Sistema **G**estor de **B**ases de **D**atos (**SGBD**): Se define como una colección de datos relacionados entre si, estructurados y organizados, y un conjunto de programas que acceden y gestionan esos datos, la colección de datos se llama **B**ase de **D**atos (**BD**).

Servicios SGBD

- Creación y definición BD.
- Manipulación de los Datos.
- Acceso controlado a Datos.
- Mantener integridad y consistencia.
- Acceso compartido a la BD.
- Mecanismos de respaldo y recuperación.

Arquitectura SGBD

- Usuarios
- Nivel Lógico
  - Nivel externo o de Visión
  - Nivel conceptual
- Nivel Físico
  - Nivel interno o físico

Tipos de independencia de datos (Arquitectura SGBD)

- Independencia Lógica: Capacidad de modificar el esquema conceptual sin alterar los esquemas externos ni programas de aplicación.
- Independencia Física: Capacidad de modificar el esquema interno sin tener que alterar ni el esquema conceptual, ni los externos.

Componentes de los SGBD

Lenguajes SGBD

- Lenguaje de definición de datos (**LDD** o **DDL**): Para especificar el esquema de la BD, las vistas de usuarios y estructuras de almacenamiento.
- Lenguaje de manipulación de datos (**LMD** o **DML**): Para leer y actualizar los datos de la BD.

Diccionario de Datos

- Es el lugar donde se deposita información acerca de todos los datos que forman la BD, describe la BD y los objetos que la forman.
- Proporciona información de la Estructura lógica y física de la BD, Las definiciones de todos los objetos de la BD (tablas, vistas indices, procedimientos, etc.), Información de las restricciones de integridad, Privilegios y roles otorgados a los usuarios, Auditoria de información.
- Cumple las siguientes características: Soportar descripciones de los modelos conceptual, lógico, interno y externo, estar integrado dentro del SGBD, Apoyar la transferencia eficiente de la información al SGBD, Estar almacenado en un medio con acceso directo para la fácil recuperación de información.

Seguridad e integridad de los Datos

- Garantiza la protección de datos contra accesos no autorizados, solo usuarios autorizados acceden a la BD.
- Los SGBD ofrecen mecanismos para implementar restricciones de integridad en la BD.
- Proporciona herramientas y mecanismos para planificación y realización de copias de seguridad y restauración.
- Debe ser capaz de recuperar la BD en caso de que algún suceso la dañe.
- Debe asegurar el acceso concurrente y ofrecer mecanismos para conservar la consistencia de los datos en caso de que varios usuarios utilicen la BD a la vez.

El administrador de la BD

En los sistemas de gestión de bases de datos existen diferentes categorías de usuarios, según los privilegios y permisos que tienen sobre los objetos que forman la BD:

- Los usuarios de la categoría DBA (Database administrator).
- Los usuarios de la categoría RESOURCE.
- Los usuarios del tipo CONNECT.

Administrador de la Base de datos (DBA) TAREAS

- Instalar el SGBD en el sistema informático.
- Crear las BBDD que se vayan a gestionar.
- Crear y mantener el esquema de la BD.
- Arrancar y parar SGBD y cargar las BBDD con las que se va a trabajar.
- Colaborar con el administrador del S.O. En tareas de ubicación, dimensionado y control.
- Colaborar en las tareas de formación de usuarios.
- Establecer standares de uso, políticas de acceso y protocolos de trabajo diario.
- Suministrar la información necesaria sobre la BD a equipos de análisis y programación.
- Efectuar tareas de explotación (resolución de dudas, controlar en tiempo real los accesos, anomalías etc., copias de seguridad, restauración, auditorias, ajustar y optimizar la BD).

Modelos de Datos

- Modelos lógicos basados en objetos (nivel conceptual y externo)
  - Modelo Entidad-Relación
  - Modelo orientado a objetos
- Modelos lógicos basados en registros (nivel conceptual y físico)
  - Modelo Relacional
  - Modelo de Red
  - Modelo Jerárquico
- Modelos físicos de datos (nivel físico)
  - Modelo unificador
  - Modelo de memoria de elementos

Modelo Entidad-Rlacion

- Entidad : Es un objeto del mundo real por ejemplo Alumno, Cientes.
- Conjunto de entidades : Es un grupo de entidades del mismo tipo.
- Entidad Fuerte : El la Entidad que no depende de otra Entidad para existir, por ejemplo Alumno.
- Entidad Débil : Es la que necesita de otra para existir, por ejemplo Notas.
- Atributos o Campos : La información que describe las Entidades, Nº matricula, Nombre, dirección, etc.
- Dominio : Es el conjunto de valores permitidos para cada Atributo.
- Identificador o Superclave : El conjunto de atributos que identifican de forma única cada entidad, ejemplo DNI.
- Clave Candidata : Es cada una de las superclaves formada por el mínimo numero de campos.
- Clave Primaria o Principal : Es la clave candidata seleccionada y que no varia en el tiempo, se subraya.
- Clave Ajena o Foránea : Es cuando esta en una entidad y es clave principal de otra entidad.
- Relación : Es la asociación de varias Entidades y tiene nombre de verbo.

Grados de Relación

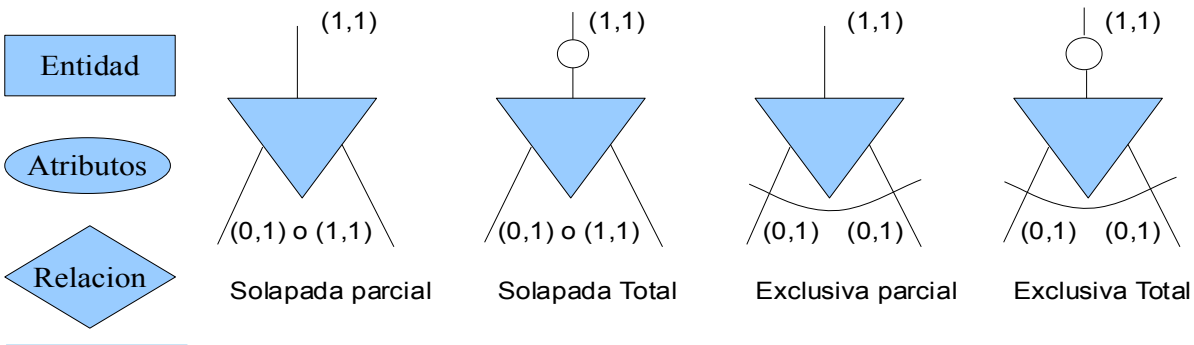
- Grado 1 o Reflexivo : La Relación de la Entidad con sigo misma.
- Grado 2 o Binaria : Es la Relación entre dos Entidades.
- Grado 3 o Ternaria : Es la Relación en la que participan 3 entidades.

Cardinalidad de una Relación

- 1:1, Uno a Uno : A cada elemento de la Entidad 1 le corresponde un elemento de la Entidad 2.
- 1:N, Uno a Muchos : A cada elemento de la Entidad 1 le corresponde uno o mas elementos de la Entidad 2.
- M:N, Muchos a Muchos : A cada elemento de la Entidad 1 le corresponde uno o mas elementos de la Entidad 2, y a cada elemento de la Entidad 2 le corresponde uno o mas elementos de la Entidad 1.

Tipos Jerarquías

- Solapada Parcial : Un empleado puede ser Ingeniero y Arquitecto y ademas otra categoría no incluida.
- Solapada Total : Un empleado puede ser Ingeniero y Arquitecto pero no otra categoría que no este incluida.
- Exclusiva Parcial : Un empleado puede ser Ingeniero o Arquitecto y ademas otra categoría no incluida.
- Exclusiva Total : Un empleado puede ser Ingeniero o Arquitecto, pero solo una de las dos.



Union Entidades y Relaciones

El modelo Relacional desarrollado por E.G. Codd para IBM, a finales de los años sesenta. Propone un modelo basado en la teoría matemática de las relaciones, con el objetivo de mantener la independencia de la estructura lógica respecto al modo de almacenamiento y otras características de tipo físico.

Modelo relacional Objetivos

- Independencia física de los datos. El modo de almacenamiento de los datos no debe influir en su manipulación lógica.
- Independencia lógica de los datos. Los cambios que se realicen en los objetos de la base de datos no deben repercutir en los programas y usuarios que accedan a la misma.
- Flexibilidad. Para presentar a los usuarios los datos de la forma mas adecuada a la aplicación que utilicen.
- Uniformidad. en la presentación de las estructuras lógicas de los datos, que son tablas, lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- Sencillez. Pues las características anteriores, así como un lenguaje de usuario sencillos, hacen que este modelo sea fácil de comprender y utilizar por el usuario.

12 Reglas de Codd

- Regla de información
- Regla de acceso garantizado
- Tratamiento sistemático de valores nulos
- Catalogo en linea dinamico basado en el modelo relacional
- Regla de sublenguaje completo de datos
- Regla de actualización de vista
- Inserción, actualización y supresión de alto nivel
- Independencia física de los datos
- Independencia lógica de los datos
- Independencia de la integridad
- Independencia de la distribución
- Regla de no subersion

Estructura Modelo Relacional

- Dominios. Se define dominio como el conjunto finito de valores homogéneos (todos del mismo tipo) y atómicos (son indivisibles) que puede tomar cada atributo, Dominios generales son los que los valores están comprendidos entre un máximo y un mínimo y Dominios restringidos los que pertenecen a un conjunto de valores específicos (sexo H o M).
- Atributos. Se define atributo como el papel o rol que desempeña un dominio en una relación, aporta un significado semántico a un dominio.
- Relaciones. La relación se representa mediante una tabla con filas y columnas, se utilizan para almacenar información sobre los objetos que se representan en la base de datos y esta formada por Atributos (columnas) y Tuplas (Filas), de las tablas se derivan los conceptos de Cardinalidad (numero de filas de la tabla), Grado (numero de columnas de la tabla), Valor (la intersección entre una fila y una columna) y Valor Null (ausencia de información).
- Claves. En una relación no hay tuplas repetidas, se identifican de un modo único mediante los valores de sus atributos, la clave debe cumplir dos requisitos: Identificación univoca (en cada fila de la tabla el valor de la clave ha de identificarla de forma univoca), No redundancia (no se puede descartar ningún atributo de la clave para identificar la fila).
- Esquema de la base de datos. Una base de datos relacional es un conjunto de relaciones normalizadas, para representar un esquema de una base de datos se debe dar el nombre de sus relaciones, los atributos de estas, los dominios sobre los que se definen estos atributos, las claves primarias y las claves ajenas.

Elementos característicos relación

- No admiten filas duplicadas
- Las filas y columnas no están ordenadas
- La tabla es plana. En el cruce de una fila y una columna solo puede haber un valor, no se admiten atributos multivaluados.

Tipos de Claves

- Clave candidata de una relación: El conjunto de atributos que identifican univoca y minimamente cada tupla de la relación.
- Clave primaria o principal (PK): Clave candidata escogida para identificar las tuplas de la relación, no tiene valores nulos.
- Clave alternativa: Claves candidatas no escogidas como claves primarias.
- Clave ajena de una relación R1 (FK): Conjunto de atributos cuyos valores coinciden con los valores de la clave primaria de de otra relación R2.

Transformación esquema E-R a esquema Relacional

- Toda Entidad se transforma en una Tabla.
- Todo Atributo se transforma en columna dentro de una tabla.
- El identificador único de la Entidad se convierte en clave primaria.
- Toda relación N:M se transforma en una tabla que tendrá como clave primaria la concatenación de los atributos clave de las entidades que asocia.

Transformación Relaciones 1:N

- Transformar la relación en una tabla. Se hace como si se tratara de una relación N:M, esta solución se realiza cuando se prevé que en un futuro la relación se convierta en N:M y cuando la relación tiene atributos propios, también se crea una tabla si la cardinalidad es opcional, es decir (0,1) y (0,M), la clave de esta tabla es de la entidad del lado muchos.
- Propagar la clave. Este caso se aplica cuando la cardinalidad es obligatoria, es decir cuando tenemos cardinalidad (1,1) y (0,M o (1,M), se propaga el atributo de cardinalidad 1 a la de cardinalidad N y también los atributos propios de la relación.

Transformación Relaciones 1:1

- Transformar la relación en una tabla. Si las entidades tienen cardinalidades (0,1), la relación se convierte en una tabla.
- Propagar la clave. Si una de las relaciones posee cardinalidad (0,1) y la otra (1,1), conviene propagar la clave de la de la entidad con cardinalidad (1,1) a la de cardinalidad (0,1), si ambas tienen cardinalidad (1,1) se puede propagar la clave de cualquiera de ellas a la otra.

Transformación Relaciones Reflexivas o recursivas

- Si la relación es 1:1 , la clave de la entidad se repite (una como clave primaria y otra como clave ajena (a la ajena se le cambia el nombre, al no poder tener 2 claves de nombre igual)
- Si la relación es 1:M habrá 2 casos, si la entidad muchos es obligatoria se procede como en el caso 1:1, si no es obligatoria se crea una nueva tabla con la clave del lado muchos y ademas se propaga la clave a la nueva tabla como clave ajena.
- Si es N:M: Se trata igual que en las relaciones binarias, la tabla resultante tendrá dos veces la clave primaria de la entidad del lado muchos mas los atributos de la relación si los hubiera.

Transformación Jerarquías

- Al no haber mecanismos para la representación de las jerarquías estas se deben eliminar, hay varias formas:
- Integrar todas las entidades en una única eliminando a los subtipos.
- Eliminación del supertipo: todos los atributos del supertipo pasan a los subtipos y la clave genérica pasa a cada uno de los subtipos, solo puede aplicarse a jerarquias totales y exclusivas.
- Insertar una relación 1:1 entre el supertipo y cada uno de los subtipos: Los atributos se mantienen y cada subtipo se identificara con la clave ajena del supertipo, los subtipos mantendrán si la relación es exclusiva la cardinalidad mínima 0 y si es solapada 0 o 1.

Transformación Relaciones N-arias

- En este tipo de relaciones se agrupan 3 o mas entidades, cada entidad se convierte en una tabla y la relación en otra, las claves de cada entidad pasaran a la tabla de la relación y se tendrá en cuenta.
- Si la relación es M:M:M es decir todas la entidades participan en cardinalidad máxima M, la clave de la tabla resultante es la unión de las claves de cada entidad mas los atributos de la relación si los tuviera.
- Si la relación es 1:M:M es decir una de las entidades participa con cardinalidad máxima 1, la clave de esta entidad pasa a formar parte de la tabla de la relación solo como un atributo mas.

La **Normalizacion** consiste en aplicar unas reglas a las relaciones obtenidas tras el paso de E/R a Relacional, esto se hace para:

a) Evitar redundancias

b) Evitar problemas de actualización de los datos en las tablas

c) Proteger la integridad de los datos

Dependencia funcional es una relación entre atributos de una misma relación (Tabla).

$X \rightarrow Y$ Ejemplo: Año nacimiento → Edad

Dependencia funcional transitiva

Si $X \rightarrow Y$ Ejemplo: Año nacimiento → Edad

Si $Y \rightarrow Z$ entonces $X \rightarrow Z$ Edad → Carnet de conducir entonces Año nacimiento → Carnet conducir

Reglas Normalizacion, se dice que una relación esta en forma normal si satisface un cierto conjunto especifico de restricciones impuestas por la regla de normalizacion correspondiente.

Primera forma normal. **1FN**, Se dice que una relación esta en 1FN, si y solo si los valores que componen cada atributo de una tupla son atómicos (es decir toma un único valor, o lo que es lo mismo no existen grupos repetitivos), las formas de eliminar los grupos repetitivos es repetir los atributos con un solo valor para cada valor del grupo repetitivo o poner cada uno de ellos en una relación aparte heredando la clave primaria de la relación en la que se encontraba.

NSS	Nombre	Puesto	Salario	Emails
111	Pepe	Jefe Area	3000	josep@ecn.es jefez@gmail.com
222	Josu	Admtivo	1500	jsanchez@ecn.es
333	Miren	Admtiva	1500	mlopez@ecn.es miren@gmail.com
….				
…..				

1FN

NSS	Nombre	Puesto	Salario	Emails
111	Pepe	Jefe Area	3000	josep@ecn.es
111	Pepe	Jefe Area	3000	jefez@gmail.com
222	Josu	Admtivo	1500	jsanchez@ecn.es
333	Miren	Admtiva	1500	mlopez@ecn.es
333	Miren	Admtiva	1500	miren@gmail.com
….				
…..				

Segunda forma normal. **2FN**, Se dice que una relación esta en 2FN si y solo si satisface la 1FN y cada atributo de la relación que no esta en el clave depende funcionalmente de forma completa de la clave primaria de la relación, Si una relación esta en 1FN y su clave primaria es simple (tiene un solo atributo) entonces también esta en 2FN.

NSS	Nombre	Puesto	Salario	Emails
111	Pepe	Jefe Area	3000	josep@ecn.es
111	Pepe	Jefe Area	3000	jefez@gmail.com
222	Josu	Admtivo	1500	jsanchez@ecn.es
333	Miren	Admtiva	1500	mlopez@ecn.es
333	Miren	Admtiva	1500	miren@gmail.com
….				
…..				

2FN

NSS	Nombre	Puesto	Salario
111	Pepe	Jefe Area	3000
222	Josu	Admtivo	1500
333	Miren	Admtiva	1500
…..	……	…….	…….
…..	……	…….	…….
….	……	…….	…….

Emails	NSS (FK)
josep@ecn.es	111
jefez@gmail.com	111
jsanchez@ecn.es	222
mlopez@ecn.es	333
miren@gmail.com	333
…….	….

Tercera forma normal. **3FN**, Una relación esta en tercera forma normal si y solo si, esta en 2FN y ademas cada atributo que no esta en la clave primaria no depende transitivamente de la clave primaria, es decir los atributos de la relación no dependen unos de otros, dependen únicamente de la clave, para ello se eliminan los atributos que dependen transitivamente y se ponen en una nueva relación con una copia de su determinante (el atributo o atributos no clave de los que dependen).

NSS	Nombre	Puesto	Salario
111	Pepe	Jefe Area	3000
222	Josu	Admtivo	1500
333	Miren	Admtiva	1500
…..	……	…….	…….
…..	……	…….	…….
….	……	…….	…….

Emails	NSS (FK)
josep@ecn.es	111
jefez@gmail.com	111
jsanchez@ecn.es	222
mlopez@ecn.es	333
miren@gmail.com	333
…….	….

3FN

NSS	Nombre	Puesto(FK)
111	Pepe	Jefe Area
222	Josu	Admtivo
333	Miren	Admtiva
…..	……	…….
…..	……	…….
….	……	…….
…..	…….	…….

Puesto	Salario
Jefe Area	3000
Admtivo	1500
Admtiva	1500
…….	…….
…….	…….
…….	…….
…….	…….

Emails	NSS (FK)
josep@ecn.es	111
jefez@gmail.com	111
jsanchez@ecn.es	222
mlopez@ecn.es	333
miren@gmail.com	333
…….	….
…….	…..

Algebra Relacional consiste en aplicar un conjunto de operadores mediante los cuales se podrán realizar operaciones de inserción, borrado, modificación y consulta de tuplas.

Unarias

Selección: $\sigma_{condicion}$ (TABLA) , El resultado es un subconjunto de filas de una tabla (determinadas por la condición) con todas las columnas y se pueden incluir en la condición los operadores and, or y not.

Proyección: $\Pi_{COL1, COL2}$ (TABLA) , El resultado es una nueva tabla con el subconjunto de columnas indicado, las filas duplicadas solo aparecen una vez.

Binarias

Union: **TABLA1 ∪ TABLA2**, El resultado de la unión es otra tabla con las filas de ambas tablas (con el mismo numero de columnas y dominios compatibles), las filas repetidas aparecen una sola vez.

Diferencia: **TABLA1 – TABLA2**, El resultado es otra tabla con las filas de una de una y no las otra (deben tener el mismo numero de columnas y dominios compatibles).

Producto cartesiano: **TABLA1 Ξ TABLA2**, El resultado es otra tabla con la suma de columnas de ambas tablas y y el conjunto formado por todas las filas de ambas tablas (Se puede hacer hacer entre tablas con diferente numero de columnas, en el resultado no puede haber columnas con el mismo nombre).

Derivadas

Intersección: **TABLA1 ∩ TABLA2**, El resultado es otra tabla con las filas que aparecen en ambas tablas y las columnas de una de ellas (Las tablas tienen que tener mismo numero de columnas y dominios compatibles).

Combinación o Join: (**TABLA1 * TABLA2**) $_{condicion}$, La combinación de dos tablas con respecto a una condición, o el producto cartesiano cumpliendo una condición determinada.

Cociente: **TABLA1 : TABLA2**, Se realiza entre dos tablas (tabla1 y tabla2) con diferentes columnas que cumplan las siguientes condiciones, la tabla1 tiene que tener columnas de la tabla2 y el numero de columnas ha de ser mayor que la tabla2, la tabla2 ha de tener como mínimo una fila

VENTAS

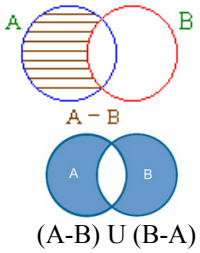
Codi	fecha	cantidad
5100	18/11/11	100
5200	19/11/11	120
5100	19/11/11	45

ARTICULOS

Código	denominación	existencias	pvp
5100	patatas	500	3,2
5200	cebollas	250	4,1

VENTAS2

Codi	fecha	cantidad
5300	18/11/11	100
5200	19/11/11	120



(A-B) U (B-A)

Selección: $\sigma_{codi=5200}$ (VENTAS)

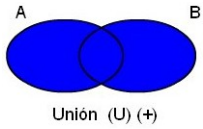
Codi	fecha	cantidad
5200	19/11/11	120

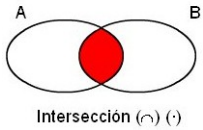
Proyección: $\Pi_{Codi, cantidad}$ (VENTAS)

Codi	cantidad
5100	100
5200	120
5100	45

Union: VENTAS ∪ VENTAS2

Codi	fecha	cantidad
5100	18/11/11	100
5200	19/11/11	120
5100	19/11/11	45
5300	18/11/11	100





Diferencia: VENTAS –VENTAS2

Codi	fecha	cantidad
5100	18/11/11	100
5100	19/11/11	45

Combinación: (VENTAS * ARTICULOS) $_{codi=Codigo}$

Codi	fecha	cantidad	denominación	existencias	pvp
5100	18/11/11	100	patatas	500	3,2
5200	19/11/11	120	cebollas	250	4,1
5100	19/11/11	45	patatas	500	3,2

Intersección:VENTAS ∩ VENTAS2

Codi	fecha	cantidad
5200	19/11/11	120

Producto cartesiano: VENTAS X ARTICULOS

Codi	fecha	cantidad	Código	denominación	existencias	pvp
5100	18/11/11	100	5100	patatas	500	3,2
5100	18/11/11	100	5200	cebollas	250	4,1
5200	19/11/11	120	5100	patatas	500	3,2
5200	19/11/11	120	5200	cebollas	250	4,1
5100	19/11/11	45	5100	patatas	500	3,2
5100	19/11/11	45	5200	cebollas	250	4,1

Cociente: TABLA1 : TABLA2

TABLA1

Modelo	Color
Seat	Rojo
Renault	Verde
Ford	Azul
Seat	Azul

TABLA2

Color
Rojo
Azul

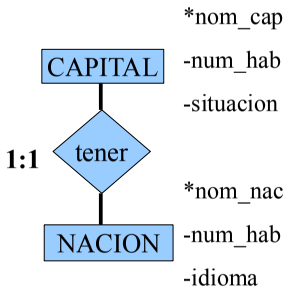
TABLA1 : TABLA2 ⟹

Modelo
Seat

PASO DEL ESQUEMA E/R AL MODELO RELACIONAL (1)

TIPO 1:1

- Cardinalidad obligatoria (1,1) en ambas entidades: Se propaga la clave de cualquiera de ellas a la otra tabla.

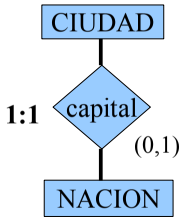


TCAPITAL (nom_cap, num_hab, situación)

TNACION (nom_nac, num_hab, idioma)

TNACION (nom_nac, num_hab, idioma, nom_cap(FK))

- Si una de las entidades posee cardinalidad opcional (0,1) y la otra (1,1): Se propaga la clave de la entidad (1,1) a la tabla de la entidad con (0,1).

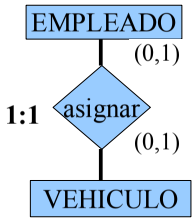


TCIUDAD (nom_ciu, km^2, num_hab)

TNACION (nom_nac, num_hab, idioma)

TNACION (nom_nac, num_hab, idioma, nom_ciu(FK))

- Cardinalidad opcional (0,1) en ambas entidades: La relación se convierte en una tabla cuya clave principal sera la concatenación de las claves de las entidades que asocia.



TEMPLEADO (num_emp, nombre, telef)

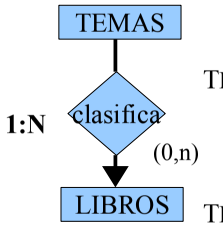
TVEHICULO (matricula, marca, modelo, cv)

TASIGNAR (nun_emp(FK), matricula(FK))

- Si la relación tiene atributos propios también pasaría a la tabla. por ejemplo fecha de asignación del vehículo.

TIPO 1:N

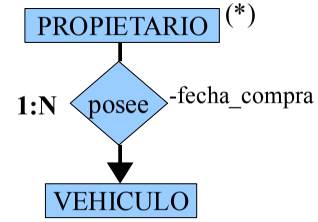
- Cardinalidad obligatoria (1,1) y (0,n) o (1,n): Se propaga la clave en el sentido de la flecha



TTEMAS (cod_tema, descrip)

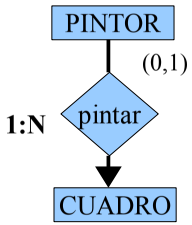
TLIBROS (cod_libro, autor, isbn, titulo)

TLIBROS (cod_libro, autor, isbn, titulo, cod_tema(FK))



TPOSEER (dni(FK), num_bast(FK), fecha_compra)

- Si la cardinalidad es opcional (0,1) y (0,n) o (1,n) o si la relación (*) tiene atributos: Se transforma la relación en una tabla y la clave sera la del lado muchos.



TPINTOR (nombre, estilo, pais)

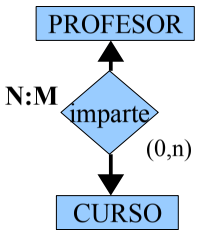
TCUADRO (nom_cuadro, motivo, tipo)

TPINTAR (nom_cuadro (FK), nombre (FK))

Los cuadros anónimos no están, Si lo están los cuadros que tengan autor.

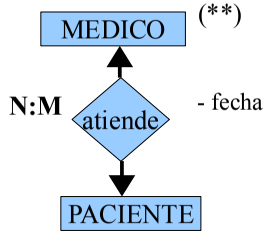
TIPO N:M

- Se transforma la relación en una tabla que tiene como clave primaria, la concatenación de las claves de las entidades que asocia.



TIMPARTE (nom_profesor (FK), cod_curso(FK))

TATIENDE (nom_medico (FK), nombre_paciente(FK), fecha)(**)



(**) A veces también deberán formar parte de la clave algún atributo de la relación.

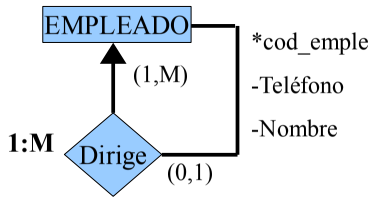
PASO DEL ESQUEMA E/R AL MODELO RELACIONAL (2)

Reflexivas o Recursivas

- Son relaciones binarias, de una entidad consigo misma, lo normal es que se convierta en dos tablas, una para la entidad y otra para la relación, Casos:

- Si la Relación es 1:1 : La clave de la entidad se repite y la tabla resultante tendrá dos veces ese atributo, una como clave primaria y otra como clave ajena y no se hace segunda tabla.

- Si la Relación es 1:M : En el caso que la Entidad muchos sea siempre obligatoria se procede como en el caso 1:1. Si no es obligatoria se cre una nueva tabla cuya clave sera la de la entidad del lado muchos y ademas se propaga la clave a la nueva tabla como clave ajena.

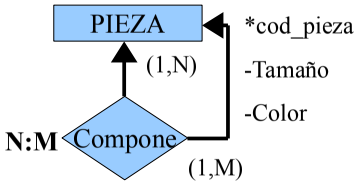


TEMPLEADO (Cod_emple, Teléfono, Nombre)

TEMPLEADO (Cod_emple, Teléfono, Nombre)

TDIRIGE (Cod_emple(FK), Cod_Direc(FK))

- Si la Relación es N:M : Se trata igual que las relaciones binarias, la tabla resultante de la relación tendrá 2 veces la clave primaria de la entidad del lado muchos, mas los atributos de la relación si los hubiera.



TPIEZA (Cod_pieza, Tamaño, Color)

TPIEZA (Cod_pieza, Tamaño, Color)

TCOMPONE_PIEZA (Cod_pieza(FK), Cod Piez Com(FK))

Relaciones N-arias

- En este tipo de relaciones se agrupan 3 o mas entidades y para pasar al modelo de datos relacional cada entidad se convierte en tabla, así como la relación que va a contener los atributos propios de ella mas las claves de todas las entidades, la clave de la tabla resultante sera la concatenación de las claves de las entidades, a tener en cuenta:

- Si la relación es M:M:M la clave de la tabla resultante es la unión de las claves de las tablas que relaciona.

- Si la relación es 1:M:M, la clave de la entidad con relación 1 pasa a formar parte de la tabla resultante pero solo como un atributo mas.

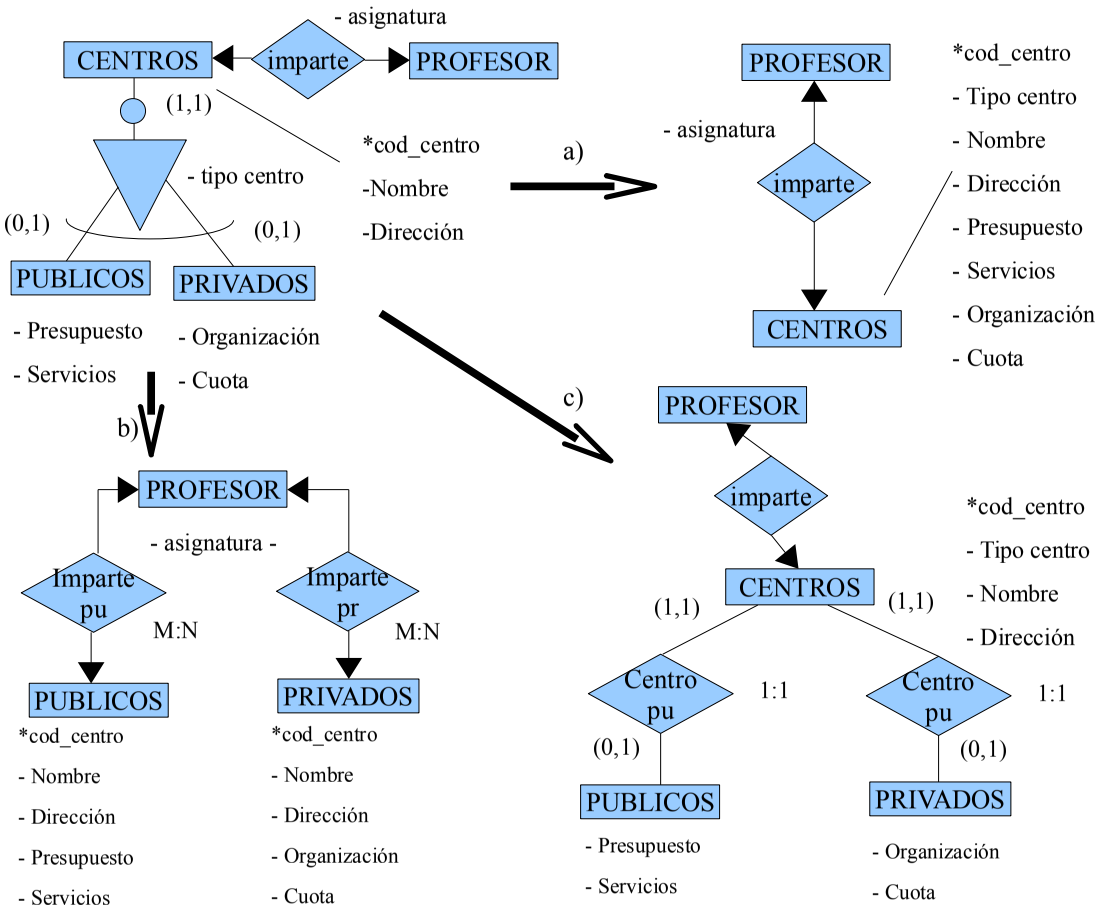
Relaciones jerárquicas

- El modelo Relacional no dispone de mecanismos para la representación de las relaciones jerárquicas, así que se tienen que eliminar, se aplican las siguientes reglas:

a) Integrar todas las Entidades en una única eliminando a los subtipos, esta nueva entidad contendrá todos los atributos del supertipo, todos los de los subtipos y los atributos discriminatorios, puede aplicarse a cualquier tipo de jerarquía, la ventaja es la simplicidad al reducir todo a una entidad y la desventaja que se generan demasiados valores nulos en los atributos opcionales.

b) Eliminación del supertipo, transfiriendo los atributos del supertipo a cada uno de los subtipos, solo se puede aplicar a jerarquías totales y exclusivas, los inconvenientes son redundancia en la información y el numero de relaciones aumenta.

c)Insertar una relación 1:1 entre el supertipo y cada uno de los subtipos. Los atributos se mantienen y cada subtipo se identificara con la clave ajena del supertipo.



Resumen SQL

Tipo de Dato:

- **VARCHAR2**(tamaño) : Almacena cadena de caracteres de longitud variable, entre () el max.
- **CHAR**(tamaño) : Almacena caracteres con una longitud fija especificada.
- **NUMBER**(precisión, escala) : Almacena datos numéricos

Operadores:

- + : Suma
- - : Resta
- * : Multiplicación
- / : División
- = : Igual a
- > : Mayor que
- >= : Mayor o igual que
- < : Menor que
- <= : Menor o igual que
- != : Distinto
- <> : Distinto
- **MAX** : Devuelve el valor máximo de una serie de valores.
- **AND** : Devuelve el valor TRUE cuando las dos condiciones son verdaderas
- **OR** : Devuelve el valor TRUE cuando una de las condiciones es verdadera
- **NOT** : Devuelve el valor TRUE si la condición es falsa
- % : Comodín, representa cualquier cadena de 0 o mas caracteres.
- _ : Marcador de posición que representa a un carácter cualquiera.
- **LIKE** : Para comparación de cadenas de caracteres con inclusión de % y _ .
- **NULL** : Para saber si el valor de una columna es nulo se usa columna **IS NULL** y si se quiere saber si no es nulo, se usa **IS NOT NULL** .
- **IN** : Para comprobar si una expresión pertenece o no (NOT) a un conjunto de valores.
- **BETWEEN** : Comprueba si un valor esta dentro de un rango de valores.

Consulta de los Datos, Sentencias:

SELECT columna **FROM** tabla **WHERE** condición **ORDER BY** columna/constante/expresión/función**;**

- **DESC** : Describe, da una descripción de la Tabla, se usa seguido del nombre de la Tabla.
- **FROM** : Especifica la tabla o lista de tablas de las que se recuperan los datos.
- **WHERE** : Obtiene las filas que cumplen la condición.
- **ORDER BY** : Especifica el criterio de clasificación del resultado de la consulta.
- **DESC** : Cuando se usa con ORDER BY, ordena de forma descendente.
- **DISTINCT** : Aparecen solo las filas que son distintas.
- * : Representa todas las columnas de la tabla.
- "alias" : Se utiliza para dar otro nombre a la columna en la presentación.

DESC Alumnos ;
Descripción de la tabla Alumnos.

SELECT Nombre **FROM** Alumnos ;
Visualiza los datos de la columna 'Nombre' de la Tabla 'Alumnos'.

SELECT Nombre, Apellidos **FROM** Alumnos **WHERE** (Nota>=5) AND (Curso=1) ;
Visualiza las columnas 'Nombre' y 'Apellidos' de la Tabla 'Alumnos' con la condición Nota<=5 y Curso=1

SELECT Nombre, Apellidos **FROM** Alumnos **WHERE** Nombre='Pedro' ;
Visualiza las columnas 'Nombre' y 'Apellidos' de la Tabla 'Alumnos' con la condición Nombre = Pedro

SELECT Nombre, Apellidos **FROM** Alumnos **ORDER BY** Nombre **DESC** ;
Visualiza las columnas 'Nombre' y 'Apellidos' de la Tabla 'Alumnos' ordenada por el Nombre en descendente

SELECT DISTINCT Dept_N **FROM** Emple ;
Visualiza solo las filas distintas de la columna 'Dept_N' de la Tabla 'Emple'.

SELECT * **FROM** Alumnos ;
Selecciona todas las columnas de la Tabla Alumnos.

SELECT Dept_N "Departamento" **FROM** Emple ;
El nombre de la columna Dept_N sera visualizado como Departamento.

SELECT Apellido **FROM** Emple **WHERE** Apellido **LIKE** 'A%o%' ;
Obtiene aquellos apellidos que empiezan por una 'A' y contienen una 'o' en su interior.

SELECT Apellido **FROM** Emple **WHERE** Comision **IS NOT NULL** ;
Consulta de los Apellidos de Empleados cuya Comisión no sea nula.

SELECT Apellido **FROM** Emple **WHERE** Dept_N **IN** (10, 30) ;
Consulta de los Apellidos de Empleados cuyo departamento sea 10 o 30.

SELECT Apellido **FROM** Emple **WHERE** Salario **BETWEEN** 1500 AND 2000 ;
Consulta de los Apellidos de Empleados cuyo Salario este comprendido entre 1500 y 2000.

SELECT Nombre, (Nota1+Nota2+Nota3)/3 "Nota media" **FROM** Notas_Alumnos **WHERE** Notas2<6 ;
De una tabla de Notas, selecciona lo alumnos cuya Nota2 es menor que 6, presenta la columna del nombre y otra columna que llamamos Nota media (por medio del alias), con el resultado de la media de las columnas Nota1, Nota2 y Nota3.

Operadores en SubConsultas:

Estos se usan cuando se sabe que el resultado es único.

- **=** : Igual a
- **>** : Mayor que
- **>=** : Mayor o igual que
- **<** : Menor que
- **<=** : Menor o igual que
- **!=** : Distinto
- **<>** : Distinto
- **IN** : Para comprobar si una expresión pertenece o no (NOT) a un conjunto de valores. Este se usa cuando no se sabe si el resultado es único o va a dar mas de un resultado.

SELECT APELLIDO, OFICIO, SALARIO, FECHA_ALT **FROM** EMPLE **WHERE** OFICIO=**(SELECT** OFICIO **FROM** EMPLE **WHERE** APELLIDO='JIMENEZ'**) OR** SALARIO>=**(SELECT** SALARIO **FROM** EMPLE **WHERE** APELLIDO='FERNANDEZ'**)** ;
Subconsulta, Queremos sacar Apellido, Oficio,.....de la Tabla EMPLE cuando tengan el mismo Oficio que Jimenez o Salario igual o superior al que tiene Fernandez, entonces tenemos que hacer una consulta sobre otras 2 consultas como se ve en las instrucciones.

SELECT OFICIO, APELLIDO **FROM** EMPLE **WHERE** DEPT_NO=20 **AND** OFICIO **IN** (**SELECT** OFICIO **FROM** EMPLE **WHERE** DEPT_NO=(**SELECT** DEPT_NO **FROM** DEPART **WHERE** DNOMBRE='VENTAS'));
En esta Subconsulta como devuelve mas de un valor se utiliza el operador **IN**.

Subconsultas correlacionadas:

El for,mato es el siguiente: **SELECT** * **FROM** EMPLE **E WHERE** SALARIO = (**SELECT MAX** (SALARIO) **FROM** EMPLE **WHERE** DETP_NO = **E**.DEPT_NO);
Obtener los datos de los empleados cuyo salario sea el máximo salario de su departamento, para referenciar a dicho departamento se necesita un alias **(E)**.

Combinación de Tablas en una consulta:

El formato es el siguiente: **SELECT** Columna **FROM** Tabla1, Tabla2 …. **WHERE** Tabla1.Columna = Tabla2.Columna;

SELECT APENOM "Alumno", NOMBRE "Asignatura" **FROM** ALUMNOS, ASIGNATURAS, NOTAS **WHERE** ALUMNOS.DNI=NOTAS.DNI **AND** NOTAS.COD=ASIGNATURAS.COD **AND** NOMBRE='FOL';

Obtener los alumnos que estudien la asignatura de FOL, para poder hacer esta consulta hay que combinar las tablas de Alumnos, Notas y Asignaturas.

Funciones de valores simples:

- **ABS** (n) : Devuelve el valor absoluto de 'n', es siempre un numero positivo.
- **CEIL** (n) : Obtiene el valor entero inmediatamente superior o igual a 'n'.
- **FLOOR** (n) : Lo opuesto a CEIL, devuelve el valor entero inmediatamente inferior o igual a 'n'.
- **MOD** (m, n) : Devuelve el resto resultante de dividir 'm' entre 'n'.
- **NVL** (valor, expresión) : Si 'valor' es nulo lo sustituye por la 'expresión', si no lo es devuelve 'valor'.
- **POWER** (m, exponente) : Calcula la potencia de un numero, devuelve el valor de 'm' elevado a un 'exponente'.
- **ROUND** (numero [,m]) : Devuelve el valor de 'numero' redondeado a 'm' decimales, si se omite 'm' devuelve 'numero' con 0 decimales y redondeado.
- **SIGN** (valor) : Indica el signo del 'valor', si menor que 0 devuelve _1, si mayor que 0 devuelve 1.
- **SQRT** (n) : Devuelve la raíz cuadrada de 'n', el valor de 'n' no puede ser negativo.
- **TRUNC** (numero, [m]) : Devuelve 'numero' truncado a 'm' decimales, si 'm' es negativo trunca por la izquierda del punto decimal, si se omite 'm' devuelve 'numero' con 0 decimales.

SELECT ABS (-46) **FROM** DUAL; Devuelve valor: **46**

SELECT CEIL (2.5) **FROM** DUAL; Devuelve valor: **3**

SELECT FLOOR (2.5) **FROM** DUAL; Devuelve valor: **2**

SELECT MOD (10,3) **FROM** DUAL; Devuelve valor: **1**

SELECT SALARIO + **NVL** (COMISION, 0) **FROM** EMPLE;

SELECT POWER (3,2) **FROM** DUAL; Devuelve valor: **9**

SELECT ROUND (-33.67,1) **FROM** DUAL; Devuelve valor: **-33,7**

SELECT TRUNC (-33.67,1) **FROM** DUAL; Devuelve valor: **-33,6**

SELECT SQRT (9) **FROM** DUAL; Devuelve valor: **3**

Funciones de grupos de valores:

- **AVG**(n) : Calcula el valor medio de 'n' ignorando los valores nulos.
- **COUNT** (* | expresión) : Cuenta el numero de veces que la expresión evaluá algún dato con valor no nulo, la opción '*' cuenta todas las filas seleccionadas.
- **MAX** (expresión) : Calcula el máximo valor de la 'expresión'.
- **MIN** (expresión) : Calcula el mínimo valor de la 'expresión'.
- **SUM** (expresión) : Obtiene la suma de valores de la 'expresión' distintos de nulos.
- **VARIANCE** (expresión) : Obtiene la varianza de los valores de 'expresión' distintos de nulos.

Se puede usar DISTINCT y ALL en este tipo de funciones, aunque su uso mas normal es con COUNT, de la siguiente manera: COUNT (* | [DISTINCT] expresión) o COUNT (* | [ALL] expresión)

SELECT AVG (SALARIO) **FROM** EMPLE **WHERE** DEPT_NO=10;

SELECT COUNT (COMISION) **FROM** EMPLE;

SELECT MAX (SALARIO) **FROM** EMPLE;

SELECT MIN (SALARIO) **FROM** EMPLE;

SELECT SUM (SALARIO) **FROM** EMPLE;

SELECT VARIANCE (SALARIO) **FROM** EMPLE;

SELECT COUNT (**DISTINCT** OFICIO) "Oficio" **FROM** EMPLE;

Funciones de listas:

- **GREATEST** (valor1, valor2, ...) : Obtiene el mayor valor de la lista.
- **LEAST** (valor1, valor2, ...) : Obtiene el menor valor de la lista. :

SELECT NOMBRE_ALUMNO **GREATEST** (NOTA1, NOTA2, NOTA3) "Nota Mayor", **LEAST** (NOTA1, NOTA2, NOTA3) "Nota Menor" **FROM** DUAL;

Funciones de cadenas de caracteres:

- **CHR** (n) : Devuelve el carácter cuyo valor en binario es equivalente a 'n'.
- **CONCAT** (cad1, cad2) : Devuelve 'cad1' concatenada con 'cad2'. es equivalente al operador ||.
- **LOWER** (cad) : Devuelve la cadena 'cad' con todas sus letras convertidas a minúsculas.
- **UPPER** (cad) : Devuelve la cadena 'cad' con todas sus letras convertidas a mayúsculas.
- **INITCAP** (cad) : Convierte la cadena 'cad' a tipo titulo, la primera letra de cada palabra de 'cad' a mayúsculas y el resto a minúsculas.
- **LPAD** (cad1, n [, cad2]) : Devuelve 'cad1' con longitud 'n' y ajustado a la derecha, 'cad2' es la cadena con la que se rellena por la izquierda, si se suprime 'cad2' el carácter de relleno es blanco.
- **RPAD** (cad1, n [, cad2]) : Devuelve 'cad1' con longitud 'n' y ajustado a la izquierda, 'cad2' es la cadena con la que se rellena por la derecha, si se suprime 'cad2' el carácter de relleno es blanco.
- **LTRIM** (cad, n [, set]) : Devuelve 'cad' con el grupo de caracteres 'set' omitidos por la izquierda, si se omite el segundo parámetro devuelve la misma cadena y si tiene blancos los omite por la izquierda.
- **RTRIM** (cad, n [, set]) : Devuelve 'cad' con el grupo de caracteres 'set' omitidos por la derecha, si se omite el segundo parámetro devuelve la misma cadena y si tiene blancos los omite por la derecha.
- **SUBSTR** (cad, m [,n]) : Devuelve la subcadena de 'cad' que abarca desde la posición indicada en 'm' hasta tantos caracteres como indique el numero'n',
- **TRANSLATE** (cad1, cad2, cad3) :Devuelve 'cad1' con los caracteres encontrados en 'cad2' y sustituidos por por los caracteres de 'cad3', los caracteres que no están en 'cad2' no se cambian.
- **REPLACE** (cad, cadena_búsqueda [,cadena_sustitución]) : Devuelve 'cad' con cada ocurrencia de 'cadena-búsqueda' sustituida por 'cadena_sustitución'.

SELECT CHR (75), **CHR** (65) **FROM** DUAL;

SELECT CONCAT (**LOWER** (APELLIDO) || ' es ', **UPPER** (OFICIO)) “Puestos Empleados” **FROM** EMPLE;

SELECT CONCAT (**INITCAP** (APELLIDO) || ' es ', **INITCAP** (OFICIO)) “Puestos Empleados” **FROM** EMPLE;

SELECT LPAD (NOMBRE, 30, ' . ') "Izquierda", **RPAD** (NOMBRE, 30, ' . ') "Derecha" **FROM** EMPLE;

SELECT RTRIM (TUTULO, ' .* ') **FROM** MISTEXTOS;

SELECT LTRIM (TUTULO, ' * ') **FROM** MISTEXTOS;

SELECT APELLIDO "Apellido", **SUBSTR** (APELLIDO, 1, 1) "Inicial Apellido" **FROM** EMPLE;

SELECT TRANSLATE ('OGRO', 'O', 'AS') **FROM** DUAL;

SELECT REPLACE (APELLIDO, 'EZ' , 'O') "Apellido empleados" **FROM** EMPLE;

Funciones que devuelven valores numéricos:

- **ASCII** (cad) : Devuelve el valor ASCII de la primera letra de la cadena 'cad'.
- **LENGTH** (cad) : Devuelve el numero de caracteres de 'cad'.
- **INSTR** (cad1, cad2 [, comienzo [, m]]) :Devuelve la posición de la 'm_esima' ocurrencia de 'cad2' en 'cad1', empezando la búsqueda en la posición 'comienzo'. Por omisión, empieza buscando en la posición 1.

SELECT ASCII ('A') FROM DUAL**;**

SELECT TEMA, **LENGTH** (TEMA) "Longitud Tema" **FROM** LIBRERIA;

SELECT INSTR (AUTOR, (',')) "Apellido" **FROM** LIBROS;

Funciones para el manejo de fechas:

- **SYSDATE** : Devuelve la fecha del sistema.
- **ADD_MONTHS** (fecha, n) : Devuelve la fecha 'fecha' incrementada en 'n' meses.
- **LAST_DAY** (fecha) : Devuelve la fecha del ultimo día del mes que contiene 'fecha'.
- **NEXT_DAY** (fecha, cad) : Devuelve la fecha del primer día de la semana indicado por 'cad' Después de la fecha indicada por 'fecha', el día de la semana se indica por su nombre, lunes (monday).
- **MONTHS_BETWEEN** (fecha1, fecha2) : Devuelve la diferencia en meses entre las fechas 'fecha1' y 'fecha2'.

SELECT SYSDATE FROM DUAL;

SELECT FECHA_ALT, **ADD_MONTHS** (FECHA_ALT, 12) **FROM** EMPLE **WHERE** DEPT_NO=10;

SELECT FECHA_ALT, **LAST_DAY** (FECHA_ALT) **FROM** EMPLE **WHERE** DEPT_NO=10;

SELECT TRUNC (**MONTHS_BETWEEN** (**SYSDATE**, '18/11/1964') / 12) "Edad actual" **FROM** DUAL;

Funciones de conversión:

- **TO_CHAR** (fecha, 'formato') : Convierte una 'fecha' (de tipo DATE) a tipo VARCHAR2 en el 'formato' especificado.
- **TO_CHAR** (numero, 'formato') : Convierte una 'numero' (de tipo NUMBER) a tipo VARCHAR2 en el 'formato' especificado.
- **TO_DATE** (cad, 'formato') : Convierte 'cad' de tipo VARCHAR2 o CHAR, a un valor tipo DATE según el 'formato' especificado.
- **TO_NUMBER** (cadena [,'formato']) : Convierte la 'cadena' a tipo NUMBER según el 'formato' especificado.

SELECT NOMBRE, FECHANAC, **TO_CHAR** (FECHANAC, '"Nacio el " dd " de " month " de " yyyy') "Fecha Formateada" **FROM** NACIMIENTOS;

SELECT TO_NUMBER ('123.456', '999G999') “No Convierte” **FROM** DUAL;
En este caso no convierte al tener el numero ya marcado el separador de miles.

SELECT TO_DATE ('01012006') **FROM** DUAL;

Mascaras de formato de caracteres

- year año en ingles.
- month Nombre del mes (ENERO).
- mon Abreviatura de 3 letras del nombre del mes (ENE).
- day Nombre del día de la semana (LUNES).
- a.m/p.m. Muestra a.m. o p.m. dependiendo del momento del dia.
- b.c./a.d. Indicador para el año (antes de cristo o después de cristo).

Mascaras de formato numéricas

- **cc** Valor del siglo.
- **yyyy** Año sin signo.
- **yyy** Últimos 3 dígitos del año, si solo se ponen 2 son 2 dígitos y con una y 1 dígito.
- **q** Numero del trimestre.
- **ww** Numero de semana del año.
- **w** Numero de semana del mes.
- **mm** Numero de mes.
- **ddd** Numero del día del año.
- **dd** Numero del día del mes.
- **d** Numero del día de la semana.
- **hh** Hora (1-12).
- **hh24** Hora (1-24).
- **mi** Minutos.
- **ss** Segundos.
- **ssss** segundos transcurridos desde medianoche.
- **j** Juliano

Tablas de formatos numéricos

- 9 Devuelve el valor con el numero especificado de dígitos, ejemplo 999 para 3 dígitos.
- $ Devuelve el valor con el signo dolar a la izquierda, ejemplo $9999.
- D Devuelve el carácter decimal en la posición especificada, ejemplo 9D99, seria 1,23.
- G Devuelve el carácter de miles en la posición indicada, ejemplo 9G999, seria 1.000.
- L Devuelve el símbolo de la moneda local en la posición indicada, ejemplo 999L, 100€.
- **,** Devuelve la coma en la posición especificada, ejemplo 9,999, seria 1,234.
- **.** Devuelve el punto en la posición especificada, ejemplo 9.999, seria 1.000.
- y otros tantos mas.....

Otras Funciones:

- **DECODE** (var, val1, cod1, val2, cod2 ..., valor_por defecto): Esta función sustituye un valor por otro, si "var" es igual a cualquier valor de la lista ("val1", "val2"..) devuelve el correspondiente código ("cod1", "cod2"...), en caso contrario da el valor señalado por defecto.
- **GROUP BY** Consulta de datos según grupos determinados, por ejemplo salario medio de cada departamento.
  HAVING Es una condición al "GROUP BY" para controlar cual de los conjuntos de filas se visualiza.
- **COUNT** (*) Usada con GROUP BY realiza el contaje de las filas determinadas por este.

SELECT APELLIDO, OFICIO, **DECODE** (**UPPER** (OFICIO), 'PRESIDENTE', 1, 'EMPLEADO', 2, 5) "Código" **FROM** EMPLE;
De la tabla EMPLE selecciona todas las filas y codifica el OFICIO, si el OFICIO es PRESIDENTE codifica con un 1, si es EMPLEADO con un 2, si es otra cosa con un 5.

SELECT DEPT_NO, OFICIO, **COUNT**(*) **FROM** EMPLE **GROUP BY** DEPT_NO, OFICIO;
Visualiza a partir de la tabla EMPLE el numero de empleados que hay en cada departamento.

Combinación externa (OUTER JOIN)

Es una variedad de combinación de tablas, que permite seleccionar algunas filas de una tabla aunque estas no tengan correspondencia con las filas de la otra tabla con la que se combina:

- **SELECT** tabla1.colum1, tabla1.colum2, tabla2.colum1 **FROM** tabla1, tabla2 **WHERE** tabla1.colum1 = tabla2.colum1**(+)**; — Se seleccionan todas las filas de la tabla1, aunque no tengan correspondencia con las filas de la tabla2; se denota con el símbolo (+) detrás de la columna de la tabla2 (que es donde no se encuentran las filas)

SELECT D.DEPT_NO, DNOMBRE, **COUNT**(E.EMP_NO) **FROM** EMPLE E, DEPART D **WHERE** E.DEPT_NO **(+)** = D.DEPT_NO **GROUP BY** D.DEPT_NO, DNOMBRE;
Se muestran todos los departamentos aunque no tengan empleados, gracias a añadir (+), si no se añade este símbolo solo se muestran los departamentos que tienen empleados (el 40 que no tiene empleados no aparece).

Operadores UNION, INTERSECT y MINUS:

- **UNION** Combina los resultados de 2 consultas, no muestra filas duplicadas.
- **UNION ALL** Combina los resultados de 2 consultas y muestra las filas duplicadas.
- **INTERSECT** Realiza la intersección, devuelve las filas que son iguales en las 2 consultas, no muestra filas duplicadas.
- **MINUS** Hace la resta, devuelve las filas que están en la 1ª Select, pero no están en la segunda.

SELECT NOMBRE **FROM** ALUM **UNION** SELECT NOMBRE **FROM** NUEVOS;
Combina las tablas ALUM y NUEVOS y muestra la columna NOMBRE

SELECT NOMBRE **FROM** NUEVOS **INTERSECT SELECT** NOMBRE **FROM** ANTIGUOS;
Realiza la intersección de las Tablas NUEVOS y ANTIGUOS

SELECT NOMBRE **FROM** ALUM **MINUS** (**SELECT** NOMBRE **FROM** NUEVOS **UNION SELECT** NOMBRE **FROM** ANTIGUOS);
Hace la Unión de las tablas NUEVOS y ANTIGUOS y el resultado lo resta de la tabla ALUM

Manipulación de Datos INSERT, UPDATE, DELETE:

- **INSERT INTO** Tabla (columna1, columna2, ...) **VALUES** (valor1, valor2, ..) — Se utiliza para insertar datos en las tablas (valor1 va con columna1). Si el valor es alfanumérico va entre ' '.
- **INSERT INTO** Tabla (columna1, columna2, …) **SELECT** (columna1, columna2,...) **FROM** Tabla (clausulas de SELECT) — Inserción de datos mediante una los datos obtenidos con SELECT.
- **UPDATE** Tabla **SET** columna=valor **WHERE** condición — Se utiliza para modificar datos en las tablas. Si el valor es alfanumérico va entre ' '.
- **UPDATE Tabla** **SET** columna=(clausulas de SELECT) **WHERE** condición — Modificar datos mediante los datos obtenidos con SELECT. Si el valor es alfanumérico va entre ' '.
- **DELETE FROM** Tabla **WHERE** condición — Eliminación de una o varias filas de una tabla. Condición necesaria para especificar que filas se borran. La condición se puede realizar con una SELECT.

INSERT INTO PROFESORES (COD_CENTRO, DNI, APELLIDOS, ESPECIALIDAD)
VALUES (22, 23444800, 'Gonzalez Sevilla, Miguel A.', 'HISTORIA');

INSERT INTO ALUM **SELECT * FROM** NUEVOS **MINUS SELECT * FROM** ALUM;
Inserta en la Taba ALUM los alumnos nuevos obtenidos de la selección, en este caso las dos tablas tienen las mismas columnas.

UPDATE EMPLE **SET** SALARIO=SALARIO+100, COMISION=COMISION+10 **WHERE** DEPT_NO=10;
Sube en 100€ el salario y en 10€ la comisión de los empleados del departamento 10

UPDATE CENTROS **SET** (DIRECCION, NUM_PLAZAS) = (SELECT DIRECCION, NUM_PLAZAS FRON CENTROS WHERE COD_CENTROS = 50) **WHERE** COD_CENTRO = 10;
Esta sentencia actualiza los datos de Dirección y plazas del centro con código 10 con los que hay en el centro con código 50.

DELETE FROM PROFESORES **WHERE** DNI **IN** (**SELECT** DNI **FROM** PROFESORES **MINUS SELECT** DNI **FROM** PERSONAL);
Borra a los profesores que estén en la tabla PROFESORES y que no estén en la tabla PERSONAL.

Manipulación de Datos ROLLBACK, COMMIT Y AUTOCOMMIT:

- **COMMIT** Se usa para validar los datos introducidos.

- **AUTOCOMMIT** Se usa para validar automáticamente las transacciones.
  SHOW AUTOCOMMIT Se usa para saber en que modo esta AUTOCOMMIT.
  SET AUTOCOMMIT Se usa para activar o desactivar el AUTOCOMMIT con ON y OFF

- **ROLLBACK** Esta orden aborta las transacciones volviendo la base de datos al estado del ultimo COMMIT.

Creación, Supresión y Modificación de tablas:

- **CREATE TABLE** Tabla — Crea objetos de base de datos. Tablas, vistas, sinónimos, etc.
  (
  Columna1 Tipo_dato [NOT NULL], — Define columna, tipo dato y si es NO NULL (no puede tener valores nulos)
  Columna2 Tipo_dato [DEFAULT], — con DEFAULT define que ponga un valor por omisión.
  ….....
  CONSTRAINT nombre **PRIMARY KEY** (Columna1, Columna2,....), — Define la/las columnas que son clave primaria.
  CONSTRAINT nombre **FOREIGN KEY** (Columna1, Columna2,....), — Define la clave ajena y la tabla que referencia.
  REFERENCES Tabla **ON DELETE CASCADE** — la ultima parte actualiza una con la otra.
  CONSTRAINT nombre **CHECK** (Columna BETWEEN 10 AND 20) — Define una restricción (valor entre 10 y 20).
  CONSTRAINT nombre **UNIQUE** (Columna) — Define que no se repita la fila dentro de la columna.
  CONSTRAINT nombre …..... — El resto de restricciones si las hay
  );

- **CREATE TABLE** Tabla — Crea objetos de base de datos. Tablas, vistas, sinónimos, etc
  AS SELECT …...; — A partir de una selección con la clausula AS y el SELECT correspondiente.

- **DROP TABLE** Tabla; — Elimina la tabla indicada.

- **DROP TABLE** Tabla — Elimina la Tabla indicada y las restricciones de integridad referencial que
  CASCADE CONSTRAINTS; — remitan al la clave primaria en otra tabla o tablas.

- **TRUNCATE TABLE** Tabla; — Elimina el contenido de la tabla indicada (pero mantiene la estructura).

- **ALTER TABLE** Tabla — Modifica tablas.
  ADD (Columna1 valor, Columna2 valor, ..), — Añade campos nuevos
  MODIFY (Columna1 valor, Columna 2 valor,..), — Modifica columnas
  DROP COLUMN (Columna1, Columna 2), — Elimina columnas
  ADD CONSTRAINT nombre **PRIMARY KEY** (Columna1,..), — Añade restricciones
  DROP CONSTRAINT nombre **PRIMARY KEY** (Columna1,..), — Elimina restricciones
  DISABLE CONSTRAINT nombre **PRIMARY KEY** (Columna1,..); — Des-habilita restricciones
  ENABLE CONSTRAINT nombre **PRIMARY KEY** (Columna1,..); — Habilita restricciones

CREATE TABLE ALUMNOS07
(
NUMERO_MATRICULA NUMBER(6) NOT NULL,
NOMBRE VARCHAR2(15) NOT NULL,
EDAD NUMBER(2) NOT NULL ,
FECHA DATE DEFAULT SYSDATE,
DIRECCION VARCHAR2(30),
LOCALIDAD VARCHAR2(15),
COD_PROVINCIA NUMBRE(2), NOT NULL,
CONSTRAINT PK_NUMERO **PRIMARY KEY**(NUMERO),
CONSTRAINT EDAD CHECK (EDAD BETWEEN 18 AND 35),
CONSTRAINT FK_COD_PROVINCIA FOREIGN KEY(COD_PROVINCIA)
REFERENCES PROVIN ON DELETE CASCADE
) ;

CREATE TABLE COPIAOFICINAS
AS SELECT * FROM OFICINAS ;

DROP TABLE ALUMNOS07;

ALTER TABLE TIENDAS2
ADD CONSTRAINT PK_ELNIF **PRIMARY KEY** (NIF);

ALTER TABLE USUARIOS
DROP COLUMN CUOTA_FAMILIAR ;

Comando para que liste las tablas del usuario:
SELECT TABLE_NAME FROM USER_TABLES;

Creación VISTAS:

- **CREATE VIEW** 'Nombre vista' — Crea una vista con el nombre indicado como si de una tabla se tratase
  AS SELECT Columna1, Columna 2, ... — Se definen las columnas que estarán en la vista
  FROM Tabla — Define de que tabla se hace la vista.
  WHERE Condición — Define una condición, para seleccionar determinadas filas.
- **CREATE OR REPLACE VIEW** — Crea o Reemplaza una vista creada
  'Nombre vista' (Name1, Name2,...) — asignamos nuevos nombres a las columnas que se seleccionan.
  AS SELECT Columna1, Columna 2, ... — Se definen las columnas que estarán en la vista
  FROM Tabla — Define de que tabla se hace la vista.
  WHERE Condición — Define una condición, para seleccionar determinadas filas.
- **CREATE VIEW** 'Nombre vista' — Crea una vista con el nombre indicado como si de una tabla se tratase
  AS SELECT Columna1, Columna 2, ... — Se definen las columnas que estarán en la vista
  FROM Tabla — Define de que tabla se hace la vista.
  WHERE Condición — Define una condición, para seleccionar determinadas filas.

  WITH CHECK OPTION — Esta opción que se pone al final, Sirve para que todas las operaciones que se hagan en la vista cumplan las condiciones con la que se creo, siempre que haya una condición se pondrá.

  WITH READ ONLY — Esta opción que se pone al final, Sirve para que solo sea de lectura la vista (SELECT) y no se pueda modificar.
- **DROP VIEW** 'Nombre vista' ; — Con esta sentencia borramos una vista creada.

CREATE VIEW ACTIVIDADCUOTA3500 **AS SELECT** CODIGO_ACTIVIDAD, DESCRIPCION **FROM** ACTIVIDADES **WHERE** CUOTA **> 3500** ;

Comando para visualizar las vistas del usuario:
SELECT VIEW_NAME, TEXT FROM USER_VIEWS;


UNDER
CONSTRUCTION
UNDER CONSTRUCTION