**Presentación del manual**

La cualificación profesional es el "conjunto de competencias profesionales con significación en el empleo que pueden ser adquiridas mediante formación modular u otros tipos de formación, así como a través de la experiencia laboral" (Ley 5/2002 de las Cualificaciones y de la Formación Profesional) .

Cada cualificación se organiza en unidades de competencia, siendo la unidad de competencia el agregado mínimo de competencias profesionales, susceptible de reconocimiento y acreditación parcial.

Así mismo, cada unidad de competencia lleva asociado un módulo formativo, donde se describe la formación necesaria para adquirir esa unidad de competencia.

Siguiendo esta secuencia, el presente manual "Operaciones de mantenimiento y consulta de datos", está basado en los contenidos del Módulo formativo: "Creación y mantenimiento de componentes software en sistemas de planificación de recursos empresariales y de gestión de relaciones con clientes" asociado a la Unidad de Competencia: "Realizar y mantener componentes software en un sistema de planificación de recursos empresariales y de gestión de relaciones con los clientes." según el Real Decreto correspondiente.

**MóDULO FORMATl'"VO: · Cl'eación y mantenimiento de··.componentes**

**:softwal'e:enssitemas .cfe'::etanificación de ·.recursos empresariales• y de nestión de l'elaciones:con clientes** .

Nivel: 3

Código: MF1215\_3

Asociado a la UC: UC1215\_3 Realizar y mantener componentes software en un sistema de planificación de recursos empresariales y de gestión de relaciones con los clientes.

Horas: 210

**UlíHDÁD ·FoRlllíATIVA: Operaciones de mantenimiento y consulta de datos**

Nivel: 3

Código: UF1888

Asociado a la UC: UC1215\_3 Horas: 90

# •

***UF 1888 OPERAC I ONES DE MANTEN IMIENTO Y CONSULTA DE DATOS***

o

**ÍNDICE**

UNIDAD DIDACTICA 1. MODELOS DE DATOS EN SISTEMAS ERP-CRM 9

1. [Modelos de datos tipos y características 11](#_TOC_250018)
   1. [Características del modelo de datos 14](#_TOC_250017)
2. [Definición del modelo de datos 18](#_TOC_250016)
   1. [Modelo relacional, definición de mecanismos que aseguren la integridad referencial 18](#_TOC_250015)

[2.2. Definición de tablas 25](#_TOC_250014)

[2.3. Conexión entre tablas 26](#_TOC_250013)

[RECUERDA 29](#_TOC_250012)

[Preguntas de Autoevaluación 31](#_TOC_250011)

UNIDAD DIDACTICA 2. DISEÑO DE DATOS •••.............•........•......••. 33

1. Definición de oqjetos y estructuras de datos, características 36
   1. [Definición de los elementos que componen el diseño de la base de datos 45](#_TOC_250010)
   2. Definición de relaciones entre Jos elementos 46
2. Creación, modificación y borrado de objetos y estructuras de datos 47
   1. [Mantenimiento de la base de datos 48](#_TOC_250009)
   2. Realización de operaciones entre tablas que garanticen la creación, modificación y

borrado de registros entre las tablas definidas en el modelo 57

[RECUERDA 71](#_TOC_250008)

[Preguntas de Autoevaluación ........................................................... ,.,, 73](#_TOC_250007)

UNIDAD DIDACTICA 3. DEFINICIÓN DE LOS TIPOS DE DATOS ..•.••••••...•. 75

1. Definición, tipos de datos y características semanticas 77
   1. Definición de tipos de datos que se utilizaran en el modelo de datos 77
   2. [Definición de constantes y variables en el modelo de datos 82](#_TOC_250006)
2. [Extensión del modelo de datos en sistemas ERP y CRM 83](#_TOC_250005)
   1. [Definición de tablas en el sistema 85](#_TOC_250004)
   2. [Definición de campos en las tablas configuradas, establecer claves primarias e índices de ordenación 86](#_TOC_250003)
   3. [Definición de las relaciones entre las tablas configuradas 89](#_TOC_250002)

[RECUERDA 91](#_TOC_250001)

[Preguntas de Autoevaluación 93](#_TOC_250000)

UNIDAD DIDACTICA 4. DICCIONARIO DE DATOS EN SISTEMAS ERP-CRM 95

1. Objetos del diccionario de datos 97
   1. Funcionalidades del lenguaje de programación establecido 99
   2. Definición de librerías, clases y métodos del lenguaje de programación 100
2. Herramientas para la creación y el mantenimiento del diccionario de datos 105
   1. Definición de herramientas de documentación 118

RECUERDA 119

a

Preguntas de Autoevaluación 121

UNIDAD DIDACTICA 5. DISEÑOS DE PANTALLA 123

1. Definición de pantallas de recogida de datos 125
   1. Herramientas para el diseño de tablas y mantenimientos 125
   2. Configuración de pantallas de entrada de datos 133
2. Herramientas de creación de mantenimientos 134
   1. Configuración de herramientas para crear tablas de datos 135

RECUERDA 139

Preguntas de Autoevaluación 141

UNIDAD DIDACTICA 6. OPERACI ONES DE CONSULTA ••••••••••.•••••••••••••. ...•. 143

1. Herramientas de búsqueda ágiles para el usuario 145
   1. Definición de las funcionalidades de los protocolos de consulta 145
   2. Configuración de pantallas de búsqueda 148
   3. Establecimiento criterios de búsqueda para cadenas o números 150

RECUERDA 153

Preguntas de Autoevaluación 155

UNIDAD DIDACTICA 7. FORMULARIOS E INFORMES EN SISTEMAS ERP-CRM ..157

1. Formularios 159
   1. Caracterfsticas de los formularios e informes 159
   2. Búsqueda de herramientas estándar que faciliten la obtención de ta información 160
   3. Exportación de la información a otros sistemas, por ajemplo, Excel 162
2. Arquitecturas de informes, elementos de informes 165
   1. Especificaciones para el diseño de informes 168
3. Herramientas para ta creación de formularios e informes 169
   1. . Desarrollo de un generador de informes para la obtención de información 174

RECUERDA 177

Preguntas de Autoevaluación 179

UNIDAD DIDACTICA 8. ACCESOS A LA INFORMACI ÓN 181

1. Accesos mediante dispositivos ODBC 183
   1. Definición del protocolo ODBC 183
   2. Utilización de consultas en lenguaje SQL 185
2. Generación de gráficos 189

RECUERDA 195

Preguntas de Autoevaluación 197

ACT 1V1DADES PRACT 1CAS .••••....•..••••••••••••.••••••••. 199

Actividad Práctica RP1 201

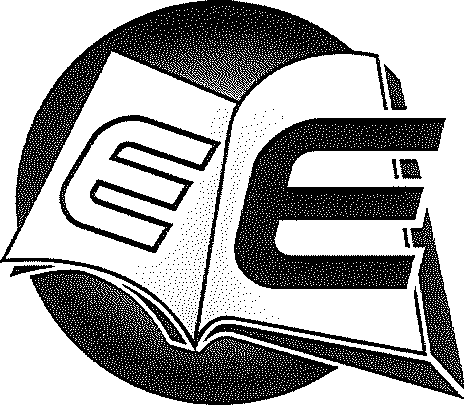
RESPUESTAS A LAS PREGUNTAS DE AUTOEVALUACI ÓN ...................••.••••••.•••.• 203

# •

## Modelos de datos en Sistemas

**UD1**

**ERP-CRM**



UF1888 Operaciones de Mantenimiento y Consulta de **Datos**

•

1. Modelos de datos tipos y características

El diseño de bases de datos se concentra en Ja forma en que la estructura de bases de dalos se usará para guardar y administrar datos del usuario final. El modelado de datos, primer paso para diseñar una base de datos, se refiere al proceso de crear un modelo específico de datos para el dominio de un problema determinado. (Un dominio de problema es un área claramente definida dentro del ambiente real, con ámbito y fronteras bien delimitados, que se ha de manajar sistemáticamente) .

Un **modelo de datos** es una representación relativamente sencilla, por Jo general gráfica, de estructuras de datos reales más complajas. En términos generales, un modelo es una abstracción de un oQjeto o hecho real más complejo. La función principal de un modelo es ayudar a que el lector entienda las complajidades del ambiente real.

Dentro del ambiente de una base de datos, el modelo representa estructuras de datos y sus características, relaciones, restricciones, transformaciones y otras construcciones con el propósito de sostener un dominio de problema específico.

El modelado de datos es un proceso iterativo y progresivo. Se empieza con una comprensión sencilla del dominio del problema y a medida que aumente ésta, también se incrementa el nivel de detalle del modelo de datos. Si se hace en forma apropiada, el modelo de dalos final es en efecto un "plano" que contiene todas las instrucciones para construir una base de datos que va a satisfacer todas las necesidades del usuario final. Este plano es narrativo y gráfico en su naturaleza, lo cual significa que contiene descripciones de texto en lenguaje sencillo, no ambiguo y claro, que describe los elementos principales de los datos.

Un modelo de datos listo para implementación debe contener al menos los siguientes componentes:

* + Una **descripción de la estructura** de datos que guardará los datos del usuario final.
  + Un **conjunto de reglas** que se pueden hacer cumplir para garantizar la integridad de los datos.
  + Una **metodología de manipulación** de datos para apoyar las transformaciones de los datos reales.

Los modelos de datos pueden facilitar la interacción entre el diseñador, el programador de aplicaciones y el usuario final. Un modelo bien diseñado puede incluso promover un majar entendimiento de la organización para la cual se desarrolló el diseño de la base de datos. En pocas palabras, los modelos de datos son una herramienta de comunicación.

**Tipos de modelos de datos**

La búsqueda de una majar administración de datos ha llevado a varios modelos que tratan de resolver los defectos críticos del sistema de archivos. Estos modelos representan escuelas de pensamiento en torno a qué es una base de datos, qué debe hacer, los tipos de estructuras que debe emplear, así como la tecnología que debería usarse para implementar estas estructuras. A continuación vamos a ver los principales modelos de datos.

**Modelos lógicos**

Son orientados a las operaciones más que a la descripción de una realidad.

Usualmente están implementados en algún manajador de Base de Datos.

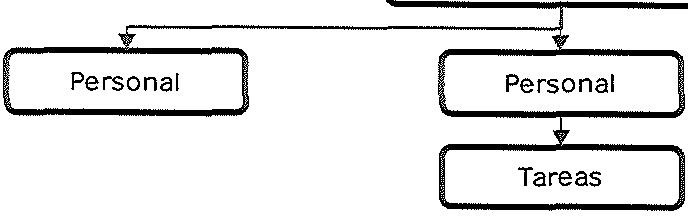
*Modeloj erárquico*

Es utilizado por los primeros gestores de datos. Es también conocido como modelo de árbol ya que se utiliza una estructura en árbol para organizar los datos.

La información es organizada en ordenjerárquico, en el que la relación entre las entidades siempre es del tipo padre/h!jo. De este modo hay una serie de nodos que contienen atributos y que se relacionarán con nodos h)jos de tal forma que puede haber más de un h)jo para el mismo padre, pero solo un padre para un h!jo.

En este modelo los datos se almacenan en estructuras lógicas conocidas como segmentos. Los segmentos se relacionan entre sí utilizando arcos.

Departamento

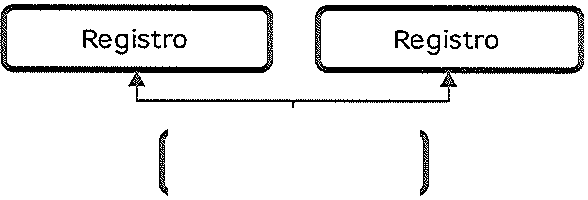


•

*Modelo de red*

Es un modelo ampliamente aceptado. Organiza la información en registros, denominados nodos, y enlaces. En los registros se almacenan los datos, mientras que los enlaces permiten relacionarlos. Las bases de datos en red son parecidas a lasjerárquicas, pero en este caso puede haber más de un padre.

En este modelo se pueden representar perfectamente cualquier tipo de relación entre los datos pero esto dificulta su manejo.



Enlace

*Modelo relacional*

En este modelo los datos se organizan en tablas donde los datos se relacionan. Es el modelo más popular de todos, por ello vamos a verlo con más detalle en el siguiente punto.

*Modelo de bases de datos orientados a of¿jetos*

Tras la aparición de la programación orientada a objetos, surge la necesidad de crear bases de datos adaptadas a este lenguaje. La programación orientada a oqjetos permite la cohesión de datos y procedimientos, haciendo que se diseñen estructuras que poseen datos en las que se definen los procedimientos que pueden realizar con los datos. En las bases de datos orientadas a objetos se utiliza esta misma idea.

A través de esta idea se pretenden que estas bases de datos consigan arreglar las limitaciones de las relacionales.

Se supone por tanto que estas son las bases de datos de tercera generación, pero siguen sin reemplazar a las relacionales, aunque son el tipo de base de datos que más está creciendo en los últimos años.

*Bases de datos of¿jeto-relacionales*

Pretenden ser un híbrido entre el modelo relacional y el orientado a objetos. El problema de la base de datos orientados a objetos es que requieren de una

reinversión de capital y esfuerzo para convertir las bases de datos relacionales en bases de datos orientadas a oQjetos.

En las bases de datos objeto relacional se intenta conseguir una compatibilidad relacional dando la posibilidad de integrar majoras de la orientación a objetos.

Estas bases de datos se basan en el estándar SQL 99. En este estándar se añade a las bases relacionales la posibilidad de almacenar procedimientos de usuarios, tipos definidos por el usuario, recursos para consulta, etc.

**Modelos conceptuales**

Estos modelos representan la información de forma absolutamente independiente al sistema gestor de base de datos. Los esquemas internos de las diferentes bases de datos no captan suficientemente bien la semántica del mundo real, de ahí que primero haya que pasar por uno o dos esquemas previos más cercanos al mundo real.

Algunos de los ajemplos de modelos conceptuales son:

* Modelo entidad-relación.
* Modelo RM/T.
* Modelos semánticos.

El modelo entidad-relación es el modelo conceptual más utilizado para el diseño conceptual de bases de datos.

* 1. **Características del modelo de datos**

Los elementos básicos de todos los modelos de datos son: **entidades, atributos, relaciones y restricciones.**

Una **entidad** es cualquier cosa (una persona, lugar, cosa o hecho) acerca de la cual se han de colectar y guardar datos. Una entidad representa un tipo particular de oQjeto en el mundo real. Como una entidad representa un tipo particular de oQjeto, las entidades son "distinguibles", es decir, cada vez que se presenta una de ellas es única y distinta. Por ajemplo, una entidad CUENTE tendrá muchas ocurrencias distinguibles, por ajemplo, John Smith, Pedro Dinamita, Tom

# •

Strickland. etc. Las entidades pueden ser olajetos físicos, como clientes o productos, pero las entidades también pueden ser abstracciones, como rutas de vuelo o conciertos musicales.

Un **atributo** es una característica de una entidad. Por ajemplo, una entidad CLIENTE seria descrita por atributos tales como apellido paterno, nombre, teléfono, dirección y limite de crédito del cliente. Los atributos son el equivalente de los campos en los sistemas de archivos.

Una **relación** describe una asociación entre entidades. Por ejemplo, existe una relación entre clientes y agentes que se puede describir como sigue, un agente puede atender a numerosos clientes y cada cliente puede ser atendido por un agente. Los modelos de datos usan tres tipos de relaciones uno a muchos, muchos a muchos y uno a uno. Los diseñadores de bases de datos por lo general usan las notaciones breves 1: M o 1 \* M:N o \* ..\* y 1:1 o 1...1, respectivamente. (Aun cuando la notación M:N es una leyenda estándar para la relación de muchos a muchos, la notación M:M laminen puede usarse). Los siguientes ajemplos ilustran las distinciones entre las tres.

* + - **Relación uno a muchos (1:M o 1..**\*). Un pintor crea muchas obras, pero cada una de ellas es pintado por sólo un pintor. Así, el pintor (el "uno") está relacionado con las pinturas (las "muchas") . Por tanto, los diseñadores de bases de datos, marcan la relación (PINTOR pinta PINTURA) como 1:M. (Nótese que los nombres de entidad con frecuencia se ponen en mayúsculos como convención, de modo que se identifiquen con facilidad). Del mismo modo, un cliente (el "uno") puede generar muchas facturas, pero cada factura (el "muchos") es generada por un solo cliente. La relación (CLIENTE genera FACTURA) también seria marcada como 1:M.
    - **Relación muchos a muchos (M:N o** \*..\*) . Un empleado puede aprender muchas habilidades en el trabajo y cada habilidad de trabajo puede ser aprendida por muchos empleados. Los diseñadores de bases de datos marcan la relación (EMPLEADO aprende HABILIDAD) como M:N. Del mismo modo, un estudiante puede tomar muchas clases y cada clase puede ser tomada por muchos estudiantes, dando así la leyenda de relación M: N para la relación expresada por (ESTUDIANTE toma CLASE).
* **Relación uno a uno (1:1 o 1.. 1).** La estructura de administración de una compañía de ventas al por menor puede requerir que cada una de sus tiendas sea manejada por **un** solo empleado. A su vez, el gerente de cada tienda, que es un empleado, manaja sólo uno tiendo Por tonto, lo relación (EMPLEADO manajo TIENDA) se marca como

1:1.

La exposición precedente identificó cada relación en ambas direcciones; esto es, las relaciones son bidireccionales:

* Un CLIENTE puede generar muchas FACTURAS.
* Cada una de las muchas FACTURAS es generado sólo por un CLIENTE.

Una **restricción** se aplica a los datos. Las restricciones son importantes porque ayudan o asegurar lo integridad de datos. Las restricciones se expresan normalmente en forma de reglas. Por ajemplo:

* **El** salario de **un** empleado puede tener valores que están entre 6 000

y 350 000.

* **El** PDC (promedio de calificaciones) de **un** estudiante puede estar entre 0.00 y 4.00.
* Cada clase debe tener un y sólo un maestro.

¿Cómo identificamos correctamente entidades, atributos, relaciones y restricciones? El primer paso es identificar con claridad las reglas de negocios para el dominio de problema que se modele.

* + 1. **Reglas de negocios**

Cuando los diseñadores de bases de datos empiezan a seleccionar o determinar las entidades, atributos y relaciones que se usarán para construir un modelo de datos, pueden empezar por entender cabalmente qué tipos de datos hay en una organización, cómo se usan los datos y que marcos de tiempo se usan Pero, por si mismos, estos datos e información no dan el panorama requerido de todo el negocio. Desde el punto de vista de una base de datos, el conjunto de datos tiene sentido sólo cuando reflaja debidamente reglas de negocios.

# •

Una **regla de negocios** es una descripción breve, precisa y no ambigua de una política, procedimiento o principio dentro de una organización especifica. En cierto sentido, las reglas de negocios tienen nombre erróneo, ya que se aplican a cualquier organización, grande o pequeña: un negocio, una unidad del gobierno, un grupo religioso, un laboratorio de investigación, etc. que guarde y utilice datos para generar información.

Las reglas de negocios, derivadas de una descripción detallada de las operaciones de una organización, ayudan a crear y hacer cumplir acciones dentro del ambiente de esa organización. Las reglas de negocios deben darse por escrito y actualizarse para reflajar cualquier cambio en el ambiente operacional de la organización.

Las reglas de negocios debidamente escritas se usan para definir entidades, atributos, relaciones y restricciones. Cada vez que se vean enunciados de relación, como "un agente puede atender a vanos clientes y cada cliente puede ser atendido por un solo agente", se observan reglas de negocios aplicadas.

Para ser eficientes, las reglas de negocios deben ser fáciles de entender y ampliamente diseminadas, para asegurarse que toda persona dentro de la organización comparta una interpretación común de las reglas. Las reglas de negocios describen, en lenguaje sencillo, las características principales y distintivos de los datos como los ve la compañía. emplos de reglas de negocios son los siguientes:

* + - * Un cliente puede generar muchas facturas.
      * Una factura es generada por un solo cliente.
      * Una sesión de capacitación no puede ser programada para menos de 1O o para más de 30 empleados.

Nótese que esas reglas de negocios establecen entidades, relaciones y restricciones. Por ajemplo, las primeras dos reglas de negocios establecen dos entidades (CLIENTES y FACTURA) y una relación 1:M entre esas dos entidades. La tercera regla de negocios establece una restricción (no menos de 1O y no más de

30 personas), dos entidades (EMPLEADO y CAPACITACIÓN) y una relación entre EMPLEO y CAPACITACIÓN.

1. Definición del modelo de datos

Un **modelo de datos** es un cor}junto de conceptos que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos, a la cual denominamos esquema. Según el nivel de abstracción en el que se encuentre la estructura descrita, el modelo que permite su descripción será un modelo externo (visión que tiene de la base de datos cada usuario en particular), modelo global (enfoque del conjunto de la empresa) o modelo interno (forma en que se organizan los datos en el almacenamiento físico).

Los modelos externos nos permiten representar los datos que necesita cada usuario en particular con las estructuras propias del lenguaje de programación que va a emplear. Los modelos globales ayudan a describir los datos para el conjunto de usuarios, podríamos decir que es la información a nivel de empresa; y, por último, los modelos internos están orientados a la máquina, siendo sus elementos de descripción punteros, índices, agrupamientos, etc.

En el punto anterior vimos que existen distintos modelos de datos según los cuales la información puede ser almacenada y relacionada entre sí. Actualmente, para la mayoría de las aplicaciones de gestión que utilizan bases de datos, el modelo más empleado es el modelo relacional, por su gran versatilidad, potencia y por los formalismos matemáticos sobre los que se basa.

A continuación desarrollaremos en este punto el modelo relacional y los mecanismos que aseguren la integridad referencial.

* 1. **Modelo relacional, definición de mecanismos que aseguren la integridad referencial**

El modelo de datos relacional es posterior a los modelos jerárquicos y de red. Nació como consecuencia de los trabajos publicados en 1969-70 por E. F. Codd, aunque los primeros SGBD relacionales no aparecen hasta principios de los años ochenta.

El modelo relacional se ocupa de tres aspectos de los datos: su **estructura,** su **integridad** y su **manipulación.** En este punto nos ocuparemos de los dos primeros.

# •

La estructura lógica y el modo de realizar las operaciones de E/S en el enfoque relacional, son distintos respecto al enfoquejerárquico y de red. El modelo relacional representa la base de datos por medio de tablas llamadas **relaciones.** Algunas de las diferencias con las bases de datosjerárquicas o en red son:

* + - A la hora de operar con los datos, en lugar de hacerlo sobre registros, actúa sobre la relación (tabla).
    - Las bases de datos relacionales no permiten ni grupos repetitivos ni apuntadores en los niveles externo y conceptual, en el nivel interno el sistema tiene libertad para usar cualquier estructura que desee.
    1. **Estructura del modelo relacional**

De manera informal, una base de datos relacional es aquella cuyos usuarios la perciben como un conjunto de tablas. Por ajemplo:

3333

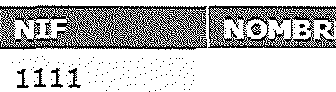
555

777

Manuel Gabriel Marco Carlos

ijel3:i§i$fu al

Málaga



Granada 11

Málaga 22

Cádiz 33

i ª'}t§#;t (•M

Teclado

Impresora Monitor

* + - 1. Tabla de PROVEEDORES B. Tabla de ARTICULOS



l e al il 1:ri Yit0

|  |  |  |
| --- | --- | --- |
| 11 | 1111 | 16,00 |
| 22 | 3333 | 280,00 |
| 33 | 3333 | 199,00 |
| 11 | 7777 | 18,00 |
| 33 | 7777 | 210,00 |

c. Tabla de PRECiOS

El modelo relacional dispone de una terminología propia que pasaremos a describir. Si suponemos que una relación puede representarse mediante una tabla (de hecho es usual y aceptable utilizar el término tabla para referirse a una relación) podríamos decir que:

* Una **tupla** corresponde a una fila de la tabla. Representa cada una de las ocurrencias de la relación (equivale a lo que conocemos como ocurrencia de un registro, en ficheros clásicos) . El número de tuplas se denomina cardinalidad, la cardinalidad varía con el tiempo.
* Un **atributo** corresponde a una columna de la tabla (equivale a un campo de un registro). El número de atributos se llama **grado.** El grado no varía con el tiempo, si añadimos un atributo a una relación, podemos considerar que se trata de otra relación nueva. Una relación de grado uno se llama unaria; de grado dos, binaria; de grado tres, ternaria; .... y una relación de grado n, n-aria.
* Un **dominio** es una colección de valores, de los cuales uno o más atributos obtienen sus valores reales. Por ajemplo, el dominio de código de artículo es el conjunto de todos los códigos de artículos posibles (equivaldría a un conjunto de valores de un tipo dado).
* **Clave candidata** es un atributo K (o conjunto de atributos) de una relación R que cumple las siguientes propiedades independientes del tiempo:

o Unicidad: no existen dos tuplas en R con el mismo valor de K. o Minimalidad: si K es compuesto, no será posible eliminar

ningún componente de K sin destruir la propiedad de unicidad.

Por ajemplo, el atributo compuesto (NIF, LOCALIDAD) no es una clave candidata de la relación PROVEEDORES, ya que podemos eliminar el atributo LOCALIDAD sin destruir la propiedad de unicidad, es decir, siguen sin existir dos tuplas con el mismo valor de NIF.

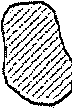
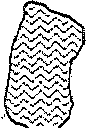
* **Clave primaria.** Es posible que una relación posea más de una clave candidata, en ese caso, se escoge una de ellas como clave primaria y el resto se denominan claves alternativas. En la práctica la elección de la clave primaria suele ser sencilla. Toda relación, sin excepción, tiene una clave primaria y suele representarse subrayando y/o añadiendo el carácter # al atributo (o conjunto de atributos) correspondiente. Por ajemplo: Artículos (código#, concepto) .

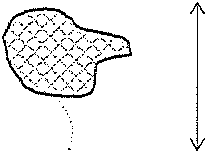
# •

* + **Clave ajena o extranjera** es un atributo (o cot}junto de atributos) de una relación R1 que es clave primaria de otra relación R2. Se utiliza para referenciar a la tupla de R2 cuya clave primaria coincida con el valor de la clave ajena de R1. Ambas claves deben definirse sobre el mismo dominio.

Por ajemplo, el atributo NIF-PROV de la relación PRECIOS es clave extranjera ya que se utiliza para referenciar a una tupla de PROVEEDORES mediante la clave primaria NIF.

R1 y R2 pueden ser la misma relación. Por ejemplo, una relación que represente un árbol genealógico de un pedigrí (Código#, Nombre, FechaNac, ...,CodPadre,CodMadre), donde CodPadre y CodMadre son claves ajenas que referencian tuplas de la misma relación.

  **Dominios**



Clave primaria

**Relación** *<*

1111

3333

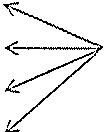
555

777

Manuel Gabriel Marco Carlos

j@e !Af·ii j @!j

Málaga Granada Mélaga Cédiz



**Tuplas**

**Cardinalidad**

**Atributos**

------ **Grados --------**

Ahora podemos dar una definición más formal de una relación: Una relación R sobre un conjunto de dominios 01,02,...,Dn (no necesariamente distintos) se compone de dos partes: una cabecera o esquema, y un cuerpo.

* La **cabecera o esquema** está formado por un conjunto f¡jo de atributos, o major dicho, de pares atributo-dominio {(A1 :01), (A2: 02),...,(An:Dn)}
* El **cuerpo** está formado por un conjunto de tuplas (n-tuplas) que varía con el tiempo. Cada tupla a su vez está formada por un conjunto de pares atributo-valor: { (A1 :vi1), (A2:vi2),...,(An:vin)} (i=1,2,...,m donde m es el número de tuplas del conjunto) . Cada v¡j es un valor del dominio Dj asociado al atributo ·

Además, las relaciones cumplen las siguientes propiedades:

* No existen tuplas repetidas. El cuerpo de la relación se ha definido como un conjunto de tuplas, y en matemáticas los conjuntos por definición no incluyen elementos repetidos. Hay que decir que muchos de los SGBD relacionales sí admiten duplicidad de tuplas.
* Las tuplas no están ordenadas. El cuerpo de la relación se ha definido como un conjunto de tuplas, y los conjuntos en matemáticas no son ordenados.
* Los atributos no están ordenados. La cabecera de la relación se ha definido como un conjunto pares atributo-dominio, y los conjuntos en matemáticas no son ordenados.

Estas propiedades son las que marcan la diferencia entre una tabla y una relación, ya que una tabla presenta las filas y las columnas en un orden, del cual carecen las relaciones. Por otro lado, una tabla podría contener filas repetidas. De todos modos, como d¡jimos anteriormente, es muy común utilizar el término tabla para referirse a una relación.

* + 1. **1 ntegridad de los datos**

En una base de datos relacional, las relaciones están st¿jetas a ciertas reglas de integridad:

* **Integridad de entidad:** "Ningún atributo que participe en una clave principal puede tener valores nulos." Coddjustifica esto del siguiente modo: En el modelo relacional, las relaciones representan entidades del mundo real y todas las entidades del mundo real son distinguibles. Las claves primarias llevan a cabo la función de identificación en el modelo relacional, por tanto un identificador nulo sería una contradicción.
* **Integridad referencial:** "No deben existir valores de clave ajena sin concordancia." Dicho de otro modo, si una relación tiene una clave ajena que referencia a otra relación (o a ella misma), cada valor de la clave ajena tiene que ser igual a un valor de la clave principal de la relación a la que referencia, o bien, ser completamente nulo.

El diseñador de la base de datos deberá poder especificar qué operaciones han de rechazarse y cuáles han de aceptarse, y en este caso, qué operaciones de compensación hay que realizar para mantener la integridad de la base de datos. Para ello el diseñador debe plantearse tres preguntas por cada clave ajena:

* ¿Puede aceptar nulos esa clave ajena? Por ajemplo, (tomando como referencia las relaciones PROVEEDORES, ARTICULOS, PRECIOS)

¿tiene sentido la existencia de un precio cuyo proveedor se desconoce? Evidentemente, no. En algunos casos esta respuesta podría ser distinta, por ajemplo, en una base de datos con las relaciones EMPLEADOS y PROYECTOS, podría existir un empleado no asignado de momento a un proyecto.

* ¿Qué deberá suceder si se intenta eliminar una tupla referenciada por una clave ajena? Por ajemplo, si se intenta eliminar un proveedor del cual existe algún precio de artículo. En general, para estos casos existen por lo menos tres posibilidades:

o **Restringir:** la operación de eliminación se restringe sólo al caso en el que no existe alguna tupla con clave ajena que la referencie, rechazándose en caso contrario. En nuestro ajemplo, un proveedor podrá ser borrado, si y sólo si, no tiene precios de artículos (por ahora, no suministra artículos).

* + **Propagar en cascada:** la operación de borrado se propaga en cascada eliminando también todas las tuplas cuya clave ajena la referencien. En nuestro ajemplo, se eliminaría el proveedor y todas las tuplas con precios de artículos suministrados por él.
  + **Anular:** se asignan nulos en la clave ajena de todas las tuplas que la reverencien y se elimina la tupla referenciada. En nuestro ejemplo, no tiene mucho sentido, pero consistiría en asignar nulos al NIF-PROV de todas las tuplas de precios pertenecientes al proveedor que queremos borrar, y posteriormente borrar al proveedor.
* ¿Qué deberá suceder si hay un intento de modificar la clave primaria de una tupla referenciada por una clave ajena? Por ajemplo, si se intenta modificar la clave de un proveedor del cual existe algún precio de artículo. En general, para estos casos también existen por lo menos tres posibilidades:
  + **Restringir:** La operación de modificación se restringe sólo al caso en el que no existe alguna tupla con clave ajena que la referencie, rechazándose en caso contrario. En nuestro ajemplo, la clave de un proveedor podrá ser modificada, si y sólo si, no tiene precios de artículos (por ahora, no suministra artículos).
  + **Propagar en cascada:** La operación de modificación se propaga en cascada modificando también la clave ajena de todas las tuplas que la referencien. En nuestro ajemplo, se modificaría el NIF del proveedor y la clave ajena NIFPROV de todos los precios de artículos suministrados por él.
  + **Anular:** Se asignan nulos en la clave ajena de todas las tuplas que la referencien y se modifica la clave de la tupla referenciada. En nuestro ajemplo, tampoco tiene mucho sentido, pero consistiría en asignar nulos al NIF-PROV de todas las tuplas de precios pertenecientes al proveedor que queremos modificar, y posteriormente modificar la clave del proveedor .

# •

Desde luego, las opciones restringir, propagar en cascada y anular, sólo representan un conjunto de casos que se presentan con bastante frecuencia en la práctica. Pero podrían adoptarse muchas otras soluciones. Por ajemplo, para un intento de borrado de un proveedor:

* Podría efectuarse un diálogo con el usuario.
* Podría grabarse información en otro archivo de antiguos proveedores.
* Podría transferirse a otro proveedor los artículos del proveedor en cuestión.
* Etc.
  1. **Definición de tablas**

Tabla en las bases de datos, se refiere al tipo de modelado de datos, donde se guardan los datos recogidos por un programa. Su estructura general se asemaja a la vista general de un programa de hqja de cálculo.

Una tabla es utilizada para organizar y presentar información. Las tablas se componen de **filas y columnas** de celdas que se pueden rellenar con textos y gráficos. Las tablas se componen de dos estructuras:

* + - **Registro:** es cada una de las filas en que se divide la tabla. Cada registro contiene datos de los mismos tipos que los demás registros. E;jemplo: en una tabla de nombres y direcciones, cada fila contendrá un nombre y una dirección.
    - **Campo:** es cada una de las columnas que forman la tabla. Contienen datos de tipo diferente a los de otros campos. En el ejemplo anterior, un campo contendrá un tipo de datos único, como una dirección, o un número de teléfono, un nombre, etc.

A los campos se les puede asignar, además, propiedades especiales que afectan a los registros insertados. El campo puede ser definido como índice o autoincrementable, lo cual permite que los datos de ese campo cambien solos o sean el principal indicar a la hora de ordenar los datos contenidos.

* 1. **Conexión entre tablas**

Una **relación** se establece haciendo coincidir los datos de las columnas de clave, normalmente las columnas con el mismo nombre en ambas tablas. En la mayor parte de los casos, la relación hace coincidir la clave principal de una tabla, que proporciona un identificador único para cada fila, con una entrada de la clave externa de Ja otra tabla. Por ajemplo, las ventas pueden asociarse a los títulos concretos vendidos creando una relación entre las columnas de identificador de título de la tabla Títulos (la clave principal) y de la tabla Ventas (la clave externa).

* + 1. **Tipos de relaciones**

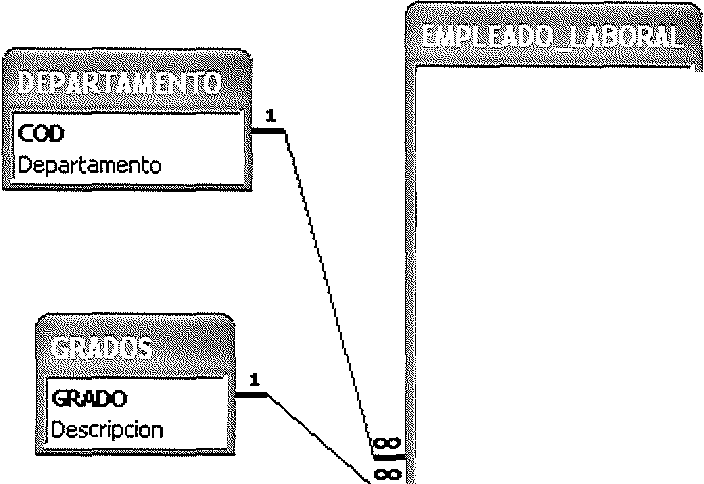
La manera en que se relacionan las tablas entre sí da lugar a comportamientos diferentes. Podemos definir tres tipos de relaciones:

* + - * Uno a muchos.
      * Muchos a muchos.
      * Uno a uno.

De todas ellas la más utilizada y recomendable en la mayoría de los casos será el modelo Uno a muchos. Nosotros vamos a ver como funciona los tres tipos de relaciones.

**Relación uno a muchos**

Por cada registro de la tabla principal (tabla de la clave principal o lado uno de Ja relación) pueden existir muchos (infinitos) registros en la tabla relacionada (tabla de la clave externa o lado infinito de la relación). La tabla relacionada no puede contener un registro que no esté relacionado con uno de la tabla principal, pero pueden haber muchos registros que estén relacionados con el mismo registro de la tabla principal: varios (infinitos) empleados de la tabla EMPLEADO\_LABORAL, pueden estar en el mismo departamento de la tabla DEPARTAMENTO ...



**N\_EM'l.E**

Cod\_Emp Especiali **Comentario remio** Fecha\_alta Titulo **Jorn\_Comp** Cat\_Lab **Porcentaje Salario Expcia** DEPART

GRADO

**Relación muchos a muchos**

Debes tener claro las llaves primarias y una tabla de unión para que se produzca la relación varios a varios entre PRODUCTOS y NUM\_VENTA. Un registro de la tabla NUM\_VENTA puede estar relacionado con varios registros de la tabla PRODUCTOS y un registro de la tabla PRODUCTOS puede estar relacionado con varios registros de la tabla NUM\_VENTA. Explicación: cuando realizamos una venta le asignamos un número (N\_VENTA) y lo almacenamos en la tabla NUM\_VENTA;

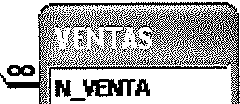
esa venta puede estar formada por uno o varios productos identificados con un numero de producto (N\_PRODUCTO) de la tabla PRODUCTOS.

Pero, ¿cómo podemos relacionar varios registros de la tabla NUM\_VENTA con varios de PRODUCTOS, y viceversa? En realidad esta relación está formada por dos relaciones de uno a muchos. Una tabla intermedia (tabla de unión VENTAS) contiene la clave principal múltiple que se forma con la conbinación de dos (o más) claves externas: N\_VENTA y N\_PRODUCTO. La combinación de estos dos campos forma un campo que no se repite. Por ajemplo: la venta 200 (N\_VENTA vale 200) se realizó con los productos 12, 14 y 36 (N\_PRODUCTO) . Si formamos con N\_VENTA y N\_PRODUCTO una clave principal obtenemos 20012, 20014 y 20036, valores no repetidos. Con otros valores de N\_VENTA se procedería igual.



N\_EMPLE

Fecha N\_CLIENTE



00

**N\_PROOUCTO**

Cantidad



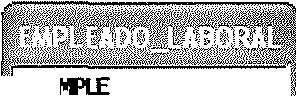
**Descripcion Precio\_Unidad** N\_PROVEEDOR

Stock foto Velocidad

Categoria

**Relación uno a uno**

Por cada registro de la tabla principal (tabla que contiene la clave principal) puede existir un sólo registro en la tabla relacionada (tabla que contiene la clave externa). La tabla relacionada no puede contener un registro que no esté relacionado con uno de la tabla principal: no puede existir un registro con FECHA\_ALTA, SALARIO, etc., si no hay un empleado con el que se relacione. Esta relación se utiliza para simplificar y organizar las tablas con muchos campos. Ver el ajemplo siguiente:



1

1

Apellido

**Nombre**

Inicial **Direccionl Direccion2** Ciudad **Provincia** Cod\_Postal Telefono

**Exento**

Cod\_Emp

Especiali **Comentario Premio** Fecha\_alta Titulo Jorn\_Comp Cat\_Lab **Porcentaje Salario Expcia** DEPART

GRADO

###### RECUERDA

* El diseño de bases de datos se concentra en Ja forma en que Ja estructura de bases de dalos se usará para guardar y administrar datos del usuario final. El modelado de datos, primer paso para diseñar una base de datos, se refiere al proceso de crear un modelo específico de datos para el dominio de un problema determinado.
* Los elementos básicos de todos Jos modelos de datos son: **entidades, atributos, relaciones y restricciones.**
* Un **modelo de datos** es un col')junto de conceptos que permiten describir, a distintos niveles de abstracción, la estructura de una base de datos, a la cual denominamos esquema. Según el nivel de abstracción en el que se encuentre la estructura descrita, el modelo que permite su descripción será un modelo externo (visión que tiene de la base de datos cada usuario en particular), modelo global (enfoque del conjunto de Ja empresa) o modelo interno (forma en que se organizan los datos en el almacenamiento físico).
* El **modelo de datos relacional** es posterior a los modelosjerárquicos y de red. Nació como consecuencia de los trabajos publicados en 1969-70 por E. F. Codd, aunque los primeros SGBD relacionales no aparecen hasta principios de los años ochenta.
* En una base de datos relacional, las relaciones están sujetas a ciertas reglas de integridad.
* **Tabla** en las bases de datos, se refiere al tipo de modelado de datos, donde se guardan los datos recogidos por un programa. Su estructura general se asemaja a la vista general de un programa de hoja de cálculo.
* Una **relación** se establece haciendo coincidir los datos de las columnas de clave, normalmente las columnas con el mismo nombre en ambas tablas.
* La manera en que se relacionan las tablas entre sí da lugar a comportamientos diferentes. Podemos definir tres tipos de relaciones: uno a muchos, muchos a muchos y uno a uno.

# •

Preguntas de Autoevaluación

1. **¿Cuál de las siguientes afirmaciones es incorrecta?**
2. El modelado de datos, primer paso para diseñar una base de datos, se refiere al proceso de crear un modelo especifico de datos para el dominio de un problema determinado.
3. Un modelo de datos es una representación relativamente sencilla, por lo general gráfica, de estructuras de datos reales más complejas.

e) El modelado de datos es un no se considera un proceso iterativo y progresivo.

1. **¿Qué tipo de modelo es conocido también como modelo de árbol ya que se utiliza una estructura en árbol para organizar los datos?**
2. Modelo de red.
3. Modelojerárquico.
4. Modelo relacional.
5. **Los elementos básicos de todos los modelos de datos son:**
6. Entidades, atributos, relaciones y restricciones.
7. Tablas, formularios, consultas, e informes.

e) Oetos, atributos, relaciones y formularios.

1. **Una tupla:**
2. Corresponde a una fila de la tabla.
3. Corresponde a una columna de la tabla.
4. Es una colección de valores.
5. **Completa la siguiente afirmación con la opción correcta:**

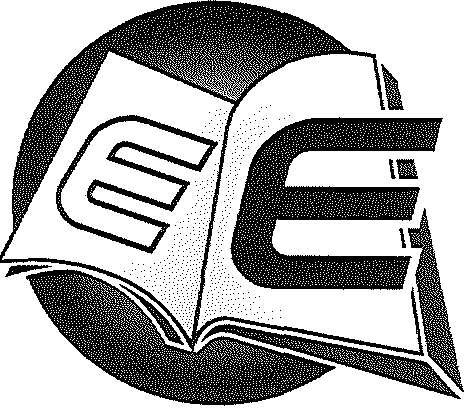
---------- en las bases de datos, se refiere al tipo de modelado de datos, donde se guardan los datos recogidos por un programa. Su estructura general se asemaja a la vista general de un programa de hqja de cálculo".

1. Atributo.
2. Entidad.
3. Tabla.

•

## Diseño de datos

UD2



UF1888 Operaciones. de Mantenimiento y Consulta de; Datas

# •

Los modelos de bases de datos tradicionales (relacional, red y jerárquico) han sido capaces de satisfacer con éxito las necesidades, en cuanto a bases de datos, de las aplicaciones de gestión tradicionales. Sin embargo, presentan algunas deficiencias cuando se trata de aplicaciones más complajas o sofisticadas como, por ajemplo, el diseño y fabricación en ingeniería (CAD/CAM, CIM), los experimentos científicos, los sistemas de información geográfica o los sistemas multimedia. Los requerimientos y las características de estas nuevas aplicaciones difieren en gran medida de las típicas aplicaciones de gestión: la estructura de los ol{jetos es más complaja, las transacciones son de larga duración, se necesitan nuevos tipos de datos para almacenar imágenes y textos, y hace falta definir operaciones no estándar, especificas para cada aplicación.

Las bases de datos orientadas a ol{jetos se crearon para tratar de satisfacer las necesidades de estas nuevas aplicaciones. La orientación a objetos ofrece flexibilidad para manejar algunos de estos requisitos y no está limitada por los tipos de datos y los lenguajes de consulta de los sistemas de bases de datos tradicionales. Una característica clave de las bases de datos orientadas a ol{jetos es la potencia que proporcionan al diseñador al permitirle especificar tanto la estructura de objetos complajos, como las operaciones que se pueden aplicar sobre dichos objetos.

Otro motivo para la creación de las bases de datos orientadas a ol{jetos es el creciente uso de los lenguajes orientados a ol{jetos para desarrollar aplicaciones. Las bases de datos se han convertido en piezas fundamentales de muchos sistemas de información y las bases de datos tradicionales son difíciles de utilizar cuando las aplicaciones que acceden a ellas están escritas en un lenguaje de programación orientado a ol{jetos como C+ +, Smalltalk o Java. Las bases de datos orientadas a objetos se han diseñado para que se puedan integrar directamente con aplicaciones desarrolladas con lenguajes orientados a objetos, habiendo adoptado muchos de los conceptos de estos lenguajes.

Los fabricantes de los SGBD relacionales también se han dado cuenta de las nuevas necesidades en el modelado de datos, por lo que las nuevas versiones de sus sistemas incorporan muchos de los rasgos propuestos para las bases de datos orientadas a ol{jetos, como ha ocurrido con lnformix y Oracle. Esto ha dado lugar al modelo relacional extendido y a los sistemas que lo implementan se les denomina sistemas objeto-relacionales. La nueva versión de SQL, SQL: 19991, incluye algunas de las características de la orientación a objetos.

Durante los últimos años se han creado muchos prototipos experimentales de sistemas de bases de datos orientadas a objetos y también muchos sistemas comerciales. Conforme éstos fueron apareciendo, surgió la necesidad de establecer un modelo estándar y un lenguaje. Para ello, los fabricantes de los SGBD orientadas a objetos formaron un grupo denominado ODMG *( Oqject Database Management Group),* que propuso el estándar ODMG-93 y que ha ido evolucionando hasta el ODMG 3.0, su última versión. El uso de estándares proporciona portabilidad, permitiendo que una aplicación se pueda ejecutar sobre sistemas distintos con mínimas modificaciones. Los estándares también proporcionan interoperabilidad, permitiendo que una aplicación pueda acceder a varios sistemas diferentes. Y una tercera ventaja de los estándares es que permiten que los usuarios puedan comparar entre distintos sistemas comerciales, dependiendo de qué partes del estándar proporcionan.

1. **Definición de objetos y estructuras de datos, características**

El desarrollo del paradigma orientado a objetos aporta un gran cambio en el modo en que vemos los datos y los procedimientos que actúan sobre ellos. Tradicionalmente, los datos y los procedimientos se han almacenado separadamente: los datos y sus relaciones en la base de datos y los procedimientos en los programas de aplicación. La orientación a objetos, sin embargo, combina los procedimientos de una entidad con sus datos.

Esta combinación se considera como un paso adelante en la gestión de datos. Las entidades son unidades autocontenidas que se pueden reutilizar con relativa facilidad. En lugar de ligar el comportamiento de una entidad a un programa de aplicación, el comportamiento es parte de la entidad en sí, por lo en cualquier lugar en el que se utilice la entidad, se comporta de un modo predecible y conocido.

El modelo orientado a oqjetos también soporta relaciones de muchos a muchos, siendo el primer modelo que lo permite. Aun así se debe ser muy cuidadoso cuando se diseñan estas relaciones para evitar pérdidas de información.

Por otra parte, las bases de datos orientadas a oqjetos son navegacionales: el acceso a los datos es a través de las relaciones, que se almacenan con los

# •

mismos datos. Esto se considera un paso atrás. Las bases de datos orientadas a oQjetos no son apropiadas para realizar consultas *ad hoc,* al contrario que las bases de datos relacionales, aunque normalmente las soportan. La naturaleza navegacional de las bases de datos orientadas a oQjetos implica que las consultas deben seguir relaciones predefinidas y que no pueden insertarse nuevas relaciones **"al vuelo".**

No parece que las bases de datos orientadas a oQjetos vayan a reemplazar a las bases de datos relacionales en todas las aplicaciones del mismo modo en que éstas reemplazaron a sus predecesoras.

Los oQjetos han entrado en el mundo de las bases de datos de formas:

* + SGBD orientados a oQjetos puros: son SGBD basados completamente en el modelo orientado a oQjetos.
  + SGBD híbridos u objeto-relacionales: son SGBD relacionales que permiten almacenar oQjetos en sus relaciones (tablas) .

Son varios los conceptos del paradigma orientado a oQjetos en programación, no obstante nosotros sólo vamos a centrarnos en definir que es un oQjeto, ya que los términos de clase y métodos serán vistos posteriormente en otra unidad didáctica.

**¿Qué es un objeto?**

Es un elemento autocontenido utilizado por el programa.

Los valores que almacena un objeto se denominan **atributos, variables** o **propiedades.** Los oQjetos pueden realizar acciones, que se denominan métodos, servicios, funciones, procedimientos u operaciones.

Los objetos tienen un gran sentido de la privacidad, por lo que sólo dan información sobre sí mismos a través de los métodos que poseen para compartir su infamación. También ocultan la implementación de sus procedimientos, aunque es muy sencillo pedirles que los ajecuten. Los usuarios y los programas de aplicación no pueden ver qué hay dentro de los métodos, sólo pueden ver los resultados de ajecutarlos. A esto es a lo que se denomina ocultación de información o encapsulamiento de datos. Cada oQjeto presenta una interface pública al resto de objetos que pueden utilizarlo. Una de las mayores ventajas del encapsulamiento es

que mientras que la interface pública sea la misma, se puede cambiar la implementación de los métodos sin que sea necesario informar al resto de objetos que los utilizan. Para pedir datos a un oQjeto o que éste realice una acción se le debe enviar un mensaje. Un programa orientado a oQjetos es un conjunto de oQjetos que tienen atributos y métodos. Los objetos interactúan envidándose mensajes. La clave, por supuesto, es averiguar qué oQjetos necesita el programa y cuáles deben ser sus atributos y sus métodos.

Por ajemplo, **Microsoft Access** consta de los siguientes tipos de oQjetos principales:

* **Tablas.** Conjunto de datos estructurados que forman el contenido de una base de datos.
* **Consultas.** Permiten catalogar, ordenar, extraer y modificar la información contenida en una tabla o en un conjunto de tablas relacionadas, dependiendo de los tipos de consulta y de las condiciones que en ella se establezcan.
* **Formularios.** Son diseños que realizamos para poder introducir, modificar o visualizar los registros de una tabla o consulta. En los formularios podemos insertar campos calculados e imágenes.
* **1 nformes.** Son extractos de datos procedentes de tablas o consultas y que tienen como propósito principal el ser impresos en papel, aunque también pueden simplemente visualizarse en pantalla.

Todos estos conceptos los iremos viendo con más detalle a lo largo de toda la unidad formativa.

**Estructura de datos**

Una estructura de datos, o un tipo de datos estructurado, es un tipo de dato construido a partir de otros. Un dato de tipo estructurado está compuesto por una serie de datos de tipos elementales y alguna relación existente entre ellos. Normalmente, la relación suele ser de orden aunque puede ser de cualquier otro tipo.

Se dice que una estructura de datos que es homogénea cuando todos los datos elementales que la forman son del mismo tipo. En caso contrario, se dice que la estructura es heterogénea. Por ajemplo, el tipo de datos complajo es una

# •

estructura homogénea, tanto la parte real como la imaginaria se representan con datos reales.

Siempre que se utilice un dato en un programa debe estar determinado su tipo, para que el traductor sepa como debe tratarlo y almacenarlo. En el caso de datos de tipos elementales, el tipo de dato determina el espacio que se utiliza en memoria. Esto, puede no ocurrir si el dato es de un tipo estructurado.

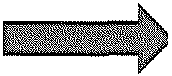
Una estructura de datos que siempre ocupa el mismo espacio en memoria, se dice que es **estática.** Por el contrario, si la memoria asignada a una determinada estructura de datos va variando durante la ajecución del programa, es decir, se realiza una asignación dinámica de memoria, se dice que es una estructura de datos **dinámica.**

**Estructura de datos estáticas**

Dentro de los tipos de estructuras de datos que ocupan siempre el mismo espacio en memoria tenemos los punteros, las cadenas y los arrays.

*Punteros*

Un puntero es un dato que indica la posición de otro dato. Su utilidad se pone de manifiesto en la construcción de estructuras de datos, ya que son ellos los que proporcionan los lazos de unión entre los elementos que constituyen las estructuras. Son importantes los punteros al principio y al final de la estructura.



**PUNTERO DATO**

Si, ocasionalmente, se necesita que un puntero no señale a ningún dato, se dice que el puntero tiene un valor nulo (puntero nulo).

*Cadenas*

Una cadena es una secuencia de caracteres que se interpretan como un dato único. Las cadenas pueden tener longitud f¡ja o variable. La longitud de la cadena se indica tanto por el número de caracteres que contiene ésta, indicado al principio de la misma, como por un carácter especial denominado fin-de-cadena. Sobre datos de tipo cadena se pueden realizar las siguientes operaciones:

* **Concatenación:** consiste en formar una cadena a partir de dos ya existentes, juntando los caracteres de ambas.
* **Extracción de subcadena:** permite formar una cadena a partir de otra ya existente. La subcadena se forma tomando un tramo consecutivo de la cadena inicial.
* **Comparación de cadenas:** es posible comparar dos cadenas. Se considera menor aquella en que el primer carácter en que difieren ambas es menor.
* **Obtención de la longitud:** la longitud de una cadena es **un** dato de tipo entero, cuyo valor es el número de caracteres que contiene ésta.

*Arrays*

El *array* (también llamado formación o matriz) , es la estructura de datos más usual. Existe en todos los lenguajes de programación y en algunos es de las pocas estructuras de datos existentes (BASIC y FORTRAN) .

Un array es una estructura de datos formada por una cantidad f)ja de datos del mismo tipo, cada uno de los cuales tiene asociado uno, o más índices, que determinan de forma unívoca la posición del dato en el array.

Podemos imaginar un array como una estructura de celdas donde se pueden almacenar valores. En la figura podemos ver una matriz de **un** sólo índice que toma valores de 1 a 7.

1 ELEMENT0 4

1 1

""\*\*\*

**\*\*\*llC**

\*\*\*\*

1 2 3 4 s 6 7

En el array de la figura siguiente utilizamos dos índices con valores entre 1 y 3, el primero, y entre 1 y 5, el segundo. Cada elemento de esta matriz está representado por un par ordenado de números, el valor de los dos índices.

# •

ELEMENTO 1,4

1

|  |  |  |
| --- | --- | --- |
|  |  | 1 \*\*\*\*\* |
|  |  | '  1 |
|  |  | 1 |

2

3

1 2 3 4 5

En general, al número de índices del array se le denomina número de dimensiones del array. La dimensión de la formación está dada por los valores máximos de los índices y, el número total de elementos es el producto de estos valores máximos. En los dos ajemplos anteriores, el número de dimensiones de los arrays son 1 y 2, las dimensiones son (7) y (3; 5) y los números totales de elementos son 7 y 15, respectivamente.

La principal operación que se puede realizar con los arrays es la selección, que consiste en especificar un elemento determinado del array. Esta operación se efectúa dando un valor para todos y cada uno de los índices del array. Con el elemento seleccionado se pueden realizar las operaciones propias de su tipo. Así, con cada elemento de un array real, una vez seleccionado, se pueden realizar las operaciones definidas para datos de tipo real (operaciones aritméticas).

**Estructura Dinámicas de datos**

Como ya se ha dicho, este tipo de estructuras ocupan un espacio en memoria que va evolucionando según el tamaño que dicha estructura vaya adquiriendo. Las estructuras de dinámicas que vamos a estudiar son: colas, pilas, listas encadenadas y árboles.

*Colas (FJFO)*

Una cola es una estructura de datos en la que el primer dato en entrar es el primer dato en salir. Es decir, es una estructura FIFO *( First In First Out).* Todo el mundo conocemos como funciona una cola, los nuevos se ponen al final, los servicios se prestan al principio y no está permitido "colarse". Las mismas reglas se aplican a las colas de datos almacenadas en la memoria de un computador.

Puntero al primer

|  |
| --- |
|  |
| **Primer elemento** |
|  |
| **Últtmo elemento** |
|  |

**elemento**

Puntero al último

**elemento**

Hay varias formas de implementar una cola en la memoria de **un** computador. Una forma simple consiste en almacenar los datos en posiciones de memoria adyacentes y utilizar punteros para el principio y el fin de la cola. Cuando un elemento se añade a la cola, el puntero de la parte posterior se ajusta para que señale al nuevo elemento. De manera similar, cuando un elemento se elimina de la cola, se ajusta el puntero delantero para que señale al nuevo primer elemento.

**El** problema de este método para implementar las colas es que las posiciones de memoria que ocupan, varían a medida que se añaden y eliminan elementos de la misma. La solución habitual consiste en asignar un área fDa para almacenar la cola y permitir que se mueva en este área de manera circular. Un área de almacenamiento de esta forma se denomina buffer circular, y puede apreciarse en la figura siguiente.

Puntero al primer elemento

|  |  |
| --- | --- |
| \*\*\*\*\*\*\*\*\*\*\*\*  Primer elemento  Último elemento |  |
|  |
| \*\*\*\*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\*\* |

<

Puntero al último elemento

Área de memoria asignada a una cola

# •

Entre las aplicaciones que tienen las colas se encuentran el almacenamiento de datos en camino, entre un procesador y un periférico, o actuar como punto intermedio en las redes de comunicación de datos.

*Pilas (UFO)*

Una pila es una colección ordenada de datos a los que sólo se puede acceder por un extremo, denominado tope o cima de la pila. La pila es una estructura en la que el último elemento en entrar será el primero en salir, es decir, es lo que se denomina estructura U FO *( Last In First Out).*

Podemos comparar esta estructura con una pila de platos colocada sobre un muelle. Cuando se añade un nuevo plato en lo alto de la pila, los demás bajan, cuando se retira un plato de la pila, los demás suben.

Igual que en el caso de las colas, van a existir dos punteros, uno que indica la posición tope de la pila, denominado puntero de pila, y otro que señala su base, denominado base de pila, y que mantiene el mismo valor mientras existe la pila. Cuando la pila esta vacía el puntero de pila tiene el mismo valor que la base de pila .

La pila es una de las estructuras más importantes en computación. Se usa en cálculos, para pasar de un lenguaje de computador a otro y, para transferir el control de una parte del programa a otra.

Las operaciones que se pueden realizar tanto con las colas como con las pilas, son las siguientes:

* + **Añadir o eliminar un elemento:** si es una cola podremos añadirlo o eliminarlo al final de la misma, y si es una pila al principio.
  + **Acceder al primer elemento:** normalmente es el único al que se va a poder acceder directamente.
  + **Acceder al elemento siguiente del último procesado:** este es el mecanismo normal de acceso tanto a colas como a pilas.
  + **Saber si está vacía:** están vacías si no contienen ningún elemento.

*Listas encadenadas*

Una lista es un conjunto ordenado de datos. Los elementos de la lista pueden insertarse o eliminarse en cualquier punto de la misma, por lo que es menos restrictiva que una pila o una cola.

La forma más sencilla de implementar una lista es hacer uso de un puntero que señale desde un dato al siguiente. También hay un puntero que señala al primer elemento de la lista, mientras que para el último se emplea un puntero nulo.

Una estructura de este tipo se denomina lista encadenada. Cada elemento de la lista consiste en una parte de datos y un puntero.

Una variación sobre la idea de una lista es el caso en el que el puntero del final de la lista señale al primer elemento. Esto crea lo que se denomina una lista circular.

Si los elementos de la lista están en orden alfabético o numérico, dichas listas se conocen como listas ordenadas.

*Arboles*

Un árbol es una estructura que implica una jerarquía, en la que cada elemento está unido a otros bajo él. Cada dato en un árbol es un nodo de dicho árbol. El nodo más alto se denomina **raíz.** Cada nodo puede estar conectado a uno o más subárboles, que también responden a la estructura de un árbol. Un nodo, en la parte inferior, del que no cuelgue ningún subárbol se denomina nodo terminal u hoja.

Un tipo especial de árboles muy usados en computación son los árboles binarios. En ellos, de cada nodo pueden colgar, a lo más, dos subárboles, denominados subárbol derecho y subárbol izquierdo, que también son árboles binarios.

La forma usual de representar los árboles supone el uso de punteros. En un árbol binario cada nodo está constituido por una parte de datos y dos punteros. Uno, o ambos punteros, pueden tener un valor nulo si del nodo no cuelgan subárboles.

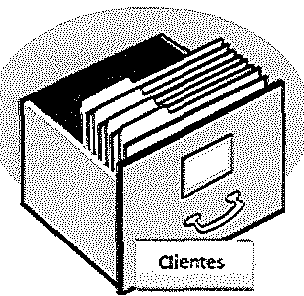
Son muy utilizados en informática. Las partes de muchos programas se enlazan como si se tratara de árboles. Los árboles se utilizan para representar operaciones aritméticas, y en búsquedas y ordenaciones .

# •

* 1. **Definición de los elementos que componen el diseño de la base de datos**

Como hemos visto con anterioridad, toda base de datos se compone de una o más **tablas.** Una tabla es una colección de datos organizados en filas y columnas. Los datos se organizan en columnas, también denominadas campos, que almacenen un tipo específico de información o tipo de datos. En la parte superior de cada columna, la primera celda se utiliza para etiquetar dicha columna.

**TABLA**



*!* **- r.mo**

*z* ***Jon;:-e***

*3* ***{s-;;be1***

*·* . ...·

1. ***vu, s***

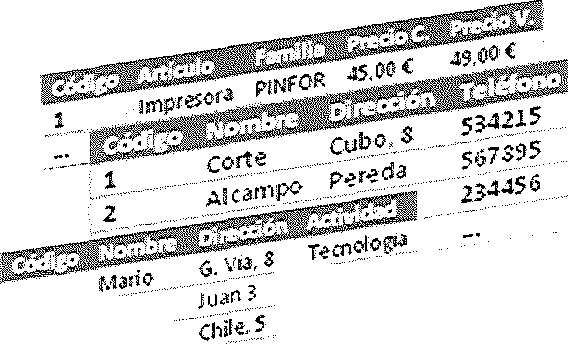
***ÍUilrt 3***

***i:iii!. --s***

***!\_ \_<;nologi\_.a***

*.! -"J; tj/* -- -

***Txtü***



**BASE DE DATOS**

1

2

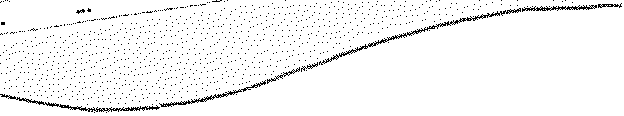
3

**Jorg:¿-** \_

¡$; -

**T-:t,.**

**Textil**

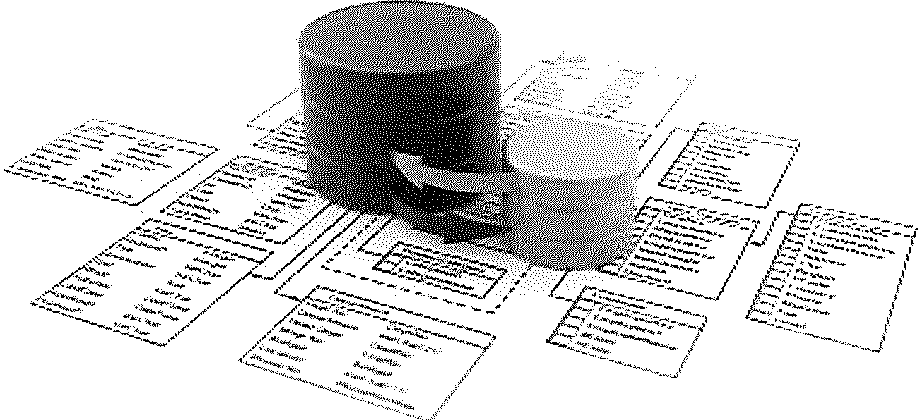


El proceso de diseño de una base de datos se guía por algunos principios. El primero de ellos es que se debe evitar la información duplicada o, lo que es lo mismo, los datos redundantes, porque malgastan el espacio y aumentan la probabilidad de que se produzcan errores e incoherencias. El segundo principio es que es importante que la información sea correcta y completa. Si la base de datos contiene información incorrecta, los informes que recogen información de la base de datos contendrán también información incorrecta y, por tanto, las decisiones que tome a partir de esos informes estarán mal fundamentadas.

Un buen diseño de base de datos es, por tanto, aquél que:

* + Divide la información en tablas basadas en temas para reducir los datos redundantes.
* Proporciona la información necesaria para reunir la información de las tablas cuando así se precise.
* Ayuda a garantizar la exactitud e integridad de la información.
* Satisface las necesidades de procesamiento de los datos y de generación de informes.
  1. **Definición de relaciones entre los elementos**

Cuando hablamos de las relaciones entre los elementos que componen el diseño de la base de datos, estamos haciendo referencia a las conexiones o relaciones que hay entre las tablas.



Como vimos anteriormente, una **relación** se establece haciendo coincidir los datos de las columnas de clave, normalmente las columnas con el mismo nombre en ambas tablas. En la mayor parte de los casos, la relación hace coincidir la clave principal de una tabla, que proporciona un identificador único para cada fila, con una entrada de la clave externa de la otra tabla. Por ajemplo, las ventas pueden asociarse a los títulos concretos vendidos creando una relación entre las columnas de identificador de título de la tabla Títulos (la clave principal) y de la tabla Ventas (la clave externa).

Recordemos que hay tres tipos de relaciones:

# •

* + - **Uno a muchos.** Cuando un registro de una tabla (tabla secundaria) sólo puede estar relacionado con un único registro de la otra tabla (tabla principal) y un registro de la otra tabla (tabla principal) puede tener más de un registro relacionado en la primera tabla (tabla secundaria).
    - **Muchos a muchos.** Cuando un registro de una tabla puede estar relacionado con más de un registro de la otra tabla y viceversa.
    - **Uno a uno.** Cuando un registro de una tabla sólo puede estar relacionado con un único registro de la otra tabla y viceversa.

En el tema anterior se habló de este tipo de relaciones y además se expuso un ajemplo de cada una de ellas para facilitar la comprensión de las mismas.

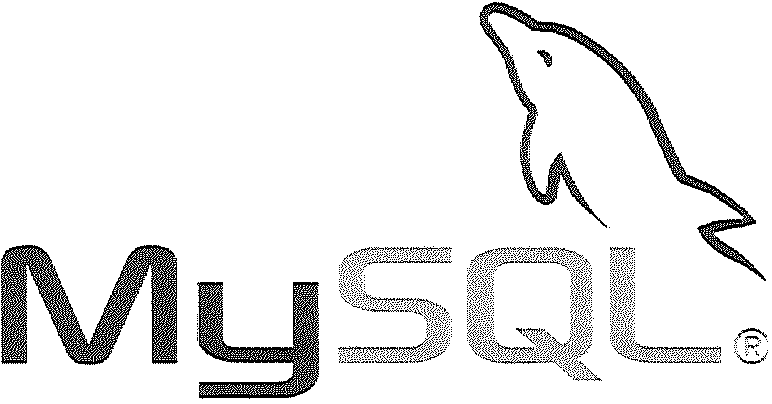
1. **Creación, modificación y borrado de objetos**

**y estructuras de datos**

En este punto vamos a desarrollar las opciones que nos proporciona **MySQL** para el mantenimiento de nuestra base de datos, así como la realización de operaciones entre tablas que garanticen la creación, modificación y borrado de registros entre las tablas definidas en el modelo.

MySQL es el servidor de bases de datos relacionales más popular, desarrollado y proporcionado por MySQL AB. MySQL AB es una empresa cuyo negocio consiste en proporcionar servicios en torno al servidor de bases de datos MySQL.

MySQL es un sistema de administración de bases de datos. Una base de datos es una colección estructurada de datos. La información que puede almacenar una base de datos puede ser tan simple como la de una agenda, un contador, o un libro de visitas, ó tan vasta como la de una tienda en línea, un sistema de noticias, un portal, o la información generada en una red corporativa. Para agregar, acceder, y procesar los datos almacenados en una base de datos, se necesita un sistema de administración de bases de datos, tal como MySQL.



MySQL es un sistema de administración de bases de datos relacionales Una base de datos relacional almacena los datos en tablas separadas en lugar de poner todos los datos en un solo lugar. Esto agrega velocidad y flexibilidad. Las tablas son enlazadas al definir relaciones que hacen posible combinar datos de varias tablas cuando se necesitan consultar datos. La parte SQL de "MySQL" significa "Lenguaje Estructurado de Consulta", y es el lenguaje más usado y estandardizado para acceder a bases de datos relacionales.

* 1. **Mantenimiento de la base de datos**

Hoy en día las bases de datos son una necesidad indispensable en la gestión de las empresas, por tal motivo se debe garantizar su disponibilidad, consistencia de datos y privacidad en todo momento.

Una base de datos sin mantenimiento puede desarrollar problemas en una o más áreas, lo cual a largo plazo puede provocar un mal rendimiento de la aplicación o incluso tiempos de inactividad y pérdida de datos. La mayoría de los responsables de TI comprenden esta necesidad pero no siempre disponen de tiempo o de personal cualificado para realizar estas tareas de mantenimiento.

**SQL Server** nos permite crear un plan de mantenimiento de nuestras bases de datos de forma muy sencilla, sin embargo es poco común que existan en la mayoría de los entornos sencillamente por desconocimiento técnico de los administradores de sistemas o por no saber cómo las bases de datos se ven afectadas a medida que se realizan operaciones sobre ellas lo cual reduce el rendimiento y a su vez los tiempo de respuesta en operaciones CRUD y especialmente en las consultas.

o

Sea que tengas una aplicación SharePoint, Dynamic CRM, Dynamic NAV, un aplicación hecha a medida, etc., vas a necesitar de un plan de mantenimiento ya que todas estas aplicaciones se ven afectadas de la misma manera. De una manera muy resumida el problema está cuando empezamos a agregar y eliminar información a nuestra base de datos, esto produce que aumente el porcentaje de fragmentación de la base de datos y de los índices por lo que la ruta que consideraba el SQL Server era la majara para realizar determinada consulta ya no lo es, para ello SQL Server se basa en plan de ajecución que en base a ciertas estadísticas le permiten saber cuál es la majar forma de atacar un consulta, pero estas estadísticas si no se actualizan en base al estado actual de la base de datos van quedando obsoletas y mientras más desfragmentación exista más cuesta realizar operaciones de escritura.

Con un poco de análisis de nuestras necesidades podemos llevar a cabo un plan de mantenimiento en pocos minutos. Veamos cómo hacerlo en SSMS.

En SQL Server Managment Studio podremos ver la carpeta Managment y dentro de ella Maintenance Plans, para usar el ayudante seleccionamos Maintenance Plans Wizard.

Sw Databases

1±1 w Serurity

1±1 Ü Server Objec:ts

1±1 w Replication El w Management

1±1 Data-tier Applications

!±l ·'- **Poíicy Management**

1±1 ;e¡¡ Data CoUection

ttl - **Resource Govemor**

El w Marntenance Plans

**tJ MaintenanceP!an**

1±1 w SQL Server Logs

:.iJí D

|  |  |
| --- | --- |
|  | "-"  New Maintenance Plan...  **Maintenance Plan \/izard** |
| View Histcry |
| Reports |
| Rerresh |

l::llj Dis

!±l D**le**

1±1 SQL Se

Hacemos clic en Next

G

**§llMffli§r$i1\_:;;µ5\_:1:, ;1,**



**SQL Server Maintenance Plan Wizard**



**This wizard helpsyou create a maintenance plan that SQL Server Af¡ent can run on a regularbasis. \'ifll:h this111zard you can perfonn routine database administration tasks such as:**

• Check database integrity

* + - **Perfonn index maintenance**
* **Update database statlstics**
* **Perform database backups**

**This •111izard wil! create maintenance plans that can be edited in SQL Seiver Management Studio. Edit maintenance p!ans to add new tasks or define workflo•N among the tasks.**

**C Do not show th!s starting page again.**

Help < Back J 1 Nexl > Fínisb , -¡ Cancel 1 .

*A*

Le damos un nombre a nuestro plan de mantenimiento ya que podremos crear tantos como queramos para distintas base de datos o conjunto de ellas.

Select Plan Properties

**How do you ""ªnt to schedule your maintenance tasks?**

**Name:**

Oescription:



*C'.* Separate schedules for each task

Singlescheduleforthe entire plan orno schedule

Schedule:



**Orange ...**

Help <Back 1 1 Next > l

# •

F\_•r.\_,,\_h \_,

eance1 j

Escogemos las tareas de mantenimiento que queremos realizar.

* **Shrink Database:** reduce espacio de la base de datos y los archivos de log.
* **Update Statistic:** asegura que el optimizador de consultas tiene la información actualizada acerca de la distribución de los datos en la base de datos, esto permite que el optimizador realice mejores cálculos acerca de cómo realizar alguna consulta.
* **Reorganized lndex:** desfragmenta y comprime los índices físicos (Clustered lndex) y no físicos.
* **Rebuild lndex:** reconstruye las páginas índices permitiendo la reorganización de los datos y búsquedas más rápidas.
* **Check Database 1 ntegrity:** confirma la integridad de los datos y índices de la base de datos.
* **History Cleanup:** elimina información histórica sobre copias de respaldos y restauraciones.
* **Maintenance Cleanup:** elimina archivos dajados por la ejecución de los planes de mantenimiento.



**Lji@i§,fld84éffii ..**

**Select Maintenance Tasks**

**Vl/hich tasks should this plan perform?**

**Select one ormore máintenanee taSks:**

o Check Database lntegrity

O Shrink Database

**O Reorganize lndex**

O Rebuíld lndex

O Update Statistics

O Oean Up History

O Execute SQL Seiver Agent Job

O Back Up Database (Ful!)

O Back Up Database {Differential)

O Back Up Database {Transaction log)

**D Maintenance Oeanup Task**

***.,V* The Oieck Database Jntegrit;• task peiforms intema! consistency checks of the data and index pages 'Nithin the database.**

Help 1  Cancel l

*.&*

Escogemos el orden en que se ejecutan las tareas.

. :.:1d1



U::tdijMi§,61,19§ §§1:1,f lr

' .'

Select Maintenance Task Orde

"

ln which order should these tasks be performed?

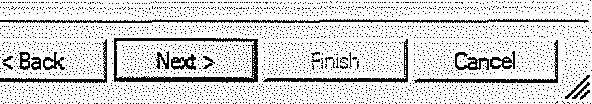
Select !heoróerforthe tasks to execute:

Shnnk Database Reorganize lndex Update Statistics



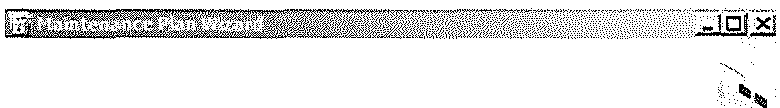
MoveOown...

*.,V* The Shrink Database task reduces the disk space consumed bythe database and log Files b}0 removing empt;• data and log pages.



Help

Para cada tarea se nos pedirá que escqjamos la base de datos, podemos escoger una, varias o todas.



Define Shrink O..tabase Task

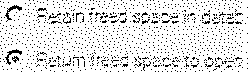
Co!'lf:gure the maíntenancto ta&<:.



*r* System databases



 *(""* All userd {e:Kduding master.model. lnsáb.

-)

r . venture'Norks



 C .:..civentl.n'eYlorlcsD\'V

r A...-!ventWorksD'.'V200S.R2

r AC"enture'NorksL*T*

'NorkslT200SR2

'

..:::J

r Ignore databa:ses <hcre the sta!:e.IS oat onl!ne

, ,\_. \_... c.n:.1 -1

Para cada tarea será necesario escoger las opciones que correspondan.



Define Shrink Database Task

**Configure the maintenance task.**

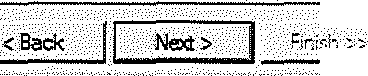
Oatabases:

Shrink database When it grows beyond:

jsn

MB

..mount of free space to remain ,;'\er-Shrink:



*r* Ratain freed spaceindatabase iies

!-'. Retum freed space to operating S)'Slem

Schedule:

Help

Cancel J J

Después de configurar todas las tareas se creara el plan de ajecución.



**Oick Stop to interrupt the operation.**

.::1fi1EJ

*v*



**Llji@h!L,ki,l§§ij§l,1,ff··**

Ma ntenance Plan Wizard Progress

*5* Total

4 Success

O Sror

*O* **Vl/amlng**

Oetails:



Status Message

**Creating maintenance plan "Maintenan** ... **Success**



·--- ---------- -- -----·------

-------------------------------

® **Adding tasks to the maintenance plan Success**

-----·---------------------.---------- ----------- ·-·----------

©! **Adding scheduling options Success**

----- --------- -----------·-------------------------- ·- -- ----------

@ **Adding reporting options Success**

---------------- ------------ --------------

**maintenance plan "Maintenanc ...**

.\_ S\_to\_P\_ ,I R\_.\_e \_·n, \_"\_" ,l

Oose 1

.-4

Podremos ver en el diseñador el plan de mantenimiento que hemos creado.

|  |  |
| --- | --- |
| ShñnkData-  Shrink Databa...  2;J Databases: A..•  Limit: 50 MB  Free space: ... | |
| . |  |

Reorganize -

Update Stat.... Update Statis... Databases: A•• .

. Object: Table...

Ali existing st.•. Sean type: Fu...

¡¡¡ **Reorganize** in.•.

*<;Jh:,* Databases: A...

Object: Table•.• Compact larg..•

En la parte superior del diseñador en el icono con forma de calendario podemos configurar la programación de un nuestro plan de mantenimiento.

l'·iBAdd Subplan

I



1 23

1 Name IM.ainte.· .na.n..ceP...lan1.------------------





1 Subplan DescriPt;on Schedule

Subplan\_l

# •

**E"HEl=fi S#i:ll@i·:!JM!!J@lliri:rl!lIi.::#ll·iil!='Aftj-i!j..2 ·§á:;•·i:.f &:M: I @·-** : .ii ! ;;¡dldlti!l:\4l¿¡y:5¡¡¡;:¡:§:[;2(::"·-·· - . -\_ **fQfXf** i

**-i:**

**NO!f!1': ]!4§1@4 1.i@Jj§4fit#M** '·;,, .,, S·: ""v-,

-ín::-;!".'.'I': -;1;.:::.Y1':'.:r<':: --·-··-------· "..fü .. ---- -- --- " -- "- ·.,---------··-.

,,;al:: **j** $' *.';ii2ZT1* **:::J** ;nin **j** :;i.,t. 1"7;'N1 **3**

"°""'"" --------- ---------------

##### ""°"''

¡;-:j3-<o)m



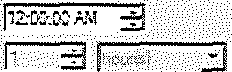
r M°"""Y



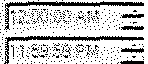
º""-"" .......................T.......... ..... .............. . ..... ....................

1

***r.* OCCIJI'$ O'XC** «-:



*r* o-...,.,,.,



1

°""""" --...-..... --·-----------·-.. ·--·....-.......\_,, ..\_,,\_,....,

S:M<io!e: j *U* ·Z'11 3 *f"'* Endáatc: J { C•C{!I ;:J

1

: "--", ----j"0-'=c--=a--n=-.,:..,: -:-.:-.,:,.,Su.nd:oy\_.:.,l·",·.\_,j.),Q, -e,.n,,d,d,.,--=--=--=- :.:.: - : . -:- .,:,.\_,-.:..:,:s.;s,,:,,,--:'B"" 1

OK J

Finalmente lo mismo que hemos realizado con el SSMS lo podemos realizar con SQL Server lntegration Services y el SQL Server Business lntelligence Development Studio.

,\_.\_. Microsoft SQL Server 2008 R2

J. mport and Export Data (32-bit)

*:;¿* mport and Export Data (64-bit)

SQL Server Business Intelngenoe Develo

·20 SQL Server Management Studio

. **Analysis Services**

Configuration Tools

, ntegration Servioes

**Performance Tools**

Cuando creemos un proyecto podremos ver que en las herramientas tenemos una sección dedicada a tareas del plan de mantenimiento. Solo es necesario arrastrar las tareas que queremos ajecutar.



**Too ox**

**... . x**

1 1±1 **Conbol Bow Items**

**Mt9AiUJQUQ\_lilGUtlJMMu:Jl**

lf Poiriter

@3 Back UpDal:abaseTask

i..J cneckDatabase!íltegrityTask

Exea;te SQLSer:ver AgentJób ••.

3,0 ExeaJte T-5QL.StatementTask

* Historv CleanupTask

1!E.Ma1ntenance Cleanup Task

5, Notify Operator Task

lfü.Rebuild!íldexTask

>d,. Reorganize.IndexTask i:;;,.1 .ShiinkDalabaseTask

Update.StatisticsTask

Rebuld Index Task

**T**

..l.

- Update

-

' StatisticsTask

**T**

,l.

Shrink Database Task

Si hace clic derecho> Editar sobre cualquier tarea la podemos configura exactamente igual como lo hacíamos con SSMS.

IJ

.

,, *,f*

!

Check Database Integrity Task

"

•

**1"' ;;;;;;;;:;:,;;:;:::;:;;;:;:;;:;:r-t:;::::::;;:r:;:;:;:;:;:::::;:;:n**

¡m

m

= íld Index

''"'"""""""""'"""" 1''"""""":f'"' """ -'

Edit .. -

Disable

Add Precedence Constraint

**Group** -'

Autosize Execute Task

Edit Breakpoints...

**Zoom**

di, Cut Copy

21 :0$:2:

X Delete

**Rename**

hril

-1asi

-.

#### •

r ·- ---""--.-

Select Ali

--

i **Properties**

Con esto generaremos un paquete de SSIS que podremos ejecutar cada vez que queramos.

* 1. **Realización de operaciones entre tablas que garanticen la creación, modificación y borrado de registros entre las tablas definidas en el modelo**

Como d)jimos al principio del epígrafe el servidor MySQL nos da la opción de realizar operaciones entre las tablas que garanticen la creación, modificación y borrado de registros entre las tablas definidas en el modelo.

* + 1. **Creación de tablas**

Para la creación de tablas debemos ir a la sentencia **CREATE TABLE.**

La sintaxis de esta sentencia es muy complaja, ya que existen muchas opciones y tenemos muchas posibilidades diferentes a la hora de crear una tabla. Las iremos viendo paso a paso, y en poco tiempo sabremos usar muchas de sus posibilidades.

En su forma más simple, la sentencia CREATE TABLE creará una tabla con las columnas que indiquemos. Crearemos, como ajemplo, una tabla que nos permitirá almacenar nombres de personas y sus fechas de nacimiento. Deberemos indicar el nombre de la tabla y los nombres y tipos de las columnas:

mysql> USE prueba Dat;abase changed

mysql> CRE}TE T.'.BLE gente (nombre VARCHAR(40), fecha DATE); Query OK, O rows affected (0.53 sec)

mysql>

Hemos creado una tabla llamada "gente" con dos columnas: "nombre" que puede contener cadenas de hasta 40 caracteres y "fecha" de tipo fecha.

Podemos consultar cuántas tablas y qué nombres tienen en una base de datos, usando la sentencia **SHOW TABLES:**

mysql> SHOW TABLES;

+------------------+

1 Tables\_in\_prueba 1

+---- -- --- - -- -+

1 gente

+-------------- ---+

l row in set (0.01 sec)

mysql>

Pero tenemos muchas más opciones a la hora de definir columnas. Además del tipo y el nombre, podemos definir valores por defecto, permitir o no que contengan valores nulos, crear una clave primaria, indexar...

La sintaxis para definir columnas es:

nombre\_col tipo [NOT NULL 1 NULL] [DEFAULT valor\_por\_defecto] [AUTO\_INCREMENT ] [[PRIMARY] KEY] [COMMENT 'string']

[definición\_referencia]

•

* + 1. **Modificación de tablas**

Llegamos ahora a un punto interesante, una base de datos sin datos no sirve para mucho, de modo que veremos cómo agregar, modificar o eliminar los datos que contienen nuestras bases de datos.

**Inserción de nuevas filas**

La forma más directa de insertar una fila nueva en una tabla es mediante una sentencia **INSERT.** En Ja forma más simple de esta sentencia debemos indicar la tabla a la que queremos añadir filas, y los valores de cada columna. Las columnas de tipo cadena o fechas deben estar entre comillas sencillas o dobles, para las columnas numéricas esto no es imprescindible, aunque también pueden estar entrecomilladas.

mysql> INSERT INTO gente VALUES ('Fulano' ,'1974-04-12');

Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO gente VALUES ('Mengano','1978-06-15'); Query OK, 1 row affected (0.04 sec)

mysql> INSERT INTO gente VALUES

-> ('Tulano', '2000-12-02'),

-> ('Pegano', '1993-02-10');

Query OK, 2 rows affected (0.02 sec) Records : 2 Duplícates: O Warníngs: O

mysql> SELECT \* FROM gente;

+---------+------------+

J nombre J fecha

+---------+------------+

Fulano 1974-04-12

Mengano 1978-06-15

Tulano 2000-12-02

Pegano 1993-02-10

+---------+------------+

4 rows ín set (0.08 sec)

mysql>

Si no necesitamos asignar un valor concreto para alguna columna, podemos asignarle el valor por defecto indicado para esa columna cuando se creó Ja tabla, usando la palabra DEFAULT:

mysq.1> INSERT INTO ciudad2 VALUES ('Peri.1.lo ', DEFAULT);

Query OK, 1 row affected (0.03 sec)

mysq.1> SELECT *+.* FROM ciudad2;

+----- ---+--- -------+

1 nombre 1 pob.lacion 1

+---------+-- -- -----+

1 Peri.l.lo 1 5000 1

+----- ---+--- -------+

1 row in set (0.02 sec)

mysq.1>

En este caso, como habíamos definido un valor por defecto para población de 5000, se asignará ese valor para la fila correspondiente a 'Perillo'.

Otra opción consiste en indicar una lista de columnas para las que se van a suministrar valores. A las columnas que no se nombren en esa lista se les asigna el valor por defecto. Este sistema, además, permite usar cualquier orden en las columnas, con la ventaja, con respecto a la anterior forma, de que no necesitamos conocer el orden de las columnas en la tabla para poder insertar datos:

mysq.1> INSERT INTO ciudad5 (pob.lacion,nombre) VALUES

|  |  |  |
| --- | --- | --- |
| -> | (7000000, | 'Madrid'), |
| -> | (9000000, | 'París'), |
| -> | (3500000, | 'Berlin'); |

Query OK, 3 rows affected (0.05 sec) Records : 3 Dup.licates : O Warnings: O

mysq.1> SELECT *+.* FROM ciudad5;

+--- ---+------ -+-----------+

1 e.lave 1 nombre 1 pob.lacion 1

+-------+--------+-----------+

1 Madrid 7000000

1. 1 París
2. 1 Berlín

9000000 1

3500000 1

+- - ---+--------+-- - -- ---+

3 rows in set (0.03 sec)

mysq.1>

Cuando creamos la tabla "ciudadS" definimos tres columnas: 'clave', 'nombre' y 'poblacion' (por ese orden). Ahora hemos insertado tres filas, en las que hemos omitido la clave, y hemos alterado el orden de 'nombre' y 'poblacion'. El valor de la 'clave' se calcula automáticamente, ya que lo hemos definido como auto-incrementado.

# •

Existe otra sintaxis alternativa, que consiste en indicar el valor para cada columna:

mysql> INSERT INTO ciudad5

-> SET nombre='Roma', poblacion=SOOOOOO; Query OK, 1 row affected (0.05 sec)

mysql> SELECT \* FROM ciudad5 ;

+-------+--------+----- -----+

1 clave 1 nombre 1 poblacion 1

+----- -+---- ---+-----------+

1 Madrid 7000000

2 París 9000000

3 Berlín 3500000

Roma 8000000

+--- - -+--- ----+-- -- -----+

4 rows in set (0.03 sec)

mysql>

Una vez más, a las columnas para las que no indiquemos valores se les asignarán sus valores por defecto. También podemos hacer esto usando el valor DEFAULT.

Para las sintaxis que lo permiten, podemos observar que cuando se inserta más de una fila en una única sentencia, obtenemos un mensaje desde MySQL que indica el número de filas afectadas, el número de filas duplicadas y el número de avisos.

Para que una fila se considere duplicada debe tener el mismo valor que una fila existente para una clave principal o para una clave única. En tablas en las que no exista clave primaria ni índice de clave única no tiene sentido hablar de filas duplicadas. Es más, en esas tablas es perfectamente posible que existan filas con los mismos valores para todas las columnas.

Por ajemplo, en *mitabla5* tenemos una clave única sobre la columna 'nombre' :

mysql> INSERT INTO mitabla5 (id, nombre) v. UES

-> **( 1, r Carlos** 1 ) *I*

-> ( 2 ,.. **1 FeJ.ipe** 1 ) ,,.

-> ( 3 , • **...a..ntoni.o** 1 ) ,

-> **(4, 1 Carlos 1 ) ,**

-> ( 5, ' J'uan ' ) ;

ERROR 1062 (23000): Duplicate entry 'Carlos' for key 1 mysql>

Si intentamos insertar dos filas con el mismo valor de la clave única se produce un error y la sentencia no se ejecuta. Pero existe una opción que podemos usar para los casos de claves duplicadas: ON DUPLICATE KEY UPDATE. En este caso podemos indicar a MySQL qué debe hacer si se intenta insertar una fila que ya existe en la tabla. Las opciones son limitadas: no podemos insertar la nueva fila, sino únicamente modificar la que ya existe. Por ajemplo, en la tabla 'ciudad3' podemos usar el último valor de población en caso de repetición:

mysql> INSERT INTO ciudad3 (nombre, poblacion) VALUES

-> ('Madrid', 7000000);

Query OK, 1 rows affected (0.02 sec)

mysql> INSERT INTO ciudad3 (nombre, poblacion) VALUES

-> ('París', 9000000),

-> ('Madrid', 7200000)

-> *ON* DUPLICATE KEY UPDATE poblacion=VALUES (poblacion); Query OK, 3 rows affected (0.06 sec)

Records: 2 Duplicates : 1 Warnings: O

mysql> SELECT \* FROM ciudad3;

+--------+-----------+

1 nombre 1 poblacion 1

+--------+-----------+

1 Madrid 1 7200000 1

1 París 1 9000000 1

+--------+-----------+

2 rows in set (0.00 sec)

mysql>

En este ajemplo, la segunda vez que intentamos insertar la fila correspondiente a 'Madrid' se usará el nuevo valor de población. Si en lugar de VALUES(poblacion) usamos poblacion el nuevo valor de población se ignora. También podemos usar cualquier expresión:

mysql> INSERT INTO ciudad3 (nombre, poblacion) VALUES

-> ('París', 9100000)

-> *ON* DuPLICATE KEY UPDATE poblacion=poblacion; Query OK, 2 rows affected (0.02 sec)

mysql> SELECT \* FROM ciudad3;

+--------+-----------+

1 nombre 1 poblacion 1

+--------+-----------+

1 Madrid 1 7200000 1

1 París 1 9000000 1

+--------+-----------+

2 rows in set (0.00 sec)

mysql> INSERT INTO ciudad3 (nombre, poblacion) VALUES

-> ('París', 9100000)

-> *ON* DUPLICATE KEY UPDATE poblacion=O;

Query OK, 2 rows affected (0.01 sec)

mysql> SELECT \* FROM ciudad3;

+--------+-----------+

1 nombre 1 poblacion 1

+--------+-----------+

1 Madrid 1 7200000 1

1 París 1 O 1

+--------+-----------+

2 rows in set (0.00 sec)

mysql>

**Reemplazar filas**

Existe una sentencia REPLACE, que es una alternativa para INSERT, que sólo se diferencia en que si existe algún registro anterior con el mismo valor para una clave primaria o única, se elimina el viajo y se inserta el nuevo en su lugar.

REPLACE [LOW\_PRIORITY 1 DELAYED]

[INTO] tbl\_name [(col\_name, ...)] VALUES (,...), (...),...

REPLACE [LOW\_PRIORITY 1 DELAYED]

[INTO] tbl\_name SET col\_name=,

mysql> REPLACE INTO ciudad3 (nombre, poblacion) VALUES

-> ('Madrid', 7200000),

-> ('París', 9200000),

-> ('Berlín', 6000000);

Query OK, 5 rows affected (0.05 sec) Records : 3 Duplicates : 2 Warnings: O

mysql> SELECT \* FROM ciudad3;

+------ -+- ---------+

1 nombre 1 poblacion 1

+--------+---- - -- -+

Berlín

1 Madrid 1

1 París 1

6000000 1

7200000 1

9200000 1

+- -- ---+--- -- ----+

3 rows in set (0.00 sec)

mysql>

En este ajemplo se sustituyen las filas correspondientes a 'Madrid' y 'París', que ya existían en la tabla y se inserta la de 'Berlín' que no estaba previamente.

Las mismas sintaxis que existen para INSERT, están disponibles para REPLACE:

mysql> REPLACE INTO ciudad3 VALUES ('Roma', 9500000);

Query OK, l rows affected (0.03 sec)

mysql> REPLACE INTO ciudad3 SET norobre='Londres', poblacion=lOOOOOOO; Query OK, l row affected (0.03 sec)

mysql> SELECT \* FROM ciudad3;

+---------+-----------+

1 nombre 1 poblacion 1

+---------+-----------+

|  |  |
| --- | --- |
| **Berlín** | 6000000 |
| **Londres** | 10000000 |
| Madrid | 7200000 |
| París | 9200000 |

Roma 9500000

+---------+-----------+

5 rows in set (O.DO sec)

mysql>

**Actualizar filas**

Podemos modificar valores de las filas de una tabla usando la sentencia UPDATE. En su forma más simple, los cambios se aplican a todas las filas, y a las columnas que especifiquemos.

UPDATE [LOW\_PRIORITY ] [IGNORE] tbl\_na!tle

SET col\_na!tlel=exprl [, col\_na!tle2=expr2 ...] [WHERE where\_definition]

[ORDER BY ...]

[LIIIT row\_count]

Por ajemplo, podemos aumentar en un 10% la población de todas las ciudades de la tabla ciudad3 usando esta sentencia:

mysql.> UPDATE ciudad3 SET poblacion=poblacion\*l.10; Query OK, 5 rows affected (0.15 sec)

Rows matched: 5 Changed: 5 Warnings: O

mysql> SELECT \* FROM ciudad3;

+---------+-----------+

1 nombre 1 poblacion 1

+---------+-----------+

|  |  |
| --- | --- |
| Ber.lín | 6600000 |
| Londres | 11000000 |
| Madrid | 7920000 |
| **París** | 10120000 |
| Roma | 10'!50000 |

+---------+-----------+

5 rows in set (O.DO sec) mysql>

Podemos, del mismo modo, actualizar el valor de más de una columna, separándolas en la sección SET mediante comas:

mysql.> UPDATE ciudad5 SET clave=clave+lO, poblacion=poblacion\*0.97; Query OK, 4 rows affected (0.05 sec)

Rows matched: 'l Changed: 4 Warnings: O

mysql> SELECT \* FROM ciudad5;

+-------+--------+-----------+

1 clave 1 nombre 1 poblacion 1

+-------+------ -+-----------+

|  |  |  |
| --- | --- | --- |
| 11 | Madrid | 6790000 |
| 12 | París | 8730000 |
| 13 | Berlín | 3395000 |

14 Roma 7760000

+-------+--------+-----------+

'l rows in set (O.DO sec)

mysql>

En este ajemplo hemos incrementado el valor de la columna ' clave' en 10 y

disminuido el de la columna 'poblacion' en un 3%, para todas las filas.

Pero no tenemos por qué actualizar todas las filas de la tabla. Podemos limitar el número de filas afectadas de varias formas.

La primera es mediante la cláusula WHERE. Usando esta cláusula podemos establecer una condición. Sólo las filas que cumplan esa condición serán actualizadas:

mysql> UPDATE ciudadS SET poblacion=poblacion\*l.03

-> **WHERE nombre= 1 Roma 1 ;**

Query OK, 1 row affected (O.OS sec)

Rows matched: 1 Changed: 1 Warnings: O

mysql> SELECT \* FROM ciudadS;

+- -- --+--------+- ------ --+

1 clave 1 nombre 1 poblacion 1

+----- -+---- ---+----- -----+

11 Madrid 6790000

1. **París**
2. Berlin
3. Roma

8730000

3395000

7992800

+-------+---- ---+- ---------+

4 rows in set (0.00 sec)

mysql>

En este caso sólo hemos aumentado la población de las ciudades cuyo nombre sea 'Roma'. Las condiciones pueden ser más complajas. Existen muchas funciones y operadores que se pueden aplicar sobre cualquier tipo de columna, y también podemos usar operadores booleanos como AND u OR. Veremos esto con más detalle en otros capítulos.

Otra forma de limitar el número de filas afectadas es usar la cláusula LIMIT. Esta cláusula permite especificar el número de filas a modificar:

mysql> UPDATE ciudad5 SET clave=clave-10 LIMIT 2; Query OK, 2 rows affected (0.05 sec)

Rows matched: 2 Changed: 2 Warnings: O

mysql> SELECT \* FROM ciudad5;

+-------+--------+----- ------+

1 clave 1 nombre 1 poblacion 1

+-------+--------+-----------+

1 Madrid 6790000

2 **París**

13 Berlín

14 Roma

8730000

3395000

7992800

+-------+-------+-----------+

4 rows in set (0.00 sec)

mysql>

En este ajemplo hemos decrementado en 1O unidades la columna clave de las dos primeras filas.

Esta cláusula se puede combinar con WHERE, de modo que sólo las 'n' primeras filas que cumplan una determinada condición se modifiquen.

Sin embargo esto no es lo habitual, ya que, si no existen claves primarias o únicas, el orden de las filas es arbitrario, no tiene sentido seleccionarlas usando sólo la cláusula LIMIT.

La cláusula LIMIT se suele asociar a la cláusula ORDER BY. Por ajemplo, si queremos modificar la fila con la fecha más antigua de la tabla 'gente', usaremos esta sentencia:

mysql> UPD.O.TE gente SET fecha="l985-04-12" ORDER BY fecha LIMIT l; Query OK, 1 row affected, 1 warning (0.03 sec)

Rows matched: 1 Changed: 1 Warnings: 1

mysql> SELECT \* FROM gente;

+---------+------------+

1 nombre 1 fecha

+---------+------------+

Fulano 1985-04-12

Mengano

Tulano

1978-06-15

2000-12-02

Pegano 1993-02-10

+---------+------------+

4 rows in set (0.00 sec)

mysql>

Sí queremos modificar la fila con la fecha más reciente, usaremos el orden inverso, es decir, el descendente:

mysql> UPD..>TE gente SET fecha="200l-l2-02" ORDER BY fecha DESC LIMIT l; Query OK, l row affected (0.03 sec)

Rows matched: l Changed: l Warnings: O

mysql> SELECT • FROM gente;

+- -------+------------+

1 nombre 1 fecha

+---------+------------+

Fulano 1985-04-12

Mengano 1978-06-15

Tulano 2001-12-02

Pegano 1993-02-10

+---------+------------+

4 rows in set (0.00 sec)

mysql>

**Eliminar filas**

Para eliminar filas se usa la sentencia DELETE. La sintaxis es muy parecida a la de **UPDATE:**

DELETE [LOW\_PRIORITY ) [QUICK) [IGNORE) FROM table name

[WHERE where\_definition) [ORDER BY ..•)

[LIMIT row\_c·ount)

La forma más simple es no usar ninguna de las cláusulas opcionales:

mysql> DELETE FROM ciudad3;

Query OK, 5 rows affected (0.05 sec) mysql>

De este modo se eliminan todas las filas de la tabla.

Pero es más frecuente que sólo queramos eliminar ciertas filas que cumplan determinadas condiciones. La forma más normal de hacer esto es usar la cláusula WHERE:

# •

mysql> DELETE FROM ciudad5 WHERE clave=2; Query OK, 1 row affected (0.05 sec)

mysql> SELECT \* FROM ciudad5;

+-------+--------+- --- -----+

1 clave 1 nombre 1 poblacion 1

+----- -+--------+- ---------+

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | Madrid |  | 6790000 1 |
| 13 | 1 Ber.lín | 1 | 3395000 1 |
| 14 | 1 Roma | 1 | 7992800 1 |

+-------+--------+--- --- ---+

3 rows in set (0.01 sec)

mysql>

También podemos usar las cláusulas LIMIT y ORDER BY del mismo modo que en la sentencia UPDATE, por ejemplo, para eliminar las dos ciudades con más población:

mysql> DELETE FROM ciudad5 ORDER BY poblacion DESC LIMIT 2;

Query OK, 2 rows affected (0.03 seci

mysql> SELECT \* FROM ciudad5;

+-------+- ------+--- --- ---+

1 clave 1 nombre 1 poblacion 1

+- -----+-- -----+--------- -+

13 1 Ber.lín 1 3395000 1

+- -----+--- ----+-----------+

1 row in set (0.00 sec)

mysql>

**Vaciar una tabla**

Cuando queremos eliminar todas las filas de una tabla, vimos en el punto anterior que podíamos usar una sentencia DELETE sin condiciones. Sin embargo, existe una sentencia alternativa, TRUNCATE, que realiza la misma tarea de una forma mucho más rápida.

La diferencia es que DELETE hace un borrado secuencial de la tabla, fila a fila. Pero TRUNCATE borra la tabla y la *vuelve* a crear vacía, lo que es mucho más eficiente.

mysql> TRUNO.TE ciudad5;

Query OK, 1 row affected (0.05 sec) mysql>

* + 1. **Borrado de tablas**

A veces es necesario eliminar una tabla, ya sea porque es más sencillo crearla de nuevo que modificarla, o porque ya no es necesaria.

Para eliminar una tabla se usa la sentencia **DROP TABLE.**

La sintaxis es simple:

DROP T."..BLE (IF EXISTS] tbl nallle (, tbl\_nallle]

Por ajemplo:

mysql> DROP T.1>.BLE ciudad6;

Query OK, O rows affected (0.75 sec) rnysql>

Se pueden añadir las palabras IF EX ISTS para evitar errores si la tabla a eliminar no existe.

mysql> DROP TABLE ciudad6;

ERROR 1051 (42502) : Unknown table 'ciudad6' mysql> DROP TABLE IF EXISTS ciudad6;

**Query OK, O rows affected, 1 warning (0.00 sec)**

mysql>

# •

###### RECUERDA

* + - * Un **objeto** es un elemento autocontenido utilizado por el programa. Los valores que almacena un oQjeto se denominan atributos, variables o propiedades. Los oQjetos pueden realizar acciones, que se denominan métodos, servicios, funciones, procedimientos u operaciones.
      * Una **estructura de datos,** o un tipo de datos estructurado, es un tipo de dato construido a partir de otros. Un dato de tipo estructurado está compuesto por una serie de datos de tipos elementales y alguna relación existente entre ellos. Normalmente, la relación suele ser de orden aunque puede ser de cualquier otro tipo.
      * Dentro de los tipos de estructuras de datos que ocupan siempre el mismo espacio en memoria tenemos los **punteros,** las **cadenas** y los **arrays.** Las estructuras dinámicas ocupan un espacio en memoria que va evolucionando según el tamaño que dicha estructura vaya adquiriendo. Las estructuras de dinámicas son: **colas, pilas, listas encadenadas y árboles.**
* **MySQL** es el servidor de bases de datos relacionales más popular, desarrollado y proporcionado por MySQL AB. MySQL AB es una empresa cuyo negocio consiste en proporcionar servicios en torno al servidor de bases de datos MySQL.
* Hoy en día las bases de datos son una necesidad indispensable en la gestión de las empresas, por tal motivo se debe garantizar su disponibilidad, consistencia de datos y privacidad en todo momento.
* **SQL Server** nos permite crear un plan de mantenimiento de nuestras bases de datos de forma muy sencilla, sin embargo es poco común que existan en la mayoría de los entornos sencillamente por desconocimiento técnico de los administradores de sistemas o por no saber cómo las bases de datos se ven afectadas a medida que se realizan operaciones sobre ellas lo cual reduce el rendimiento y a su vez los tiempo de respuesta en operaciones CRUD y especialmente en las consultas.
* MySQL nos da la opción de realizar operaciones entre las tablas que garanticen la creación, modificación y borrado de registros entre las

tablas definidas en el modelo.

* Para la creación de tablas debemos ir a la sentencia **CREATE TABLE.**
* Para eliminar una tabla se usa la sentencia **DROP TABLE.**

Preguntas de Autoeval uación

1. **Confirma si la siguiente afirmación es verdadera o falsa:**

*"Los valores que almacena un ol:¿jeto se denominan atributos, variables o propiedades".*

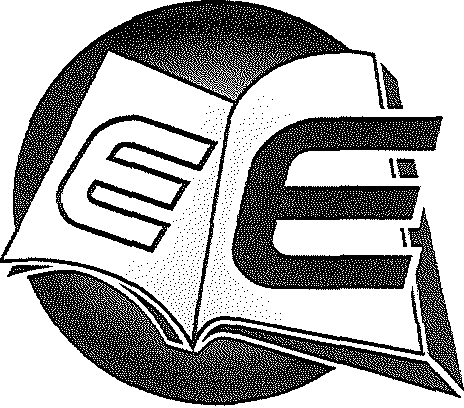
1. Verdadero.
2. Falso.
3. **Dentro de los tipos de estructuras de datos que ocupan siempre el mismo espacio en memoria tenemos:**
4. Las colas (FIFO).
5. Las pilas (UFO) .
6. Los arrays.
7. **¿Qué es una cola?**
8. Una estructura de datos en la que el primer dato en entrar es el primer dato en salir.
9. Una colección ordenada de datos a los que sólo se puede acceder por un extremo.

e) Un conjunto ordenado de datos.

1. **Una tabla es una colección de datos organizados en:**
2. Atributos y variables.
3. Filas y columnas.
4. Punteros y cadenas.
5. **Para la creación de tablas con MySQL debemos ir a la sentencia:**
6. CREATE TABLE.
7. NEW TABLE.
8. INSERT TABLE.

**Defi ición de los tipos de datos**

UD3



UF1888 Operaciones de Mantenimiento y C,onsulta de Datos

# •

* 1. Definición, tipos de datos y características semánticas

Después de la fase de diseño de una base de datos, y una vez se ha realizado el paso a tablas del mismo, en necesario crear las tablas correspondientes dentro de la base de datos. Para cada campo de cada una de las tablas, es necesario determinar el tipo de datos que contiene; de esta forma ajustaremos el diseño de la base de datos, y conseguiremos un almacenamiento óptimo con la menor utilización de espacio.

En este epígrafe vamos a describir cada uno de los tipos de datos que puede tener un campo en MySQL, así como la definición de constantes y variables en el modelo de datos.

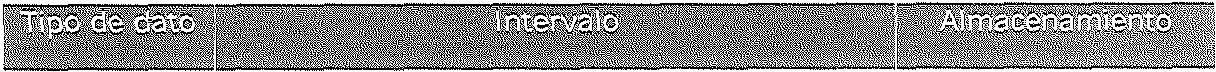
Debemos tener en cuenta que cada aplicación como por ejemplo Microsoft Excel y Open Office, entre otros, tendrán sus propios tipos de datos.

* + 1. **Definición de tipos de datos que se utilizarán en el modelo de datos**

Los tipos de datos que puede haber en un campo en MySQL, se pueden agrupar en tres grandes grupos:

* + - * Tipos de datos numéricos.
      * Tipos de fecha y hora.
      * Tipos de cadena.

**1.1.1. Datos numéricos**



De -128 a 127 (signed)

TINYINT

SMALLINT

De O a 255 (unsigned)

De -32768 a 32767 (signed) De O a 65535 (unsigned)

1. byte
2. bytes

G

MEDIUMINT

INT INTEGER

BIGINT

BIT

BOOL BOOLEAN

FLOAT [(M,0)]

FLOAT (p)

DOUBLE [(M,0)]

REAL[(M,O)] DOUBLE PRECISION

DECIMAL [(M[,O])]

DEC [*(M[.*O])] NUMERIC [(M[,O])]

FIXED[(M[.D])]

De -8388608 a 8388607 (signed) De O a 16777215 (unsigned)

De -2147483648 a 2147483647 (signed)

De O a 4294967295 (unsigned)

De -9223372036854775808 a 9223372036854775807

(signed)

De O a 18446744073709551615 (unsigned)

Equivalente a TINYINT(1)

Equivalente a TINYINT(1) Valor o = False

**Valor 1 = True**

De -3.402823466E+38 a -1.175494351 E-38, O, y desde 1.175494351E-38 a 3.402823466E+38

**M es el número total de dígitos y *O* es el número de dígitos después del punto decimal. Si se omite *M* y *O,* los valores se almacenan en los límites permitidos por el hardware (unas 7 posiciones decimales)**

*p* representa la precisión en bits, MySQL usa este valor sólo para determinar si se debe usar FLOAT o DOUBLE **para el tipo de datos resultante.**

**Si *p* está entre O a 24, el tipo de datos se convierte en** FLOAT (sin M *ó* D). Si *p* está entre 25 a 53, el tipo de datos se convierte a DOUBLE (sin M *ó* D).

**En realidad este tipo de datos es proporcionado por**

MySQL por compatibilidad con ODBC

De -1.7976931348623157E+308 a

-2.2250738585072014E-308, o, y desde 2.2250738585072014E-308 a

1.7976931348623157E+308

***M es* el número total de dígitos y *D* es el número de dígitos después del punto decimal. Si se omite *M* y *D,* los valores se almacenan en los límites permitidos por el hardware (unas 15 posiciones decimales)**

Equivalente a DOUBLE, con la excepción de que si está activado el modo REAL\_AS\_FLOAT, REAL es un sinónimo de FLOAT en lugar de DOUBLE

**Número en coma flotante sin empaquetar. Se comporta como una columna CHAR. El número se almacena como una cadena, usando un caracter para cada dígito del valor.**

**El rango maximo es el mismo que para el tipo DOUBLE**

# •

1. bytes
2. bytes

8 bytes

1 byte

1 byte

4 bytes

4 bytes si o <= *p* <= 24, 8 bytes si 25 < = *p* <=

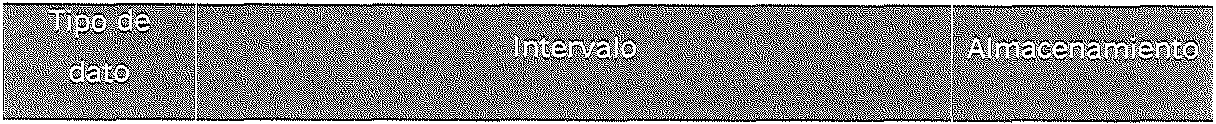
53

8 bytes

4 Bytes *ó* 8 bytes

*M+2* bytes sr *o >* o M+1 bytes sr *D* = o ***0+2 ,* si *M < D***

* + 1. **Tipos de fecha y hora**



DATE

DATETIME

TIMESTAMP

*[ (M,0)]*

TIME

YEAR ((214)]

Fecha, con rango desde '1000-01-01' a '9999-12-31' con

formato 'YYYY-MM-DD'

Fecha y hora, con rango desde '1000-01-01 00:00:00' a '9999- 12-31 23:59:59' con formato 'YYYY-MM-DD HH:MM:SS'

Fecha y hora, el rango *va* desde '1970-01-01 00:00:01' UTC a '2038-01-19 03:14:07' UTC, El formato de almacenamiento depende del tamaño del campo

Hora, con rango desde '-838:59:59' a '838:59:59', con el formato 'HH:MM:SS'

Año en dos o cuatro dígitos, para cuatro dígitos, el rango es de 1901 a 2155, para dos dígitos es de 70 a 69 (representando

desde 1070 a 2069)

3 bytes

8 bytes

4 bytes

3 bytes

1 byte

* + 1. **Cadenas de caracteres**

CHAR (M)

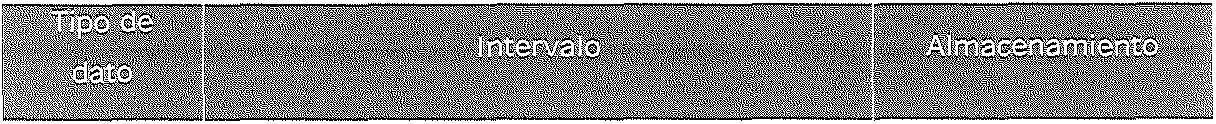
VARCHAR (M)

BINARY (M)

VARBINARY

(M)

Almacena una cadena de longitud füa. La cadena podrá



**contener desde O a 255 caracteres**

**Almacena una cadena de longitud variable. La cadena**

**podrá contener desde O a 255 caracteres**

**Similar a CHAR, excepto que contiene cadenas de caracteres binarias en lugar de cadenas no binarias. Es decir, que contienen cadenas de bytes en lugar de cadenas de caracteres. Esto significa que no tienen conjunto de caracteres, y la comparación y ordenación se basa en los valores numéricos de los bytes en los valores**

**Similar a VARCHAR, excepto que contiene cadenas de caracteres binarias en lugar de cadenas no binarias. Es decir, que contienen cadenas de bytes en lugar de cadenas de caracteres. Esto significa que no tienen**

**conjunto de caracteres, y la comparación y ordenación**

**se basa en los valores numéricos de los bytes en los**

Mbytes (tanto si se ocupan

**como si no)**

**Tamaño contenido del**

campo más 1 byte

Mbytes, o < = *M <=* 255

**Tamaño contenido del**

campo más 1 byte

**valores**

TEXT

TINYTEX

MEDIUMTEXT

LONGTEXT

BLOB

TINYBLOB

MEDIUMBLOB

**Tipo de datos no binario que puede contener una cantidad variable de datos. Sirve para almacenar texto** (gran cantidad). Hasta 65535 caracteres

**Tipo de datos no binario que puede contener una cantidad variable de datos. Sirve para almacenar texto** (gran cantidad). Hasta 255 caracteres

**Tipo de datos no binario que puede contener una cantidad variable de datos. Sirve para almacenar texto** (gran cantidad). Hasta 16.777.215 caracteres

**Tipo de datos no binario que puede contener una cantidad variable de datos. Sirve para almacenar texto** (gran cantidad). Hasta 4.294.967.295 caracteres

**Tipo de datos binario que puede contener una cantidad variable de datos. Permite almacenar ficheros (de** cualquier tipo). Hasta 65535 bytes

**Tipo de datos binario que puede contener una cantidad variable de datos. Permite almacenar ficheros (de** cualquier tipo). Hasta 255 bytes

**Tipo de datos binario que puede contener una cantidad variable de datos. Permite almacenar ficheros (de** cualquier tipo). Hasta 16.777.215 bytes

*Longitucl* + 2 bytes,

**mientras *L* < 216**

*Longitucl* + 1 bytes,

**mientras *L* < 2ª**

*Longitucl* + 3 bytes,

**mientras *L* < 224**

*Longitucl* + 4 bytes,

**mientras *L* < 232**

*Longitucl* + 2 bytes,

**mientras *L* < 216**

*Longitucl* + 1 bytes,

**mientras *L* < 28**

*Longitucl* + 3 bytes,

**mientras *L* < 224**

LONGBLOB

**Tipo de datos binario que puede contener una cantidad *Longitud*** + 4 **bytes, variable de datos. Permite almacenar ficheros (de mientras *L*** < 2s2 cualquier tipo). Hasta 4.294.967.295 bytes

Es un tipo de datos de cadena con un valor elegido de 1 6 2 bytes, dependiendo

ENUM (valor1, valor2, ...)

SET (valor1, valor2, ...)

**una lista de valores permitidos que se enumeran explícitamente en la especificación de la columna al** crear la tabla. Acepta hasta 65535 valores distintos

**Es un tipo de datos de cadena que puede contener ninguno, uno ó varios valores de una lista previamente** establecida (al crear la tabla). La lista puede tener un **maximo de 64 valores**

del número de valores de ENUM

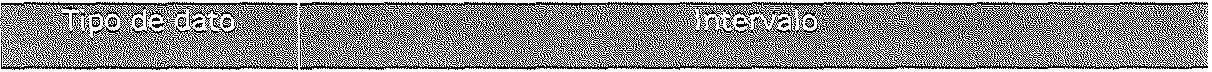
1, 2, 3, 4, ó 8 bytes, **dependiendo del número de miembros del conjunto**

* + 1. **Tipos de datos espaciales**

MySQL implementa extensiones espaciales siguiendo la especificación del Consorcio Open GIS (OGC), un consorcio internacional de más de 250 compañías, agencias y universidades que participan en el desarrollo de soluciones conceptuales públicamente disponibles y que pueden ser útiles para todo tipo de aplicaciones que

# •

manajan datos espaciales. MySQL implementa un subconjunto del entorno SQL con Tipos Geométricos propuesto por el OGC. Este término se refiere a un entorno SQL que ha sido extendido con un conjunto de tipos geométricos. Una columna SQL con valores geométricos se implementa como una columna que tiene un tipo geométrico. Las especificaciones describen un conjunto de tipos geométricos SQL, así como las funciones para analizar y crear valores geométricos sobre esos tipos.



Geometry es la clase base de lajerarquía para el modelo geométrico

GEOMETRY

**OpenGIS. Es una clase no instancíable, pero tiene unas cuantas propiedades que son comunes para todos los valores geométricos creados con cualquiera** de las subclases de Geometry

POINT

**Un POINT es una geometría que representa una ubicación única en un espacio de coordenadas (valor de la coordenada X, valor de la coordenada Y)**

LINESTRING

**Un LINESTRING es una Curva con interpolación linear entre puntos. Un** LINESTRI NG tiene coordenadas de segmentos, definidos por cada par **consecutivo de puntos**

POLYGON

**Un POLYGON es una superficie planar que representa una geometría multicara. Se define por un único límite exterior y cero o mas límites interiores, donde cada límite interior define un agujero en el polígono**

MULTIPOINT

**Un MULTIPOINT es una colección de geometrías compuesta de elementos Point. Los puntos no están conectados ni ordenados de ningún modo**

MULTILINESTRING

Una MULTILINESTRI NG es una colección de geometrías MultiCurve compuesta de elementos LINESTRI NG

MULTIPOLYGON

Un MULTIPOLYGON es un O!<ieto MultiSurface compuesto de elementos POLYGON

Una GEOMETRYCOLLECTION es una geometría que consiste en una colección

**de una o más geometrías de cualquier clase.**

GEOMETRYCOLLECTION Todos los elementos en una GEOMETRYCOLLECTION deben estar en el mismo **Sistema de Referencia Espacíal (es decir, en el mismo sistema de coordenadas)**

CURVE

**Una clase CURVE es una geometría unidimensional, normalmente representada por una secuencia de puntos. Las subclases particulares de CURVE definen el tipo de interpolación entre puntos. CURVE es una clase no instancia ble**

MULTICURVE

**Una clase MULTICURVE es una colección de geometrías que se compone de elementos CURVE. MULT!CURVE es una clase no instanciable**

SURFACE

**Una clase SURFACE es una geometría bidimensional. Es una clase no instanciable. Su úníca subclase instanciable es POLYGON**

* 1. **Definición de constantes y variables en el modelo de datos**

Llegados a este punto, es fundamental clarificar dos términos muy importantes dentro de la programación informática; éstos son el concepto de constante y de una variable.

* + 1. **Definición de constante**

En programación, una **constante** es un valor que no puede ser alterado durante la ajecución de un programa.

Una constante corresponde a una longitud f!.ia de un área reservada en la memoria principal del ordenador, donde el programa almacena valores f!.ios.

Por ajemplo: el valor de pi = 3.1416

Por conveniencia, el nombre de las constantes suele escribirse en mayúsculas en la mayoría de lenguajes.

* + 1. **Definición de variables**

En programación, una **variable** está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Ese espacio contiene una cantidad o información conocida o desconocida, es decir un valor.

El nombre de la variable es la forma usual de referirse al valor almacenado: esta separación entre nombre y contenido permite que el nombre sea usado independientemente de la información exacta que representa. El identificador, en el código fuente de la computadora puede estar ligado a un valor durante el tiempo de ejecución y el valor de la variable puede por lo tanto cambiar durante el curso de la ajecución del programa.

El concepto de variables en computación puede no corresponder directamente al concepto de variables en matemática. El valor de una variable en computación no es necesariamente parte de una ecuación o fórmula como en matemáticas. En computación una variable puede ser utilizada en un proceso

# •

repetitivo: puede asignársele un valor en un sitio, ser luego utilizada en otro, más adelante reasignársele un nuevo valor para más tarde utilizarla de Ja misma manera. Procedimientos de este tipo son conocidos con el nombre de **iteración.**

En programación de computadoras, a las variables, frecuentemente se le asignan nombres largos para hacerlos relativamente descriptivas para su uso, mientras que las variables en matemáticas a menudo tienen nombres escuetos, formados por uno o dos caracteres para hacer breve en su transcripción y manipulación.

El espacio en el sistema de almacenaje puede ser referido por distintos identificadores diferentes. Esta situación es conocida entre Jos angloparlantes como *"aliasing"* y podría traducirse como *"sobrenombramiento"* para los hispanoparlantes. Asignarle un *valor* a una variable utilizando uno de los identificadores cambiará el *valor* al que se puede acceder a través de los otros identificadores.

Los compiladores deben reemplazar los nombres simbólicos de las variables con la real ubicación de los datos. Mientras que el nombre, tipo y ubicación de una variable permanecen f!.ios, los datos almacenados en la ubicación pueden ser cambiados durante la ajecución del programa.

Las variables pueden ser de longitud:

* + - * **Fija:** cuando el tamaño de la misma no variará a lo largo de la ejecución del programa. Todas las variables, sean del tipo que sean tienen longitud f!.ia, salvo algunas excepciones - como las colecciones de otras variables (arrays) o las cadenas.
* **Variable:** cuando el tamaño de la misma puede variar a lo largo de la ajecución. Típicamente colecciones de datos.
  1. **Extensión del modelo de datos en sistemas ERP y CRM**

Los módulos de un sistema ERP varían dependiendo de las características de la empresa, pues son muy diferentes los requerimientos en organizaciones en las

•

que, por ajemplo, su principal negocio es la producción, la distribución o bien los servicios. Algunos de los módulos más comunes son:

* **Gestión Financiera.** Habitualmente agrupa las funciones de Contabilidad, Tesorería, Presupuestos y Activos Fijos
* **Gestión de Ventas/Compras.** Incluirá la funcionalidad referida a la gestión de la cadena de suministro, aprovisionamientos, gestión del ciclo de ventas desde la presentación de ofertas hasta la facturación, etc.
* **Gestión de la Distribución y Logística.** Permite al usuario la gestión de almacenes en sus distintas variantes.
* **Gestión y planificación de la Producción.** Control y gestión de los procesos de producción
* **Gestión de Proyectos.** Control y gestión de los proyectos en sus distintas fases.
* **Gestión de Recursos Humanos.** Gestión de la Empresa con sus empleados (datos personales, control de presencia, etc.

Como podemos ver, la mayoría de los ERPs adoptan una estructura modular que soporta los diferentes procesos de una empresa: el modulo de gestión financiera, el módulo de compras, el módulo de gestión de ventas, el módulo de recursos humanos, etc.

**MRP**

Plan de búsqueda de material

**CSM**

Gestión de

**suministros**

**CRM**



**ERP**

Base de datos con el plan de recursos de la empresa

Gestión de recursos clientes

**HRM FRM**

Gestión de recursos Gestión de recursos humanos financieros

Todos estos módulos están interconectados y comparten una base de datos común, garantizando de este modo la coherencia e integración de los datos generados.

# •

El hecho de que estos productos sean modulares posibilita la implantación del sistema por etapas, reduciendo el impacto global en la organización al facilitar la transición desde los sistemas anteriores. Normalmente, el primer módulo que se pone en marcha es el financiero y, posteriormente, se van integrando los restantes, dependiendo de las características particulares de cada empresa

El sistema básico del ERP está formado por las aplicaciones técnicas y la arquitectura necesaria para servir de plataforma al resto de los módulos.

Proporciona herramientas de administración para controlar tanto el sistema en sí (rendimiento, comunicación con otras aplicaciones y otros sistemas, etc.), como la base de datos que constituye el núcleo del producto.

Las principales plataformas de servidores son Microsoft y UNIX, mientras que las bases de datos más utilizadas son Oracle, Microsoft SQL

Server e IBM. Asimismo, la mayor parte de los sistemas ERP disponen de lenguajes de programación propietarios de cuarta generación, que facilitan el desarrollo y adaptación de aplicaciones a la medida de cada cliente.

* + 1. **Definición de tablas en el sistema**

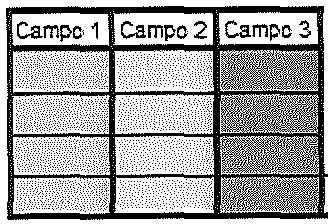
Recordemos que en la mayoría de las bases de datos la información se almacena en **TABLAS,** que ordenan los datos en filas y columnas. Las filas se denominan **REGISTROS** y las columnas se llaman **CAMPOS.**

Las tablas, en la base de datos, se refieren al tipo de modelado de datos, donde se guardan los datos recogidos por un programa. Su estructura general se asemaja a la vista general de un programa de hQja de cálculo.

De acuerdo a los requerimientos que tengamos para la creación de nuestra base de datos, debemos identificar adecuadamente los elementos de información y dividirlos en entidades (temas principales) como pueden ser las sucursales, los productos, los clientes, etc.

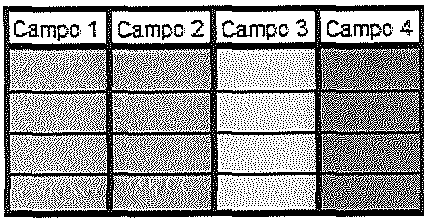
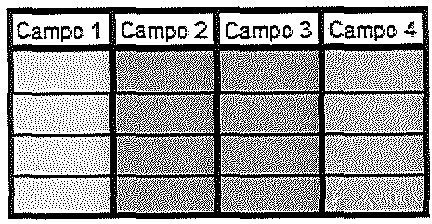
Para cada uno de los objetos identificados crearemos una tabla. Si en una base de datos los objetos principales son los empleados y los departamentos de la empresa entonces tendremos una tabla para cada uno de ellos. Si en otra base de datos los oetos principales son los libros, autores y editores entonces necesitaremos tres tablas en nuestra base de datos.

**Productos**



Campo 4

**Sucursales Clientes**

* + 1. **Definición de campos en las tablas configuradas, establecer claves primarias e índices de ordenación**

En el contexto de una tabla de base de datos relacional, una **columna** es un conjunto de valores de datos de un simple tipo particular, uno por cada fila de la tabla. Las columnas proporcionan la estructura según la cual se componen las filas.

El término **campo** es frecuentemente intercambiable con el de columna, aunque muchos consideran más correcto usar el término campo (o valor de campo) para referirse específicamente al simple elemento que existe en la intersección entre una fila y una columna.

Por ajemplo, una tabla que representa compañías pudo tener las siguientes columnas:

* + - * ID (Identificador entero, único a cada fila)
      * Nombre (texto)
      * Dirección 1 (texto)
      * Dirección 2 (texto)
      * Ciudad (Identificador entero, proviene de una tabla separada de ciudades, de la que cualquier información del estado o del país puede ser tomada)
      * Código postal (texto)

# •

* + - * + 1 ndustria (identificador entero, Proviene de una tabla separada de industrias)
        + Etc.

Cada fila proporcionaría un valor de los datos para cada columna y después sería entendida como solo simple valor de datos estructurado, en este caso representando a una compañía. Más formalmente, cada fila puede ser interpretada como una variable relacional, compuesta por un coajunto de tuplas, con cada tupla consistiendo en los dos elementos: el nombre de la columna relevante y el valor que esta fila proporciona para esa columna.

En una tabla no puede haber dos columnas con el mismo nombre pero ese nombre sí se puede utilizar en otra tabla.

Los campos que forman parte de una relación de tabla se denominan **claves.** Una clave consta por lo general en un campo, pero puede estar compuesta por más de un campo. Existen dos tipos de claves:

* **Clave principal.** Es una columna (o combinación de columnas) que permite identificar de forma inequívoca cada fila de la tabla, por lo que no pueden haber en una tabla dos filas con el mismo valor en la columna definida como clave principal. Por lo general, hay un número de identificación único, como un número de serie o un código que funciona como la clave principal. Por ajemplo, puede tener una tabla Clientes en la que cada cliente tenga un número de identificación de cliente único. El campo de identificación de cliente es la clave principal de la tabla Clientes. Cuando una clave principal contiene más de un campo, generalmente está compuesta de campos preexistentes que, en su conjunto, proporcionan valores únicos. Por ajemplo, puede usar una combinación de apellidos, nombre y fecha de nacimiento como la clave principal de una tabla para personas.
* **Clave externa.** Una tabla puede tener además una o más claves externas. Una clave externa contiene valores que se corresponden con valores de la clave principal de otra tabla. Por ajemplo, puede tener una tabla Pedidos en la que cada pedido tenga un número de identificación de cliente que se corresponda con un registro de una tabla Clientes. El campo de identificador de cliente es una clave externa de la tabla Pedidos.

### G

1. Clientes Pedid es



1 *\l* Id 1 Tipo de pa go

....

-

Apel lid os

!

Fecha d e pa go

! **Nombre Notas**

.

**Nombre completo** 1

1

! */2*

**Tipo impositivo** -

**Dirección de correo electrónico** • Estad o de impu estos

•

.. r

* + **Compañía** § 1 **Id de situaci ón**

•

1

i Cargo Id ele cllente

,,, =

i Teléf ono d el tra bajo

**Teléfono particular**

Id d e empleado

1

1• Id d e transportista -

1 **Teléfono móvil**

1 1 **Tipo *de* contenido**

1 **Número de fax** -1 Tipo de ard>ivo

!

¡

**Dirección**

* Ciu da d i **Datos adjuntos.Fi**

1 1 El Datos adju ntos

1

i

**Estado o provincia** •

-- •'

**Datos adjuntos.Fi** -

1 C Posta l ... 1 Datos a djuntos.Fi ....

Como ajemplo, en la siguiente imagen podemos ver una clave principal identificada con el incono de clave junto al nombre de campo (1) y una clave externa (2).

Una clave principal correcta debe tener varias características:

* Identifica inequívocamente cada fila.
* Nunca debe estar vacía ni ser nula (siempre debe contener un valor).
* Los valores que contiene no suelen cambiar (lo ideal es que no cambien) .

Por último un **índice de ordenación,** es una estructura de datos definida sobre una columna de tabla (o varias) y que permite localizar de forma rápida las filas de la tabla en base a su contenido en la columna indexada además de permitir recuperar las filas de la tabla ordenadas por esa misma columna. Funciona de forma parecida al índice de un libro donde tenemos el título del capítulo y la página donde empieza dicho capítulo, en un índice definido sobre una determinada columna tenemos el contenido de la columna y la posición de la fila que contiene dicho valor dentro de la tabla.

La definición de los índices de la base de datos es tarea del administrador de la base de datos, los administradores más experimentados pueden diseñar un buen coajunto de índices, pero esta tarea es muy complaja, consume mucho tiempo y

# •

está st.¡jeta a errores, incluso con cargas de trabajo y bases de datos con un grado de complajidad no excesivo.

* + 1. **Definición de las relaciones entre las tablas configuradas**

Cuando encontramos que existe una relación entre dos tablas debemos identificar el campo de relación. Por ajemplo, en una base de datos de productos y categorías existirá una relación entre las dos tablas porque una categoría puede tener varios productos asignados. Por lo tanto el campo con el código de la categoría será el campo que establezca la relación entre ambas tablas.

**Productos**

**categoría**

**Categorías**

Gategorio

Recordemos que son tres las relaciones que podemos encontrarnos entre las tablas; estas son las siguientes:

* + - * **Uno a muchos.** Cuando un registro de una tabla (tabla secundaria) sólo puede estar relacionado con un único registro de la otra tabla (tabla principal) y un registro de la otra tabla (tabla principal) puede tener más de un registro relacionado en la primera tabla (tabla secundaria).
      * **Muchos a muchos.** Cuando un registro de una tabla puede estar relacionado con más de un registro de la otra tabla y viceversa.
* **Uno a uno.** Cuando un registro de una tabla sólo puede estar relacionado con un único registro de la otra tabla y viceversa.

Muy importante, los campos que se utilicen para establecer la relación entre las dos tablas deben ser del mismo tipo (INTEGER, TEXTO, SMALL INTEGER, etcétera). Habitualmente se suelen utilizar tipos enteros (INTEGER) para este propósito, aunque nos valdría igualmente cualquier otro tipo siempre y cuando sea el mismo en las dos tablas. Además, debes tener en cuenta lo siguiente:

* La propiedad **Tamaflo** del campo debe ser igual en ambas tablas.
* Si el campo en la tabla primaria está definido como de Valor automático en la tabla secundaria debe estar definido como **INTEGER.**
* El campo común debe ser **Clave principal** en la tabla primaria.

# •

###### RECU ERDA

* Después de la fase de diseño de una base de datos, y una vez se ha realizado el paso a tablas del mismo, en necesario crear las tablas correspondientes dentro de la base de datos. Para cada campo de cada una de las tablas, es necesario determinar el tipo de datos que contiene; de esta forma ajustaremos el diseño de la base de datos, y conseguiremos un almacenamiento óptimo con la menor utilización de espacio.
* Los tipos de datos que puede tener un campo en MySQL son:

**numéricos, de fecha y hora y de cadena.**

* MySQL implementa extensiones espaciales siguiendo la especificación del Consorcio Open GIS (OGC), un consorcio internacional de más de

250 compañías, agencias y universidades que participan en el desarrollo de soluciones conceptuales públicamente disponibles y que pueden ser útiles para todo tipo de aplicaciones que manejan datos espaciales.

* En programación, una **constante** es un valor que no puede ser alterado durante la ajecución de un programa.
* En programación, una **variable** está formada por un espacio en el sistema de almacenaje (memoria principal de un ordenador) y un nombre simbólico (un identificador) que está asociado a dicho espacio. Ese espacio contiene una cantidad o información conocida o desconocida, es decir un valor.
* La mayoría de los ERPs adoptan una estructura modular que soporta los diferentes procesos de una empresa: el modulo de gestión financiera, el módulo de compras, el módulo de gestión de ventas, el módulo de recursos humanos, etc.
* La mayoría de las bases de datos la información se almacena en **TABLAS,** que ordenan los datos en filas y columnas. Las filas se denominan **REGISTROS** y las columnas se llaman **CAMPOS.**
* En el contexto de una tabla de base de datos relacional, una **columna** es un col'}junto de valores de datos de un simple tipo particular, uno por cada fila de la tabla. Las columnas proporcionan Ja estructura según Ja cual se componen las filas.
* Normalmente todas las tablas deben tener una **clave principal** definida. Una clave principal es una columna (o combinación de columnas) que permite identificar de forma inequívoca cada fila de la tabla, por lo que no pueden haber en una tabla dos filas con el mismo valor en la columna definida como clave principal.
* Por último un **índice de ordenación,** es una estructura de datos definida sobre una columna de tabla (o varias) y que permite localizar de forma rápida las filas de la tabla en base a su contenido en la columna indexada además de permitir recuperar las filas de la tabla ordenadas por esa misma columna.
* Los campos que se utilicen para establecer la relación entre las dos tablas deben ser del mismo tipo

Preguntas de Autoevaluación

1. **¿Cuál de los siguientes se considera un dato de tipo numérico de MySQL?**
2. DATE.
3. TINYINT.
4. CHAR.
5. **¿Cuál de los siguientes se considera un dato de tipo espacial de MySQL?**
6. BINARY.
7. TIME.

e) GEOMETRY.

1. **Completa la siguiente afirmación con la opción correcta:**

"En programación, una --------- es un valor que no puede ser alterado durante la ajecución de un programa".

1. Constante.
2. Variable.

e) Tupla.

1. **¿Cómo se denominan las filas de una tabla?**
2. Campos.
3. Registros.
4. Variables.
5. **¿Cuándo se da la relación de uno a muchos?**
6. Cuando un registro de una tabla puede estar relacionado con más de un registro de la otra tabla y viceversa.
7. Cuando un registro de una tabla (tabla secundaria) sólo puede estar relacionado con un único registro de la otra tabla (tabla principal) y un registro de la otra tabla (tabla principal) puede tener más de un registro relacionado en la primera tabla (tabla secundaria).
8. Cuando un registro de una tabla sólo puede estar relacionado con un único registro de la otra tabla y viceversa .

# •