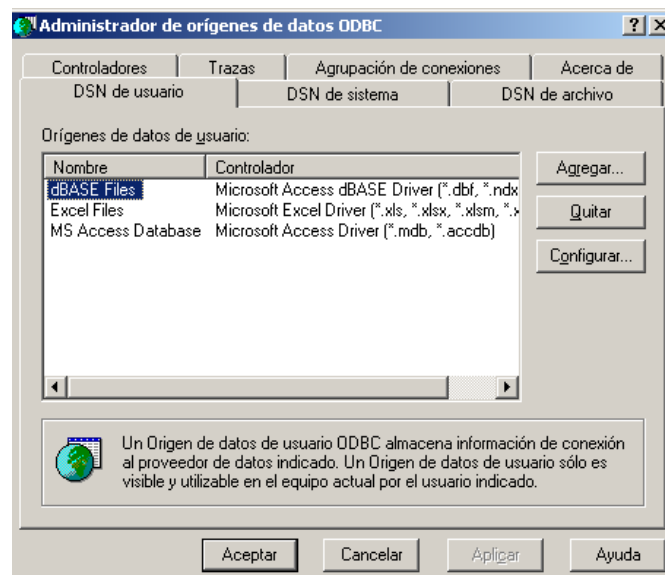
Al hacer clic en un espacio vacío del modelo físico, seleccionamos Table Display, ahí elegimos Column Datatype, se mostrará los tipos de datos elegidos a cada columna.



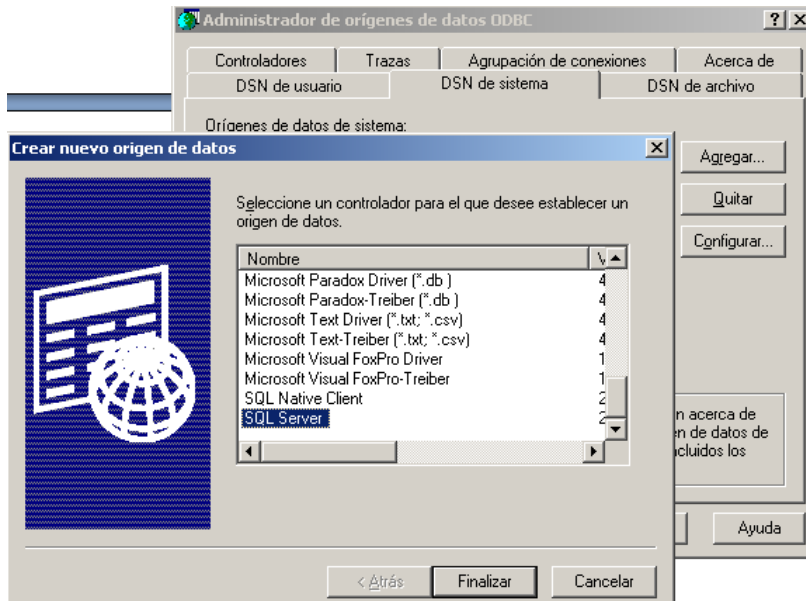
4. Ahora nos conectamos a nuestro servidor de Base de datos desde Inicio – Programas – Microsoft SQL Server 2005 – SQL Server Management Studio. Nos conectaremos con el usuario 'sa' de SQL Server Autenticación. Creamos una base de datos en blanco.



5. Para poder hacer la conexión entre Erwin y SQL Server necesitamos de un ODBC, para ello debemos crear uno desde el menú Inicio – Configuración – Panel de Control. Dentro elegimos Herramientas Administrativas – Orígenes de datos ODBC.

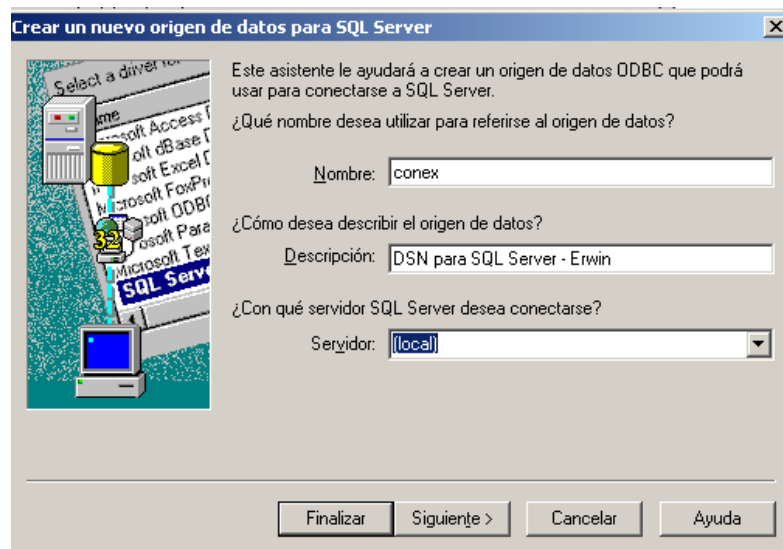


Nos ubicamos en la pestaña DSN de Sistema, en donde agregamos un nuevo DSN desde el botón Agregar...




Debemos seleccionar un controlador al cual estableceremos el origen de datos, lógicamente seleccionamos SQL Server, hacemos clic en Finalizar...

Ahora debemos colocar el nombre al DSN, puede ser cualquier nombre, igualmente le agregamos una descripción a nuestro DSN (opcional). Algo muy importante es el servidor con el que debemos conectarnos(servidor local), seleccionamos el servidor disponible y con el cual nos conectamos al motor de la base de datos.



Luego seleccionamos la forma de comprobación del id de usuario de inicio de sesión, debemos seleccionar Autenticación de SQL Server, con el usuario 'sa'.

Crear un nuevo origen de datos para SQL Server



¿Cómo desea que SQL Server compruebe la autenticidad del Id. de inicio de sesión?

☐ Con la autenticación de Windows NT, mediante el Id. de inicio de sesión de red.

☒ Con la autenticación de SQL Server, mediante un Id. de inicio de sesión y una contraseña escritos por el usuario.

Para cambiar la biblioteca de red usada para comunicarse con SQL Server, haga clic en Configuración del cliente.

[Configuración del cliente...](#)

☒ Conectar con SQL Server para obtener la configuración predeterminada de las opciones de configuración adicionales.


Id. de inicio de sesión:

Contraseña:

[< Atrás](#)
[Siguiente >](#)
[Cancelar](#)
[Ayuda](#)

Luego de ello, seleccionamos la base de datos en blanco que creamos para que sea la predeterminada, la que finalmente recibirá las tablas generadas desde Erwin.

Crear un nuevo origen de datos para SQL Server



☒ Establecer la siguiente base de datos como predeterminada:

☐ Adjuntar nombre del archivo de la base de datos:

☒ Crear procedimientos almacenados temporales para instrucciones SQL preparadas y eliminar los procedimientos almacenados:

☒ Sólo al desconectar.

☐ Al desconectar y cuando sea conveniente mientras esté conectado.

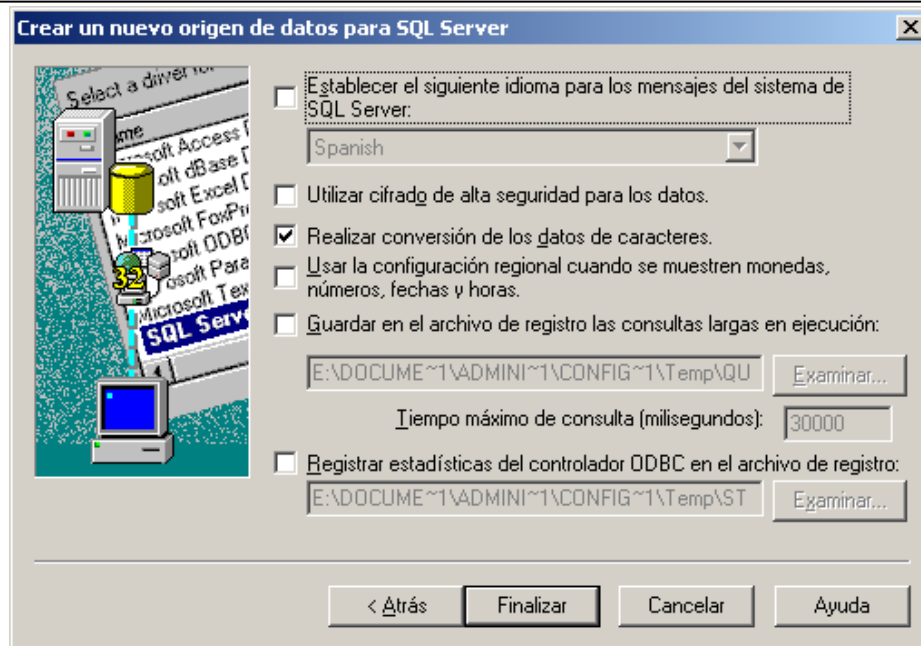
☒ Usar identificadores entrecomillados ANSI.

☒ Usar nulos, rellenos y advertencias ANSI.

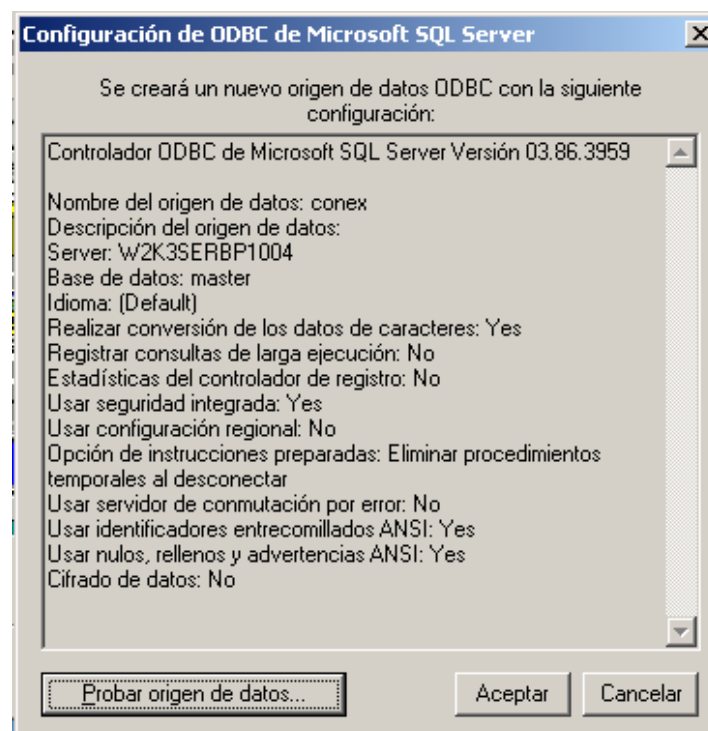
☐ Usar el servidor SQL Server de conmutación por error si el servidor SQL Server primario no se encuentra disponible.

[< Atrás](#)
[Siguiente >](#)
[Cancelar](#)
[Ayuda](#)

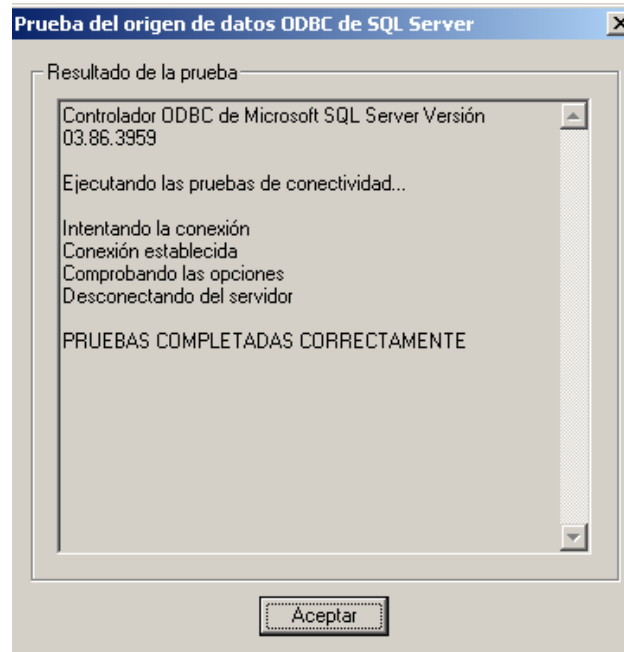
Establecemos la configuración del idioma y otros formatos de registro y estadísticas. Le damos clic en Finalizar.



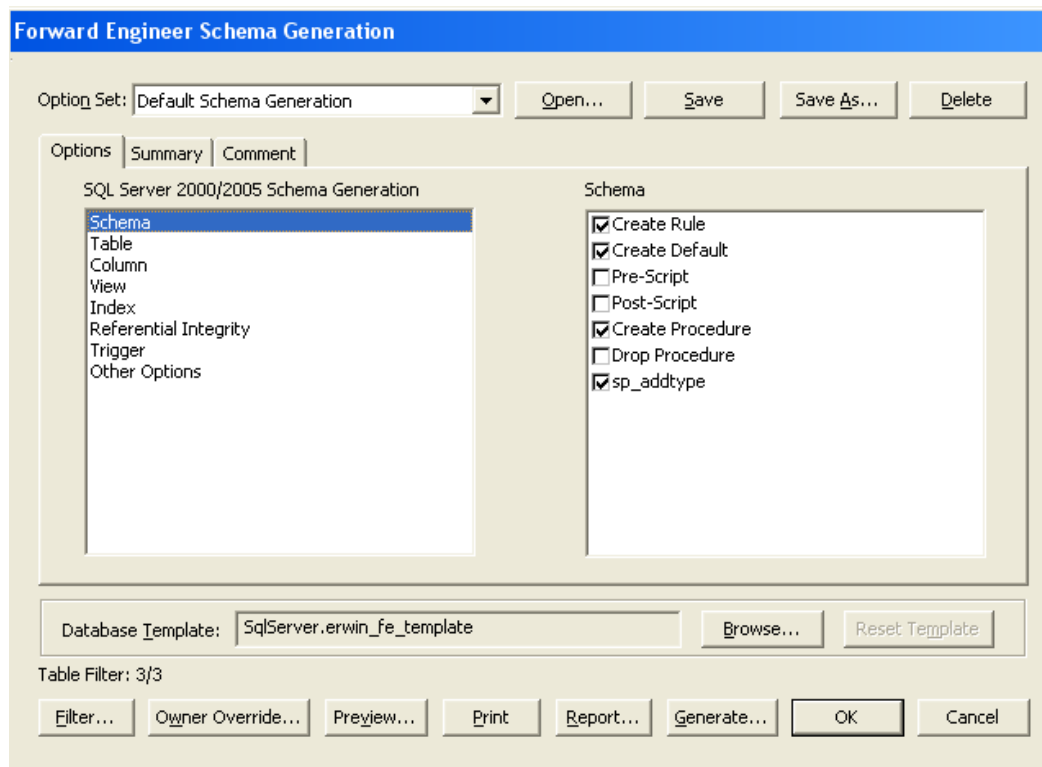
Podemos probar la conexión para ver si hemos cometido algún error, o si no tenemos una conexión disponible



Si todo sale bien, debe mostrar el mensaje Pruebas completadas correctamente...



6. Generamos la base de datos una vez creado el origen de datos ODBC, para ello en Erwin nos vamos al menú Tools – Forward Engineer – Schema Generation...



Seleccionamos los elementos que deseamos crear en la base de datos, como por ejemplo el esquema de la base de datos, las tablas a exportar, las columnas de cada tabla, los índices, etc. Cuando hayamos terminado le damos clic en Generate para generar en el servidor la base de datos.

IMPORTANTE

Para poder ejecutar la ingeniería directa hacia SQL Server, debemos tener instalado el software cliente de SQL Server, el Msdn Native Client que se instala en el sistema al momento de la instalación del servidor.

SQL Server Native Client es una interfaz de programación de aplicaciones (API) de acceso a datos independiente que se introdujo en SQL Server 2005 y que se utiliza tanto para OLE DB como para ODBC. SQL Server Native Client combina el proveedor OLE DB de SQL y el controlador ODBC de SQL en una biblioteca de vínculos dinámicos (DLL) nativa. También ofrece muchas más funciones nuevas de las que se proporcionaban en Data Access Components para Windows (DAC para Windows, anteriormente Microsoft Data Access Components o MDAC). Puede utilizar SQL Server Native Client para crear nuevas aplicaciones o mejorar las existentes incorporando las características introducidas en SQL Server 2005, como la compatibilidad con conjuntos de resultados activos múltiples (MARS), los tipos de datos definidos por el usuario (UDT), las notificaciones de consulta, el aislamiento de instantánea y el tipo de datos XML.

El controlador ODBC de SQL Server Native Client siempre se utiliza junto con el administrador de controladores ODBC que se proporciona con DAC para Windows. El proveedor OLE DB de SQL Server Native Client puede utilizarse junto con los servicios principales de OLE DB que se proporcionan con DAC para Windows, pero no se trata de un requisito; la opción de usar o no los servicios principales depende de los requisitos de la aplicación individual (por ejemplo, si se requiere la agrupación de conexiones).

Contenido:

- ✓ Construcción del DER normalizado
- ✓ Pasos para obtener el MER a partir de un DER.
- ✓ Práctica Calificada 02 - Teoría
- ✓ Normalizar un documento en sus tres formas
- ✓ Creación del DER normalizado
- ✓ Práctica Calificada 02 – Practico
- ✓ Creación de Tablas de Datos
- ✓ Identificando los tipos de datos empleados en SQL Server 2005
- ✓ Restricciones Primary Key, Default, Check, Unique, Nulos, Identidades, Foreign Keys.

MODELO LÓGICO GLOBAL – CREACIÓN DE TABLAS (DDL)

➤ CONSTRUCCIÓN DEL DER NORMALIZADO

Describiremos los pasos para llevar a cabo el diseño lógico. Ya que aquí se trata el diseño de bases de datos relacionales, en esta etapa se obtiene un conjunto de relaciones (tablas) que representen los datos de interés. Este conjunto de relaciones se valida mediante la normalización, técnica que se estudia al final del capítulo.

El objetivo del diseño lógico es convertir los esquemas conceptuales locales en un esquema lógico global que se ajuste al modelo de SGBD sobre el que se vaya a implementar el sistema. Mientras que el objetivo fundamental del diseño conceptual es la compleción y expresividad de los esquemas conceptuales locales, el objetivo del diseño lógico es obtener una representación que use, del modo más eficiente posible, los recursos que el modelo de SGBD posee para estructurar los datos y para modelar las restricciones

Los modelos de bases de datos más extendidos son el modelo relacional, el modelo de red y el modelo jerárquico. El modelo orientado a objetos es también muy popular, pero no existe un modelo estándar orientado a objetos.

El modelo relacional (y los modelos previos) carecen de ciertos rasgos de abstracción que se usan en los modelos conceptuales. Por lo tanto, un primer paso en la fase del diseño lógico consistirá en la conversión de esos mecanismos de representación de alto nivel en términos de las estructuras de bajo nivel disponibles en el modelo relacional.

➤ **PASOS PARA OBTENER UN MER A PARTIR DE UN DER**

La metodología que se va a seguir para el diseño lógico en el modelo relacional consta de dos fases, cada una de ellas compuesta por varios pasos que se detallan a continuación.

✓ **CONSTRUIR Y VALIDAR LOS ESQUEMAS LÓGICOS LOCALES PARA CADA VISTA DE USUARIO.**

1. Convertir los esquemas conceptuales locales en esquemas lógicos locales.
2. Derivar un conjunto de relaciones (tablas) para cada esquema lógico local.
3. Validar cada esquema mediante la normalización.
4. Validar cada esquema frente a las transacciones del usuario.
5. Dibujar el diagrama entidad-relación.
6. Definir las restricciones de integridad.
7. Revisar cada esquema lógico local con el usuario correspondiente.

✓ **CONSTRUIR Y VALIDAR EL ESQUEMA LÓGICO GLOBAL.**

8. Mezclar los esquemas lógicos locales en un esquema lógico global.
9. Validar el esquema lógico global.
10. Estudiar el crecimiento futuro.
11. Dibujar el diagrama entidad-relación final.
12. Revisar el esquema lógico global con los usuarios.

En la primera fase, se construyen los esquemas lógicos locales para cada vista de usuario y se validan. En esta fase se refinan los esquemas conceptuales creados durante el diseño conceptual, eliminando las estructuras de datos que no se pueden implementar de manera directa sobre el modelo que soporta el SGBD, en el caso que nos ocupa, el modelo relacional. Una vez hecho esto, se obtiene un primer esquema lógico que se valida mediante la normalización y frente a las transacciones que el sistema debe llevar a cabo, tal y como se refleja en las especificaciones de requisitos de usuario. El esquema lógico ya validado se puede utilizar como base para el desarrollo de prototipos. Una vez finalizada esta fase, se dispone de un esquema lógico para cada vista de usuario que es correcto, comprensible y sin ambigüedad.

1. Convertir los esquemas conceptuales locales en esquemas lógicos locales

En este paso, se eliminan de cada esquema conceptual las estructuras de datos que los sistemas relacionales no modelan directamente:

(a) Eliminar las relaciones de muchos a muchos, sustituyendo cada una de ellas por una nueva entidad intermedia y dos relaciones de uno a muchos de esta nueva entidad con las entidades originales. La nueva entidad será débil, ya que sus ocurrencias dependen de la existencia de ocurrencias en las entidades originales.

(b) Eliminar las relaciones entre tres o más entidades, sustituyendo cada una de ellas por una nueva entidad (débil) intermedia que se relaciona con cada una de las entidades originales. La cardinalidad de estas nuevas relaciones binarias dependerá de su significado.

(c) Eliminar las relaciones recursivas, sustituyendo cada una de ellas por una nueva entidad (débil) y dos relaciones binarias de esta nueva entidad con la entidad original. La cardinalidad de estas relaciones dependerá de su significado.

(d) Eliminar las relaciones con atributos, sustituyendo cada una de ellas por una nueva entidad (débil) y las relaciones binarias correspondientes de esta nueva entidad con las entidades originales. La cardinalidad de estas relaciones dependerá del tipo de la relación original y de su significado.

(e) Eliminar los atributos multievaluados, sustituyendo cada uno de ellos por una nueva entidad (débil) y una relación binaria de uno a muchos con la entidad original.

(f) Revisar las relaciones de uno a uno, ya que es posible que se hayan identificado dos entidades que representen el mismo objeto (sinónimos). Si así fuera, ambas entidades deben integrarse en una sola.

(g) Eliminar las relaciones redundantes. Una relación es redundante cuando se puede obtener la misma información que ella aporta mediante otras relaciones. El hecho de que haya dos caminos diferentes entre dos entidades no implica que uno de los caminos corresponda a una relación redundante, eso dependerá del significado de cada relación.

Una vez finalizado este paso, es más correcto referirse a los esquemas conceptuales locales refinados como esquemas lógicos locales, ya que se adaptan al modelo de base de datos que soporta el SGBD escogido.

2. Derivar un conjunto de relaciones (tablas) para cada esquema lógico local

En este paso, se obtiene un conjunto de relaciones (tablas) para cada uno de los esquemas lógicos locales en donde se representen las entidades y relaciones entre entidades, que se describen en cada una de las vistas que los usuarios tienen de la empresa. Cada relación de la base de datos tendrá un nombre, y el nombre de sus atributos aparecerá, a continuación, entre paréntesis. El atributo o atributos que forman la clave primaria se subrayan. Las claves ajenas, mecanismo que se utiliza para representar las relaciones entre entidades en el modelo relacional, se especifican aparte indicando la relación (tabla) a la que hacen referencia.

A continuación, se describe cómo las relaciones (tablas) del modelo relacional representan las entidades y relaciones que pueden aparecer en los esquemas lógicos.

(a) Entidades fuertes. Crear una relación para cada entidad fuerte que incluya todos sus atributos simples. De los atributos compuestos incluir sólo sus componentes.

Cada uno de los identificadores de la entidad será una clave candidata. De entre las claves candidatas hay que escoger la clave primaria; el resto serán claves alternativas. Para escoger la clave primaria entre las claves candidatas se pueden seguir estas indicaciones:

- Escoger la clave candidata que tenga menos atributos.
- Escoger la clave candidata cuyos valores no tengan probabilidad de cambiar en el futuro.
- Escoger la clave candidata cuyos valores no tengan probabilidad de perder la unicidad en el futuro.
- Escoger la clave candidata con el mínimo número de caracteres (si es de tipo texto).
- Escoger la clave candidata más fácil de utilizar desde el punto de vista de los usuarios.

(b) Entidades débiles. Crear una relación para cada entidad débil incluyendo todos sus atributos simples. De los atributos compuestos incluir sólo sus componentes. Añadir una clave ajena a la entidad de la que depende. Para ello, se incluye la clave primaria de la relación que representa a la entidad padre en la nueva relación creada para la entidad débil. A continuación, determinar la clave primaria de la nueva relación.

(c) Relaciones binarias de uno a uno. Para cada relación binaria se incluyen los atributos de la clave primaria de la entidad padre en la relación (tabla) que representa a la entidad hijo, para actuar como una clave ajena. La entidad hijo es la que participa de forma total (obligatoria) en la relación, mientras que la entidad padre es la que participa de forma parcial (opcional). Si las dos entidades participan de forma total o parcial en la relación, la elección de padre e hijo es arbitraria. Además, en caso de que ambas entidades participen de forma total en la relación, se tiene la opción de integrar las dos entidades en una sola relación (tabla). Esto se suele hacer si una de las entidades no participa en ninguna otra relación.

(d) Relaciones binarias de uno a muchos. Como en las relaciones de uno a uno, se incluyen los atributos de la clave primaria de la entidad padre en la relación (tabla) que representa a la entidad hijo, para actuar como una clave ajena. Pero ahora, la entidad padre es la de "la parte del muchos" (cada padre tiene muchos hijos), mientras que la entidad hijo es la de "la parte del uno" (cada hijo tiene un solo padre).

(e) Jerarquías de generalización. En las jerarquías, se denomina entidad padre a la entidad genérica y entidades hijo a las subentidades. Hay tres opciones distintas para representar las jerarquías. La elección de la más adecuada se hará en función de su tipo (total/parcial, exclusiva/superpuesta).

1. Crear una relación por cada entidad. Las relaciones de las entidades hijo heredan como clave primaria la de la entidad padre. Por lo tanto, la clave primaria de las entidades hijo es también una clave ajena al padre. Esta opción sirve para cualquier tipo de jerarquía, total o parcial y exclusiva o superpuesta.

2. Crear una relación por cada entidad hijo, heredando los atributos de la entidad padre. Esta opción sólo sirve para jerarquías totales y exclusivas.

3. Integrar todas las entidades en una relación, incluyendo en ella los atributos de la entidad padre, los atributos de todos los hijos y un atributo discriminativo para indicar el caso al cual pertenece la entidad en consideración. Esta opción sirve para cualquier tipo de jerarquía. Si la jerarquía es superpuesta, el atributo discriminativo será multivalorado.

Una vez obtenidas las relaciones con sus atributos, claves primarias y claves ajenas, sólo queda actualizar el diccionario de datos con los nuevos atributos que se hayan identificado en este paso.

3. Validar cada esquema mediante la normalización

La normalización se utiliza para mejorar el esquema lógico, de modo que satisfaga ciertas restricciones que eviten la duplicidad de datos. La normalización garantiza que el esquema resultante se encuentra más próximo al modelo de la empresa, que es consistente y que tiene la mínima redundancia y la máxima estabilidad.

La normalización es un proceso que permite decidir a qué entidad pertenece cada atributo. Uno de los conceptos básicos del modelo relacional es que los atributos se agrupan en relaciones (tablas) porque están relacionados a nivel lógico. En la mayoría de las ocasiones, una base de datos normalizada no proporciona la máxima eficiencia, sin embargo, el objetivo ahora es conseguir una base de datos normalizada por las siguientes razones:

- Un esquema normalizado organiza los datos de acuerdo a sus dependencias funcionales, es decir, de acuerdo a sus relaciones lógicas.
- El esquema lógico no tiene porqué ser el esquema final. Debe representar lo que el diseñador entiende sobre la naturaleza y el significado de los datos de la empresa. Si se establecen unos objetivos en cuanto a prestaciones, el diseño físico cambiará el esquema lógico de modo adecuado. Una posibilidad es que algunas relaciones normalizadas se desnormalicen. Pero la desnormalización no implica que se haya malgastado tiempo normalizando, ya que mediante este proceso el diseñador aprende más sobre el significado de los datos. De hecho, la normalización obliga a entender completamente cada uno de los atributos que se han de representar en la base de datos.
- Un esquema normalizado es robusto y carece de redundancias, por lo que está libre de ciertas anomalías que éstas pueden provocar cuando se actualiza la base de datos.
- Los equipos informáticos de hoy en día son mucho más potentes, por lo que en ocasiones es más razonable implementar bases de datos fáciles de manejar (las normalizadas), a costa de un tiempo adicional de proceso.
- La normalización produce bases de datos con esquemas flexibles que pueden extenderse con facilidad.

El objetivo de este paso es obtener un conjunto de relaciones que se encuentren en la forma normal de Boyce-Codd. Para ello, hay que pasar por la primera, segunda y tercera formas normales. El proceso de normalización se describe en el apartado 7.3.

4. Validar cada esquema frente a las transacciones del usuario

El objetivo de este paso es validar cada esquema lógico local para garantizar que puede soportar las transacciones requeridas por los correspondientes usuarios. Estas transacciones se encontrarán en las especificaciones de requisitos de usuario. Lo que se debe hacer es tratar de realizar las transacciones de forma manual utilizando el diagrama entidad-relación, el diccionario de datos y las conexiones que establecen las claves ajenas de las relaciones (tablas). Si todas las transacciones se pueden realizar, el esquema queda validado. Pero si alguna transacción no se puede realizar, seguramente será porque alguna entidad, relación o atributo no se ha incluido en el esquema.

5. Dibujar el diagrama entidad-relación

En este momento, se puede dibujar el diagrama entidad-relación final para cada vista de usuario que recoja la representación lógica de los datos desde su punto de vista. Este diagrama habrá sido validado mediante la normalización y frente a las transacciones de los usuarios.

6. Definir las restricciones de integridad

Las restricciones de integridad son reglas que se quieren imponer para proteger la base de datos, de modo que no pueda llegar a un estado inconsistente. Hay cinco tipos de restricciones de integridad.

- (a) Datos requeridos. Algunos atributos deben contener valores en todo momento, es decir, no admiten nulos.

(b) Restricciones de dominios. Todos los atributos tienen un dominio asociado, que es el conjunto los valores que cada atributo puede tomar.

(c) Integridad de entidades. El identificador de una entidad no puede ser nulo, por lo tanto, las claves primarias de las relaciones (tablas) no admiten nulos.

(d) Integridad referencial. Una clave ajena enlaza cada tupla de la relación hijo con la tupla de la relación padre que tiene el mismo valor en su clave primaria. La integridad referencial dice que si una clave ajena tiene un valor (si es no nula), ese valor debe ser uno de los valores de la clave primaria a la que referencia. Hay varios aspectos a tener en cuenta sobre las claves ajenas para lograr que se cumpla la integridad referencial.

1. ¿Admite nulos la clave ajena? Cada clave ajena expresa una relación. Si la participación de la entidad hijo en la relación es total, entonces la clave ajena no admite nulos; si es parcial, la clave ajena debe aceptar nulos.

2. ¿Qué hacer cuando se quiere borrar una ocurrencia de la entidad padre que tiene algún hijo? O lo que es lo mismo, ¿qué hacer cuando se quiere borrar una tupla que está siendo referenciada por otra tupla a través de una clave ajena? Hay varias respuestas posibles:

- Restringir: no se pueden borrar tuplas que están siendo referenciadas por otras tuplas.
- Propagar: se borra la tupla deseada y se propaga el borrado a todas las tuplas que le hacen referencia.
- Anular: se borra la tupla deseada y todas las referencias que tenía se ponen, automáticamente, a nulo (esta respuesta sólo es válida si la clave ajena acepta nulos).
- Valor por defecto: se borra la tupla deseada y todas las referencias toman, automáticamente, el valor por defecto (esta respuesta sólo es válida si se ha especificado un valor por defecto para la clave ajena).
- No comprobar: se borra la tupla deseada y no se hace nada para garantizar que se sigue cumpliendo la integridad referencial.

3. ¿Qué hacer cuando se quiere modificar la clave primaria de una tupla que está siendo referenciada por otra tupla a través de una clave ajena? Las respuestas posibles son las mismas que en el caso anterior. Cuando se escoge propagar, se actualiza la clave primaria en la tupla deseada y se propaga el cambio a los valores de clave ajena que le hacían referencia.

(e) Reglas de negocio. Cualquier operación que se realice sobre los datos debe cumplir las restricciones que impone el funcionamiento de la empresa.

Todas las restricciones de integridad establecidas en este paso se deben reflejar en el diccionario de datos para que puedan ser tenidas en cuenta durante la fase del diseño físico.

7. Revisar cada esquema lógico local con el usuario correspondiente

Para garantizar que cada esquema lógico local es una fiel representación de la vista del usuario lo que se debe hacer es comprobar con él que lo reflejado en el esquema y en la documentación es correcto y está completo.

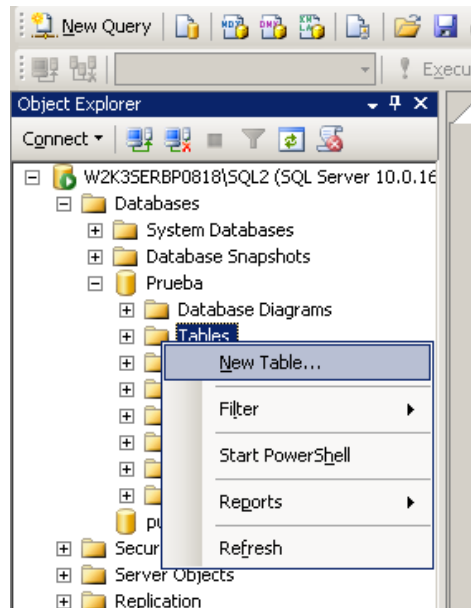
➤ PRACTICA CALIFICADA 02

Temas:

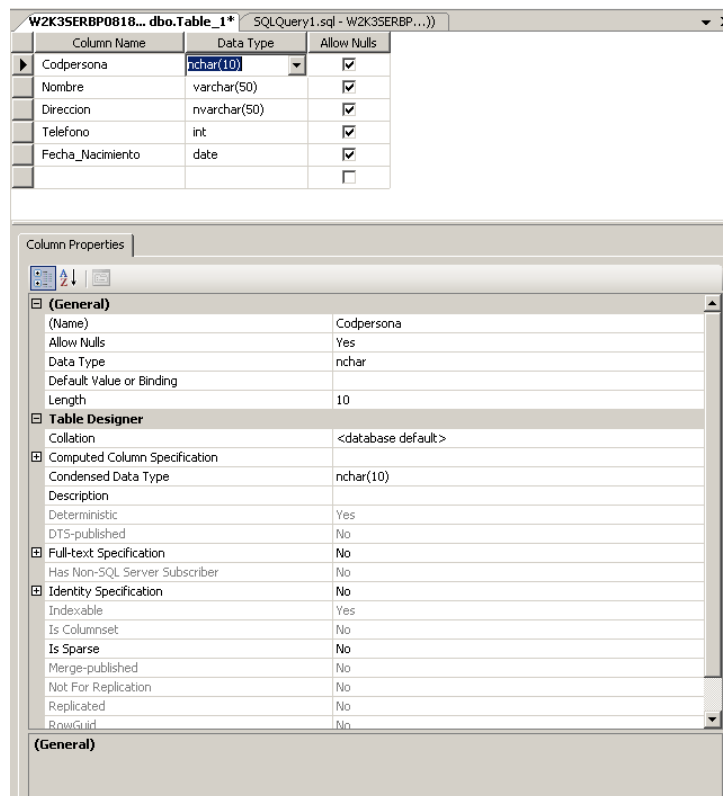
1. Abstracción de datos: Generalización, Clasificación, Agregación y Asociación. Conceptos básicos y ejemplos por cada uno.
2. Los tipos de datos disponibles para SQL Server en su versión 2008.
3. Mencione algunas características importantes que deben cumplir los DBMS. Concepto de DBMS y principales funciones.
4. Realizar el modelo de datos, lógico y físico para un caso de estudio propuesto por el profesor. Utilizar Case Erwin para su realización, el modelo físico debe mostrar tipos de datos para las columnas.
5. Crear la base de datos del modelo físico de datos del ejercicio anterior, utilizando el SQL Server 2008 para ello, las tablas deben estar relacionadas y mostrar las restricciones (Constraint) que el diseñador crea convenientes. La Base de datos debe tener un tamaño inicial, un archivo de datos y un archivo de registros (lógico) y con crecimiento porcentual para ambos tipos de archivos.

➤ CREACION DE TABLAS (WIZARD)

Una vez creada la base de datos, desplegamos la carpeta que le pertenece en el Explorador de Objetos, buscamos la carpeta Tablas y le hacemos clic derecho - Nueva Tabla...



Se desplegará la ventana de creación de tablas, en donde iremos colocando los campos, tipos de datos y propiedades o constraints correspondientes.



Al finalizar le damos clic derecho sobre la pestaña Query

➤ CREACION DE TABLAS DE DATOS (DDL)

Una vez creada la base de datos y habiéndola seleccionado para trabajar en ella, debemos empezar a crear las tablas que contendrá, para ello utilizamos la sentencia DDL Create, utilizando la siguiente sintaxis:

```
CREATE TABLE "TABLA_NOMBRE" (  
    "CAMPO_1" TIPO_DE_DATO,  
    "CAMPO_2" TIPO_DE_DATO,  
    .  
    .  
    "CAMPO_N" TIPO_DE_DATO  
)
```

➤ TIPOS DE DATOS EMPLEADOS EN SQL SERVER 2008

Cuando definimos una tabla, variable o constante debemos asignar un tipo de dato que indica los posibles valores. El tipo de datos define el formato de almacenamiento, espacio que de disco-memoria que va a ocupar un campo o variable, restricciones y rango de valores validos.

Transact SQL proporciona una variedad predefinida de tipos de datos. Casi todos los tipos de datos manejados por Transact SQL son similares a los soportados por SQL.

Tipos de datos numéricos

SQL Server dispone de varios tipos de datos numéricos. Cuanto mayor sea el número que puedan almacenar mayor será en consecuencia el espacio utilizado para almacenarlo. Como regla general se recomienda usar el tipo de dato mínimo posible. Todos los datos numéricos admiten el valor NULL.

Bit. Una columna o variable de tipo bit puede almacenar el rango de valores de 1 a 0.

Tinyint. Una columna o variable de tipo tinyint puede almacenar el rango de valores de 0 a 255.

SmallInt. Una columna o variable de tipo smallint puede almacenar el rango de valores -32768 a 32767.

Int. Una columna o variable de tipo int puede almacenar el rango de valores -231 a 231-1 .

BigInt. Una columna o variable de tipo bigint puede almacenar el rango de valores -263 a 263-1

Decimal(p,s). Una columna de tipo decimal puede almacenar datos numéricos decimales sin redondear. Donde p es la precisión (número total de dígitos) y s la escala (número de valores decimales)

Float. Una columna de datos float puede almacenar el rango de valores $-1,79 \times 10^{308}$ a $1,79 \times 10^{308}$, si la definimos con el valor máximo de precisión. La precisión puede variar entre 1 y 53.

Real. Sinónimo de float(24). Puede almacenar el rango de valores $-3,4 \times 10^{38}$ a $3,4 \times 10^{38}$,

Money. Almacena valores numéricos monetarios de -263 a $263-1$, con una precisión de hasta diez milésimas de la unidad monetaria.

SmallMoney. Almacena valores numéricos monetarios de $-214.748,3647$ a $214.748,3647$, con una precisión de hasta diez milésimas de la unidad monetaria.

Todos los tipos de datos enteros pueden marcarse con la propiedad identity para hacerlos autonuméricos.

Tipos de datos de caracter

Char(n). Almacena n caracteres en formato ASCII, un byte por cada letra. Cuando almacenamos datos en el tipo char, siempre se utilizan los n caracteres indicados, incluso si la entrada de datos es inferior. Por ejemplo, si en un char(5), guardamos el valor 'A', se almacena 'A ', ocupando los cinco bytes.

Varchar(n). Almacena n caracteres en formato ASCII, un byte por cada letra. Cuando almacenamos datos en el tipo varchar, únicamente se utilizan los caracteres necesarios, Por ejemplo, si en un varchar(255), guardamos el valor 'A', se almacena 'A', ocupando solo un byte bytes.

Varchar(max). Igual que varchar, pero al declararse como max puede almacenar $2^{31}-1$ bytes.

Nchar(n). Almacena n caracteres en formato UNICODE, dos bytes por cada letra. Es recomendable utilizar este tipo de datos cuando los valores que vayamos a almacenar puedan pertenecer a diferente idiomas.

Nvarchar(n). Almacena n caracteres en formato UNICODE, dos bytes por cada letra. Es recomendable utilizar este tipo de datos cuando los valores que vayamos a almacenar puedan pertenecer a diferente idiomas.

Nvarchar(max). Igual que varchar, pero al declararse como max puede almacenar $2^{31}-1$ bytes.

Tipos de datos de fecha

Datetime. Almacena fechas con una precisión de milisegundo. Debe usarse para fechas muy específicas. El formato que presenta es de fecha y hora (0:00:00). Ocupa 8 bytes.

Date. Almacena las fechas sin el formato de hora (a diferencia del datetime), ocupa 4 bytes.

Time. Almacena datos en formato de horas – minutos – segundos. Al igual que el date, ocupa solo 4 bytes.

SmallDatetime. Almacena fechas con una precisión de minuto, por lo que ocupa la mitad de espacio de que el tipo datetime, para tablas que puedan llegar a tener muchos datos es un factor a tener muy en cuenta.

TimeStamp. Se utiliza para marcar un registro con la fecha de inserción - actualización. El tipo timestamp se actualiza automáticamente cada vez que insertamos o modificamos los datos.

Tipos de datos binarios

Binary. Se utiliza para almacenar datos binarios de longitud fija, con una longitud máxima de 8000 bytes.

Varbinary. Se utiliza para almacenar datos binarios de longitud variable, con una longitud máxima de 8000 bytes..Es muy similar a binary, salvo que varbinary utiliza menos espacio en disco.

Varbinary(max). Igual que varbinary, pero puede almacenar 231-1 bytes

Tipo de datos XML

XML.es una de las grandes mejoras que incorpora SQL Server 2005 es el soporte nativo para XML. Como podemos deducir, este tipo de datos se utiliza para almacenar XML.

➤ RESTRICCIONES (CONSTRAINTS)

Se utiliza la cláusula CONSTRAINT en las instrucciones ALTER TABLE y CREATE TABLE para crear o eliminar índices. Existen dos sintaxis para esta cláusula dependiendo si desea Crear ó Eliminar un índice de un único campo o sise trata de un campo multiíndice. Si se utiliza el motor de datos de Microsoft, sólo podrá utilizar esta cláusula con las bases de datos propias de dicho motor. Los constraints permiten asegurar la integridad referencial de los datos en las tablas, asegurando la fidelidad de los datos, asimismo permite controlar el ingreso de los datos por parte de los usuarios, a fin de tener siempre la información deseada. Un constraint es un objeto que se adhiere a las tablas.

PRIMARY KEY

Permite asignar una clave principal a la tabla. Una tabla debe tener siempre una clave primaria con la cual indizamos la tabla. Un primary key puede tener un campo o varios campos que la componen.

Ejemplo:

```
CREATE TABLE TABLA1(  
    CAMPO_1 CHAR(5) PRIMARY KEY,  
    CAMPO_2 VARCHAR(40)  
)
```

Aquí estamos asignando el campo 1 como clave principal, sin embargo es recomendable manejar las claves mediante constraints, que podemos administrar más adelante.

```
CREATE TABLE ALUMNO (  
    CODALUM CHAR(5),  
    NOMBRE VARCHAR(40)  
    CONSTRAINT PK_TABLAALUM PRIMARY KEY (CODALUM)  
)
```

FOREIGN KEY

Con la restricción "foreign key" se define un campo (o varios) cuyos valores coinciden con la clave primaria de la misma tabla o de otra, es decir, se define una referencia a un campo con una restricción "primary key" o "unique" de la misma tabla o de otra.

La integridad referencial asegura que se mantengan las referencias entre las claves primarias y las externas. Por ejemplo, controla que si se agrega un código de la tabla Alumno en la tabla Notas, tal código exista en la tabla Alumno.

También controla que no pueda eliminarse un registro de una tabla ni modificar la clave primaria si una clave externa hace referencia al registro.

La siguiente es la sintaxis parcial general para agregar una restricción "foreign key":

```
CREATE TABLE NOTAS (  
    CODNOTA CHAR(5),  
    CODALUM CHAR(5) CONSTRAINT FK_TABLANOT FOREIGN KEY (CODALUM)  
    REFERENCES ALUMNO (CODALUM)  
)
```

- Entre Alumno y Notas hay una relación Uno a Muchos, en donde Alumno es la tabla fuerte y Notas es la tabla débil. Entonces tenemos que mediante un constraint Foreign Key creamos la relación lógica entre ambas tablas.
- Cuando agregamos cualquier restricción a una tabla que contiene información, SQL Server controla los datos existentes para confirmar que cumplen con la restricción, si no los cumple, la restricción no se aplica y aparece un mensaje de error. Por ejemplo, si intentamos agregar una restricción "foreign key" a la tabla "Notas" y existe un alumno con un valor de código para editorial que no existe en la tabla "Alumno", la restricción no se agrega.
- Actúa en inserciones. Si intentamos ingresar un registro (una nota) con un valor de clave foránea (codalum) que no existe en la tabla referenciada (alumno), SQL server muestra un mensaje de error. Si al ingresar un registro (una nota), no colocamos el valor para el campo clave foránea (codalum), almacenará "null", porque esta restricción permite valores nulos (a menos que se haya especificado lo contrario al definir el campo).
- Actúa en eliminaciones y actualizaciones. Si intentamos eliminar un registro o modificar un valor de clave primaria de una tabla si una clave foránea hace referencia a dicho registro, SQL Server no lo permite (excepto si se permite la acción en cascada, tema que veremos posteriormente). Por ejemplo, si intentamos eliminar un alumno a la que se hace referencia en Notas, aparece un mensaje de error.
- Esta restricción (a diferencia de "primary key" y "unique") no crea índice automáticamente.
- La cantidad y tipo de datos de los campos especificados luego de "foreign key" DEBEN coincidir con la cantidad y tipo de datos de los campos de la cláusula "references".
- Esta restricción se puede definir dentro de la misma tabla (lo veremos más adelante) o entre distintas tablas.
- Una tabla puede tener varias restricciones "foreign key".
- No se puede eliminar una tabla referenciada en una restricción "foreign key", aparece un mensaje de error.

- Una restricción "foreign key" no puede modificarse, debe eliminarse y volverse a crear.
- Para ver información acerca de esta restricción podemos ejecutar el procedimiento almacenado "sp_helpconstraint" junto al nombre de la tabla. Nos muestra el tipo, nombre, la opción para eliminaciones y actualizaciones, el estado (temas que veremos más adelante), el nombre del campo y la tabla y campo que referencia.
- También informa si la tabla es referenciada por una clave foránea.

LLAVES COMPUESTAS

Para asignar más de un campo como clave principal, lo haremos de la siguiente forma:

```
CREATE TABLE ALUMNO_NOTAS(  
    CODALUM CHAR(5),  
    CODNOTA CHAR(5),  
    CONSTRAINT PK_ALNOT PRIMARY KEY (CODALUM, CODNOTA),  
    CONSTRAINT FK_ALNOT1 FOREIGN KEY (CODALUM) REFERENCES  
        ALUMNO(CODALUM),  
    CONSTRAINT FK_ALNOT2 FOREIGN KEY (CODNOTA) REFERENCES  
        NOTAS(CODNOTA)  
)
```

Haremos esto cada vez que debamos crear tablas asociadas producto de una relación de Muchos a Muchos entre dos tablas, o deseamos asignar claves compuestas a una tabla.

Otro ejemplo:

Clave primaria



autor_id	titulo_id	au_ord	royaltyper
172-32-1176	PS3333	1	100
213-46-8915	BU1032	2	40
213-46-8915	BU2075	1	100
238-95-7766	PC1035	1	100
267-41-2394	BU1111	2	40

La clave primaria de la tabla TituloAutor en la base de datos pubs

DEFAULT

Especifica el valor que se asigna a la columna cuando no se ha suministrado ningún valor de forma explícita durante una acción de inserción. Las definiciones DEFAULT pueden aplicarse a cualquier columna excepto a las definidas por la propiedad IDENTITY. Las definiciones DEFAULT desaparecen cuando la tabla se elimina. Sólo los valores constantes, por ejemplo, una cadena de caracteres o una función de fecha, se pueden utilizar como valores predeterminados.

Ejemplo:

Si deseamos asignar una condición de empleado por defecto, haremos lo siguiente:

```
CREATE TABLE EMPLEADO(  
    CODEMP CHAR(5),  
    CONDICION VARCHAR(30) DEFAULT 'PRACTICANTE',  
)
```

Si al momento de insertar un registro a esta tabla no indicamos un valor para este campo, por defecto aparecerá este dato para el campo Condición.

Si deseamos indicar que la fecha de inscripción de un cliente nuevo sea la fecha actual del sistema, utilizaremos esta restricción invocando la función getdate().

```
CREATE TABLE CLIENTE(  
    CODCLIE CHAR(5),  
    FECHA_INSC DATETIME DEFAULT GETDATE(),  
)
```

Si al momento de insertar un registro a esta tabla no indicamos un valor para este campo, por defecto aparecerá la fecha actual del sistema, es decir la fecha que figura en el ordenador como dato para este campo.

CHECK

Las restricciones CHECK aseguran la integridad de dominio al limitar los valores que son aceptados para una columna. Son similares a las restricciones FOREIGN KEY en que ambas controlan los valores que son puestos en una columna. La diferencia está en cómo se determina cuáles son valores válidos. Las restricciones FOREIGN KEY toman los valores válidos de otra tabla, mientras que las restricciones CHECK determinan los valores válidos evaluando una expresión lógica que no se basa en datos de otra columna. Por ejemplo, es posible limitar el rango de valores para una columna Salario creando una restricción CHECK que permita solamente datos dentro del rango de \$15.000 a \$100.000. Esta capacidad evita el ingreso de salarios fuera del rango normal de salarios de la compañía.

Se puede crear una restricción CHECK con una expresión lógica (Booleana) que retorne TRUE (verdadero) o FALSE (falso) basada en operadores lógicos. Para permitir solamente datos que se encuentren dentro del rango de \$15.000 a \$100.000, la expresión lógica será como la siguiente:

Salario >= 15000 AND Salario <= 100000

Se puede aplicar múltiples restricciones CHECK para una sola columna. Las restricciones son evaluadas en el orden en que han sido creadas. Además, se puede aplicar una misma restricción CHECK a múltiples columnas creando la restricción a nivel de tabla. Por ejemplo, se puede usar una restricción CHECK para múltiples columnas para confirmar que cualquier fila con la columna País igual a USA tenga valor para la columna Estado que sea una cadena de dos caracteres. Esta posibilidad permite que múltiples condiciones sean controladas en un lugar.

Crear restricciones CHECK

Se pueden crear restricciones usando uno de los siguientes métodos:

- Creando la restricción cuando se crea la tabla (como parte de las definiciones de la tabla)
- Agregando la restricción a una tabla existente.

Se puede modificar o eliminar una restricción CHECK una vez que ha sido creada. Por ejemplo, se puede modificar la expresión usada por la restricción CHECK sobre una columna en la tabla.

Para modificar una restricción CHECK primero se debe eliminar la antigua restricción y luego recrearla con su nueva definición.

El siguiente comando CREATE TABLE crea una tabla Tabla1 y define la columna Col2 con un restricción CHECK que limita los valores que puede tomar la columna al rango comprendido entre 0 y 100.

```
CREATE TABLE TABLA1
(
  COL1 INT PRIMARY KEY,
  COL2 INT
  CONSTRAINT MONTO_LIMITE CHECK (COL2 BETWEEN 0 AND 100),
  COL3 VARCHAR(30)
)
```

También se puede definir la misma restricción usando restricción CHECK a nivel tabla:

```
CREATE TABLE TABLA1
(
  COL1 INT PRIMARY KEY,
  COL2 INT ,
  COL3 VARCHAR(30),
  CONSTRAINT MONTO_LIMITE CHECK (COL2 BETWEEN 0 AND 100)
)
```

Otro ejemplo:

Podemos marcar un campo sexo para que el usuario ingrese solamente un caracter: M o F.

```
SEXO CHAR(1) CONSTRAINT CHK_SX CHECK(SEXO IN('M','F'))
```

UNIQUE

Se pueden crear restricciones UNIQUE en el mismo modo que se crean restricciones PRIMARY KEY:

- Creando la restricción al momento de crear la tabla (como parte de la definición de la tabla)
- Agregando la restricción a una tabla existente, previendo que la o las columnas comprendidas en la restricción UNIQUE contengan solo valores no duplicados o valores nulos. Una tabla puede aceptar múltiples restricciones UNIQUE.

Se pueden usar los mismos comandos Transact-SQL para crear restricciones UNIQUE que los utilizados para crear restricciones PRIMARY KEY. Simplemente reemplace las palabras PRIMARY KEY por UNIQUE. Al igual que con las restricciones PRIMARY KEY las restricciones UNIQUE pueden ser modificadas o eliminadas una vez creadas.

Cuando se agrega una restricción UNIQUE a una columna (o columnas) existente en la tabla, SQL Server (por defecto) controla los datos existentes en las columnas para asegurar que todos los valores, excepto los nulos, son únicos. Si se agrega una restricción UNIQUE a una

columna que tienen valores no nulos duplicados, SQL Server genera un mensaje de error y no agrega la restricción.

SQL Server automáticamente crea un índice UNIQUE para asegurar la unicidad requerida por la restricción UNIQUE. Por lo que, si se intenta ingresar una nueva fila con valores duplicados para la columna (o combinación de columnas) especificada se genera un mensaje de error diciendo que ha sido violada la restricción UNIQUE y no se agrega la fila a la tabla. Si no se especifica un índice agrupado, se creará un índice no-agrupado por defecto cuando se crea una restricción UNIQUE.

Ejemplo:

```
CREATE TABLE CLIENTE(  
    CODCLIE CHAR(5) PRIMARY KEY,  
    RUCCLIE CHAR(11) UNIQUE  
)
```

En este caso indico que el campo Rucclie debe ser único en todos sus valores, es decir no debe haber valores repetidos en la tabla.

IDENTITY

Cada tabla puede tener sólo una columna de identificación, la que contendrá una secuencia de valores generados por el sistema que unívocamente identifican a cada fila de la tabla. Las columnas de identificación contienen valores únicos dentro de la tabla para la cual son definidas, no así con relación a otras tablas que pueden contener esos valores en sus propias columnas de identificación. Esta situación no es generalmente un problema, pero en los casos que así lo sea (por ejemplo cuando diferentes tablas referidas a una misma entidad conceptual, como ser clientes, son cargadas en diferentes servidores distribuidos en el mundo y existe la posibilidad que en algún momento para generar reporte o consolidación de información sean unidas) se pueden utilizar columnas ROWGUIDCOL como se vio anteriormente.

Ejemplo:

Si queremos crear un id de registros autogenerado correlativamente, debemos hacer esto:

```
CREATE TABLE REGISTRO_VENTAS(  
    IDREGISTRO Int IDENTITY(1,1)  
    FECHAREG DATETIME  
    CONSTRAINT PK_TABLAREG PRIMARY KEY (IDREGISTRO)  
)
```

Para colocar a un campo esta restricción, debe ser de tipo de dato integer.
IDENTITY(1,1) – Indica que comenzará desde 1 e irá incrementándose de uno en uno.

NULOS (NULL – NOT NULL)

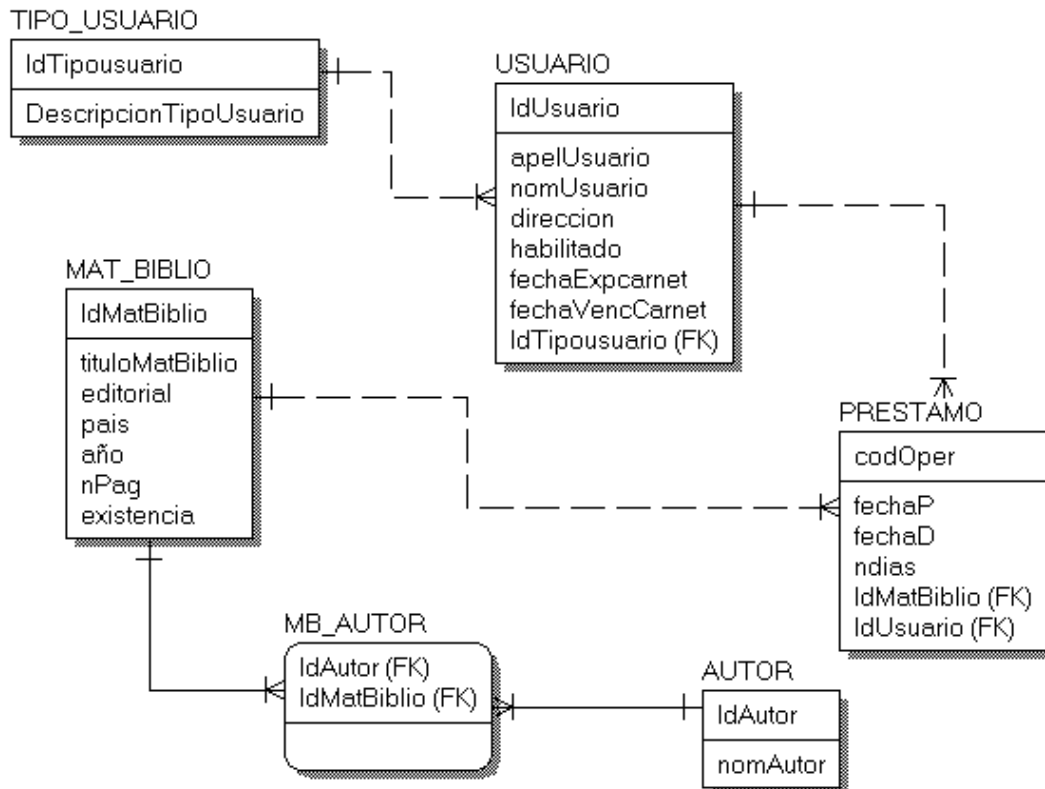
La anulabilidad de una columna determina si las filas en la tabla pueden contener valores nulos para esa columna. Un valor nulo no es lo mismo que un cero, un blanco o una cadena de caracteres de longitud cero. Un valor nulo significa que no se ha ingresado ningún valor para esa columna o que el valor es desconocido o indefinido. La anulabilidad de una columna se define cuando se crea o se modifica una tabla. Si se usan columnas que permiten o no valores nulos, se debería usar siempre la cláusula NULL y NOT NULL dada la complejidad que tiene el

SQL Server para manejar los valores nulos y no prestarse a confusión. La cláusula NULL se usa si se permiten valores nulos en la columna y la cláusula NOT NULL si no.

Las claves primarias y foráneas son NOT NULL por defecto, igualmente los campos que están marcados por las restricciones IDENTITY y DEFAULT no contendrán valores nulos.

➤ **EJERCICIO**

Crear las siguientes tablas para una base de datos de Biblioteca.



TIPO_USUARIO

IdTipousuario
DescripcionTipoUsuario

```
CREATE TABLE TIPO_USUARIO
(
  idTipoUsuario int Identity(1,1),
  descTipoUsuario varchar(20) NOT NULL,
  CONSTRAINT PK_TIPO_USUARIO PRIMARY KEY(idTipoUsuario)
)
GO
```


USUARIO

IdUsuario
apelUsuario
nomUsuario
direccion
habilitado
fechaExpcarnet
fechaVencCarnet
IdTipousuario (FK)

```
CREATE TABLE USUARIO
```

```
(  
idUsuario char(8)NOT NULL,  
idTipoUsuario int NOT NULL,  
apelUsuario varchar(35)NOT NULL,  
nomUsuario varchar(35) NOT NULL,  
direccion varchar(50)NULL,  
habilitado bit NOT NULL,  
fechaExpCarnet smalldatetime NOT NULL,  
fechaVencCarnet smalldatetime NOT NULL,  
CONSTRAINT PK_USUARIO_idUsuario PRIMARY KEY(idUsuario),  
CONSTRAINT FK_USUARIO_idTipoUsuario FOREIGN KEY (idTipoUsuario) REFERENCES  
TIPO_USUARIO(idTipoUsuario)  
)  
GO
```

MAT_BIBLIO

IdMatBiblio
tituloMatBiblio
editorial
pais
año
nPag
existencia

```
CREATE TABLE MAT_BIBLIO
```

```
(  
idMatBiblio varchar(20)NOT NULL,  
tituloMatBiblio varchar(150)NOT NULL,  
editorial varchar(50)NULL,  
pais varchar(20)NULL,  
año smalldatetime NULL,  
nPag int NULL,  
existencia int NOT NULL,  
)
```

CONSTRAINT PK_MAT_BIBLIO_idMatBiblio PRIMARY KEY(idMatBiblio)
)

PRESTAMO

codOper
fechaP
fechaD
ndias
IdMatBiblio (FK)
IdUsuario (FK)

```
CREATE TABLE PRESTAMO
(
codOper char(7)NOT NULL,
idMatBiblio varchar(20)NOT NULL,
idUsuario char(8)NOT NULL,
fechaP smalldatetime NOT NULL,
fechaD smalldatetime NOT NULL,
ndias int NOT NULL,
CONSTRAINT PK_PRESTAMO_codOper PRIMARY KEY(codOper),
CONSTRAINT FK_PRESTAMO_idMatBiblio1 FOREIGN KEY(idMatBiblio) REFERENCES
MAT_BIBLIO(idMatBiblio)
CONSTRAINT FK_PRESTAMO_idMatBiblio2 FOREIGN KEY(idUsuario) REFERENCES
USUARIO(idUsuario)
)
GO
```

AUTOR

IdAutor
nomAutor

```
CREATE TABLE AUTOR
(
idAutor char(4)NOT NULL,
nomAutor varchar(50)NOT NULL,
CONSTRAINT PK_AUTOR_idAutor PRIMARY KEY(idAutor)
)
GO
```

MB_AUTOR



```
CREATE TABLE MB_AUTOR
(
  idMatBiblio varchar(20) NOT NULL,
  idAutor char(4) NOT NULL,
  CONSTRAINT PK_MB_AUTOR_idMatBiblio_idAutor PRIMARY KEY(idMatBiblio,idAutor),
  CONSTRAINT FK_MB_AUTOR_idMatBiblio FOREIGN KEY(idMatBiblio) REFERENCES
  MAT_BIBLIO(idMatBiblio),
  CONSTRAINT FK_MB_AUTOR_idAutor FOREIGN KEY(idAutor) REFERENCES
  AUTOR(idAutor)
)
GO
```

Ojo: Esta tabla es producto de una relación Muchos a Muchos entre las tablas Mat_Biblio y Autor, entonces esta tabla es generada en el modelo físico y tiene una clave principal compuesta por las claves de ambas tablas.

PRIMARY KEY (idMatBiblio,idAutor),

Ambas son claves foráneas pero entre las dos son identificadas como las claves principales de la tabla asociada.

Semana

7

Contenido:

- Integridad Relacional
- Operaciones de Álgebra Relacional
- Operaciones Tradicionales Teoría de conjuntos
- Ejercicios
- Creación de Tablas de datos
- Restricciones
- Insert, delete y update.
- Ejercicios

ÁLGEBRA RELACIONAL – INTEGRIDAD REFERENCIAL

➤ INTEGRIDAD REFERENCIAL

La integridad referencial es una propiedad deseable en las bases de datos. Gracias a la integridad referencial se garantiza que una entidad (fila o registro) siempre se relaciona con otras entidades válidas, es decir, que existen en la base de datos. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos y relaciones mal resueltas.

Todas las bases de datos relacionales gozan de esta propiedad gracias a que el software gestor de base de datos vela por su cumplimiento. En cambio, las bases de datos jerárquicas requieren que los programadores se aseguren de mantener tal propiedad en sus programas.

✓ FUNCIONAMIENTO

Supongamos una base de datos con las entidades Persona y Factura. Toda factura corresponde a una persona y solamente una. Implica que en todo momento dichos datos sean correctos, sin repeticiones innecesarias, datos perdidos y relaciones mal resueltas.

Supongamos que una persona se identifica por su atributo DNI (Documento nacional de identidad). También tendrá otros atributos como el nombre y la dirección. La entidad Factura debe tener un atributo DNI_cliente que identifique a quién pertenece la factura.

Por sentido común es evidente que todo valor de DNI_cliente debe corresponder con algún valor existente del atributo DNI de la entidad Persona. Esta es la idea intuitiva de la integridad referencial.

Existen tres tipos de integridad referencial:

- Integridad referencial débil: si en una tupla de R todos los valores de los atributos de K tienen un valor que no es el nulo, entonces debe existir una tupla en S que tome esos mismos valores en los atributos de J.
- Integridad referencial parcial: si en una tupla de R algún atributo de K toma el valor nulo, entonces debe existir una tupla en S que tome en los atributos de J los mismos valores que los atributos de K con valor no nulo.
- Integridad referencial completa: en una tupla de R todos los atributos de K deben tener el valor nulo o bien todos tienen un valor que no es el nulo y entonces debe existir una tupla en S que tome en los atributos de J los mismos valores que toman los de K.

✓ PRIMARY KEY

La clave principal (PRIMARY KEY) nos permite asegurar la integridad de entidad (puesto que es única en cada registro) y por otro lado nos garantiza la estabilidad de las relaciones con otras tablas.

✓ FOREIGN KEY

La restricción FOREIGN KEY, se conoce como la clave externa o foránea que ya hemos explicado. Y como ya sabes es la pareja de la restricción PRIMARY KEY, y juntas cumplen con la integridad referencial.

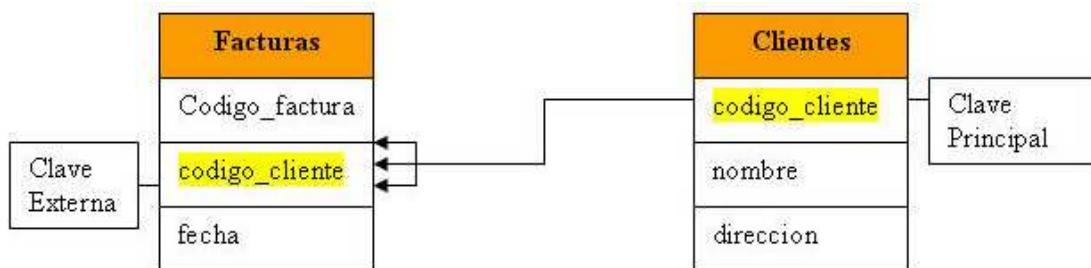
Una clave externa es una copia de la clave principal de la tabla principal, se inserta en la tabla que se pretende enlazar y con esto creamos la relación entre un par de tablas. Las claves externas pueden ser varias en una misma tabla, mientras que las principales deben ser únicas.

Para que esta relación que comentamos se cumpla, la clave principal que enlaza con la externa debe cumplir obligatoriamente que las dos columnas sean del mismo tipo.

✓ INTEGRIDAD REFERENCIAL EN CASCADA

Esta tipo de integridad que surgió con la versión 2000 de SQL Server, permite una serie de operaciones, que sólo pueden llevarse a cabo de este modo y no de otro.

Explicuemos porque a este tipo de integridad referencial se le añade el concepto de cascada. Imagina que tenemos una tabla que almacena las facturas emitidas para los clientes. Esta tabla entre sus campos contiene el campo `codigo_cliente`, que es la clave externa que se relaciona con la clave principal de la tabla clientes. Por lo tanto la tabla clientes tendrá un campo `codigo_cliente` que se relaciona con la tabla Facturas. Puedes verlo algo más claro en el siguiente gráfico:



Como ves, tenemos una relación uno a varios. Mientras en la tabla Clientes el campo `codigo_cliente` será único en cada registro, en la tabla Facturas, este mismo campo puede aparecer varias veces en diferentes registros, ya que emitimos varias facturas para el mismo cliente.

Esta relación representa una integridad referencial estricta, la cual no nos permite modificar el valor del código de cliente en la tabla clientes ya que de algún modo dejaría "huérfanos" a aquellos registros que anteriormente estaban relacionados al código de cliente que tratamos de modificar. Otra limitación que tenemos con esta integridad es que no nos permiten eliminar un cliente que tenga facturas emitidas en la tabla Facturas, por lo tanto no podemos eliminar un registro que tenga otros registros referenciados mediante estas relaciones.

Precisamente para solucionar estos problemas que hemos planteado tenemos la integridad referencial en cascada. Podemos añadir una serie de acciones que permita solventar estas complicaciones.

Podemos incluir actualizaciones en cascada, de modo que cuando modifiquemos el valor de una clave principal en la tabla principal, se modifiquen del mismo modo los valores de las claves externas en el resto de tablas enlazadas a la principal. En nuestro caso, al modificar el valor del campo `codigo_cliente` de un determinado cliente, este nuevo valor se cambiará para todos los registros referenciados en la tabla Facturas.

Igualmente podemos incluir eliminaciones en cascada, de tal forma que si eliminamos un registro de la tabla principal, se eliminen también los registros enlazados en la tabla subordinada. En nuestro caso, podríamos eliminar un cliente, y automáticamente se eliminarían todos sus registros de facturas.

Debes tener mucho cuidado a la hora de utilizar este tipo de relaciones en cascada, ya que si activamos por ejemplo la eliminación en cascada, corremos peligro de que un usuario no sea consciente de que perderá todos los registros vinculados a esa tabla.

Una clave principal, por norma general no cambiará su valor, por lo tanto no tiene sentido activar en esos casos tampoco la actualización en cascada.

✓ **INTEGRIDAD REFERENCIAL EN CASCADA**

Las restricciones de integridad referencial en cascada permiten definir las acciones que SQL Server 2005 lleva a cabo cuando un usuario intenta eliminar o actualizar una clave a la que apuntan las claves externas existentes.

Las cláusulas REFERENCES de las instrucciones CREATE TABLE y ALTER TABLE admiten las cláusulas ON DELETE y ON UPDATE. Las acciones en cascada también se puede definir mediante el cuadro de diálogo Relaciones de clave externa.

[ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }]

[ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }]

✓ **NO ACTION**

Es el valor predeterminado si no se especifica ON DELETE u ON UPDATE.

✓ **ON DELETE NO ACTION**

Especifica que si se intenta eliminar una fila con una clave a la que hacen referencia las claves externas de las filas existentes en otras tablas, se produce un error y se revierte la instrucción DELETE.

✓ **ON UPDATE NO ACTION**

Especifica que si se intenta actualizar un valor de clave en una fila a cuya clave hacen referencia las claves externas de filas existentes en otras tablas, se produce un error y se revierte la instrucción UPDATE.

✓ **CASCADE, SET NULL y SET DEFAULT**

Permiten la eliminación o actualización de valores de clave de modo que se pueda realizar un seguimiento de las tablas definidas para tener relaciones de clave externa en la tabla en la que se realizan las modificaciones. Si las acciones referenciales en cascada se han definido también en las tablas de destino, las acciones en cascada especificadas se aplican para las filas eliminadas o actualizadas. No se puede especificar CASCADE para ninguna de las claves externas o principales que tengan una columna timestamp.

✓ **ON DELETE CASCADE**

Especifica que si se intenta eliminar una fila con una clave a la que hacen referencia claves externas de filas existentes en otras tablas, todas las filas que contienen dichas claves externas también se eliminan.

✓ **ON UPDATE CASCADE**

Especifica que si se intenta actualizar un valor de clave de una fila a cuyo valor de clave hacen referencia claves externas de filas existentes en otras tablas, también se actualizan todos los valores que conforman la clave externa al nuevo valor especificado para la clave.

Nota:

CASCADE no se puede especificar si una columna timestamp es parte de una clave externa o de la clave a la que se hace referencia.

✓ **ON DELETE SET NULL**

Especifica que si se intenta eliminar una fila con una clave a la que hacen referencia las claves externas de las filas existentes de otras tablas, todos los valores que conforman la clave externa de las filas a las que se hace referencia se establecen en NULL. Todas las columnas de clave externa de la tabla de destino deben aceptar valores NULL para que esta restricción se ejecute.

✓ **ON UPDATE SET NULL**

Especifica que si se intenta actualizar una fila con una clave a la que hacen referencia las claves externas de las filas existentes de otras tablas, todos los valores que conforman la clave externa de las filas a las que se hace referencia se establecen en NULL. Todas las columnas de clave externa de la tabla de destino deben aceptar valores NULL para que esta restricción se ejecute.

✓ **ON DELETE SET DEFAULT**

Especifica que si se intenta eliminar una fila con una clave a la que hacen referencia las claves externas de las filas existentes de otras tablas, todos los valores que conforman la clave externa de las filas a las que se hace referencia se establecen como predeterminados. Todas las columnas de clave externa de la tabla de destino deben tener una definición predeterminada para que esta restricción se ejecute. Si una columna acepta valores NULL y no se ha establecido ningún valor predeterminado explícito, NULL se convierte en el valor predeterminado implícito de la columna. Todos los valores distintos de NULL que se establecen debido a ON DELETE SET DEFAULT deben tener unos valores correspondientes en la tabla principal para mantener la validez de la restricción de la clave externa.

✓ **ON UPDATE SET DEFAULT**

Especifica que si se intenta actualizar una fila con una clave a la que hacen referencia las claves externas de las filas existentes de otras tablas, todos los valores que conforman la clave externa de la fila a los que se hace referencia se establecen en sus valores predeterminados. Todas las columnas externas de la tabla de destino deben tener una definición predeterminada para que esta restricción se ejecute. Si una columna se convierte en NULL, y no hay establecido ningún valor predeterminado explícito, NULL deviene el valor predeterminado implícito de la columna. Todos los valores no NULL que se establecen debido a ON UPDATE

SET DEFAULT deben tener unos valores correspondientes en la tabla principal para mantener la validez de la restricción de clave externa.

Ejemplos.

CODCLIE	NOMBRE
C0001	RAUL LOPEZ
C0002	ABRAHAM JARA

NRO_FACTURA	FECHA_EMISION	CODCLIE	NOMBRE
F00011	20/12/2009	C0001	RAUL LOPEZ
F00012	16/01/2010	C0002	ABRAHAM JARA

En este caso tenemos dos tablas, en las cuales los clientes registrados en el sistema tienen facturas ya generadas producto de una compra que ellos realizaron. Pongámonos en los dos posibles casos de integridad referencial en cascada:

- Si queremos modificar el nombre del cliente en la tabla Cliente, entonces este dato debería actualizarse también en la tabla Factura, esto ocurrirá siempre y cuando hayamos añadido esta instrucción en el momento de la creación de la tabla. Quedaría así:

CODCLIE	NOMBRE
C0001	JUAN LAVADO
C0002	ABRAHAM JARA

NRO_FACTURA	FECHA_EMISION	CODCLIE	NOMBRE
F00011	20/12/2009	C0001	JUAN LAVADO
F00012	16/01/2010	C0002	ABRAHAM JARA

- Si queremos eliminar a un cliente en la tabla Cliente, entonces este dato debería supuestamente eliminarse también en la tabla Factura, esto ocurrirá siempre y cuando hayamos añadido esta instrucción en el momento de la creación de la tabla. ¿Pero, es correcto hacer esto?

CODCLIE	NOMBRE
C0002	ABRAHAM JARA

NRO_FACTURA	FECHA_EMISION	CODCLIE	NOMBRE
F00012	16/01/2010	C0002	ABRAHAM JARA

¡No es correcto!

No deberíamos eliminar facturas ya que esto puede generar conflictos de información más adelante, existe la llamada eliminación lógica, en la cual podemos crear un campo en la tabla

Cliente que indique si el cliente está activo o no (Podemos utilizar el tipo de dato bit, que maneja como valores el 0 y 1, es un tipo de dato boolean).

➤ ALGEBRA RELACIONAL

El álgebra relacional es un conjunto de operaciones que describen paso a paso como computar una respuesta sobre las relaciones, tal y como éstas son definidas en el modelo relacional. Denominada de tipo procedimental, a diferencia del Cálculo relacional que es de tipo declarativo.

Describe el aspecto de la manipulación de datos. Estas operaciones se usan como una representación intermedia de una consulta a una base de datos y, debido a sus propiedades algebraicas, sirven para obtener una versión más optimizada y eficiente de dicha consulta.

✓ OPERACIONES DEL ALGEBRA RELACIONAL

Las operaciones de álgebra relacional manipulan relaciones. Esto significa que estas operaciones usan uno o dos relaciones existentes para crear una nueva relación. Esta nueva relación puede entonces usarse como entrada para una nueva operación. Este poderoso concepto - la creación de una nueva relación a partir de relaciones existentes hace considerablemente más fácil la solución de las consultas, debido a que se puede experimentar con soluciones parciales hasta encontrar la proposición con la que se trabajará.

El álgebra relacional consta de nueve operaciones:

- Unión
- Intersección
- Diferencia
- Producto
- Selección
- Proyección
- Reunión
- División
- Asignación

Las cuatro primeras se toman de la teoría de conjunto de las matemáticas; las cuatro siguientes son operaciones propias del álgebra relacional y la última es la operación estándar de dar un valor a un elemento.

UNIÓN

La operación de unión permite combinar datos de varias relaciones. Supongamos que una determinada empresa internacional posee una tabla de empleados para cada uno de los países en los que opera. Para conseguir un listado completo de todos los empleados de la empresa tenemos que realizar una unión de todas las tablas de empleados de todos los países.

No siempre es posible realizar consultas de unión entre varias tablas, para poder realizar esta operación es necesario e imprescindible que las tablas a unir tengan las mismas estructuras, que sus campos sean iguales.

INTERSECCIÓN

La operación de intersección permite identificar filas que son comunes en dos relaciones. Supongamos que tenemos una tabla de empleados y otra tabla con los asistentes que han realizado un curso de inglés (los asistentes pueden ser empleados o gente de la calle). Queremos crear una figura virtual en la tabla denominada "Empleados que hablan Inglés", esta figura podemos crearla realizando una intersección de empleados y curso de inglés, los elementos que existan en ambas tablas serán aquellos empleados que han asistido al curso.

DIFERENCIA

La operación diferencia permite identificar filas que están en una relación y no en otra. Tomando como referencia el caso anterior, deberíamos aplicar una diferencia entre la tabla empleados y la tabla asistentes al curso para saber aquellos asistentes externos a la organización que han asistido al curso.

PRODUCTO

La operación producto consiste en la realización de un producto cartesiano entre dos tablas dando como resultado todas las posibles combinaciones entre los registros de la primera y los registros de la segunda. Esta operación se entiende mejor con el siguiente ejemplo:

Tabla A	
X	Y
10	22
11	25
Tabla B	
W	Z
33	54
37	98
42	100

El producto de $A * B$ daría como resultado la siguiente tabla:

Tabla A * Tabla B

10	22	33	54
10	22	37	98
10	22	42	100
11	25	33	54
11	25	37	98
11	25	42	100

SELECCIÓN

La operación selección consiste en recuperar un conjunto de registros de una tabla o de una relación indicando las condiciones que deben cumplir los registros recuperados, de tal forma que los registros devueltos por la selección han de satisfacer todas las condiciones que se hayan establecido. Esta operación es la que normalmente se conoce como consulta.

Podemos emplearla para saber que empleados son mayores de 45 años, o cuales viven en Madrid, incluso podemos averiguar los que son mayores de 45 años y residen en Madrid, los que son mayores de 45 años y no viven en Madrid, etc..

En este tipo de consulta se emplean los diferentes operadores de comparación (=, >, <, >=, <=, <>), los operadores lógicos (and, or, xor) o la negación lógica (not).

PROYECCIÓN

Una proyección es un caso concreto de la operación selección, esta última devuelve todos los campos de aquellos registros que cumplen la condición que he establecido. Una proyección es una selección en la que seleccionamos aquellos campos que deseamos recuperar. Tomando como referencia el caso de la operación selección es posible que lo único que nos interese recuperar sea el número de la seguridad social, omitiendo así los campos teléfono, dirección, etc.. Este último caso, en el que seleccionamos los campos que deseamos, es una proyección.

REUNIÓN

La reunión se utiliza para recuperar datos a través de varias tablas conectadas unas con otras mediante cláusulas JOIN, en cualquiera de sus tres variantes INNER, LEFT, RIGHT. La operación reunión se puede combinar con las operaciones selección y proyección.

Un ejemplo de reunión es conseguir los pedidos que nos han realizado los clientes nacionales cuyo importe supere 15.000 unidades de producto, generando un informe con el nombre del cliente y el código del pedido. En este caso se da por supuesto que la tabla clientes es diferente a la tabla pedidos y que hay que conectar ambas mediante, en este caso, un INNER JOIN.

DIVISIÓN

La operación división es la contraria a la operación producto y quizás sea la más compleja de explicar, por tanto comenzaré directamente con un ejemplo. Una determinada empresa posee una tabla de comerciales, otra tabla de productos y otra con las ventas de los comerciales. Queremos averiguar que comerciales han vendido todo tipo de producto.

Lo primero que hacemos es extraer en una tabla todos los códigos de todos los productos, a esta tabla la denominamos A.

Tabla A	
Código Producto	
1035	
2241	
2249	
5818	

En una segunda tabla extraemos, de la tabla de ventas, el código del producto y el comercial que lo ha vendido, lo hacemos con una proyección y evitamos traer valores duplicados. El resultado podría ser el siguiente:

Tabla B	
Código Comercial	Código Producto
10	2241
23	2518
23	1035
39	2518
37	2518
10	2249
23	2249
23	2241

Si dividimos la tabla B entre la tabla A obtendremos como resultado una tercera tabla que:

Los campos que contiene son aquellos de la tabla B que no existen en la tabla A. En este caso el campo Código Comercial es el único de la tabla B que no existe en la tabla A.

Un registro se encuentra en la tabla resultado si y sólo si está asociado en tabla B con cada fila de la tabla A

Tabla Resultado	
Código Comercial	
23	

¿Por qué el resultado es 23?. El comercial 23 es el único de la tabla B que tiene asociados todos los posibles códigos de producto de la tabla A.

ASIGNACIÓN

Esta operación algebraica consiste en asignar un valor a uno o varios campos de una tabla.

✓ LAS OPERACIONES

Básicas

Cada operador del álgebra acepta una o dos relaciones y retorna una relación como resultado. σ y Π son operadores unarios, el resto de los operadores son binarios. Las operaciones básicas del álgebra relacional son:

SELECCIÓN (σ)

Permite seleccionar un subconjunto de tuplas de una relación (**R**), todas aquellas que cumplan la(s) condición(es) **P**, esto es:

$$\sigma_P(R)$$

Ejemplo:

$$\sigma_{\text{Apellido=Gomez}}(\text{Alumnos})$$

Selecciona todas las tuplas que contengan Gómez como apellido en la relación Alumnos.

Una condición puede ser una combinación booleana, donde se pueden usar operadores como: \wedge, \vee , combinándolos con operadores $<, >, \leq, \geq, =, \neq$.

PROYECCIÓN (Π)

Permite extraer columnas(atributos) de una relación, dando como resultado un *subconjunto vertical* de atributos de la relación, esto es:

$$\Pi_{A_1, A_2, \dots, A_n}(R)$$

donde A_1, A_2, \dots, A_n son atributos de la relación **R**.

Ejemplo:

$$\Pi_{\text{Apellido, Semestre, NumeroControl}}(\text{Alumnos})$$

Selecciona los atributos Apellido, Semestre y NumeroControl de la relación Alumnos, mostrados como un subconjunto de la relación Alumnos

PRODUCTO CARTESIANO (\times)

El producto cartesiano de *dos relaciones* se escribe como:

$$R \times S$$

y entrega una relación, cuyo *esquema* corresponde a una combinación de todas las tuplas de **R** con cada una de las tuplas de **S**, y sus atributos corresponden a los de **R** seguidos por los de **S**.

Ejemplo:

Alumnos × Maestros

Muestra una nueva relación, cuyo esquema contiene cada una de las tuplas de la relación Alumnos junto con las tuplas de la relación Maestros, mostrando primero los atributos de la relación Alumnos seguidos por las tuplas de la relación Maestros.

UNIÓN (U)

La operación

$$R \cup S$$

Retorna el conjunto de tuplas que están en R, o en S, o en ambas. R y S deben ser *uniones compatibles*.

DIFERENCIA (-)

La diferencia de dos relaciones, R y S denotada por:

$$R - S$$

Entrega todas aquellas tuplas que están en R, pero **no** en S. R y S deben ser *uniones compatibles*.

Estas operaciones son fundamentales en el sentido en que (1) todas las demás operaciones pueden ser expresadas como una combinación de éstas y (2) ninguna de estas operaciones pueden ser omitidas sin que con ello se pierda información.

No básicas

Entre los operadores no básicos tenemos:

INTERSECCIÓN (∩)

La intersección de dos relaciones se puede especificar en función de otros operadores básicos:

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

La intersección, como en Teoría de conjuntos, corresponde al conjunto de todas las tuplas que están en R y en S, siendo R y S *uniones compatibles*.

REUNIÓN NATURAL (⋈) (NATURAL JOIN)

La operación Reunión natural en el álgebra relacional es la que permite reconstruir las tablas originales previas al proceso de normalización. Consiste en combinar las proyección, selección

y producto cartesiano en una sola operación, donde la condición θ es la igualdad Clave Primaria = Clave Externa (o Foranea), y la proyección elimina la columna duplicada (clave externa).

Expresada en las operaciones básicas, queda

$$R \bowtie S = \Pi_{A1, A2, \dots, An}(\sigma_{\theta}(R \times S))$$

Una reunión zeta (θ -Join) de dos relaciones es equivalente a:

$$R \bowtie_{\theta} S = \sigma_{\theta}(R \times S)$$

Donde la condición θ es libre.

Si la condición θ es una igualdad se denomina EquiJoin.

DIVISIÓN (/)

Supongamos que tenemos dos relaciones $A(x, y)$ y $B(y)$ donde el dominio de y en A y B , es el mismo.

El operador división A / B retorna todos los distintos valores de x tales que para todo valor y en B existe una tupla $\langle x, y \rangle$ en A .

Ejemplos

Suponga las relaciones o tablas:

Alumno

ID	NOMBRE	CIUDAD	EDAD
01	Pedro	Santiago	14
11	Juan	Buenos Aires	18
21	Diego	Lima	12
31	Rosita	Concepción	15
41	Manuel	Lima	17

Apoderado

ID	NOMBRE	FONO	ID_ALUMNO
054	Víctor	654644	21
457	José	454654	11
354	María	997455	31
444	Paz	747423	01

Curso

COD	NOMBRE	FECHA_INICIO	DURACION	VALOR
01142	Sicología	13-01	15	3.000
02145	Biología	15-02	12	2.500
03547	Matemáticas	01-03	30	4.000
04578	Música	05-04	10	1.500
05478	Física	20-04	15	3.200

Inscrito

ID	ID_AL	COD
1	01	05478
2	01	02145
3	11	03547
4	21	02145
5	41	03547

Mostrar los nombres de los alumnos y su apoderado

Primero, realizaremos una **combinación** entre alumnos y apoderados (pues necesitamos saber a que alumno le corresponde tal apoderado). La combinación realizará un producto cartesiano, es decir, para cada tupla de alumnos (todas las filas de *alumnos*) hará una mezcla con cada una tupla de apoderados y seleccionará aquellas nuevas tuplas en que alumnos.id sea igual a apoderados.id_alumno, esto es:

ID (alumno)	NOMBRE (alumno)	CIUDAD	EDAD	ID (apoderado)	NOMBRE (apoderado)	FONO	ID_ALUMNO
01	Pedro	Santiago	14	054	Víctor	654644	21
01	Pedro	Santiago	14	457	José	454654	11
01	Pedro	Santiago	14	354	María	997455	31
01	Pedro	Santiago	14	444	Paz	747423	01
11	Juan	Buenos Aires	18	054	Víctor	654644	21
11	Juan	Buenos Aires	18	457	José	454654	11
11	Juan	Buenos Aires	18	354	María	997455	31
11	Juan	Buenos Aires	18	444	Paz	747423	01
21	Diego	Lima	12	054	Víctor	654644	21
21	Diego	Lima	12	457	José	454654	11
21	Diego	Lima	12	354	María	997455	31
21	Diego	Lima	12	444	Paz	747423	01
31	Rosita	Concepción	15	054	Víctor	654644	21
31	Rosita	Concepción	15	457	José	454654	11
31	Rosita	Concepción	15	354	María	997455	31
31	Rosita	Concepción	15	444	Paz	747423	01
41	Manuel	Lima	17	054	Víctor	654644	21
41	Manuel	Lima	17	457	José	454654	11
41	Manuel	Lima	17	354	María	997455	31
41	Manuel	Lima	17	444	Paz	747423	01

Por tanto, el resultado final de la combinación es:

Alumnos				Apoderados			
ID (alumno)	NOMBRE (alumno)	CIUDAD	EDAD	ID (apoderado)	NOMBRE (apoderado)	FONO	ID_ALUMNO
01	Pedro	Santiago	14	444	Paz	747423	01
11	Juan	Buenos Aires	18	457	José	454654	11
21	Diego	Lima	12	054	Víctor	654644	21
31	Rosita	Concepción	15	354	María	997455	31

Ahora, aquí debemos mostrar solo el nombre del alumno y el nombre del apoderado, esto lo hacemos con un **Project** o **Proyección**, donde la tabla final sería:

II	
NOMBRE (alumno)	NOMBRE (apoderado)
Pedro	Paz
Juan	José
Diego	Víctor
Rosita	María

Resumiendo en un solo paso:

$\Pi_{Alumnos.NOMBRE, Apoderados.NOMBRE} (Alumnos \bowtie_{Alumnos.ID = Apoderados.ID_ALUMNO} Apoderados)$

Se lee: Proyecta los nombre de alumnos y nombre de apoderados de los alumnos cuyo ID sea el mismo que el ID_ALUMNO de los apoderados.

Mostrar el nombre de los alumnos inscritos y el nombre de los cursos que tomaron
[editar]

Comenzaremos con una combinación entre los inscritos y los cursos para obtener el nombre de los cursos:

$Inscritos \bowtie_{Inscritos.COD = Cursos.COD} Cursos$

Lo que nos da la tabla:

Resultado 1

ID	ID_AL	COD (inscritos)	COD (cursos)	NOMBRE	FECHA_INICIO	DURACION	VALOR
1	01	05478	05478	Física	20-04	15	3.200
2	01	02145	02145	Biología	15-02	12	2.500
3	11	03547	03547	Matemáticas	01-03	30	4.000
4	21	02145	02145	Biología	15-02	12	2.500
5	41	03547	03547	Matemáticas	01-03	30	4.000

Como podemos observar, la **combinación** solo nos entrega las combinaciones entre *Inscritos* y *Cursos* en que *COD* sea igual entre los inscritos y el curso correspondiente.

Ahora necesitamos los nombres de los alumnos inscritos. Al resultado anterior (*Resultado 1*) aplicaremos una nueva **combinación** comparando los *ID* de los alumnos para colocar el nombre adecuado con el estudiante adecuado:

Resultado 1 $\bowtie_{Resultado\ 1.ID_AL = Alumnos.ID}$ **Alumnos**

O escrito todo junto:

$(Inscritos \bowtie_{Inscritos.COD = Cursos.COD} Cursos) \bowtie_{Resultado\ 1.ID_AL = Alumnos.ID} Alumnos$

La tabla de este nuevo resultado sería:

Resultado 2

ID (inscrito)	ID_AL	COD (inscritos)	COD (cursos)	NOMBRE (curso)	FECHA_INICIO	DURACION	VALOR	ID (alumno)	NOMBRE (alumno)	CIUDAD	EDAD
1	01	05478	05478	Física	20-04	15	3.200	01	Pedro	Santiago	14
2	01	02145	02145	Biología	15-02	12	2.500	01	Pedro	Santiago	14
3	11	03547	03547	Matemáticas	01-03	30	4.000	11	Juan	Buenos Aires	18
4	21	02145	02145	Biología	15-02	12	2.500	21	Diego	Lima	12
5	41	03547	03547	Matemáticas	01-03	30	4.000	41	Manuel	Lima	17

Finalmente con una **Proyección** mostraremos el nombre del alumno y el curso inscrito:

$\Pi_{Resultado2.NOMBRE(alumno), Resultado2.NOMBRE(curso)}(\text{Resultado 2})$

Donde la tabla final sería:

NOMBRE (alumno)	NOMBRE (curso)
Pedro	Física
Pedro	Biología
Juan	Matemáticas
Diego	Biología
Manuel	Matemáticas

La expresión completa sería:

$\Pi_{Resultado2.NOMBRE(alumno), Resultado2.NOMBRE(curso)}((\text{Inscritos} \bowtie_{Inscritos.COD = Cursos.COD} Cursos) \bowtie_{Resultado1.ID_AL = Alumnos.ID} Alumnos)$

Mostrar los nombres y precios de los cursos inscritos con valor menor a 3.000 [editar]

$\Pi_{Resultado1.NOMBRE, Resultado1.VALOR}(\sigma_{Cursos.VALOR < 3000}(Cursos))$

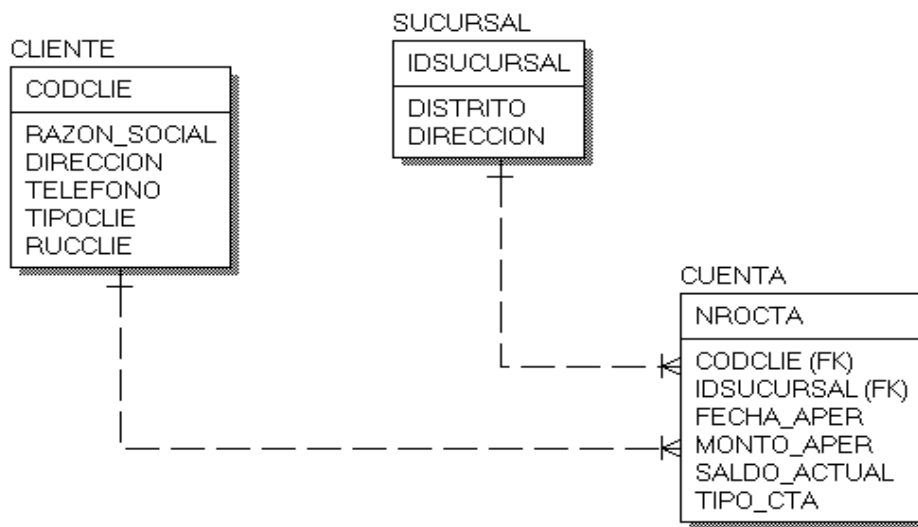
Lo que nos entregaría la tabla:

Resultado final

NOMBRE	VALOR
Biología	2.500
Música	1.500

➤ EJERCICIOS DE CREACIÓN DE TABLAS

Crear las siguientes tablas:



Aplicar las siguientes reglas de integridad de datos:

- Todo cliente debe tener un número de ruc irrepetible.

```
create table cliente(  
codclie char(5),  
razon_social varchar(50),  
direccion varchar(50),  
telefono char(7),  
tipoclie varchar(30),  
rucclie char(11) UNIQUE  
CONSTRAINT PK_CLIE PRIMARY KEY(codclie)  
)
```

- Agregar las claves foráneas a la tabla Cuentas. El tipo de cuenta puede tener sólo los valores I,II y III. Además debe tener a la fecha actual del sistema como valor por defecto para el campo fecha_aper.

```
create table sucursal(  
idsucursal char(8) primary key,  
distrito varchar(20),
```

```
direccion varchar(40)
)
```

```
create table cuentas(
nrocta char(8) primary key,
codclie char(5) constraint fk_cta1 foreign key(codclie) references
cliente(codclie),
idsucursal char(8),
fecha_aper datetime default getdate(),
monto_aper decimal(8,2),
saldo_actual decimal(8,2),
tipo_cta varchar(20) CONSTRAINT CHK_CTA CHECK(TIPO_CTA IN('I','II','III'))
CONSTRAINT FK_CTA1 FOREIGN KEY(CODCLIE) REFERENCES
CLIENTE(CODCLIE),
CONSTRAINT FK_CTA2 FOREIGN KEY(IDSUCURSAL) REFERENCES
SUCURSAL(IDSUCURSAL)
)
```

EJERCICIOS PARA RESOLVER

Interprete las siguientes reglas de negocio en tablas con restricciones.

- Las cuotas de pago para un crédito pueden ser de 3, 6, 12 y 24 meses.
- Para registrar a un nuevo socio del club, la cuota de inscripción debe ser mayor o igual a 500 nuevos soles.
- Los únicos turnos disponibles para matriculas de alumnos son mañana y tarde.

LABORATORIO # 7

Crear los modelos en Erwin y la posterior base de datos en SQL Server de los siguientes casos de estudio:

CASO 1

La empresa de formación X, desea llevar un control informatizado de los cursos que imparte así como de los profesores que participan en dichos cursos. Para ello, nos han dado las siguientes especificaciones:

- Cada curso, del que se desea conocer el título, el número de horas y el tema o los temas que trata, se identifica por un código de curso.
- Cada curso puede tener una serie de cursos cuyo realización previa es obligatoria (prerrequisito) o recomendada.
- Cada curso se puede impartir una o varias veces, en diferentes fechas y en cada edición del mismo pueden participar diferentes empleados.
- Los empleados, de los que se desea conocer su código de empleado, nombre, DNI y fecha de antigüedad en la empresa, pueden impartir y recibir cursos pero con la restricción de que en una misma edición de un curso no pueden participar como profesores y como alumnos.

CASO 2

La empresa Personal Quality desea incorporar en su política de contratación criterios de calidad del personal basados en la medición de sus habilidades o competencias.

- La empresa desea medir las competencias intelectuales de todos sus empleados y además desea conocer las competencias emocionales de sus directivos (por ejemplo, la capacidad de trabajo en grupo, la motivación, capacidad de liderazgo, etc.). De todas ellas se desea conocer: su código de identificación, su nombre y su descripción. Además, para cada competencia emocional se desea conocer, lo que se ha denominado el umbral; es decir, el valor mínimo de cada competencia por debajo del cual ningún empleado podrá ser directivo. Se requiere también que todo directivo mantenga este umbral mínimo en, al menos, 5 competencias emocionales.
- Para llevar a cabo este estudio, Personal Quality ha contactado con el Emotional Skill Center quien le ha proporcionado una batería de Test. Cada competencia está asociada a un conjunto de test que permiten medirla. Un test puede medir una única competencia. Cada test se identifica por un nombre y debe tener asociado un conjunto de preguntas, una plantilla para su corrección así como el modo en que se deberán interpretar los resultados.
- Cada empleado se identifica por un código interno. Además se quiere conocer el nombre, la dirección y un teléfono de contacto de cada empleado.

CASO 3

La gestión de una farmacia requiere poder llevar control de los medicamentos existentes, así como de los que se van sirviendo, para lo cual se pretende diseñar un sistema acorde a las siguientes especificaciones:

- En la farmacia se requiere una catalogación de todos los medicamentos existentes, para lo cual se almacenará un código de medicamento, nombre del medicamento, tipo de medicamento (jarabe, comprimido, pomada, etc.), unidades en stock, unidades vendidas y precio. Existen medicamentos de venta libre, y otros que sólo pueden dispensarse con receta médica.
- La farmacia adquiere cada medicamento a un laboratorio, o bien los fabrica ella misma. Se desea conocer el código del laboratorio, nombre, teléfono, dirección, fax así como el nombre de la persona de contacto.

- Los medicamentos se agrupan en familias, dependiendo del tipo de enfermedades a las que dicho medicamento se aplica.
- La farmacia tiene algunos clientes que realizan los pagos de sus pedidos a fin de cada mes (clientes con crédito). La farmacia quiere conocer las unidades de cada medicamento comprado (con o sin crédito) así como la fecha de compra. Además, es necesario tener los datos bancarios de los clientes con crédito, así como la fecha de pago de las compras que realizan.

Semana

8

Contenido:

- Definición y Formato para la creación de diccionario de datos.
- Ejercicios creando diccionario de Datos.
- Conceptos
- Recuperación
- Transacción
- Concurrencia – Problemas y soluciones
- Seguridad
- Manipulación de Datos
- Inserción de datos.
- Eliminación de registros
- Actualización de registros
- Conociendo el lenguaje SQL (Structured Query Language).
- Cláusulas SELECT, FROM, WHERE, ORDER BY.
- Ejercicios

DICCIONARIO DE DATOS - MANIPULACIÓN DE DATOS (DML)

➤ DICCIONARIO DE DATOS

✓ CONCEPTO

Un diccionario de datos es un conjunto de metadatos que contiene las características lógicas y puntuales de los datos que se van a utilizar en el sistema que se programa, incluyendo nombre, descripción, alias, contenido y organización.

Estos diccionarios se desarrollan durante el análisis de flujo de datos y ayuda a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño del proyecto.

Identifica los procesos donde se emplean los datos y los sitios donde se necesita el acceso inmediato a la información, se desarrolla durante el análisis de flujo de datos y auxilia a los analistas que participan en la determinación de los requerimientos del sistema, su contenido también se emplea durante el diseño.

En un diccionario de datos se encuentra la lista de todos los elementos que forman parte del flujo de datos de todo el sistema. Los elementos más importantes son flujos de datos, almacenes de datos y procesos. El diccionario de datos guarda los detalles y descripción de todos estos elementos.

✓ **FORMATO DE UN DICCIONARIO DE DATOS**

Una definición de un dato se introduce mediante el símbolo “=”; en este contexto el “=” se lee como “está definido por”, o “está compuesto de”, o “significa”. Para definir un dato completamente, la definición debe incluir:

El significado del dato en el contexto de la aplicación. Esto se documenta en forma de comentario.

La composición del dato, si es que está compuesto de otros elementos significativos. Los valores que el dato puede tomar, si se trata de un dato elemental que ya no puede ser descompuesto.

Diccionario de datos (DD) Este elemento del enfoque de base de datos es el conjunto centralizado de atributos lógicos que especifican la identificación y caracterización de los datos que se manejan en la BD. La BD contiene el valor de los datos, el DD contiene meta datos, es decir los atributos lógicos de dichos datos.

✓ **DATOS ELEMENTALES**

Son aquellos para los cuales no hay una descomposición significativa. Por ejemplo, puede ser que no se requiera descomponer el nombre de una persona en primer-nombre, apellido-materno y apellido-paterno; esto depende del contexto del sistema que se esté modelando. Cuando se han identificado los datos elementales, deben ser introducidos en el DD y proveer una breve descripción que describa el significado del dato. En el caso de que el dato tenga un nombre significativo, se puede omitir la descripción, sin embargo; es importante especificar las unidades de medida que el dato puede tomar.

Ejemplo: Peso = * peso del paciente al ingresar al hospital *

unidad: kilo, rango:2-150 *

Altura = * unidad: cm, rango: 100-200 * Sexo = * valores : [F|M] *

✓ **DATOS OPCIONALES**

Un dato opcional es aquel que puede o no estar presente como componente de un dato compuesto. Ejemplo: Dirección = calle + número + (ciudad) + (país) + (código-postal)

Selección

Indica que un elemento consiste de exactamente una opción de un conjunto de alternativas.

Ejemplos:

Sexo = [Femenino | Masculino]

Tipo-de-cliente = [Gubernamental | Académico | Industria | Otros]

Iteración

Se usa para indicar ocurrencias repetidas de un componente en un elemento compuesto.

Ejemplo:

Orden-de compra = nombre-cliente + dirección-de-envío + {artículo}

Significa que una orden de compra siempre debe contener un nombre de cliente, una dirección de envío y cero o más ocurrencias de un artículo.

✓ **EJERCICIOS**

Se pueden especificar límites superiores e inferiores a las iteraciones.

Orden-de compra = nombre-cliente + dirección-de-envío + 1{artículo}10

Significa que una orden de compra siempre debe contener un nombre de cliente, una dirección de envío y de 1 a 10 artículos.

Ejemplos de iteraciones con límites:

$a = 1\{b\}$

$a = \{b\}10$

$a = 1\{b\}10$

$a = \{b\}$

EJEMPLO REGISTRO DE EMPLEADOS = {Registro del empleado}

REGISTRO DE TIEMPOS DEL EMPLEADO = {Registro de tiempos del empleado}

Registro del empleado = * Datos de cada empleado*

Número de empleado + Información personal + Información de pago + Información de pago actual + Información anual

Registro de tiempos del empleado = Número de empleado + Nombre del empleado + Horas trabajadas

Cheque de pago del empleado = Número de empleado + Nombre de empleado + Dirección + Cantidades del pago actual + 5

Produce el cheque de pago del empleado

REGISTRO DE TIEMPOS DEL EMPLEADO

Empleado

Cheque de pago del empleado

Registro del empleado

Registro de tiempos del empleado

REGISTRO DE EMPLEADOS

El DD provee información del DER. En general, las instancias del DER corresponden a los almacenes de datos de los DFD. EJEMPLO: CLIENTES = {cliente}

cliente = nombre-cliente + dirección + número-teléfono

compra = * asociación entre un cliente y uno o más artículos *

nombre-cliente + 1{id-artículo + cantidad-artículos}

ARTÍCULOS = {artículo}

artículo = id-artículo + descripción

En el ejemplo anterior, cliente es la definición de un tipo de objeto (entidad) y una instancia del almacén de datos CLIENTES. La llave de cliente es el atributo nombre-cliente, el cual diferencia una instancia de otra. El signo @ es usado para indicar los campos llave, o bien estos campos llave se subrayan.

✓ **RAZONES PARA LA UTILIZACIÓN DE LOS DICCIONARIOS DE DATOS:**

- Para manejar los detalles en sistemas muy grandes, ya que tienen enormes cantidades de datos, aun en los sistemas más chicos hay gran cantidad de datos. Los sistemas al sufrir cambios continuos, es muy difícil manejar todos los detalles. Por eso se registra la información, ya sea sobre hoja de papel o usando procesadores de texto. Los analistas mas organizados usan el diccionario de datos automatizados diseñados específicamente para el análisis y diseño de software.
- Para asignarle un solo significado a cada uno de los elementos y actividades del sistema. Los diccionarios de datos proporcionan asistencia para asegurar significados comunes para los elementos y actividades del sistema y registrando detalles adicionales relacionados con el flujo de datos en el sistema, de tal manera que todo pueda localizarse con rapidez.
- Para documentar las características del sistema, incluyendo partes o componentes así como los aspectos que los distinguen. También es necesario saber bajo que circunstancias se lleva a cabo cada proceso y con qué frecuencia ocurren. Produciendo una comprensión más completa. Una vez que las características están articuladas y registradas, todos los participantes en el proyecto tendrán una fuente común de información con respecto al sistema.
- Para facilitar el análisis de los detalles con la finalidad de evaluar las características y determinar donde efectuar cambios en el sistema. Determina si son necesarias nuevas características o si están en orden los cambios de cualquier tipo. Se abordan las características:

✓ **CONCEPTOS BASICOS**

RECUPERACION

La recuperación significa que, si se da algún error en los datos, hay un bug de programa ó de hardware, el DBA (Administrador de base de datos) puede traer de vuelta la base de datos al tiempo y estado en que se encontraba en estado consistente antes de que el daño se causara. Las actividades de recuperación incluyen el hacer respaldos de la base de datos y almacenar

esos respaldos de manera que se minimice el riesgo de daño o pérdida de los mismos, tales como hacer diversas copias en medios de almacenamiento removibles y almacenarlos fuera del área en antelación a un desastre anticipado. La recuperación es una de las tareas más importantes de los DBA's.

La recuperación, frecuentemente denominada "recuperación de desastres", tiene dos formas primarias. La primera son los respaldos y después las pruebas de recuperación.

La recuperación de las bases de datos consiste en información y estampas de tiempo junto con bitácoras los cuales se cambian de manera tal que sean consistentes en un momento y fecha en particular. Es posible hacer respaldos de la base de datos que no incluyan las estampas de tiempo y las bitácoras, la diferencia reside en que el DBA debe sacar de línea la base de datos en caso de llevar a cabo una recuperación.

Las pruebas de recuperación consisten en la restauración de los datos, después se aplican las bitácoras a esos datos para restaurar la base de datos y llevarla a un estado consistente en un tiempo y momento determinados. Alternativamente se puede restaurar una base de datos que se encuentra fuera de línea sustituyendo con una copia de la base de datos.

Si el DBA (o el administrador) intentan implementar un plan de recuperación de bases de datos sin pruebas de recuperación, no existe la certeza de que los respaldos sean del todo válidos. En la práctica, los respaldos de la mayoría de los RDBMSs son raramente válidos si no se hacen pruebas exhaustivas que aseguren que no ha habido errores humanos o bugs que pudieran haber corrompido los respaldos.

TRANSACCION

Una transacción es una unidad lógica de procesamiento de la base de datos que incluye una o más operaciones de acceso a la base de datos, que pueden ser de inserción eliminación, modificación o recuperación. Las transacciones pueden delimitarse con sentencias explícitas `begin transaction` y `end transaction`.

Un SGBD se dice transaccional, si es capaz de mantener la integridad de los datos, haciendo que estas transacciones no puedan finalizar en un estado intermedio. Cuando por alguna causa el sistema debe cancelar la transacción, empieza a deshacer las órdenes ejecutadas hasta dejar la base de datos en su estado inicial (llamado punto de integridad), como si la orden de la transacción nunca se hubiese realizado.

Para esto, el lenguaje de consulta de datos SQL (Structured Query Language), provee los mecanismos para especificar que un conjunto de acciones deben constituir una transacción.

BEGIN TRAN: Especifica que va a empezar una transacción.

COMMIT TRAN: Le indica al motor que puede considerar la transacción completada con éxito.

ROLLBACK TRAN: Indica que se ha alcanzado un fallo y que debe restablecer la base al punto de integridad.

En un sistema ideal, las transacciones deberían garantizar todas las propiedades ACID; en la práctica, a veces alguna de estas propiedades se simplifica o debilita con vistas a obtener un mejor rendimiento.

PROPIEDADES

Toda transacción debe cumplir cuatro propiedades ACID:

- Atomicidad (Atomicity): es la propiedad que asegura que la operación se ha realizado o no, y por lo tanto ante un fallo del sistema no puede quedar a medias.
- Consistencia (Consistency): es la propiedad que asegura que sólo se empieza aquello que se puede acabar. Por lo tanto, se ejecutan aquellas operaciones que no van a romper la reglas y directrices de integridad de la base de datos.
- Aislamiento (Isolation): es la propiedad que asegura que una operación no puede afectar a otras. Esto asegura que la realización de dos transacciones sobre la misma información nunca generará ningún tipo de error.
- Permanencia (Durability): es la propiedad que asegura que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

La atomicidad frente a fallos se suele implementar con mecanismos de journaling, y la protección frente a accesos concurrentes mediante bloqueos en las estructuras afectadas. La serialibilidad viene garantizada por la atomicidad. La permanencia se suele implementar forzando a los periféricos encargados de almacenar los cambios a confirmar la completa y definitiva transmisión de los datos al medio (generalmente, el disco).

La forma algorítmica que suelen tener las transacciones es la siguiente:

```
iniciar transacción (lista de recursos a bloquear)
ejecución de las operaciones individuales.
if (todo_ok){
    aplicar_cambios
}
else{
    cancelar_cambios
}
```

En cualquier momento, el programa podría decidir que es necesario hacer fallar la transacción, con lo que el sistema deberá revertir todos los cambios hechos por las operaciones ya hechas. En el lenguaje SQL se denomina COMMIT a aplicar cambios y ROLLBACK a cancelar cambios.

Las transacciones suelen verse implementadas en sistemas de bases de datos y, más recientemente, se han visto incorporadas a como gestiona un sistema operativo la interacción con un sistema de archivos (como varias características de las bases de datos, debido a que son muy similares arquitectónicamente).

CONCURRENCIA

En sistemas multiusuario, es necesario un mecanismo para controlar la concurrencia. Se pueden producir inconsistencias importantes derivadas del acceso concurrente (3 problemas).

Las transacciones de los usuarios se podrían ejecutar de manera concurrente y podrían acceder y actualizar los mismos elementos de la BD.

Problemas y Soluciones

Problemas:

- Actualización perdida
- Actualización temporal (lectura sucia)

- Resumen incorrecto

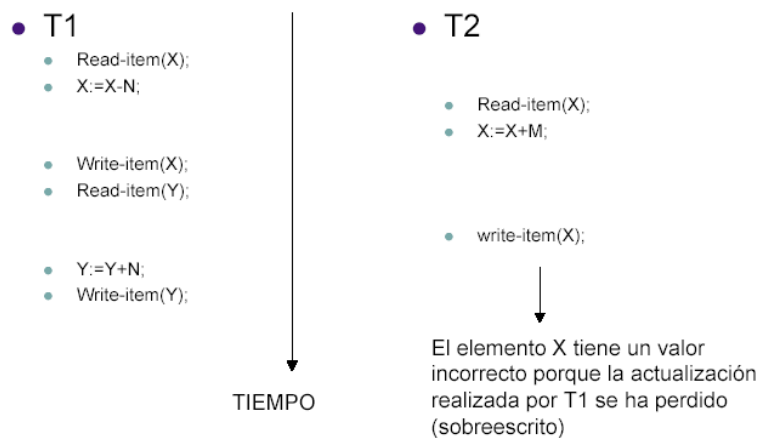
Ejemplo

Sistema de BD de reservas en una línea área.

T1: transfiere N reservas de un vuelo, cuyo número de asientos reservados está almacenado en el elemento de la BD llamado X, a otro vuelo, cuyo número de asientos reservados está almacenado en el elemento de la BD llamado Y.

ACTUALIZACIÓN PERDIDA

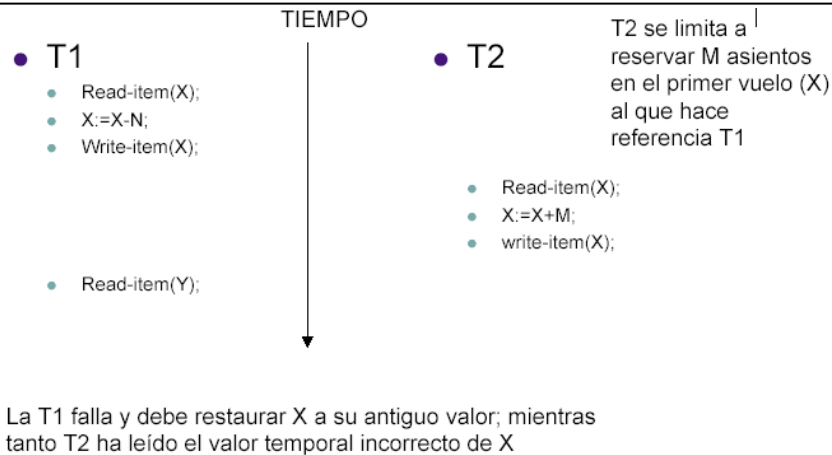
- Esto ocurre cuando las transacciones que tienen acceso a los mismos elementos de la BD tienen sus operaciones intercaladas de modo que hacen incorrecto el valor de algún elemento.
- T1 y T2 se introducen al mismo tiempo y sus operaciones se intercalan.



- El valor final del elemento X es incorrecto, porque T2 lee el valor de X ANTES de que T1 lo modifique en la BD, con lo que se pierde el valor actualizado que resulta de T1.
- Si $X=80$ al principio, $N=5$ (T1 transfiere 5 reservas de asientos del vuelo que corresponde a X al vuelo que corresponde a Y) y $M=4$ (T2 reserva 4 asientos en X), el resultado final debería ser $X=79$, pero es $X=84$, porque la actualización de T1 que eliminó 5 asientos de X se ha perdido.

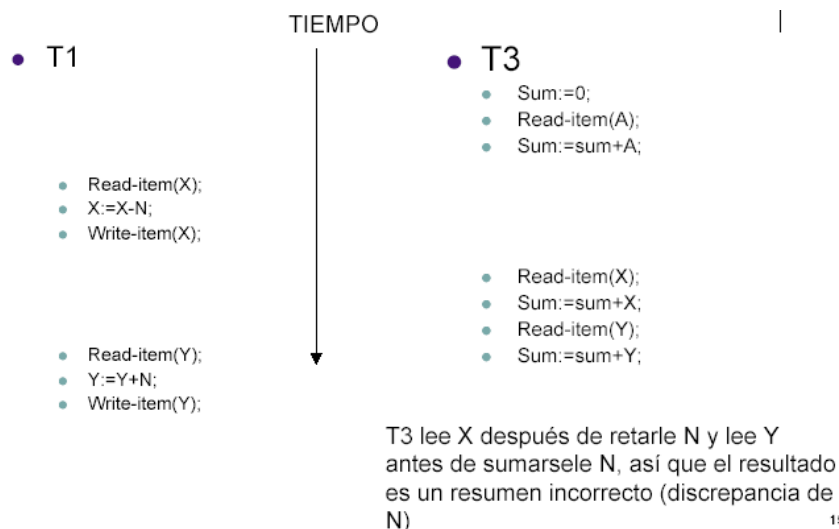
ACTUALIZACIÓN TEMPORAL

- Esto ocurre cuando una transacción actualiza un elemento de la BD y luego la transacción falla por alguna razón. Otra transacción tiene acceso al elemento actualizado antes de que se restaure a su valor original.
- T1 actualiza el elemento X y después falla antes de completarse, así que el sistema debe cambiar X otra vez a su valor original.
- Antes de que pueda hacerlo, la transacción T2 lee el valor “temporal” de X, que no se grabará permanentemente en la BD debido al fallo de T1.
- El valor que T2 lee de X se llama dato sucio, porque fue creado por una transacción que no se ha completado ni confirmado todavía.



RESUMEN INCORRECTO

Si una transacción está calculando una función agregada de resumen sobre varios registros mientras otras transacciones están actualizando algunos de ellos, puede ser que la función agregada calcule algunos valores antes de que se actualicen y otros después de actualizarse



SEGURIDAD

Seguridad significa la capacidad de los usuarios para acceder y cambiar los datos de acuerdo a las políticas del negocio, así como, las decisiones de los encargados. Al igual que otros metadatos, una DBMS relacional maneja la seguridad en forma de tablas. Estas tablas son las "llaves del reino" por lo cual se deben proteger de posibles intrusos

➤ LENGUAJE DE DEFINICION 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 de la misma llevar a cabo las tareas de consulta o manipulación de los datos, organizados por el modelo de datos adecuado.

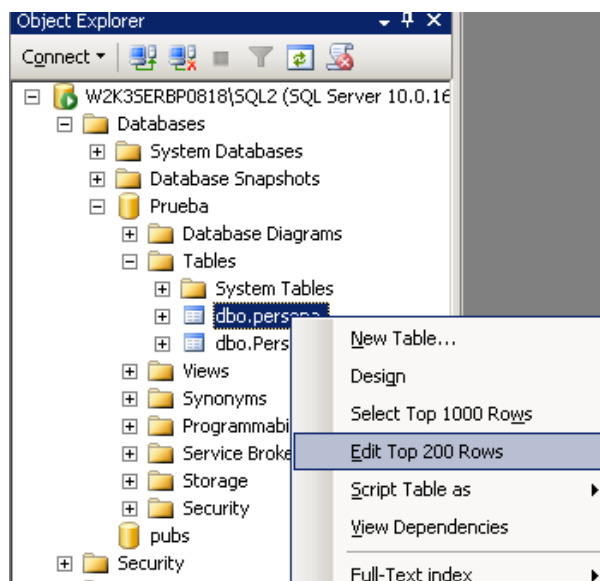
El lenguaje de manipulación de datos más popular hoy en día es SQL, usado para recuperar y manipular datos en una base de datos relacional. Otros ejemplos de DML son los usados por bases de datos IMS/DL1, CODASYL u otras.

Tenemos 4 sentencias DML's:

- **Insert into:** Sentencia que permite insertar registros de datos a las tablas de lavase de datos. Está muy ligado a la estructura de la tabla.
- **Delete:** Permite eliminar registros de datos de las tablas.
- **Update:** Sentencia que permite hacer modificaciones a los datos de las tablas.
- **Select:** La sentencia más poderosa del SQL, permite hacer consultas y recuperación de registros de datos de las tablas.

✓ INSERCIÓN DE REGISTROS DE DATOS

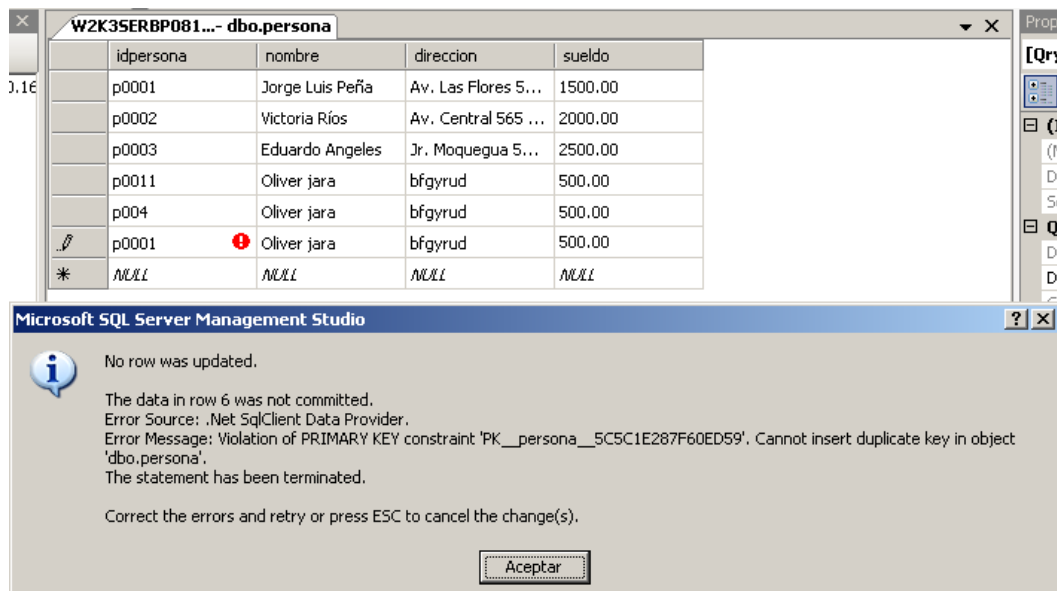
Para insertar un registro de datos, hacemos clic derecho sobre la tabla en el Explorador de Objetos, seleccionamos Editar las primeras 200 filas (Edit top 200 Rows)...



Luego de ello procedemos a insertar los registros uno a uno...

W2K3SERBP081...- dbo.persona				
	idpersona	nombre	direccion	sueldo
	p0001	Jorge Luis Peña	Av. Las Flores 5...	1500.00
	p0002	Victoria Ríos	Av. Central 565 ...	2000.00
	p0003	Eduardo Angeles	Jr. Moquegua 5...	2500.00
	p0011	Oliver jara	bfgyrud	500.00
	p004	Oliver jara	bfgyrud	500.00
▶	p05	Oliver jara	bfgyrud	500.00
*	NULL	NULL	NULL	NULL

Si insertamos algún dato que no corresponde a la definición de la tabla, o intentamos insertar una clave duplicada, nos aparece una ventana de mensaje de error...



✓ SENTENCIA DML INSERT INTO

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

Forma básica

```
INSERT INTO "tabla" ("columna1", ["columna2,..."]) VALUES ("valor1", ["valor2,..."])
```

Las cantidades de columnas y valores deben ser las mismas. Si una columna no se especifica, le será asignado el valor por omisión. Los valores especificados (o implícitos) por la sentencia INSERT deberán satisfacer todas las restricciones aplicables. Si ocurre un error de sintaxis o si alguna de las restricciones es violada, no se agrega la fila y se devuelve un error.

Ejemplo

```
INSERT INTO agenda_telefonica (nombre, numero) VALUES ('Roberto Jeldrez', '4886850');
```

Cuando se especifican todos los valores de una tabla, se puede utilizar la sentencia acortada:

```
INSERT INTO "tabla" VALUES ("valor1", ["valor2,..."])
```

Ejemplo:

Asumiendo que 'nombre' y 'numero' son las únicas columnas de la tabla 'agenda_telefonica'

```
INSERT INTO agenda_telefonica VALUES ('Roberto Jeldrez', '4886850');
```

Formas avanzadas

Inserciones en múltiples filas

Una característica de SQL (desde SQL-92) es el uso de constructores de filas para insertar múltiples filas a la vez, con una sola sentencia SQL:

```
INSERT INTO "tabla" ("columna1", ["columna2,..."]) VALUES ("valor1a", ["valor1b,..."]),  
("value2a", ["value2b,..."]),....
```

Ejemplo

Asumiendo ese 'nombre' y 'numero' son las únicas columnas en la tabla 'agenda_telefonica':

```
INSERT INTO agenda_telefonica VALUES ('Roberto Fernández', '4886850'), ('Alejandro Sosa',  
'4556550')
```

Que podía haber sido realizado por las sentencias

```
INSERT INTO agenda_telefonica VALUES ('Roberto Fernández', '4886850');
```

```
INSERT INTO agenda_telefonica VALUES ('Alejandro Sosa', '4556550');
```

Notar que las sentencias separadas pueden tener semántica diferente (especialmente con respecto a los triggers), y puede tener diferente rendimiento que la sentencia de inserción múltiple.

✓ EJERCICIO

Insertar registros a la siguiente tabla:

```
CREATE TABLE PERSONA(  
    CODPER CHAR(6),  
    NOMBREPER VARCHAR(40),  
    SEXO CHAR(1),  
    NRORUC CHAR(11),  
    FECHA_INSC DATETIME DEFAULT GETDATE(),  
    CONSTRAINT PK_CLIE PRIMARY KEY(CODPER),  
    CONSTRAINT CK_CLIE CHECK(SEXO IN('M','F'))  
)
```

Insertando registros...

```
INSERT INTO CLIENTE VALUES ('PE0001','Arturo Perez','M','41254687841','12/20/2010')
```

Con esto nos aparecerá un mensaje que dice: 1 fila insertada...

Pero vemos que el campo Fecha_insc tiene un constraint default, entonces podemos obviar este campo ya que tendrá este valor de todas maneras...

```
INSERT INTO CLIENTE (codper, nombreper, sexo, nruruc) VALUES ('PE0002','ARTURO PEREZ','M')
```

Veamos otro ejemplo con una tabla llamada Distrito:

```
CREATE TABLE DISTRITO  
(  
    CODIS CHAR(5) PRIMARY KEY,  
    DESCRIPCION VARCHAR(50)  
)
```

Insertando registros...

```
INSERT INTO DISTRITO VALUES('D0001','LA MOLINA')  
INSERT INTO DISTRITO VALUES('D0002','LA VICTORIA')  
INSERT INTO DISTRITO VALUES('D0003','LINCE')  
INSERT INTO DISTRITO VALUES('D0004','SURQUILLO')  
INSERT INTO DISTRITO VALUES('D0005','SAN ISIDRO')
```

Tener en cuenta:

- El orden de los datos debe ser el mismo orden que tienen los campos de las tablas, de lo contrario SQL Server nos mostrará un mensaje que indica que se está respetando la estructura definida de la tabla.
- Los valores numéricos para aquellos campos definidos con un tipo de dato numérico como Decimal o Int, se colocan sin comilla. Sólo llevarán comillas aquellos datos de texto y fecha
- No se ingresan aquellos datos de campos marcados con la restricción Identity.

Veamos un ejemplo de un registro mal insertado:

Utilizando nuevamente la tabla Persona:

```
INSERT INTO CLIENTE VALUES ('PE0001','Arturo Perez','M','12/20/2010')
```

Estamos cometiendo un grave error, ya que no estamos respetando la estructura de la tabla, porque no estamos indicando un valor para el campo RUCCLIE, que va después del campo SEXO, SQL Server no permitirá el ingreso de este registro.

Además estamos reingresando una clave ya repetido en el campo CODPER, según la integridad referencial de la tabla, este campo es una llave primaria por lo que no puede existir dos códigos repetidos.

Debemos ingresar este siguiente registro entonces:

```
INSERT INTO CLIENTE VALUES ('PE0003','Arturo Perez','M','65302104710','12/20/2010')
```

Debemos insertar los registros que correspondan a los campos, tomando en cuenta tipo de dato, restricción y reglas de negocio para tener siempre los datos correctos, las cuales nos entregarán siempre la información correcta requerida.

✓ **ELIMINACION DE REGISTROS**

Utilizamos la Sentencia SQL Delete, que permite eliminar uno o más registros de una tabla, esto lo haremos cuando ya no necesitemos dicho registro.

Forma Básica

DELETE FROM TABLA WHERE CONDICION

Descripción de las cláusulas

- FROM: Palabra clave opcional que se puede utilizar entre la palabra clave DELETE y el destino (tabla, vista o rowset)
- FROM <tabla>: Especifica una cláusula FROM adicional. Esta extensión de Transact-SQL para DELETE permite especificar datos y eliminar las filas correspondientes de la tabla en la primera cláusula FROM. Se puede utilizar esta extensión, que especifica una combinación, en lugar de una subconsulta en la cláusula WHERE para identificar las filas que se van a quitar.
- WHERE: Especifica las condiciones utilizadas para limitar el número de filas que se van a eliminar. Si no se proporciona una cláusula WHERE, DELETE quita todas las filas de la tabla.

Ejemplo:

Tenemos la siguiente tabla:

CODPER	NOMBRE	DIRECCION
PE0001	ALAN GARCIA	AV.ELSOL 345
PE0002	ERICK TORRES	JR. LAMPA 456
PE0003	JUAN VARGAS	AV. LOS ROBLES 564

Si queremos eliminar a la persona Juan Vargas, ejecutaremos la siguiente sentencia:

```
DELETE FROM Persona WHERE codper='PE0001'
```

Si no indicamos una cláusula Where, se eliminarán todos los registros sin excepción, la tabla quedaría así:

CODPER	NOMBRE	DIRECCION
PE0001	ALAN GARCIA	AV.ELSOL 345
PE0002	ERICK TORRES	JR. LAMPA 456

Tener en cuenta la integridad referencial de datos, que indica que no se podrá eliminar un registro cuya clave está relacionada con algún otro registro de una tabla relacionada, siempre y cuando se realice una eliminación en cascada, como hemos visto en sesiones anteriores.

➤ **ACTUALIZACION DE REGISTROS**

Utilizamos la sentencia SQL Update, que permite modificar los datos de los registros de las tablas, ideal cuando se desea actualizar un dato importante o corregir un dato erróneo.

Forma Básica.

UPDATE TABLA SET CAMPO='NUEVOVALOR' WHERE CONDICION

Descripción de las cláusulas:

- UPDATE: Aquí indicamos el nombre de la tabla sobre la cual se realizará alguna modificación de dato.
- SET: Cláusula que indica el campo sobre el cual se hará el cambio de valor, colocaremos el nuevo valor que contendrá el campo, aquí no es importante el valor que tenía hasta ese momento.
- WHERE: Especifica las condiciones utilizadas para limitar el número de filas que se van a actualizar. Si no se proporciona una cláusula WHERE, DELETE actualiza todas las filas de la tabla.

Ejemplo:

Tenemos la siguiente tabla:

CODPER	NOMBRE	DIRECCION
PE0001	ALAN GARCIA	AV.ELSOL 345
PE0002	ERICK TORRES	JR. LAMPA 456
PE0003	JUAN VARGAS	AV. LOS ROBLES 564

Para modificar la dirección de la persona Erick Torres, debemos ejecutar la siguiente instrucción:

UPDATE Persona SET Dirección = 'Av. Las Torres 453 Cercado' WHERE codper='pe0002'

En la cláusula Where debemos indicar algún dato de algunos de los campos que permita identificar al registro que debe actualizar, si indicamos mal esta condición, modificaremos registros que no deseábamos cambiar. La tabla queda así:

CODPER	NOMBRE	DIRECCION
PE0001	ALAN GARCIA	AV.ELSOL 345
PE0002	ERICK TORRES	AV. LAS TORRES 453 CERCADO'
PE0003	JUAN VARGAS	AV. LOS ROBLES 564

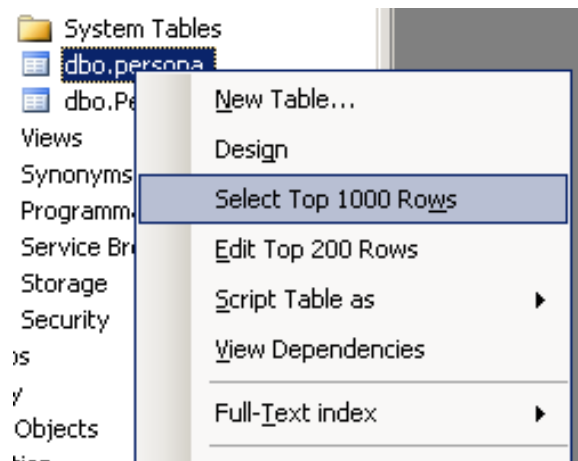
Tener en cuenta la integridad referencial de datos, que indica que se puede actualizar un dato de la tabla principal y de los registros de las tablas relacionadas, siempre y cuando se realice una actualización en cascada, como hemos visto en sesiones anteriores.

➤ CONSULTA DE DATOS

El proceso más importante que podemos llevar a cabo en una base de datos es la consulta de los datos. De nada serviría una base de datos si no pudiéramos consultarla. Es además la operación que efectuaremos con mayor frecuencia.

Para consultar la información SQL pone a nuestra disposición la sentencia SELECT.

Para consultar los registros de una tabla, podemos hacer clic derecho y seleccionamos Seleccionar las primeras 1000 filas (Select top 1000 Rows)...



Veremos la consulta con el script generado....

SQLQuery1.sql - W2K3SERBP...))

```
/****** Script for SelectTopNRows command from SSMS *****/  
SELECT TOP 1000 [idpersona]  
    , [nombre]  
    , [direccion]  
    , [sueldo]  
FROM [Prueba] . [dbo] . [persona]
```

Results Messages

	idpersona	nombre	direccion	sueldo
1		Oliver jara	bfgyrud	500.00
2	p0001	Jorge Luis Peña	Av. Las Flores 567 San Juan	1500.00
3	p0002	Victoria Ríos	Av. Central 565 Lince	2000.00
4	p0003	Eduardo Angeles	Jr. Moquegua 567 Cercado	2500.00
5	p0011	Oliver jara	bfgyrud	500.00
6	p004	Oliver jara	bfgyrud	500.00

Pero veamos ahora cómo crear una consulta por medio de la sentencia DML Select...

LA SENTENCIA SELECT

La sentencia SELECT nos permite consultar los datos almacenados en una tabla de la base de datos.

El formato de la sentencia select es:

```
SELECT [ALL | DISTINCT ]
        <nombre_campo> [{,<nombre_campo>}]
FROM <nombre_tabla>|<nombre_vista>
    [{,<nombre_tabla>|<nombre_vista>}]
[WHERE <condicion> [{ AND|OR <condicion>}]]
[GROUP BY <nombre_campo> [{,<nombre_campo >}]]
[HAVING <condicion>[{ AND|OR <condicion>}]]
[ORDER BY <nombre_campo>|<indice_campo> [ASC | DESC]
    [{,<nombre_campo>|<indice_campo> [ASC | DESC ]}]]
```

Veamos por partes que quiere decir cada una de las partes que conforman la sentencia.

SELECT. Palabra clave que indica que la sentencia de SQL que queremos ejecutar es de selección.

ALL. Indica que queremos seleccionar todos los valores. Es el valor por defecto y no suele especificarse casi nunca.

DISTINCT. Indica que queremos seleccionar sólo los valores distintos.

FROM. Indica la tabla (o tablas) desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se denomina a la consulta "consulta combinada" o "join". En las consultas combinadas es necesario aplicar una condición de combinación a través de una cláusula WHERE.

WHERE. Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admiten los operadores lógicos AND y OR.

GROUP BY. Especifica la agrupación que se da a los datos. Se usa siempre en combinación con funciones agregadas.

HAVING. Especifica una condición que debe cumplirse para los datos. Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Su funcionamiento es similar al de WHERE pero aplicado al conjunto de resultados devueltos por la consulta. Debe aplicarse siempre junto a GROUP BY y la condición debe estar referida a los campos contenidos en ella.

ORDER BY: Presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC (orden ascendente) y DESC (orden descendente). El valor predeterminado es ASC.

Para formular una consulta a la tabla tCoches (creada en el capítulo de tablas) y recuperar los campos matricula, marca, modelo, color, numero_kilometros, num_plazas debemos ejecutar la siguiente consulta. Los datos serán devueltos ordenados por marca y por modelo en orden ascendente, de menor a mayor.

```
SELECT matricula, marca, modelo, color, numero_kilometros, num_plazas FROM tCoches
ORDER BY marca,modelo
```


La palabra clave FROM indica que los datos serán recuperados de la tabla tCoches. Podríamos haber especificado más de una tabla, pero esto se verá en el apartado de consultas combinadas.

También podríamos haber simplificado la consulta a través del uso del comodín de campos, el asterisco "**".

```
SELECT * FROM tCoches ORDER BY marca,modelo
```

El uso del asterisco indica que queremos que la consulta devuelva todos los campos que existen en la tabla.

LA CLÁUSULA WHERE

La cláusula WHERE es la instrucción que nos permite filtrar el resultado de una sentencia SELECT. Habitualmente no deseamos obtener toda la información existente en la tabla, sino que queremos obtener sólo la información que nos resulte útil en ese momento. La cláusula WHERE filtra los datos antes de ser devueltos por la consulta.

En nuestro ejemplo, si queremos consultar un coche en concreto debemos agregar una cláusula WHERE. Esta cláusula especifica una o varias condiciones que deben cumplirse para que la sentencia SELECT devuelva los datos. Por ejemplo, para que la consulta devuelva sólo los datos del coche con matrícula M-1525-ZA debemos ejecutar la siguiente sentencia:

```
SELECT matricula, marca, modelo, color, numero_kilometros, num_plazas FROM tCoches  
WHERE matricula = 'M-1525-ZA'
```

Cuando en una cláusula Where queremos incluir un tipo texto, debemos incluir el valor entre comillas simples.

Además, podemos utilizar tantas condiciones como queramos, utilizando los operadores lógicos AND y OR. El siguiente ejemplo muestra una consulta que devolverá los coches cuyas matrículas sean M-1525-ZA o bien M-2566-AA.

```
SELECT matricula, marca, modelo, color, numero_kilometros, num_plazas FROM tCoches  
WHERE matricula = 'M-1525-ZA' OR matricula = 'M-2566-AA'
```

Además una condición WHERE puede ser negada a través del operador lógico NOT. La siguiente consulta devolverá todos los datos de la tabla tCoches menos el que tenga matrícula M-1525-ZA.

```
SELECT matricula, marca, modelo, color, numero_kilometros, num_plazas FROM tCoches  
WHERE NOT matricula = 'M-1525-ZA'
```

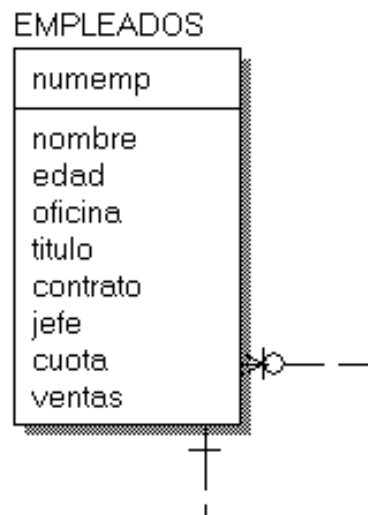
Forma básica.

La forma más básica del Select es la siguiente.

```
SELECT [Campos][*] FROM Tabla WHERE CONDICION
```

LABORATORIO # 8 – EJERCICIOS

Tenemos la siguiente tabla en nuestro modelo, contiene una relación recursiva, que indica que existen empleados que dependen de otros, el jefe.



La tabla en SQL Server...

numemp	nombre	edad	oficina	titulo	contrato	jefe	cuota	ventas
101	Antonio Viguer	45	12	representante	20/10/86	104	300.000 Pts	305.000 Pts
102	Alvaro Jaumes	48	21	representante	10/12/86	108	350.000 Pts	474.000 Pts
103	Juan Rovira	29	12	representante	01/03/87	104	275.000 Pts	286.000 Pts
104	José González	33	12	dir ventas	19/05/87	106	200.000 Pts	143.000 Pts
105	Vicente Pantalla	37	13	representante	12/02/88	104	350.000 Pts	368.000 Pts
106	Luis Antonio	52	11	dir general	14/06/88		275.000 Pts	299.000 Pts
107	Jorge Gutiérrez	49	22	representante	14/11/88	108	300.000 Pts	186.000 Pts
108	Ana Bustamante	62	21	dir ventas	12/10/89	106	350.000 Pts	361.000 Pts
109	María Sunta	31	11	representante	12/10/99	106	300.000 Pts	392.000 Pts
110	Juan Víctor	41		representante	13/01/90	104		76.000 Pts

Diccionario de datos de la tabla Empleado:

Tabla empleados con los siguientes campos:

- numemp: número del empleado
- nombre : nombre y apellidos del empleado
- edad : edad del empleado
- oficina : número de la oficina donde trabaja el empleado, p.ej. Antonio Viguer trabaja en la oficina 12 de Alicante
- titulo : el cargo que desempeña el empleado
- contrato : fecha en que se contrató al empleado
- jefe: número de su jefe inmediato, p.ej. El jefe de Antonio Viguer es José González. Observar que Luis Antonio no tiene jefe, es el director general.
- cuota : cuota del empleado, sería el importe mínimo de ventas que debe alcanzar el empleado en el año
- ventas : importe de ventas realizadas durante este año

Ejecutemos algunas sentencias:

1. Insertar un nuevo registro de datos.

```
INSERT INTO Empleados VALUES ('111', 'Oliver Jara', '30', '11', 'dir general', '25/01/2010', ' ', '300.00,85.00')
```

2. Modificar la edad del empleado Luis Antonio quien tiene 40 años pero aparece con 52 años.

```
UPDATE Empleados SET edad='40' WHERE nombre = 'Luis Antonio'
```

3. Eliminar al empleado Juan Víctor.

```
DELETE FROM Empleados WHERE nombre = 'Juan Víctor'
```

4. Mostrar en una consulta a aquellos empleados que tienen como jefe al señor José Gonzales cuyo código es el 104.

```
SELECT * FROM Empleados WHERE numemp = '104'
```

Podemos hacerlo también así:

```
SELECT * FROM Empleados WHERE nombre = 'José Gonzales'
```

5. Mostrar en una consulta a todos aquellos Empleados que tengan el título de representante o directores de ventas.

```
SELECT * FROM Empleados WHERE titulo = 'representante' OR titulo = 'dir ventas'
```

Debemos colocar los datos exactamente igual a como se encuentran en los registros de la tabla.

6. Asignar una nueva oficina a todos aquellos que tienen la número 12 actualmente, su nueva oficina será la número 22.

```
UPDATE Empleados SET oficina = '22' WHERE oficina = '12'
```

7. Mostrar en una consulta el nombre y número de todos los empleados y el de sus respectivos jefes.

```
SELECT numemp, nombre, jefe FROM Empleados
```

8. Mostrar en una consulta a todos los empleados en un listado ordenado ascendente según el número de oficina.

```
SELECT * FROM Empleados ORDER BY oficina
```

EJERCICIO DE INSERCIÓN DE REGISTROS

Ejecutar las siguientes tablas e inserción de registros...

```
CREATE TABLE CLIENTES(  
IDCLIENTE INT IDENTITY NOT NULL,  
NOMCLIENTE VARCHAR(18) NOT NULL,  
APECLIENTE VARCHAR(25) NOT NULL,  
DIRCLIENTE VARCHAR(50)  
PRIMARY KEY (IDCLIENTE)  
)
```

```
CREATE TABLE EMPLEADO(  
IDEMPLEADO INT IDENTITY NOT NULL,  
NOMEMPLEADO VARCHAR(20) NOT NULL,  
APEEMPLEADO VARCHAR(25) NOT NULL,  
FECINGRESO SMALLDATETIME NOT NULL,  
SUELDO DECIMAL(10,2) NULL CHECK (SUELDO > 0)  
PRIMARY KEY (IDEMPLEADO)  
)
```

```
CREATE TABLE PEDIDO(  
IDPEDIDO VARCHAR(7) NOT NULL,  
IDCLIENTE INT NOT NULL,  
IDEMPLEADO INT NOT NULL,  
FECPEDIDO SMALLDATETIME NOT NULL,  
TOTAL_PEDIDO DECIMAL(8,2) CHECK (TOTALPEDIDO < 2000)  
FOREIGN KEY (IDCLIENTE) REFERENCES CLIENTES,  
FOREIGN KEY (IDEMPLEADO) REFERENCES EMPLEADO  
)
```

Registros a insertar...

```
Insert Clientes (NomCliente,ApeCliente,DirCliente) Values('ROSA','SOLIS','BARRANCO')  
Insert Clientes (NomCliente,ApeCliente,DirCliente) Values('CARLOS','MORI','LIMA')  
Insert Clientes (NomCliente,ApeCliente,DirCliente) Values('JUAN','ROJAS','ATE')  
Insert Clientes (NomCliente,ApeCliente,DirCliente) Values('DANIELA','CASTRO','COMAS')  
Insert Clientes (NomCliente,ApeCliente,DirCliente) Values('LOURDES','VILLENAL','LIMA')
```

```
Insert Empleado (NomEmpleado,ApeEmpleado,FecIngreso)  
Values('CARLOS','TORRES','17/04/2000')  
Insert Empleado (NomEmpleado,ApeEmpleado,FecIngreso)  
Values('SOFIA','CONTRERAS','18/03/2002')  
Insert Empleado (NomEmpleado,ApeEmpleado,FecIngreso)  
Values('RAUL','FLORES','05/06/2001')
```

```
Insert Pedido values('0000001',1,1,'04/11/2008',100)  
Insert Pedido values('0000002',2,1,'05/08/2008',200)  
Insert Pedido values('0000003',1,2,'14/06/2008',500)
```

Comprobamos los datos insertados...

```
SELECT * FROM CLIENTES  
SELECT * FROM EMPLEADO  
SELECT * FROM PEDIDO
```

Contenido:

- Revisión de Proyectos (Teórico)
 - DER Normalizado
 - MER (Elaborado en ERWIN).
 - Diccionario de Datos
- Práctica Calificada 03 - Teoría
 - Normalización
 - DER
 - Álgebra Relacional (03 tablas)
- Revisión de Proyectos. (Laboratorio)
 - Ejecución de Script (SQL)
 - Creación de Base de datos con sus archivos
 - Creación de tablas con restricciones CHECK
 - Inserción de registros (15 por tabla)
 - Relaciones
- Práctica Calificada
 - Creación de una BD simple
 - Creación de tablas con integridad de datos y referencial (5 Tablas)
 - Ingreso de datos
 - Actualización

REVISION DE PROYECTOS

➤ REVISION DE PROYECTOS – TEORICO

El objetivo de este proyecto es que los alumnos apliquen todo lo aprendido en el curso mediante la creación de un DER normalizado, su posterior MER elaborado en Erwin y el diccionario de datos posterior.

Aquí se presenta un caso propuesto para el desarrollo del proyecto. Se recomienda asignar este proyecto con algunas semanas de anticipación.

ESPECIFICACIONES:**Para el desarrollo teórico**

- ✓ Crear el diseño de la base de datos utilizando Erwin 7.1, aplicando todos los conocimientos aplicados en clase.
- ✓ El diseño debe estar debidamente normalizado por lo menos hasta la 3ra forma normal de datos. Aplicar técnicas de Denormalización en caso crea el grupo conveniente.
- ✓ Presentación mediante exposición de trabajo, se tomará en cuenta la presentación personal del grupo.

CASO DE ESTUDIO**SISTEMA DE CONTROL DE PROYECTOS POR PERSONAL**

La compañía Rosales S.A. desea implementar un sistema de control, para lo cual se nos proporciona la siguiente información: La compañía tiene un conjunto de departamentos. Cada departamento tiene un conjunto de empleados, un conjunto de proyectos y un conjunto de oficinas. Cada empleado tiene una historia de empleos (trabajos que ha desempeñado anteriormente), para cada empleo existe una categoría de salarios (conjunto de salarios recibidos mientras tenía ese empleo).

Requerimientos:

- Se desea tener control sobre los departamentos, presupuestos asignados y los gerentes al mando.
- Acerca del empleado se requiere mantener información acerca de los trabajos efectuados anteriormente, fechas, salarios (actuales y anteriores), así como sus datos personales.
- Información al día de los proyectos actualmente ejecutándose, así como el personal asignado a cada proyecto.
- Información sobre las oficinas, así como sus dimensiones.
- Generación de reportes gerenciales mensuales.

Presentar el Modelo Entidad Relación en clase, sustentando las técnicas de normalización, y técnicas de Abstracción de datos (Generalización – Clasificación).

➤ REVISION DE PROYECTOS EN LABORATORIO

- Crear la base de datos utilizando SQL Server 2005, respetando la integridad referencial de datos.
- Aplicar restricciones (los 4 tipos) y datos insertados (mínimo 5 registros para tablas de control y 20 registros para tablas de transacciones)
- Mostrar consultas sql aplicadas en clase (Select, update y delete).

➤ **PRACTICA CALIFICADA N° 3 – Teoría**

1. Aplicar la técnica de normalización del siguiente documento...

[illegible]

2. Realiza el Modelo Entidad relación del siguiente caso de estudio.

La empresa Personal Quality desea incorporar en su política de contratación criterios de calidad del personal basados en la medición de sus habilidades o competencias.

La empresa desea medir las competencias intelectuales de todos sus empleados y además desea conocer las competencias emocionales de sus directivos (por ejemplo, la capacidad de trabajo en grupo, la motivación, capacidad de liderazgo, etc.). De todas ellas se desea conocer: su código de identificación, su nombre y su descripción. Además, para cada competencia emocional se desea conocer, lo que se ha denominado el umbral; es decir, el valor mínimo de cada competencia por debajo del cual ningún empleado podrá ser directivo.

Se requiere también que todo directivo mantenga este umbral mínimo en, al menos, 5 competencias emocionales.

Para llevar a cabo este estudio, Personal Quality ha contactado con el Emotional Skill Center quien le ha proporcionado una batería de Test. Cada competencia está asociada a un conjunto de test que permiten medirla. Un test puede medir una única competencia. Cada test se identifica por un nombre y debe tener asociado un conjunto de preguntas, una plantilla para su corrección así como el modo en que se deberán interpretar los resultados.

Cada empleado se identifica por un código interno. Además se quiere conocer el nombre, la dirección y un teléfono de contacto de cada empleado

3. Aplicar algunas operaciones del álgebra relacional de las siguientes tablas.

numemp	nombre	edad	oficina	titulo	contrato	jefe	cuota	ventas
101	Antonio Viguer	45	12	representante	20/10/86	104	300.000 Pts	305.000 Pts
102	Alvaro Jaumes	48	21	representante	10/12/86	108	350.000 Pts	474.000 Pts
103	Juan Rovira	29	12	representante	01/03/87	104	275.000 Pts	286.000 Pts
104	José González	33	12	dir ventas	19/05/87	106	200.000 Pts	143.000 Pts
105	Vicente Pantalla	37	13	representante	12/02/88	104	350.000 Pts	368.000 Pts
106	Luis Antonio	52	11	dir general	14/06/88		275.000 Pts	299.000 Pts
107	Jorge Gutiérrez	49	22	representante	14/11/88	108	300.000 Pts	186.000 Pts
108	Ana Bustamante	62	21	dir ventas	12/10/89	106	350.000 Pts	361.000 Pts
109	María Sunta	31	11	representante	12/10/99	106	300.000 Pts	392.000 Pts
110	Juan Víctor	41		representante	13/01/90	104		76.000 Pts

oficina	ciudad	region	dir	objetivo	ventas
11	Valencia	este	106	575.000 Pts	693.000 Pts
12	Alicante	este	104	800.000 Pts	735.000 Pts
13	Castellon	este	105	350.000 Pts	368.000 Pts
21	Babajoz	oeste	108	725.000 Pts	836.000 Pts
22	A Coruña	oeste	108	300.000 Pts	186.000 Pts
23	Madrid	centro	108		
24	Madrid	centro	108	250.000 Pts	150.000 Pts
26	Pamplona	norte			
28	Valencia	este		900.000 Pts	0 Pts

➤ **PRACTICA CALIFICADA Nº 3 - Laboratorio**

1. Crear la base de datos del siguiente MER, debe tener todas las propiedades que corresponden.

Deseamos administrar la información correspondiente un pequeño banco con una red de sucursales en todo el territorio nacional.

El banco ofrece a sus clientes cuentas corrientes y cuentas de ahorro. Un cliente tiene al menos una cuenta, aunque puede tener varias de cualquiera de los dos tipos. Cada cuenta puede pertenecer a un sólo cliente (el titular de la cuenta), o bien a dos (el segundo de ellos siendo el co-titular de la cuenta).

De los clientes interesa el nombre y dirección y se identifican por un código. Los clientes pueden ser personas reales u organizaciones. De las personas guardaremos su edad, fecha de nacimiento y sexo; en cambio de las organizaciones guardaremos su tipo (empresa, institución pública, etc...), un representante y su número de empleados.

Cada cuenta se identifica por un código cuenta cliente (en siglas, CCC) formado por el identificador del banco, la sucursal y el número de cuenta (dentro de dicha sucursal).

De cada cuenta interesa conocer su saldo actual y su saldo medio.

Cada sucursal se identifica por su número. Además tiene una dirección, un código postal y una ciudad.

Los empleados del banco se identifican por su DNI. También interesa conocer su nombre, fecha de nacimiento, sexo y la sucursal en la cual trabajan (un empleado no puede trabajar en más de una sucursal).

2. Crear las tablas correspondientes de la base de datos.
3. Insertar 5 registros a cada tabla de la base de datos creada.
4. Realizar algunas modificaciones de registros mediante la sentencia Update.

BIBLIOGRAFIA

- **Base de Datos Orientado a Objetos**, Microsoft
- **Fundamentos de Base de Datos, Tercera Edición. McGraw-Hill**
Abraham Silberschatz
- **Comunidad de CA Data Modeler**, <http://www.cacomvip.ca.com>.
Ca 2010.
- **Introducción a la Base de datos, Modelo relacional, Primera edición.**
Poins Capota Olga – Marín Ruiz Nicolaz. Año 2005
- **SQL Server 2005 – SQL Transact, Colección recursos informáticos.**
Gerome Gabillaud.