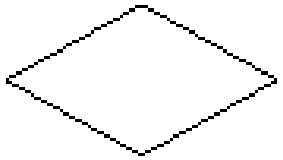
Modelo entidad-relación

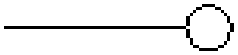
- Utilizado para el diseño conceptual de bases de datos.
- Permiten describir la realidad mediante un conjunto de representaciones gráficas.
- Se basa los conceptos de entidad, relación y atributo.

Símbolos

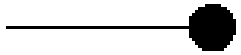
entidad



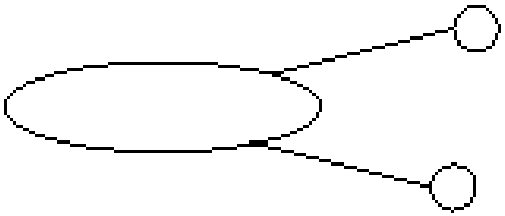
relación



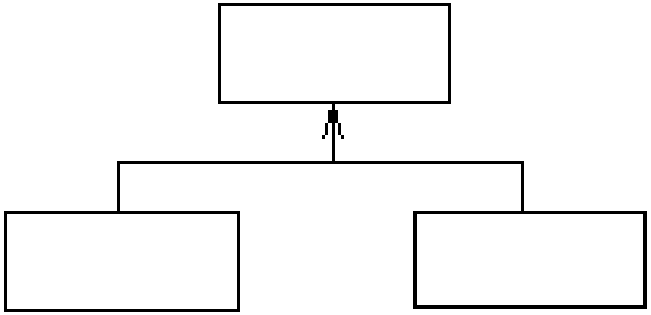
atributo



identificador



atributo compuesto



jerarquía de generalización

Entidad

Se define como entidad a cualquier tipo de objeto o concepto sobre el que se puede dar información :
(cosa, persona, concepto abstracto o suceso.)
Ejemplo: coches, casas, empleados, clientes, empresas, etc.

Entidad

- Se representan gráficamente mediante rectángulos
- Su nombre aparece en el interior.
- Un nombre de entidad sólo puede aparecer una vez en el esquema conceptual.

Tipos de entidades

Hay dos tipos de entidades:

Entidad débil: es una entidad cuya existencia depende de la existencia de otra entidad, no tiene atributo que la determine.

Entidad fuerte: es una entidad que tiene atributo determinante

Relación

- Es una correspondencia o asociación entre dos o más entidades.
- Cada relación tiene un nombre que describe su función.
- Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior.

Tipo de Relación

- Las entidades que están involucradas en una determinada relación se denominan *entidades participantes*.
- El número de participantes en una relación es lo que se denomina *grado* de la relación.
- **Relación binaria :** relación en la que participan dos entidades.
- **Relación múltiple:** relación en la que participan 3 entidades .

Cardinalidad

- La *cardinalidad* con la que una entidad participa en una relación especifica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ocurrencia de dicha entidad.

MAPEO

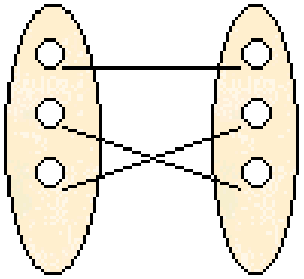
1.UNA A UNA: Una entidad de A puede asociarse únicamente con una entidad de B.
2.UNA A MUCHAS: Una entidad de a puede asociarse con cualquier cantidad de entidades de B.
3.MUCHAS A UNA: Cualquier cantidad de entidades de A puede asociarse con una entidad de B.
4.MUCHAS A MUCHAS: Cualquier cantidad de entidades de a puede asociarse con cualquier cantidad de **entidades en B.**

Ejemplo

UNA A UNA

Alumnos Tesis

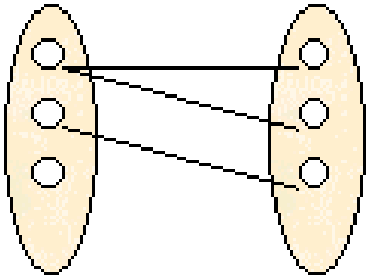
A B



UNA A MUCHAS

Carreras Alumnos

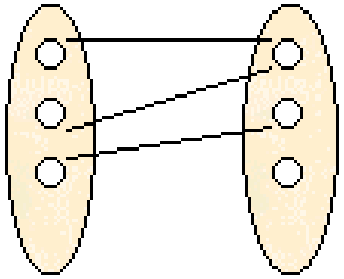
A B



MUCHAS A UNA

Alumnos Carreras

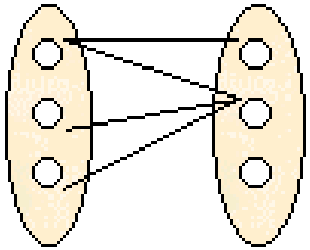
A B



MUCHAS A MUCHAS

Alumnos Materias

A B



Totalidad

- La participación de una entidad en una relación es *obligatoria (total)* si la existencia de cada una de sus ocurrencias requiere la existencia de, al menos, una ocurrencia de la otra entidad participante. Si no, la participación es *opcional (parcial).* Las reglas que definen la cardinalidad de las relaciones son las *reglas de negocio*.

Atributo

- Es una característica de interés o un hecho sobre una entidad o sobre una relación.
- Los atributos representan las propiedades básicas de las entidades y de las relaciones.
- Toda la información extensiva es portada por los atributos.
- Gráficamente, en el modelo cuelgan de las entidades o relaciones a las que pertenecen.

Atributo

- Cada atributo tiene un conjunto de valores asociados denominado *dominio*.
- El dominio define todos los valores posibles que puede tomar un atributo.
- Puede haber varios atributos definidos sobre un mismo dominio

Tipo de atributo

- Los atributos pueden ser simples o compuestos.
- ***Atributo simple*** : atributo que tiene un solo componente, no se puede dividir en partes más pequeñas que tengan un significado propio.
- ***Atributo compuesto*** : atributo con varios componentes, cada uno con un significado por sí mismo.

Tipo de atributos

- Un grupo de atributos se representa mediante un atributo compuesto cuando tienen afinidad en cuanto a su significado, o en cuanto a su uso.
- Un atributo compuesto se representa gráficamente mediante un óvalo.

Tipo de atributos

- Los atributos también pueden clasificarse en monovalentes o polivalentes.
- Un ***atributo monovalente*** es aquel que tiene un solo valor para cada ocurrencia de la entidad o relación a la que pertenece.
- Un ***atributo polivalente*** es aquel que tiene varios valores para cada ocurrencia de la entidad o relación a la que pertenece.
- A estos atributos también se les denomina ***multivaluados***,

identificador

- Un identificador de una entidad es un atributo o conjunto de atributos que determina de modo único cada ocurrencia de esa entidad.
- Un identificador debe cumplir dos condiciones:
  1. No pueden existir dos ocurrencias de la entidad con el mismo valor del identificador.
  2. Si se omite cualquier atributo del identificador, la condición anterior deja de cumplirse.
- Toda entidad fuerte tiene al menos un atributo determinante

Diseño de una Base de Datos

- El primer paso en el diseño de una base de datos es la producción del esquema conceptual. Normalmente, se construyen varios esquemas conceptuales, cada uno para representar las distintas visiones que los usuarios tienen de la información. Cada una de estas visiones suelen corresponder a las diferentes áreas funcionales de la empresa como, por ejemplo, producción, ventas, recursos humanos, etc.

Diseño de una Base de Datos

- Estas visiones de la información, denominadas ***vistas***, se pueden identificar de varias formas. Una opción consiste en examinar los diagramas de flujo de datos, que se pueden haber producido previamente, para identificar cada una de las áreas funcionales. La otra opción consiste en entrevistar a los usuarios, examinar los procedimientos, los informes y los formularios, y también observar el funcionamiento de la empresa.

Diseño de una Base de Datos

- A los esquemas conceptuales correspondientes a cada vista de usuario se les denomina ***esquemas conceptuales locales***. Cada uno de estos esquemas se compone de entidades, relaciones, atributos, dominios de atributos e identificadores. El esquema conceptual también tendrá una documentación, que se irá produciendo durante su desarrollo. Las tareas a realizar en el diseño conceptual son las siguientes:

Diseño Conceptual

1. Identificar las entidades.
2. Identificar las relaciones.
3. Identificar los atributos y asociarlos a entidades y relaciones.
4. Determinar los dominios de los atributos.
5. Determinar los identificadores.
6. Determinar las jerarquías de generalización (si las hay).
7. Dibujar el diagrama entidad-relación.
8. Revisar el esquema conceptual local con el usuario.

Problema a resolver

- La asociación "Amigos de la Fiesta" desea recoger en una base de datos toda la información acerca de las corridas de toros que se celebran en España y de todos los datos relacionados con ellas.

Requerimientos

- Se desea tener información acerca de cada corrida, identificada conjuntamente por un número de orden, la feria en la que se celebra y el año de celebración (por ejemplo: orden = 2, feria = San Isidro, Año = 1999).

Requerimientos

- En una determinada corrida actúan una serie de toreros (mínimo 1 y máximo 3) de los que desea guardar su DNI, nombre, apodo y fecha en que tomó la alternativa (fecha en la que se convirtió en matador de toros). Además se desea saber quien fue el torero que le dio la alternativa (padrino) en su día (un torero puede dar la alternativa a varios toreros o a ninguno).

Requerimiento

- En cada corrida un torero obtiene una serie de premios (cuántas orejas, cuántos rabos y si salió por la puerta grande o no) de los que se desea mantener información.

Requerimiento

- Cada torero puede tener un apoderado del que es protegido. A su vez, un apoderado lo puede ser de varios toreros. De él se desea saber su DNI, nombre, dirección y teléfono.

Requerimiento

- Una corrida se celebra en una plaza de toros de la que se desea saber su nombre que se supone único, localidad, dirección y aforo. En una misma plaza se pueden celebrar varias corridas de toros.

Requerimiento

- En cada corrida son estoqueados al menos 6 toros. Cada toro viene identificado por el código de la ganadería a la que pertenece, el año en que nació y un número de orden. Además se desea mantener información acerca de su nombre y color así como el orden en que fue toreado.

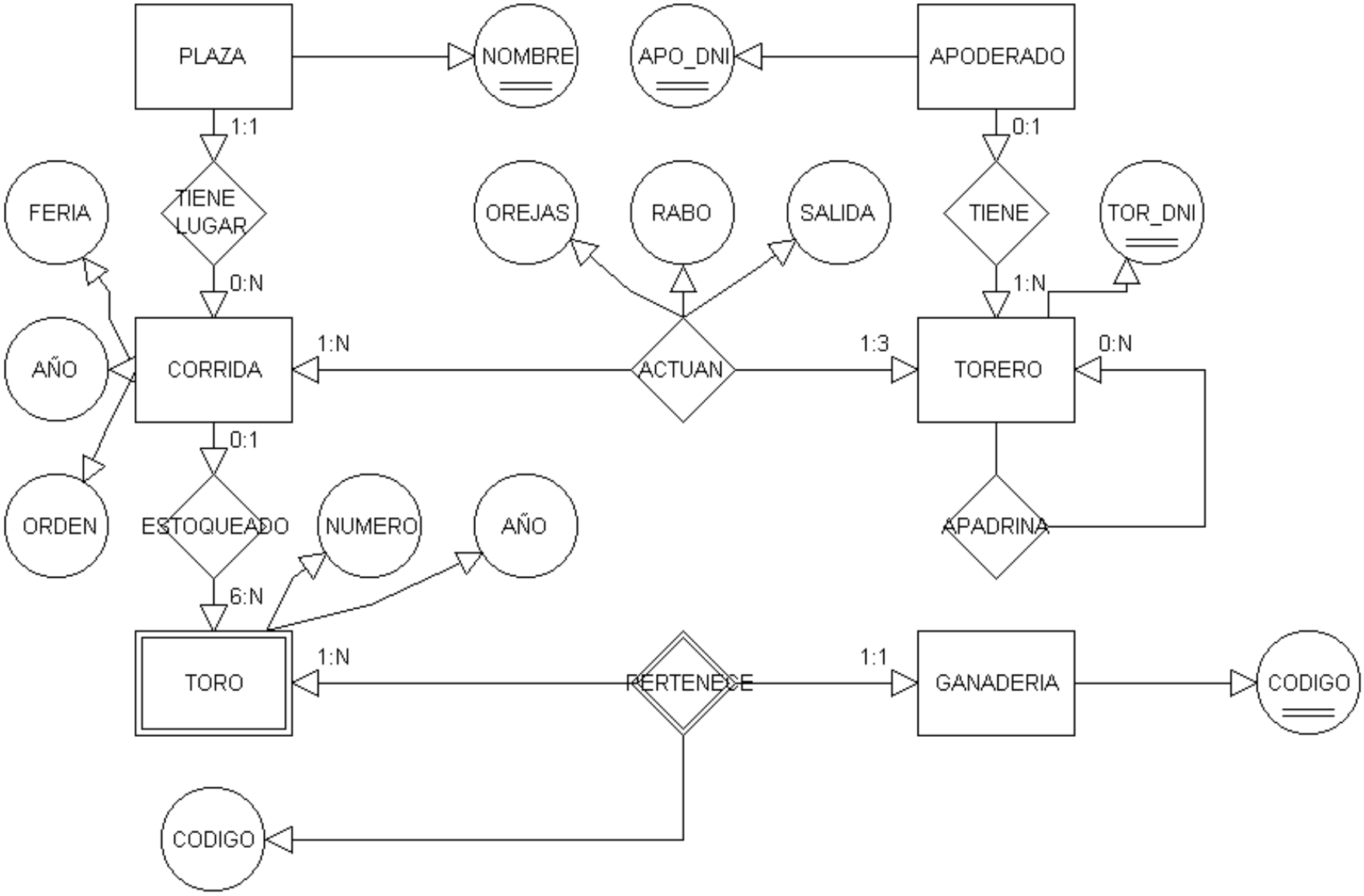
Requerimiento

- Cada toro pertenece a una ganadería determinada. De cada ganadería se pretende saber su código, localidad y antigüedad (fecha de creación).

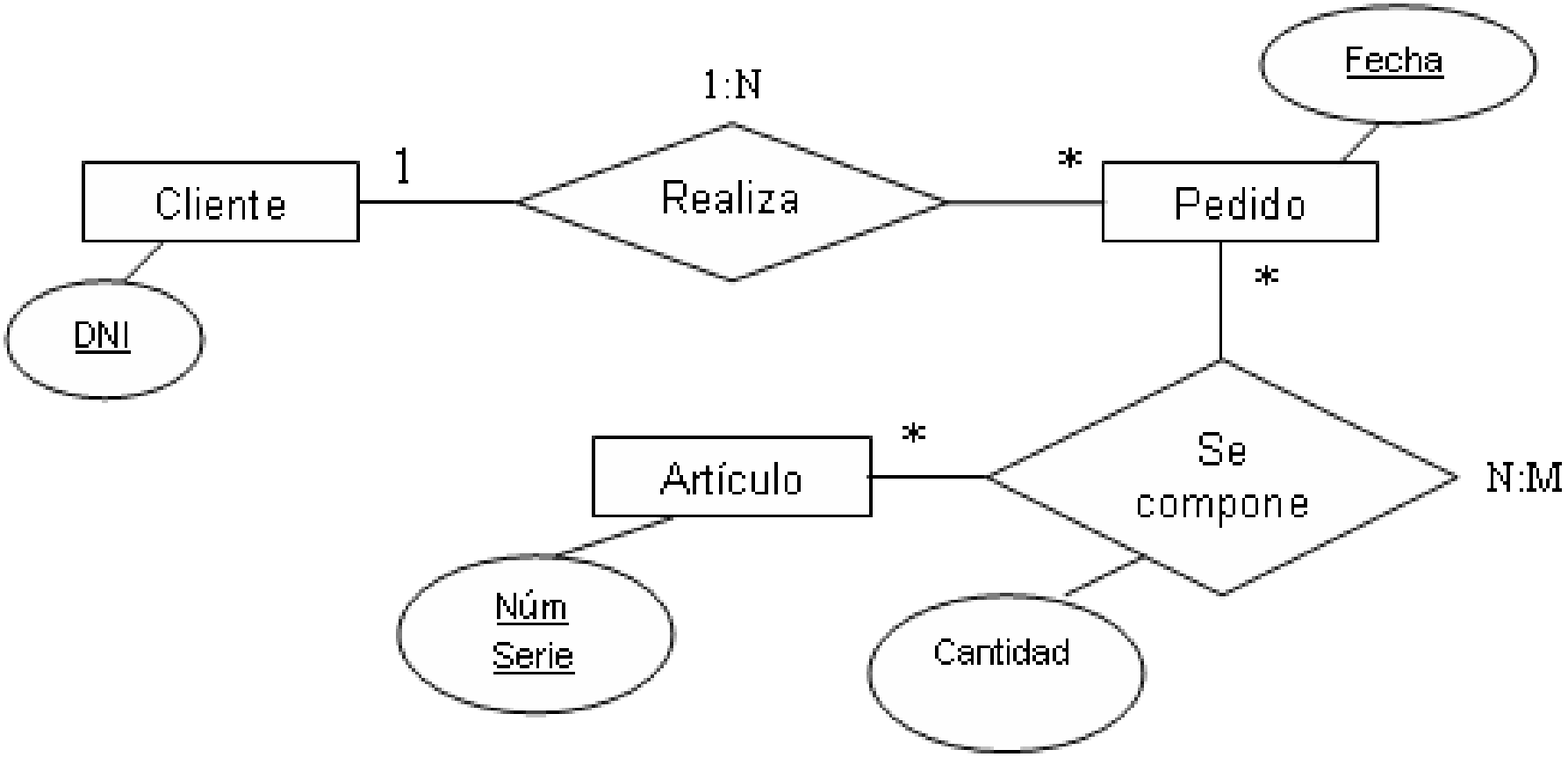
Comentarios

- A falta de determinados atributos, el modelo entidad - relación de este enunciado sería el siguiente:

Solución



Para Discutir
Pedido



Modelado Entidad-Relación

- Es ***una*** *técnica* para el modelado de datos que consiste en los siguientes pasos:
  1. Se parte de una descripción textual del problema o sistema de información a automatizar (los requisitos).
  2. Se hace una lista de los sustantivos y verbos que aparecen.
  3. Los sustantivos son posibles entidades o atributos.
  4. Los verbos son posibles relaciones.
  5. Analizando las frases se determina la cardinalidad de las relaciones y otros detalles.
  6. Se elabora el diagrama (o diagramas) entidad-relación.
  7. Se completa el modelo con listas de atributos y una descripción de otras restricciones que no se pueden reflejar en el diagrama.

Modelo implementable

- Son necesarias otras técnicas para lograr un modelo directamente implementable
  - Transformación de relaciones múltiples en binarias.
  - Normalización de relaciones (algunas relaciones pueden transformarse en atributos y viceversa).
  - Conversión en tablas (en caso de utilizar una base de datos relacional).
- Etc.

Modelo relacional

- es un modelo de datos basado en la lógica de predicado y en la teoría de conjuntos.
- Este modelo considera la base de datos como una colección de relaciones.
- De manera simple, una relación representa una tabla donde cada fila representa una colección de valores que describen una entidad.
- Cada fila se denomina tupla o registro y cada columna campo.

Ventajas del modelo

- Garantiza herramientas para evitar la duplicidad de registros, a través de campos claves o llaves.

1. Garantiza la integridad referencial: Así al eliminar un registro elimina todos los registros relacionados dependientes.
2. Favorece la normalización por ser más comprensible y aplicable.

Modelo Relacional

- Se basa en describir la información usando tablas. Estas tablas se intentan estructurar de forma que cumplan unos formatos

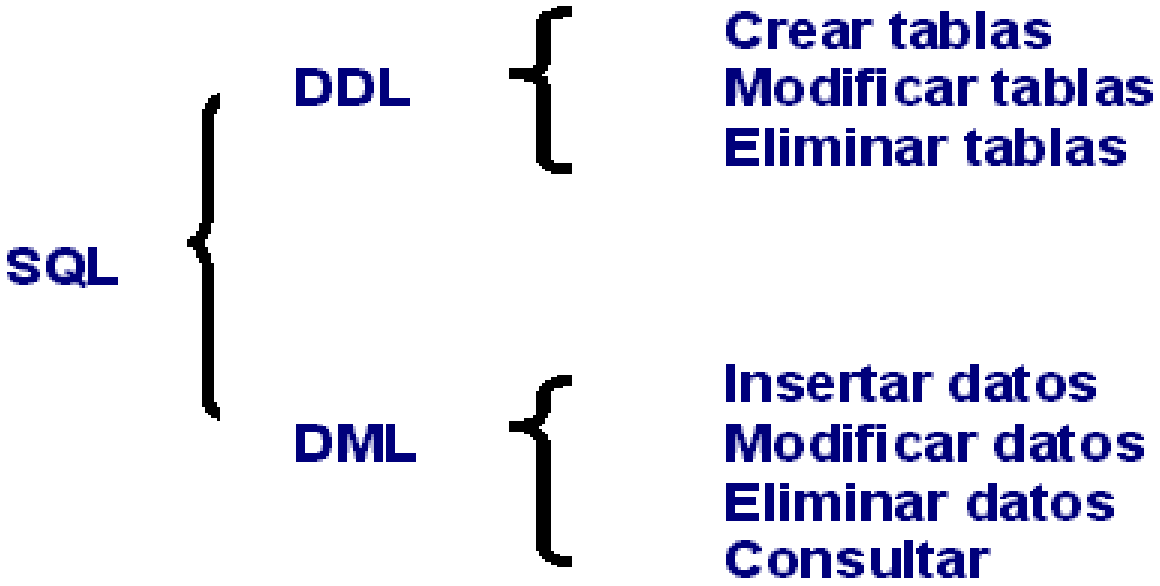
Formas Normales.

- Cuanto más alta la forma normal, mas estrictos son los criterios que cumple la tabla y más fácil resulta tratarla.
- Primera FN: No hay campos múltiples
- Segunda FN: No hay dependencias parciales
- Tercera FN: No hay dependencias transitivas

Lenguajes

Existen diversos lenguajes mediante los cuales se puede construir un modelo relacional.

SQL(structured query languaje), lenguaje de consulta estándar para la manipulación y definición de los datos.



DDL

- **Crear Tablas**
- CREATE TABLE<nombre de la tabla>
  (
  <campo1> (<tipo>[,NO NULL]),
  <campo2> (<tipo>[,NO NULL]),...
)

DDL

- **Tipos Validos**
- CHAR (<LONG>)[VAR]
- FLOAT
- INTEGER
- SMALLINT

DDL

- **Crear Indices** CREATE [UNIQUE] INDEX <nom.indice> ON < nom>tabla>
  (
  <nomcampo1> [ASC/DES],
  <nomcampo2> [ASC/DES],...
)

DDL

- **Modificar Tablas (expandirlas)**
- EXPAND TABLE <nom.tabla>
- ADD FIELD <nom.campo> (<TIPO>[NO NULL])
-

DDL

- **Eliminar Tablas**
- DROP TABLE <nom.tabla>
- **Borrar Indices**
- DROP <nom>indice>

DDL Ejemplo crear tabla

- **Tablas**
- CREATE TABLE persona
  (
  nombre(CHAR(40) VAR, NO NULL),
  edad(SMALLINT, NONULL),
  estatura(FLOAT, NO NULL),
  telefono(CHAR(7))
)

DDL Ejemplo crear indice a la tabla

Crear Índices

- **Por nombre**
- CREATE INDEX ind_nom ON persona
  (
  nombre
)

DDL

Crear Índices

- **Por estatura sin llaves repetidas, descendente**
  CREATE UNIQUE INDEX ind_est ON persona
  (
  estatura desc
)
- **Por edad(primero mas jovenes); edades repetidas, poe estatura(primero la mas alta)**
  CREATE INDEX ind_ed_est ON persona
  (
  edad,estatura desc
)

DML

Modificaciones

- EXPAND TABLE persona
- ADD FIELD direccion(CHAR(30), VAR, NO NUL

DML

Modificar datos

- UPDATE <nom.tabla>
  SET <campo1> = <campo1>,
  <campo2> = <campo2>,...
  [WHERE <condicion>]

DML

Eliminar datos

- DELETE<nom.tabla>
  [WHERE <condicion>]

DML

Insertar datos

- INSERT INTO <nom.tabla> [(<campo1>,<campo2>...):] < <valor1>,<valor2>...>

DML

Consultas en una tabla

- SELECT [UNIQUE] <lista de campos/*>
  FROM <nom>tabla>
  [WHERE <condicion>]
  [ORDER BY <campo> [asc/des]]

FUNCIONES

- **Funciones integradas en select/where**
  COUNT(*) conteo
  SUM(<campo>) total(acumulador)
  AVG(<campo>) promedio
  MAX(<campo>) maximo
  MIN(<campo>) minimo

ESTRUCTURA DE LA TABLA

CREATE TABLE persona
(nombre(CHAR(40) VAR, NO NULL),
edad(SMALLINT, NONULL),
estatura(FLOAT, NO NULL),
telefono(CHAR(7))

- **Ejemplos:**
- Agregar una persona en la tabla
  - **Insert into persona**
  - **< 'juan',15,1,75,'2-15-15','forjadores'>**
- Íncrementar edad de todas las tuplas
  - **Update persona**
  - **Set edad=edad+1**

Operadores validos

=	igual	operacionales
]=	diferente	
<	menor	
<=	menor o igual	
>	mayor	
>=	mayor o igual	

NOT	logicos
AND	
OR	

Base ejemplo

Clientes

Nc	Nombre	Domicilio	Estado
320	Juan	Forjadores	Infantil
145	Pedro	Catolica	Adulto
215	Jesus	5 mayo	Adolescente
328	Luis	independencia	Adulto
416	Jose	morelos	Adolescente

Rentas

Nc	clave	Fecha	Dias
320	A716	7/10/97	3
320	B942	7/10/97	3
215	A320	7/10/97	2
328	C518	8/10/97	1
416	B415	8/10/97	1

Videos

Clave	Titulo	Clasificacion	Costo
A320	La roca	B	12.00
B415	Tornado	B	12.00
A716	Batman	A	15.00
B524	Contra cara	B	15.00
C318	Seven	C	12.00

Construcción de la base ejemplo

- Create table clientes
  (
  nc(integer,NO NULL),
  nombre(char(20)VAR,NO NULL)
  domicilio(char(40)VAR,NO NULL)
  estado(char(15)VAR,NO NULL)
)
- insert into clientes
  <320,'juan','forjadores','infantil'>
  <145,'pedro','catolica','adulto'>

Construcción de la base ejemplo

- create tabla videos
  (
  clave(char(4),NO NULL),
  titulo(char(20)VAR,NO NULL),
  clasificacion(char(1)VAR,NO NULL),
  costo(float,NO NULL)
)
- insert into videos
  <'A320','la roca','b',12.00>
  <'b415','tornado','b',12.00>

Construcción de la base ejemplo

- Create table renta
  (
  nc(integer,NO NULL),
  clave(char(4),NO NULL),
  fecha(char(8),NO NULL),
  dias(smallint,NO NULL)
)
- Insert into renta
  <320,'A716','7/10/97',3>
  <320','b716','7/10/97'.3>

Consultas sobre la base ejemplo

- Muestre el nombre y estado de los clientes cuyo numero de credencial es mayor a 100

- select nombre estado
  form clientes
  where nc>100

Consultas sobre la base ejemplo

- Se desea conocer la cantidad de clientes de cada estado
- select estado, count(*)
  from clientes
  grop by estado

Consultas sobre la base ejemplo

3. Se desea consultar los nombres de las peliculas que cuestan menos de 15.00 que no son infantiles.

- Select titulo
  From videos
  Where (costo < 15.00 and clasificacion ù = "a")

Consultas en varias tablas

- select [unique] <lista campos /*>
  from <lista de tablas>
  where <condicion>

Se desea conocer la lista de los distintos títulos rentados el 8 oct 97
Select unique video.tiulo
From renta,videos
Where (renta.fecha = '8/oct/97' and renta.clave = video.clave)

Consultas y subconsultas

- Se desea mostrar los titulos de las peliculas que han sido rentadas por lo mas de dos dias
- Select videos.titulo
  From videos,renta
  Where (videos.clave=renta.clave) and (renta.dias>2)
- En subconsultas
- select videos.titulo
  form videos
  where clave in (select clave
  from rentas
  where renta.dias >2)

Consultas, subconsultas

- se desea conocer el domicilio de los clientes que han rentado peliculas para adolescentes.
- Select clientes.domicilio
  Form clientes
  Where nc in (select nc
  From renta
  Where clave in (select clave
  From videos
  Where clasificacion='b'
  ))

En uniones

Una union permite consultar los resultados de dos o mas tablas en una sola salida; cuando los resultados de las tablas son semejantes (muestran la misma informacion) se suprimen las salidas redundantes, operando asi como una union de conjuntos.

Select <lista de campos/*>
From <tabla1>
Where<condicin1>

Union

Select <lista campos2/*>
From<tabla2>
Where<condicion2>

- se desea obtener una lista de clientes y de películas se desea incorporar solo a los clientes infantiles y a las películas que pueden ser rentadas por estos.

```
Select nombre
   From clientes
   Where estado='infantil'
Union
Select titulo
   From videos
   Where clasificacion = 'a'
```

MODIFICACIÓN DE LA BASE DE DATOS.

Eliminación.

Una solicitud de eliminación se expresa de forma muy parecida a una consulta. Sin embargo, en vez de presentar tuplas al usuario, quitamos las tuplas seleccionadas de la base de datos. Sólo podemos eliminar tuplas completas; no podemos eliminar únicamente valores de determinados atributos.

MODIFICACIÓN DE LA BASE DE DATOS.

Inserción.

Para insertar datos en una relación, bien especificamos la tupla que sa va a insertar o escribimos una consulta cuyo resultado sea un conjunto de tuplas que se va a insertar.

MODIFICACIÓN DE LA BASE DE DATOS.

Actualización.

En ciertas ocasiones podemos querer cambiar un valor en una tupla sin cambiar todos los valores de la tupla. Si hacemos estos cambios usando eliminación e inserción, es posible que no podamos conservar los valores que no queremos cambiar. En lugar de ello, usamos el operador actualización

VISTAS

- No existen físicamente
- Se forman mediante la selección y/o filtrado de los componentes de otras tablas
- Uuna vista puede ser definida en base a una lista previa.

VISTAS

- **Formato de definición de vistas**

- DEFINE VIEW<nombre vista> [(identif_campo1, identif_campo2,...)] AS<operación de consulta>

Ejemplo:

- Crear una vista para obtener los nombres y domicilios de los clientes adultos, es deseable el establecimiento de las cabeceras nombre del cliente, domicilio del cliente.
- DEFINE VIEW cliente_adulto
  (nombre del cliente, domicilio del cliente)
  As(select nombre,domicilio
  From clientes
  Where estado = 'adulto')

- Como puede verse, la especificación de los identificadores es opcional; si estos se omiten se asumirán los nombres de los campos extraidos en la consulta.
- La operación de consulta permite todos los formatos validos de consulta en SQL con excepción del groop by.
- Cuando una vista es definida en base a otra, se se dice que es dependiente de esta por lo tanto, se suprimirá automáticamente la vista dependiente si se suprime la vista original.

Eliminación de vistas

- Drop view <nombre tabla>

•Ejemplo: supongamos que se desea crear una vista dependiente de la vista cliente adulto que contenga solamente a los clientes que viven sobre forjadores. Se desean los mismos campos y la vista será llamada cliente_adulto_forjadores.

Eliminación de vistas

- Define view cliente_adulto_forjadores
  AS (select *
  From cliente_adulto
  Where domicilio_del_cliente like 'forjadores%')
  ~
  ~
  drop view cliente_adulto
- (se eliminara tambien la vista cliente_adulto_forjadores, puesto que es dependiente de cliente_adulto.)

Eliminación de vistas

- La eliminación de una tabla provoca también la eliminación automática de todas las listas que se hayan definido haciendo referencia a ella.