Universidad de Cádiz

Escuela Superior de Ingeniería

Ingeniería Informática

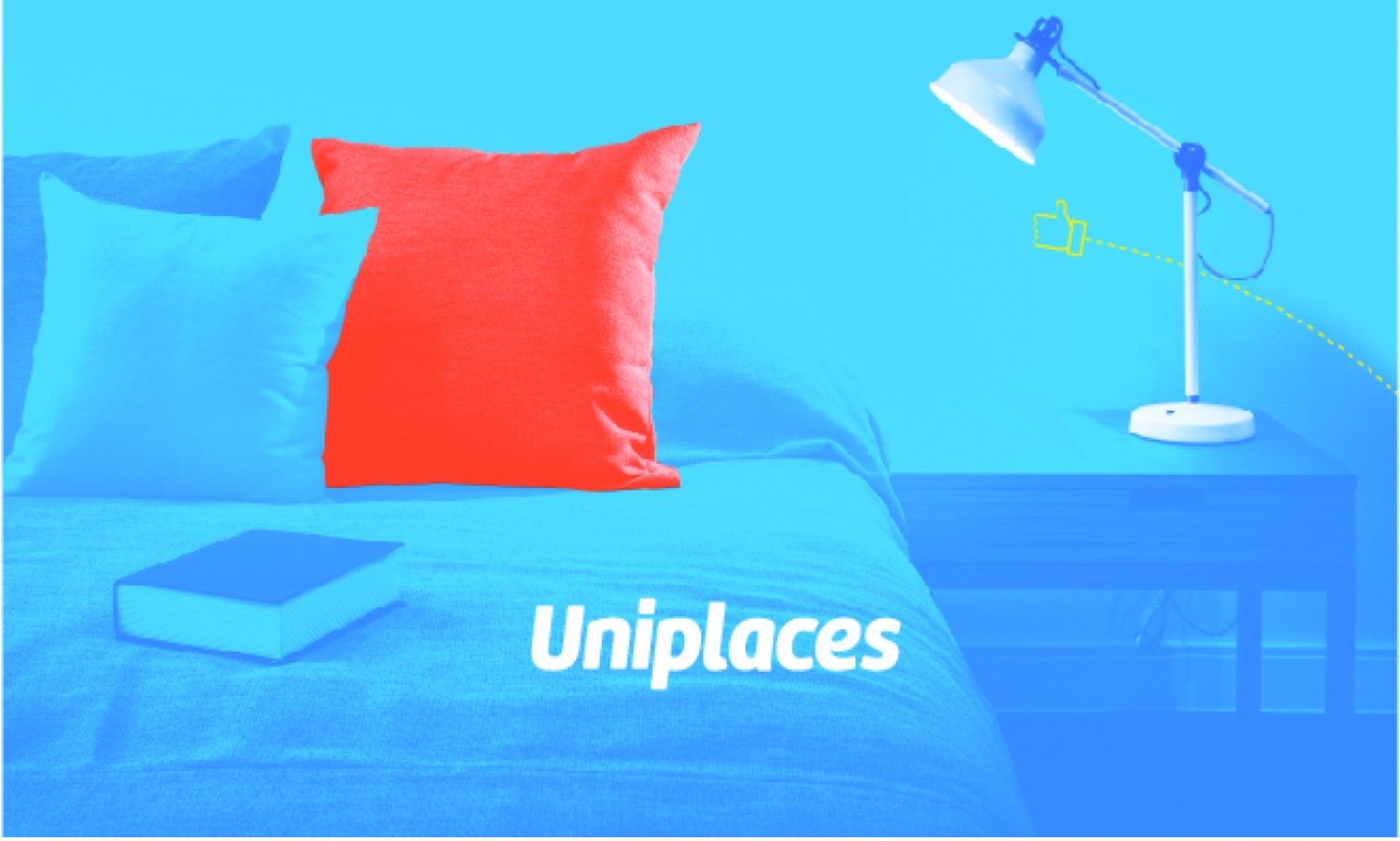
Bases de Datos

Apuntes de la asignatura

Antonio Robles Sorroche

Alumno de la asignatura

Busca tu residencia o piso de estudiantes en Uniplaces



ÍNDICE

TEORÍA

1

CONCEPTOS BÁSICOS

1.1 La información

1.1.1 ¿Qué es la información?

Es el resultado de procesar datos. La información es el conjunto organizado de datos que constituye sobre un cierto fenomeno o ente. La información permite resolver problemas y tomar decisiones, ya que su uso racional es la base del conocimiento.

1.1.2 Datos

Los datos son el antecedente necesario para llegar al conocimiento exacto de una cosa o para deducir las consecuencias legítimas de un hecho.

1.2 Cualidades de la información

Las cualidades que debe poseer la información y que hacen de ella un recurso fundamental de las organizaciones y de los individuos son básicamente:

precision: se puede definir como el porcentaje de información correcta sobre la información total del sistema (ficheros, datos...). Si queremos que los resultados sean precisos debemos también suministrarle datos precisos, no pudiendo pretender en los resultados una precisión mayor a la de los datos de entrada.

Oportunidad: Tiempo transcurrido desde el momento que se produjo el hecho que originó el dato hasta el momento en el que la información se pone a disposición del usuario.

Compleción: la información ha de ser completa para poder cumplir sus fines. La compleción absoluta es imposible de conseguir, se suele pretender en los sistemas de información que alcance un nivel suficiente.

Segura: la información ha de estar protegida ante su deterioro, así como frente a accesos no autorizados.

Coherente: debe ser coherente en si misma, además de consistente con las reglas semanticas propias del mundo real.

Integra: Debe ser coherente y precisa.

1.3 Sistema de Información

Podemos definir el sistema de información como un cto. de elementos, ordenadamente relacionados entre si de acuerdo con unas reglas que aporta al sistema objeto la información necesaria para el cumplimiento de sus fines, para lo cual tendrá que recoger, procesar, y almacenar datos procedentes tanto de la misma organicación como de fuentes externas, facilitando la recuperación, elaboración y presentación de los mismos.

1.4 Componentes de un sistema de Información

Contenido Dato: Referenciales, o factuales (Estruct, o no estruct)

Hardware: Cpu, periférico

Software: SO, SGDB, Control de comunicaciones, tratamientos específicos.

Admin: Area datos, Area informática

Usuarios: Informáticos, o no informáticos.

1.5 Sistema de Ficheros

Fichero: cto. de bloques físico, cto. de bloques del mismo tipo.

SGF: Sistema que gestiona un cto. de ficheros específicos para una o varias aplicaciones.

Aplicación: Cto. de programas que utiliza información almacenada en ficheros con formato para cada la aplicación.

Sistemas orientados hacia el proceso: Los datos se almacenan en ficheros diseñados para cada aplicación.

Las aplicaciones se analizan e implantan con entera independencia unas de las otras, y los datos no se suelen transferir entre ellas. Con este planteamiento se produce una ocupación inutol de memoria así como un aumento de los tiempos de procesos, así como enormes inconsistencias en estos sistemas, ya que la actualización de los ficheros hay que hacerlas una a una, en los diferentes bloques. Así, la depedencia física de los datos y el soporte hace que la flexibilidad y la adaptabilidad sea bastante escasa.

1.6 Sistema de Bases de Datos

Los sistemas de ficheros no son optimos cuando se presentan demandas inesperadas de información o cuando se pretende tener un verdadero sistema de información orientado a la toma de decisiones. Necesidad de una gestión más racional del conjunto de datos, surgiendo así un nuevo enfoque que se apoya sobre una base de datos.

Sistema de base de datos: Es un sistema de mantenimiento de registros por ordenador, cuyo proposito general es registrar y mantener la información por separado. Se compone de datos, software, hardware, y usuarios.

Definición de Base de Datos: Colección compartida de datos, lógicamente relacionados, junto con una descripcion de esos datos, que están diseñados para satisfacer las necesidades de información de una organización.
Los sistemas de bases de datos presentan una multitud de ventajas frente a los sistemas clásicos de ficheros.

- DATOS: Una ventaja es la independencia de los datos respecto a los tratamientos, y viceversa. Un cambio en cualquiera de ambos no supone un cambio de la bas de datos por completo. Proporciona flexibilidad de adaptaciónsin excesivos costes. Los datos también proporcionan una mejor disponibilidad de los datos para el cto de usuarios. Ya no hay usuarios propietarios de información, ya que esta se comparte entre el cto de aplicaciones. La eficiencia a la hora de la recogida de datos también es mayor, ya que sólo se recogen y validan los datos una vez.
- Los resultados: Mayor coherencia, ya que en los tratamientos se utilizan los mismos datos y todos ellos son coherentes y comparables. Tienen un mayor valor informativo, puesto que en cto es superior a la suma del valor informativo de los elementos individuales que lo constituyen. Mayor documentación de la información normalizada.
- Los usuarios: Acceso más rápido y sencillo de los usuarios finales, más facilidad para compartir los datos por el cto de usuarios, e implantación larga y dificil.

También presenta desventajas, como instalación costosa, necesidad de personal especializado para la creación y mantenimiento de la base de datos, y una implantación larga y dificil, por lo que aparece una falta de rentabilidad a corto plazo.

2

SISTEMA DE GESTIÓN DE BD

2.1 Definicion de SGBD

Un SGBD es un software que permite a los usuarios definir, crear y mantener una base de datos. Interfaz entre usuario y BD. Necesita ser definido (especificar los tipos de datos, las estructuras y las restricciones para los datos), construir (almacenar los datos concretos sobre algun medio de almacenamiento controlado por el SGBD), y manipular (consultar la base de datos para recuperar unos datos específicos, actualizar la BD y generar informes.

2.2 Objetivos de un SGBD

- Independencia física: capacidad de modificación de las estructuras de almacenamiento sin afectar a las definiciones lógicas de los datos.
- Independencia lógica: capacidad de modificación de las estructuras lógicas de los datos sin modificar las aplicaciones de los usuarios.
- Manipulacion de los datos: Manejo de los datos utilizando lenguajes procedimentales o no.
- Eficacia de los accesos a los datos: poder acceder a los datos de la mejor manera posible segun cada usuario.
- Administración centralizada de los datos: permite una administración coherente y eficaz de los datos.
- No redundancia de los datos: compartición de las estructuras de los datos por usuario/aplicacion
- Coherencia de los datos: Los datos están sujetos a reglas de integridad que deben cumplir.
- Compartición de los datos: acceso concurrente aplicacion/usuario
- Seguridad de los datos: protección de los datos contra accesos no autorizados.

2.3 Operaciones típicas de un SGBD

Que afectan a la totalidad de los datos: creación, reestructuración, y consulta a la totalidad.

Que afecten a ciertos datos: Actualización y consulta selectiva.

2.4 Funciones de un SGBD

Descripción o Definición: Debe permitir al diseñador especificar:

- los elementos de datos de la BD
- estructuras de datos

- las relaciones que existen entre los datos
- las reglas de integridad semantica
- las características de tipo físico
- las vistas lógicas de usuario.

se realiza con un DDL específico del SGBD y debe suministrar los medios para definir los tres niveles de la arquitectura (externa, lógica e interna).

Manipulación. La consulta a la BD puede ser de dos tipos; en totalidad, o consulta selectiva. La funcion de actualización incluye insercion de nuevos elementos, borrado cuando haya desaparecido algunos elementos, y modificación de los registros que han sufrido cambios.
Esta función se llevará a cabo con un DML que facilite los instrumentos para la realización de tareas.

Control: Reune todas las interfaces DCL que necesitan los diferentes usuarios para comunicarse con la BD y proporcionar un cto de procedimientos para el administrador tales como servicio, seguridad física, y protección contra accesos no autorizados.

2.5 Componentes del entorno de un SGBD

Maquina Son

- Software: herramienta de 4º generacion SQL
- Hardware: Los SGBD tienden a ser intependientes al SO y al Hardware

Humana Son

- Procedimientos: Instrucciones o reglas que gobiernan en el diseño y utilización de la BD
- Personas: Admin de Datos, Admin de BD, Diseñador, Programador, y usuarios finales.

Datos Lazos de union entre personas y máquinas.

2.6 Clasificación de los SGBD

2.6.1 Segun el MD en el que se basa

- Relacional
- Orientado a Objetos
- Objeto-Relacional

2.6.2 Numero de usuarios a los que da servicio

- Monousuario: da servicio a un sólo usuario, normalmente en ordenadores personales.
- Multiusuario: Atienden a varios usuarios al mismo tiempo.

2.6.3 Lugar donde se almacenan los datos

- Centralizado: Los datos se almacenan en un sólo sitio.
- Distribuido: La base de datos real y el software del SGBD pueden estar distribuidos por varios sitios y conectados por una conexion de red.

2.6.4 Propósito

- General: se diseña y se construye el SGBD para varias aplicaciones.
- Específico: Se construye el SGBD para una sola aplicación.

Las bases de datos tienen las siguientes características: separación entre los programas y los datos, manejo de multiples vistas de usuario, y empleo de un catálogo para el almacenamiento de la BD.
Para ello se propoe una arquitectura en 3 niveles:

1. Nivel interno: esquema interno que describe la estructura física de almacenamiento de la BD.
2. Nivel conceptual: define la estructura de toda la Base de Datos sin especificar las estructuras necesarias.
3. Nivel Externo

La definicion de la BD en cada uno de los niveles se denomida esquema. Los tres esquemas no son más que descripciones de los datos, lo únicos datos que existen realmente están en el nivel físico.

2.7 Independencia de los datos

Podemos definir como la capacidad de modificar el esquema en un nivel superior del SBD sin tener que modificar el nivel inmediatamente superior. Para ello definimos dos tpos de intependencia de datos:

Independencia lógica de los datos: Capacidad de modificar el esquema conceptual sin tener que modificar los esquemas externos ni programas de aplicación. Podemos usarlo tanto para ampliar, como reducir la BD.

Independencia física de los datos: Capacidad de modificar el esquema interno sin tener que modificar el conceptual.

2.8 Modelo de Datos

El modelo de datos es un cto. de conceptos que permiten describir a distinotos niveles de abstraccion la estructura de una base de datos, a lo cual llamamos esquema. Segun el nivel de abstracción en que se encuentre la estructura, el modelo que permite su descripción será:

- Externo: representar los datos que necesita cada usuario en particular con las estructuras propias del lenguaje de programación a emplear.
- Global: describir los datos para el cto. de usuarios, podríamos decir que es la información a nivel de empresa.
- Interno: orientados a la máquina, siendo sus elementos de descripción punteros, índices...

2.8.1 Modelos Globales

- Modelo conceptual o de Alto Nivel: Facilitan la descripción global del cto. de información de la empresa con independencia de la máquina. Son modelos de análisis, no de implementación.
- Modelo convencional: Orientados a describir los datos a nivel lógico para el SGBD, por lo que sus conceptos son propios de cada SGBD

Los modelos de datos, tanto lógicos como físicos, son intrumentos que se aplican a los datos para obtener el esquema.
Tenemos que distinguir entre los conceptos esquema y ocurrencia. El esque,a es la descripción de la estructura de la BD, invariante en el tiempo. Una ocurrencia o instancia de ese esquema son los datos almacenados en ese momento, y puede cambiar en el tiempo.

2.8.2 Definicion de Modelo de Datos

Un MD es una colección integrada de conceptos para describir y manipular datos, las relaciones existentes entre los mismos, y las restricciones aplicables a todos los datos, todo ello dentro de una organización. Las propiesades de un MD son:

- Estáticas
  - Elementos permitidos: Mo son los mismos para todos los MD, pero generalmente son Objetos, Asociaciones entre objetos, propiedades de los objetos, dominios...
  - Elementos NO permitidos: Restricciones inherentes (propias del propio modelo) y restricciones de integridad (facilidades para que el desarrollador pueda representar el esquema lo más fielmente posible a la realidad.)
- Dinámicas: Los valores que toman los distintos objetos de un esquema en un momento determinado t_i reciben el nombre de ocurrencia del esquema (BD_i)

  $$t_i \neq t_j \rightarrow BD_i \neq BD_j$$

  La componente dinámica del modelo consta de un cto. de operadores que se definen sobre la estructura del correspondiente MD ya que no todas admiten las mismas operaciones. La aplicación de una ocurrencia de un esquema transdorma esta otra en una ocurrencia, tal que $O(BD_i) = BD_j$. Una operación tiene dos componentes:
  - Localización: Consiste en localizar una ocurrencia de un pbjeto indicando el camino o bien un cto. de ocurrencias especificando una condición.
  - Accion: se realizan sobre las ocurrencias previamente localizadas mediante una operación de localizaciones y pueden consistir en una recuperacion o una actualización.

2.8.3 Restricciones de integidad

Las restricciones de integridad o semanticas son condiciones que limitan el cto. de ocurrencias válidas de un esquema. Nos referimos por semánticas al dignificado de los datos, y por integridad a la correccion de los datos y a su consistencia respecto al mundo real del que proceden.
las ventajas de la integracion de las restricciones junto a la definicion de los datos son:

- Las restricciones se guardan una sóla vez, y no entre diferentes programas, así nunca hay inconsistencia
- Elimina cargas de programación al no tener que definirlas para cada programa
- El diseñador será quien lo haga y las guarde junto con las definiciones de los datos que deben cumplirlas, evitando redundancias.
- Las RI deben estar descritas en el esquema de la BD y los MDdeben tener herramientas para representarlas
- Los SGBD que soportan a estos MD tienen que reconocerlas y poder gestionarlas.

2.8.4 Clasificación de los modelos de datos

Modelos de Alto Nivel o conceptuales Consta de conceptos muy cercanos al mundo real, como son los ctos. de entidades y las relaciones entre ellas. Modelo ER.

Modelos de Implementación o lógicos Proporcionan conceptos entendidos por los usuarios finales. Son los más usados por los SGBD comerciales. Modelo relacional de datos.

Modelo de bajo nivel o físico Describe los detalles de almacenamiento de los datos en el ordenador. MD para usuarios especializados.

2.9 Lenguajes

Necesitamos diferentes tipos de lenguajes y procedimientos para comunicarnos con la BD.

DDL (Lenguaje de Definicion de Datos) Lenguaje descriptivo que permite al DBA y/o usuarios describir el esquema completo de una BD (Lógico, externo y físico). No necesitan apoyarse en ningún lenguaje de programación, son autocontenidos. La compilación de las órdenes es un cto. de tablas almacenadas en el catálogo del sistema, también llamado DD o Diccionario de Datos.

DML (Lenguaje de Manipulacion de Datos Proporciona un cto. de operaciones que nos permiten manipular y consultar los datos almacenados en la BD. Los DML pueden ser:

- Procedimentales: Lenguaje que indica al sistema qué datos necesita y cómo deben recuperarse
- No procedimentales: Lenguaje que indica al sistema que datos necesita pero no como deben recuperarse.

DCL (Lenguaje de contro de Datos) Lenguaje que permite llevar un contro de la seguridad del sistema, tanto sobre usuarios como sobre los datos.

SQL Lenguaje de Consulta estructurado. Nace junto a los SGBD basados en el modelo de datos relacional. Es el lenguaje más utilizado tanto en los SGBD como en los SGF.
Está basado en Álgebra y Cálculo relacional. Es completo (DML, DDL y DCL) y su uso es interactivo y embebido.

3

DISEÑO CONCEPTUAL

3.1 Entidades

Entidades una cosa del mundo real con esistencia independiente. Una entidad puede ser objetos con existencia física o objetos con existencia conceptual.

3.2 Atributo

Cada entidad tiene propiedades específicas, llamadas atributos, que las describen. Una entidad particular tendrá un valor para cada uno de sus atributos. Los valores constituyen una parte decisiva de los datos. Los tipos de atributo son:

Simples o Compuestos Los atributos compuestos pueden dividirse en componentes más pequeños, que representan atributos más básicos con su propio significado independiente. El valor es la concatenación de los vaores del atributo. Los atributos simples son atributos no divisibles.

Monovaluados o Multivaluados Los atributos monovaluados tienen un sólo valor para una entidad en particular. Los atributos multivaluados pueden tener distinto numero de valores para cada entidad. Puede tener límite inferior o superior.

Almacenados o Derivados En los atributos derivados podemos calcular su valor a partir de otra información almacenada, es decir, de esos Atributos almacenados.

Nulos Nulos o NULL son aquellas entidades que no poseen ningún valor aplicable a ningún atributo.

3.3 Tipo de Entidad

Las entidades que poseen los mismos atributos se agrupan formando un cto. de entidades del mismo tipo.

- Un tipo de entidad se describe por su nombre y la lista de nombres de sus atributos (esquema).
- Una ocurrencia de un tipo de entidad es una entidad con valores para cada atributo.

$$Persona(32598784P, Perez, Florinda, 658987456, AvdEstherGadeschiS/N)$$

El cto. de ocurrencias forma la instancia.

3.4 Atributos Clave

Una clave candidata de una relación es un cto. de atributos que identifica unívocamente y mínimamente cada tupla de la relación.
Una relación puede tener más de una clave candidata, entre las cuales se debe distinguir:

Clave primaria Aquella clave candidata que el usuario escogerá por consideraciones ajenas al modelo relacional para identificar las tuplas de la relación. Se representa SUBRAYADA.

Clave Alternativa Son claves que no han sido elegidas como clave primaria

3.5 Dominio

Conjunto de valores que puede tomar un atributo. Todo atributo atómico tiene asociado un único dominio sobre el cual puede tomar valores válidos.

3.6 Relaciones

Una relación es una asociación entre entidades que están relacionadas de alguna manera en el mundo real tratado. Un tipo de relacion es un cto. de relaciones del mismo tipo.
Los tipos de relaciones segun:

Segun el grado : numero de entidades que participan en la relación.

- Binario: Intervienen dos tipos de entidad.
- Ternario: Intervienen tres tipos de entidad.
- Reflexio: interviene un único tipo de entidad.

Segun el rol : Todo tipo de entidad que participa en un tipo de relación juega un papel específico en la relación.

Cardinalidad : limitaciones en las combinaciones de entidades que pueden aparticipar en una misma entidad. Pueden ser 1:1, 1:N, o M:N

Participación : Especifica si toda la extensión de un tipo de entidad participa en un tipo de relación o sólo parte.

3.6.1 Tipo de entidad debil

Aquella que no tiene suficientes atributos para formar una clave primaria. Se representa por un doble rectángulo.
Una entidad débil siempre tiene una restricción de participación total en la relación en la que le une a su entidad dominante o fuerte. Dependencias:

- En existencia: no puede existir sin la ocurrencia de la entidad fuerte.
- En identificación: las ocurrencias del tipo de entidad no se pueden identificar.

3.6.2 Generalización y especificación

Se emplea para resaltar las características comunes de varios conjuntos de entidades, o bien para especificar de forma más correcta las características de alguno de sus subconjuntos.
Diremos que el cto. de entidades A es una especialización del cto. de entidades B si el cto. de entidades A está incluido dentro del cto. de entidades B. Se puede decir que B es una generalización de A.

3.6.3 Agregación

Herramienta que se utiliza para expresar relaciones entre relaciones o relaciones entre relaciones y cto. de entidades.
se puede considerar como un tipo de entidad genérica sin especificar su estructura interna, como una caja negra donde sólo debemos conocer las claves primarias de los tipos de entidades a los que integra.

3.6.4 Notación

- Tipos de entidades: Se representan con rectánguos.
- Atributos: Se representan con elipses/circulos.
- Atributo Clave primaria: subrayado

3.7 Diseño Conceptual

Las etapas del diseño conceptual son:

Análisis de requisitos : Pretende analizar de forma más minuciosa y sistemática las especificaciones de requisitos, para identificar más claramente los requisitos relacionados con la información manipulada por el sistema. Eliminaremos:

- Ambiguedades en los requisitos recabados
- Complementar los requisitos
- Dotarlos de una estructura
- Entender realmente el significado de todos los términos

Diseño del esquema conceptual Realizar un refinamiento y estructuración sucesivos del esquema percibido para obtener el esquema conceptual

La utilización de un MD de alto nivel independiente a la implementación proporciona

- Entendimiento completo de la estructura, semantica, interrelciones y restricciones de la BD.
- Descripción del contenido de la BD
- Mejor para obtener el esquema conceptual: es más general y expresivo, y sirve de vehiculo de comunicación entre usuarios, diseñadores y analistas.

El paso de esquema descriptivo a un primer esquema conceptual tiene dos enfoques:

- Linguistico:
  Sustantivo que actúa como sujeto → tipo de entidad
  Nombre propio → ocurrencia de un tipo de entidad
  Verbo transitivo → tipo de relacion
  Preposicion o frase preposicional entre dos nombres → tipo de relación o asociación entre una entidad y atributo.
- Categorización de los objetos:
  Si un objeto tiene más propiedades, además de su nombre, y/o describe un tipo de objeto de datos con existencia autónoma es conveniente usar un tipo de entidad para representarlo.
  Si un concepto tiene una estructura simple sin propiedades y/o describe a un objeto de datos al que se le asigna un valor, es mejor representarlo como un atributo de otro concepto al cual se refiere.
  Si un concepto proporciona una relación lógica entre dos o más entidades, y/o hace posible la seleccion de una entidad a través de una referencia a un atributo de otra entidad conviene representarlo como relación.

3.7.1 Documentación

Necesidad de una documentación para la interpretación del esquema conceptual. La documentación se puede organizar como un diseño de datos. Se requieren dos tablas: Entidades, con Nombres, Atributos e Identificadores (CP), y Relaciones, con Nombres, Entidades involucradas, y atributos.
Las restricciones del universo del discurso se agrupan en una tabla.

3.7.2 Problema en el diseño conceptual

El esquema es correcto cuando se usan adecuadamente los elementos del modelo ER:

- Sintacticamente los conceptos se representan correctamente en el esquema.
- Semanticamente correctos: Los elementos se usan de acuerdo a sus definiciones.
- Redundancia: Un esquema es redundante cuando una relacion R_1 entre dos entidades poseen el mismo contenido de información que una ruta de relaciones $(R_2, R_3, etc...)$ que conecte los mismos pares de ocurrencias que R_1,o cuando el valor de un atributo puede calcularse del valor de otro atributo.

4

MODELO DE DATOS RELACIONAL

4.1 Estructura

Representar a una BD como un cto. de tablas o relaciones. Una BD relacional se divide en tres partes y cada una de ellas tiene sus propios términos especiales:

- Estructura: cto. de relaciones que forman la BD.
- Integridad: cto. de restricciones de la BD.
- Manipulacion: cto. de operaciones sobre la información contenida en la BD.

4.2 Atributo

Los atributos de una relación se corresponden con los nombres de las columnas de dicha relación. Toda relación está formada por un cto. de atributos que describen los significados de las entradas de datos en las columnas. Los nombres de atributos deben ser únicos.

4.3 Esquema

El esquema de una relación R denotado por $R(A_1, A_2, A_3...A_n)$ está constituido por un nombre de relación R y una lista de atributos $A_1, A_2, A_3...A_n$. Los atributos de un esquema de relación son a su vez un cto. y no una lista, no hay un orden específico de aparición. El diseño de una BD relacional consiste en un cto de esquemas de relaciones, llamado esquema relacional de la BD.

4.4 Tuplas e instancias

Son registros de una relación y suelen tener valor para cada uno de los atributos de la relación. Representan objetos y su relación, clases. Todas las tuplas de una misma relacón son diferentes por lo tanto el valor del identificador único ha de ser distinto para cada una de ellas. El cto de todas las tuplas de una relación en un momento determinado se denomina instancia.

4.5 Dominios

Cada componente de cada tupla debe ser atómico y de algun tipo elemental de datos. Un dominio es indivisible en lo que el modelo relacional de datos se refiere. Un dominio es un cto. de valores todos del mismo tipo y es donde se extraen los valores reales para los atributos. Un atributo está definido cobre un único dominio. Deben estar especificados en la descripción de la BD.

4.6 Relaciones

Una relación es un subconjunto del producto cartesiano de un listado de dominios, no necesariamente todos distintos $X_{i=1}^{n} D_i$
Toda relación tiene un nombre, un esquema, y una instancia. El esquema está formado por un cto. fijo de atributos que se representan por parejas atributo-dominio $(A_1 : D_1), (A_2 : D_2), ..., (A_n : D_n)$ donde n es el grado o número de atributos de la relación.
La instancia es el cto. de tuplas en un momento dado. Es normal omitir el nombre del atributo en contextos informales o cuando el usuario lo conozca.

4.6.1 Propiedades

- No existen tuplas repetidas
- Las tuplas no están ordenadas
- Los atributos no están ordenados
- Todos los valores de los atributos son atómicos.

4.6.2 Tipos de relaciones

Persistentes Son aquellas relaciones cuya definicion permanece en la base de datos, borrandose sólamente mediante acción explícita del usuario.

Relaciones Base: Existen por si mismas, no en funcion de otras relaciones, y se crean especificando explícitamente su esquema de relacion.
Vistas: Relaciones derivadas que se definen dando un nombre a una expresión de consulta. Se podría decir que son relaciones virtuales, en el sentido de que no tiene datos almacenados, si no que lo único que se almacena es su definicion en términos de otras relaciones con nombre, las cuales pueden ser relaciones base, o otras vistas instantáneas.

Instantaneas: Relaciones derivadas al igual que las vistas, es decir, que se definen en términos de otras relaciones, aunque estas tiene datos propios almacenados, resultado de ejecutar la consulta especificada o de guardar una relación base.

Temporales A diferencia de las relaciones persistentes, una relacón temporal desaparece de la BD en un cierto momento sin necesidad de una acción de borrado específica del usuario. Son resultados de consultas, resultados intermedios (de alguna expresion anidada), y resultados temporales.

4.6.3 Clave primaria

Toda relación tiene un identificador único formado por un subconjunto del cto. de atributos de su esquema.
Sea R el esquema de la relación y CP su clave primaria, entonces se cumple que CP está contenido en R.
Una relacion puede tener más de un identificador. La clave primaria es elegida por el DBA entre las claves candidatas.

4.6.4 Regla de Integridad de las Relaciones

El modelo relacional tiene dos reglas de integridad:

Regla de integridad de las relaciones ningun componente de la CP de una relación puede sportar nulos

Regla de Integridad referencial Se especifica entre dos relaciones y sirve para mantener la consistencia entre tuplas de las dos relaciones. Una tupla en una relación que haga referencia a otra relación deberá referirse a una tupla existente en esa relación.

4.6.5 Clave foranea

Atributo o cto. de atributos de una relación R_2 cuyos valores deben concordar con los de la clave primaria de una relación R_1, donde R_1 y R_2 no tienen que ser distintas.
Si una relación R_2 tiene un descriptor que es una clave candidata de la relación R_1, todo valor de dicho descriptor debe concordar con un valor de la clave candidata referenciada de R_1 o bien ser nulo.
Formalmente, toda clave foranea debe satisfacer estas dos propiedades independientes en el tiempo: Cada valor de CF es nulo del todo o bien no nulo del todo, y existe una relación base R_1 con CP tal que cada valor no nulo de CF es identico al valor de CP en alguna tupla de R_1.
CP y CF deben estar definidas sobre el mismo dominio.
La CF no necesita ser un componente de la CP de la relacion que la contiene.
Una relacion puede incluir una CF cuyos valores, no nulos, deben concordar con los valores de la CP de esa misma relación.
Las CF aceptan nulos.
Las CF nos permiten interrelacionar relaciones.

4.6.6 Restricciones

En el modelo relacional, al igual que en otros modelos, existen restricciones, es decir, estructuras u ocurrencias no permitidas.

Restriccions inherentes Son obligadas por el propio modelo

- No hay tuplas iguales
- el orden de las tuplas no es significativo
- el orden de los atributos no es significativo
- cada atributo sólo puede tomar un único valor del dominio sobre el que está definido.

Restricciones semanticas o de usuario Facilidades que el modelo ofrece a los usuarios con el fin de que estos puedan reflejar en el esquema la semantica del mundo real de la forma más precisa posible.

- Clave primaria: permite declarar un atributo o un cto. de atributos como CP de una relación, su valor no podra repetirse, ni ser nulo.
- Unicidad: Mediante la cual se indica que valores en un cto. de atributos no pueden repetirse en una relación.
- Obligatoriedad NOT NULL: los atributos no pueden ser nulos.
- Integridad referencial: Si una relación R_2 tiene un descriptor que es una clave candidata de la relación R_1, todo valor de ese descriptor debe concordar con un valor de la clave candidata referenciada de R_1 o bien ser nulo.

4.6.7 Valores nulos en el modelo relacional

Se puede definir el valor nulo como una señal utilizada para representar información desconocida, inaplicable, e inexistente, no válida, no proporcionada, indefinida, etc.
La necesidad de valores nulos o marcas en las bases de datos es evidente por diversas razones, como crear tuplas con valores desconocidos por el momento, añadir un nuevo valor a la tupla, o atributos inaplicables en ciertas tuplas.
El tratamiento de nulos exige:

- Operaciones de comparación: se plantea el problema de saber si dos valores nulos son iguales o no.
- Operaciones aritméticas: se considera nulo el resultado de sumar, restar, multiplicar, o dividir cuando alguno de los operandos toma valores nulos.
- Operaciones algebráicas: Hay problemas en algunas operaciones como la union externa.
- Funcion de agregación: Los valores nulos no entran a formar parte de las tuplas evaluadas. Media aritmética.

4.7 Lenguaje de Consultas

Sirven para que el usuario manipule la información de la BD. Se clasifican en:

Procedimentales El usuario indica las operaciones y la secuencia en que se deben realizar. Álgebra relacional.

NO Procedimentales El usuario indica lo que quiere obtener sin especificar cómo obtenerlo. Cálculo relacional.

Los lenguajes de consulta pasaron a llamarse DML ya que también permite insertar, eliminar, y modificar la información. Los SGBD disponen de DML, DCL, DDL, y lenguajes huespedes como por ejemplo SQL.

4.8 12 reglas de Codd

1. Representación de la Información: Toda información debe representarse en forma de tabla, principio básico del MR.
2. Acceso Garantizado: todo dato debe ser accesible mediante una combinación de un nombre de tabla, un valor de su clave, y el nombre de una columna.
3. Tratamiento sistemático de valores nulos: han de ser tratados sistemáticamente por el sistema, el cual ha de ofrecer las posibilidades necesarias para su tratamiento.
4. Catálogo en linea basado en el modelo relacional: la representacion de la descripción de la BD debe ser igual a la de los otros datos, y su acceso debe realizarse por el mismo medio del lenguaje relacional.
5. Sublenguaje de datos completo: debe existir un lenguaje que permita un completo manejo de la BD.
6. Actualización de vistas: toda vista teóricamente autorizable debe poder ser actualizada por el sistema
7. Insercion, actualización, y eliminación de alto nivel: todas las operaciones de manipulación de datos deben operar sobre ctos. de filas.
8. Independencia física de los datos: el acceso lógico debe mantenerse incluso cuando cambien los métodos de acceso o la forma de almacenamiento.
9. Independencia lógica de los datos: los programas de aplicación no deben verse afectados por cambios realizados en el esquema de la BD.
10. Independencia de integridad: las reglas de integridad de una BD deben ser definibles por medio del sublenguaje de datos relacional y habrá de almacenarse en el DD de la BD, no en los programas de aplicación.
11. Independencia de distribución: debe existir un sublenguaje de datos que pueda soportar BD distribuidas sin alterar los programas de aplicación cuando se distribuyen datos.
12. Regla de la no subversión: si un SGBD soporta un lenguaje de bajo nivel que permita el acceso fila a fila, este no puede utilizarse para saltarse las reglas de integridad expresadas por medio del lenguaje de más alto nivel.

5

ÁLGEBRA Y CÁLCULO RELACIONAL

5.1 Álgebra Relacional

El modelo de datos relacional tiene asociado dos partes:

- Estática: Formada por las estructuras que almacenan los datos y las restricciones que soportan estos datos.
- Dinámica: Permite la transformación entre estados de la BD.

El modelo relacional actúa sobre conjuntos de tuplas y mediante lenguajes de manipulación relacionales asocia una sintaxis concreta a las operaciones. Se dividen en dos tipos:

- Algebráicos: Los cambios de estado se especifican mediante las operaciones. Los operandos son relaciones y el resultado es una relación. Se conoce como álgebra relacional a "una colección de operaciones que sirven para manipular relaciones enteras."
- Predicativos: Los cambios de estado se especifican mediante predicados que definen el estado objetivo sin indicar las operaciones que hay que realizar para llegar al mismo. Basados en cálculo de predicados.

El álgebra relacional consta de un cto. de operadores que aplicados a las relaciones dan como resultado nuevas relaciones. Propiedad de cierre.
Una operación consiste en aplicar un operador O a una relación R para obtener otra relación R_0 a la cual se le pueda aplicar otro operador.
El operador se aplica a una extensión de R, r(R), no al esquema R. Codd definió ocho operadores divididos en dos grupos de cuatro:

- Operadores conjuntistas tradicionales: union, intersección, diferencia, y producto cartesiano.
- Operadores relacionales especiales: Selección, proyección, producto natural, division.

Clasificación de operadores:

Operadores fundamentales Son esenciales, y no pueden obtenerse a partir de otros. Pueden ser unarios o binarios. Sin ellos el álgebra no sería un lenguaje completo. Estos son proyección, selección, producto cartesiano, union de conjuntos, diferencia de conjuntos.

Operadores adicionales Son operadores binarios y se caracterizan porque simplifican las expresiones, ya que las operaciones que ellos representan sí pueden ser expresadas por combinación de operadores fundamentales. Son Interescción de ctos., productos de reunion y división.

Los operadores fundamentales, atendiendo al número de operandos de cada operador, pueden dividirse en:

- Unarios: si el operador tiene una única relación como operando. Proyección y selección.
- Binarios: Si el operador tiene dos relaciones como operandos. Producto cartesiano, unión y diferencia de ctos.

5.1.1 ¿Para qué sirve el álgebra?

Obtención de datos utilizando sus operaciones. Ayuda a escribir expresiones simplificando alguna de ellas. Patrón para otros lenguajes, ya que podemos decir que un lenguaje es completo si cualquier expresión puede expresarse en álgebra.

5.2 Cálculo relacional

La diferencia fundamental entre un lenguaje algebráico y un lenguaje predicativo es que en el primero hay que especificar qué operadores se tienen que aplicar a las relaciones para obteres el resultado, mientras que en los segundos sólo es preciso indicar cual es el resultado que se quiere obtener, expresándolo mediante cálculo de predicados de primer orden.
Se divide en calculo relacional de dominios y de tuplas.

5.2.1 Cálculo relacional de Tuplas

Se deduce del cálculo de predicados de primer orden, dando las siguientes interpretaciones:

- Las variables se asocian a tuplas
- Las constantes se asocian a los valores de los dominios subyacentes a los atributos
- Los operadores son los permitidos de comparación, los lógicos $\neg$, $\wedge$, $\vee$, $\forall$ y $\exists$.

5.2.2 Calculo relacional de dominios

El cálculo relacional orientado a dominios está basado en el cálculo de predicados de primer orden. Las variables de dominio se definen sobre un dominio, tomando en cáda momento un valor de este.

5.3 Poder expresivo del Algebra y del Cálculo

- El álgebra y el cálculo, tanto de tuplas como de dominios, tiene el mismo poder de expresión
- Tenemos que restringir las expresiones del cálculo a expresiones sanas. Pasamos de fórmulas insanas a sanas, restringiendo el dominio de la variable t. DOM(t) = S.
- El álgebra relacional no presenta estos problemas, pues los operadores algebráicos son finitos y los resultados de estas operaciones también.
- Las expresiones algebráicas son sanas.
- Cualquien expresión de cálculo equivalente a otra de álgebra será una fórmula sana.
- Se denomina cálculo restringido de tupla o dominio a todas las expresiones de cálculo que tienen alguna expresión algebráica equivalente.
- Cualquien expresión de álgebra se puede expresar en cálculo restringido por lo que ambos lenguajes son equipotentes.

6

DISEÑO LÓGICO

6.1 Dependencias funcionales

La nocion de dependencia funcional es una generalización del concepto clave. La teoría de normalización se basa en este concepto. Una dependencia funcional no puede ser demostrada, pero sí afirmada por observación del mundo real al que pertenece. Nos muestran algunas interrelaciones importantes entre los atributos. Son invariantes en el tiempo mientras ni cambie el problema en el mundo real. Forma unas restricciones en el cto. de las relaciones.

Dada una relación R se dice que el atributo R.y $\in$ R es funcionalmente dependiente de otro atributo R.x $\in$ R, y se expresa R.x $\rightarrow$ R.y si y solo si cada valor de R.x tiene asociado a él exáctamente un valor de R.y para cualquier extensión de la relación R.

Se tienen en cuenta atributos de una misma relación. Los atributos funcionalmente dependientes pueden ser simples o compuestos y pueden formar parte de la clave primaria de alguna clave candidata. No se hace referencia al número de tuplas en las que el atributo R.x tiene un mismo valor.

Si x es una clave candidata, se cumple que $\forall$ y, x $\rightarrow$ y $\nRightarrow$ y $\rightarrow$ x. El diseño de una BD está basado en la teoría de la normalización, y esta a su vez en las dependencias funcionales entre atributos.

6.1.1 Propiedades de las dependencias funcionales

Reflexiva R.b $\subseteq$ R.a $\Rightarrow$ R.a $\rightarrow$ R.b, $\forall$ R.a $\Rightarrow$ R.a $\rightarrow$ R.a

Aumentativa a $\rightarrow$ b $\Rightarrow$ (a+c) $\rightarrow$ (a+b)

Transitiva a $\rightarrow$ b $\wedge$ b $\rightarrow$ c $\Rightarrow$ a $\rightarrow$ c

Union a $\rightarrow$ b $\wedge$ a $\rightarrow$ c $\Rightarrow$ a $\rightarrow$ (b+c)

Pseudotransitiva a $\rightarrow$ b $\wedge$ (b+c) $\rightarrow$ d $\Rightarrow$ (a+c) $\rightarrow$ d

Descomposicion a $\rightarrow$ bc $\Rightarrow$ a $\rightarrow$ b $\wedge$ a $\rightarrow$ c

6.2 Dependencias multivaluadas

- Si x,y $\in$ R y x $\rightarrow\!\!\!\rightarrow$ y en R, y $\exists t_1, t_2$ / $t_1[x] = t_2[x] \rightarrow \exists t_3, t_4$ / $z = R - (x \cup y)$
- Si x $\rightarrow\!\!\!\rightarrow$ $y \Rightarrow x \rightarrow\!\!\!\rightarrow y|z$
- Un valor de x puede tener varios valores de y, los cuales se relacionaran con dicho x. A cada valor de x pueden corresponder varios valores de y y varios valores de z, y que estos sean independientes.

6.2.1 Propiedades de las dependencias multivaluadas

Repeticion R.a → R.b ⇒ R.a →→ R.b

Complementacion R.a →→ R.b ⇒ R.a →→ (R-(R.a∪R.b))

Aumento multivaluado R.a →→ R.b ∧ R.c ∈ R.d ⇒ R.(a+d) →→ R.(b+c)

Trans. Multivaluada R.a →→ R.b ∧ R.b →→ R.c ⇒ R.a →→ R.(c-b)

Condensacion R.a →→ R.b, R.c ∈ R.b, R.d → R.c ∧ R.d ∪ R.b ⇒ ∅ R.a → R.b

Union R.a →→ R.b ∧ R.b →→ R.c ⇒ R.a → R(b+c)

Interseccion R.a →→ R.b ∧ R.b →→ R.c ⇒ R.a → R(b∩c)

Diferencia R.a →→ R.b ∧ R.b →→ R.c ⇒ R.a → R(b-c)

6.3 Dependencias de reunion

Dependencia entre relaciones y su descomposición sin pérdidas de 3 o más relaciones. Casos raros y difíciles de detectar.

6.4 Peligros en el diseño

Para realizar un buen diseño sin redundancias ni pérdidas de información debemos tener en cuenta:

- Diseñar el esquema de relación donde sea fácil explicar su significado. Los atributos deben ser propiedad del mismo cto. de entidades o relaciones existentes entre dichos ctos.
- Los esquemas de relación deben estar diseñados de tal forma que no existan anomalías de inserción, eliminación, o modificaciones en las relaciones.
- Evitar al diseñar la posibe inserción de nulos.
- Las relaciones deben poder interrelacionarse por atributos que sean CP o CF

6.5 Proceso de Normalización

Se basa en que los datos son independientes de las aplicaciones que les gestionan, generando un numero de relaciones, tales que tengan los atributos necesarios para representar la entidad o la relación entre relaciones.
Las ventajas de la normalización son;

- Facilidad de uso: datos agrupados en tablas que identifican un objeto o una relación.
- Flexibilidad: relacionar información de distintas tablas con álgebra relacional.
- Precisión: la interrelación entre tablas consigue información exacta.
- Seguridad: facili mantener controles de acceso a los datos.
- Facilidad de implementación: las tablas se almacenan en ficheros planos.
- Independencia de datos: estructuras de datos y aplicaciones son independientes.
- Claridad: la información se representa de forma sencilla.
- Facilidad de gestión: fácil al estar en álgebra y cálculo.
- Mínima redundancia: sólo necesaria para interpretar las tablas.
- Máximo rendimiento: sólo se usa la información necesaria.

6.6 Formas normales de Codd

1FN R está en 1FN si sus dominios no tienen elementos que sean conjuntos.

2FN R está en 2FN si está en 1FN y todo atributo no primario tiene una dependencia funcional plena con la CP.

3FN R está en 3FN si está en 2FN y ninguno de sus atributos no primarios tienen dependencia transitivas respecto de la CP.

FNBC R está en FNBC si está en 3FN y si toda dependencia funcional tiene a una clave por determinante.

PRÁCTICAS

7

SQL*PLUS

7.1 Formateo de consultas

7.1.1 Cabeceras y pies de página

btitle Coloca un título que puede estar compuesto de varias lineas en la parte inferior de cada página del informe.

ttitle Coloca un título superior, numera las páginas y coloca la fecha actual.

Podemos añadir formato de alineación, y añadir variables como el número de linea (SQL.LNO), de página (SQL.PNO), código de error (SQL.SQLCODE) o de usuario (SQL.USER)...

7.1.2 Formateo de columnas

La orden COLUMN permite cambiar las cabeceras y formatos de cualquier columna o consulta SELECT.

COL[UMN] [nombre — expresión [opción]
Expresión corresponde a columnas virtuales. Las diferentes opciones son algunas como ALI[AS] para renombrar, HEA[DING] para definir las cabeceras de las columnas, o CLE[AR] para eliminar todo valor de la columna. Ninguna cambia la definición de la columna en la tabla.

7.2 Variables del sistema

Cada una de estas variables controla algún aspecto de las operaciones de SQL*PLUS. Se establecen con la órden SET y se comprueba su contenido con SHOW:

SET establece un aspecto del entorno de SQL*PLUS para la sesión actual:

SET variable_sistema valor

variables son ECHO, LIN[ESIZE], NEWP[AGE], SPA[CE]...

SHOW nos muestra el valor de una variable en el sistema.

SHOW variable_sistema [,variable_sistema,...]

Las diferentes variables de SHOW son SPOOL, ALL, PARAMETERS...

7.3 Entrada y salida de datos

7.3.1 Definición de variables

Se realiza con la orden DEFINE, permite especificar variables y asignarles un valor tipo char.

DEFINE [nom_variable] — [nom_variable = 'texto']

Podemos borrar la definición de una variable con UNDEFINE

UNDEFINE nom_variable

7.3.2 Uso de operadores con variables

Las variables pueden no definirse con anterioridad. Para ello usamos los operadores & y &&

&[&]nombre_variable

Variable no definida previamente :

- &var1: es posible dejar indeterminado el nombre de una columna o tabla, y dejar que SQL nos solicite dicho valor antes de ejecutar la sentencia. Si VERIFY está activo nos mostrará el valor anterior y el actual suministrado por el usuario. Con & solicitamos una situación de la variable.
- &&var1: el sistema pide al usuario un valor para la variable var1 quedando definida con ese valor a partir de ese momento.

Variable definida previamente Los dos operadors tienen el mismo funcionamiento en ese caso.

7.3.3 Uso de variables con distintas órdenes

& se puede usar para recibir distintos argumentos cuando utilizamos alguna orden de ficheros. Cuando ejecutamos START y @ es posible pasarles distintos argumentos al fichero a ejecutar.

START — @ nombre_fichero [arg1,arg2,...,argN]

7.4 Ficheros de órdenes

Son ficheros del SO que contienen órdenes de SQL. Cuando el usuario inicia sesión en SQL*PLUS, $Oracle^{TM}$ comprueba si existe en el subdirectorio actual un fichero llamado "login.sql". Si existe lo ejecuta y realiza las órdenes que contenga.

Se ejecutará cualquier acción de SQL contenida en cualquier fichero del subdirectorio, muy util si queremos crear un entorno de trabajo adecuandolo a nuestras necesidades.

8

MANIPULACIÓN DE DATOS

El lenguaje de Manipulación de Datos (DML) se usa para realizar las operaciones de mantenimiento y consulta de bases de datos. Está formado por 4 órdenes

8.1 La orden SELECT

Se utiliza para realizar cualquier consulta a una o varias tablas de nuestro o de otros esquemas siempre que tengamos privilegios. Puede incorporar varias consultas anidadas, afectar a múltiples tablas, usar funciones, etc...

8.1.1 Proyección de una tabla

Cualquier consulta a una o varias tablas de la BD deberá constar como mínimo de una sentencia formada por la orden SELECT y la cláusula FROM.

- Tras la orden SELECT aparecerán la lista de columnas que queremos proyectar, separadas por comas y en el orden que queremos que aparezca.
- En la cláusula FROM indicaremos los nombres de las tablas cuyas columnas queremos seleccionar, tambien separadas por comas.

8.1.2 Selección de filas de una tabla

Podemos seleccionar una serie de tuplas concretas y no todas las que forman la tabla. Este subconjunto de tuplas son aquellas que cumplen unas determinadas condiciones, que pueden tener criterios distintos, y se especifican en la cláusula WHERE. A diferencia de SELECT y FROM esta cláusula es opcional y se coloca a continuación de la cláusula FROM.

El criterio se establece mediante una expresión lógica compuesta por una serie de condiciones, cada una de las cuales toma el valor falso o verdadero, combinando con los operadores lógicos NOT, AND, u OR. Las condiciones a usar son:

- Comparación con un valor ($\leq, \ll, \geq, \gg, =, != $)
- Comparación en un intervalo (BETWEEN)
- Comparacion con una lista de valores (IN)
- Comparación con un patrón (LIKE)
- Test sobre la indeterminación de un valor (IS NULL)
- Test de todos o al menos uno (ALL/ANY)
- Test de existencia (EXISTS)

Los operadores lógicos han de mantener un cierto orden. SQL efectúa primero las comparaciones, después las negaciones con NOT, después los AND, y finalmente los OR. Para mayor seguridad es conveniente el uso de paréntesis.

8.1.3 Operador BETWEEN

Permite seleccionar las filas cuya columna especificada contenga un valor que se encuentre dentro de un intervalo determinado, incluyendo a los propios valores que determinan el intervalo. Equivale a combinar operaciones AND. Su negación es NOT BETWEEN.

8.1.4 Operador IN

Permite seleccionar las filas cuya columna contiene un valor incluido en una lista de valores dada. Equivalente a combinar operaciones OR. Su negación es NOT IN.

8.1.5 Operador LIKE

Permite seleccionar las filas que contienen en la columna seleccionada (de tipo alfanumérico) un valor coincidente con el patrón dado.

Con los operadores = o IN el valor de una columna debe coincidir exactamente con las constante que sigue. Si se produce un error de codificación, o se ha olvidado el valor exacto, no es posible encontrar la fila buscada por este medio. Por lo tanto hemos de recurrir a la búsqueda con patrón. Su negación es NOT LIKE.

8.1.6 Operador IS NULL

Si el valor de una columna no ha sido inicializado esta no interviene jamás en una selección por comparación de valores. Es posible saber si ha sido inicializado con IS NULL. Su negación es IS NOT NULL.

8.1.7 Eliminación de registros repetidos

SQL por defecto no elimina las repeticiones de las ocurrencias devueltas. Puede obtenerse sin dichas repeticiones añadiendo la cláusula DISTINCT tras la orden SELECT.

8.1.8 Renombrar columnas

El nuevo nombre o alias irá entre comillas dobles tras el nombre de la columna en la órden SELECT.

8.1.9 Clasificación de filas

Cuando no se determina el orden en que se desean obtener las filas, puede suceder que se obtengan resultados diferentes en ejecuciones diferentes de la misma orden.

Si quisiésemos ordenar los resultados sería preciso añadir la cláusula ORDER BY seguida del nombre de las columnas sobre las cuales se desea elaborar la clasificación separadas por comas.

Añadimos "asc" o "desc" para ordenar en orden asendente y descendente respectivamente (Por defecto se ordena de forma ascendente). En caso de existencia de nulos los agrupa al principio si "desc" y al final si "asc". La columna de ordenación debe forzosamente formar parte de la orden SELECT.

8.2 La orden INSERT

Se emplea para añadir nuevas filas a una tabla ya existente. Su formulación más sencilla sirve para añadir una fila cada vez. Para ello, se citará el nombre de las columnas que hay que inicializar y el valor que se les desea dar.

INSERT INTO nombretabla [(col1, col2,...,colN)]
VALUES (valor1, valor2,...,valorN)

El orden de los valores situados entre paréntesis debe corresponderse exactamente con el orden de las columnas de la tabla. Si no habría que indicar las columnas.

8.2.1 Inserción parcial

En este caso debemos fijar qué columnas reciben los valores, pudiendo no corresponderse en orden con la tabla.

8.2.2 Inserción desde otra tabla

Pueden insertarse datos en una tabla usando los registros de una o varias tablas siempre que asociemos a INSERT una consulta SELECT en lugar de VALUES.

INSERT INTO tabla1 [(col1,col2,...,coln)]
SELECT columna — expresion
FROM tabla2
[WHERE criterio de selección]

La tabla2 ya debe existir, y los tipos de datos corresponderse entre tablas.

8.3 La orden UPDATE

Esta orden cambia o actualiza los datos existentes de una tabla, especificando sus nuevos valores en las columnas seleccionadas. Pueden cambiarse todos los valores de una columna para todos los registros, o bien únicamente algunos valores para registros específicos usando la cláusula WHERE.

La modificación es una expresión que admite constantes, nombres de columnas, e incluso consultas anidadas. La formación general es multifila, para modificar una única fila, basta con añador un criterio de selección.

UPDATE nombre tabla
SET col1 = nuevovalor1 [,col2 = nuevovalor2,...]
[WHERE criterio de selección]

8.3.1 Cláusula SET

En la cláusula SET, cuyo ejemplo hamos visto antes, es posible el empleo de consultas anidadas, tal que:

SET col1 = (SELECT col2 FROM...)
Si la consulta anidada se asocia a la cláusula SET no devuelve ningún valor y la columna admite nulos, dejará todos los nulos. También podemos incluir la consulta anidada en la cláusula WHERE.

8.4 La orden DELETE

Se usa para borrar los registros de una tabla. Según el criterio de selección borraremos uno o más registros.

DELETE [FROM] tabla
[WHERE criterio de selección]

Si no contiene una cláusula WHERE todas las filas de la tabla serán borradas, pero la tabla seguirá existiendo, y podrían insertarse nuevos datos usando INSERT. DELETE afecta al registro completo.

8.4.1 Diferencias entre DROP, DELETE, y TRUNCATE

DELETE Sin un criterio de selección hace que se eliminen todos los datos, pero la definición de la tabla continúa en el diccionario de datos.

DROP además elimina la definición del diccionario de datos, liberando espacio.

TRUNCATE Borra todos las tuplas de la tabla pero no borra la definición del diccionario de datos. Además puede liberar espacio proporcionandoselo a Oracle.

9

FUNCIONES Y EXPRESIONES

9.1 Expresiones

Las expresiones aritméticas nos permiten realizar cálculos numéricos con datos pudiendo combinar varias columnas usando los cuatro operadores aritméticos; +,-,*,/. Si se trata de una operación de visualización, basta con escribir la expresión aritmética en la cláusula SELECT. Puede emplearse también en WHERE o HAVING.

9.2 Tabla dual

Proporcionada por $Oracle^{TM}$, se emplea para probar funciones o realizar cálculos rápidos. Se crea en el momento de la instalación y consta de una única fila y una única columna.

9.3 Funciones

Manipula un cto. de datos y devuelven un resultado. Las funciones se dividen en dos tipos:

Funciones de registros únicos :

- Funciones numéricas
- Funciones de caracteres
- Funciones de fecha
- Funciones de conversión
- Otras funciones

Funciones de grupo

Las funciones de registros aparecen en la lista de la orden SELECT (suponiendo que esta no contenga una cláusula GROUP BY) y en WHERE. Las de grupo aparecen en la lista de la orden SELECT y en HAVING, siempre que la consulta no contenga GROUP BY.

9.3.1 Funciones numéricas

Pueden aparecer en la cláusula SELECT, WHERE u ORDER BY. Se les puede aplicar diferentes funciones como por ejemplo abs(n), sqrt(n), cos(n), sin(n), tan(n)...

9.3.2 Funciones de caracteres

Podemos aplicar funciones especiales para manipular series de caracteres. Estas pueden devolver un tipo caracter (por ejemplo concat(cad1,cad2) que devuelve cad1 y cad2 unidas) o bien devolver un tipo numérico (por ejemplo lenght(cad) devuelve la longitud de la cadena).

9.3.3 Funciones de conversión

Convierten un valor de un tipo de dato en otro. Usadas generalmente para dar formato a las columnas. Por ejemplo to_date(ch[, fmt[, 'ndl']]) convierte ch de tipo char o varchar2 a un tipo fecha.

9.3.4 Otras funciones

Existen otras funciones que no podemos incluir en los otros grupos, aplicandolos a la tabla Dual. Son nvl(exp1,exp2), que devuelve exp2 si exp1 es nulo y viceversa, uid, que retorna un entero como ID de usuario, y user que retorna el nombre de usuario de $Oracle^{TM}$.

9.3.5 Funciones de grupos

Las funciones de grupos se efectuan sobre un cto de valores de una columna dentro de un grupo de filas.

Un grupo es un subconjunto de filas de una tabla en la que el valor de una columna es constante. Los grupos se especifican mediante la clásula GROUP BY seguida del nombre de la columna sobre la cual se efectua la agrupación. Cuando no existe la cláusula GROUP BY el grupo lo forman todas las filas seleccionadas.

Son funciones de grupo avg([distinct / all] n), count(* / [distinct / all] exp), max(n/exp)...

El argumento contiene el nombre de la columna sobre el cual debe ejercerse el cálculo. Si el argumento viene precedido de distinct se eliminarán antes las repeticiones de valores. Los NULL no se tienen en cuenta.

9.4 Consultas por grupos

9.4.1 Cláusula GROUP BY

Reordena la tabla que resulta de una consulta SELECT en un número mínimo de grupos tales que, en cáda uno, el valor de la columna tenga el mismo valor. No afecta a la organización de la tabla. Cuando se emplea la cláusula GROUP BY las funciones de grupo se calculan para cada grupo. Esta cláusula permite la visualización del valor de la columna común, seguida de los valores de las funciones aplicadas a cada grupo

Se usa para definir múltiples grupos de filas en una consulta SELECT. También puede añadirse a la consulta una cláusula WHERE para establecer una selección de filas, o ORDER BY para ordenar por distintas columnas las consultas.

GROUP BY col1 [,col2...colN]

9.4.2 Cláusula HAVING

Equivale a la cláusula WHERE pero aplicada a grupos. Esta cláusula no puede emplearse si antes no ha sido especificada una cláusula GROUP BY. La cláusula HAVING se utiliza para restringir los grupos seleccionados en el resultado de la consulta un vez realizada esta con la cláusula GROUP BY.

HAVING nombrefuncion o expresión.

9.5 Los valores NULL y la expresión nvl

El valor NULL es un valor indeterminado, por lo tanto, los valores nulos no se utilizan cuando se evalúan expresiones o funciones. Si un valor nulo se emplea en un cálculo aritmético, el valor final será siempre nulo. Para sustituir temporalmente el valor nulo por cualquier otro se debería usar la función nvl, por ejemplo:

```
SELECT avg(nvl(art_peso,0))
FROM artículos;
```

10

CONSULTAS ANIDADAS

La información de una consulta SELECT puede filtrarse segun un criterio definido en una cláusula WHERE o HAVING. Podemos definir sus criterios de selección a través de otra consulta SELECT. Teóricamente no hay límites de niveles de anidación. Las consultas anidadas pueden usarse tanto en WHERE como en HAVING. El tipo de dato que devuelve la consulta anidada debe corresponderse con el dato que se esté comparando. Una consulta anidada no puede contener ORDER BY.

SELECT col1,...,colN
FROM tabla
WHERE col3 operador (SELECT ...)

10.1 Devolución de un sólo valor

Una subconsulta puede devolver un único valor que sirve como comparación de la consulta de la cláusula WHERE principal. La consulta interior puede obtener valores desde otra tabla diferente a la referenciada en la cláusula SELECT exterior.

10.2 Comparación con operadores lógicos

Podemos combinar dos anidadas mediante los operadores OR o AND. Las columnas devueltas por la consulta interior han de coincidir con las del WHERE de la consulta exterior tanto en su número como en el tipo de dato.

10.3 Devolución de múltiples filas

Para ellos debe incluirse en una cláusula WHERE que tenga un operador de lista IN o uno de los operadores ALL o ANY. Ciertas consultas pueden retomar una lista de valores, en tales consultas deberá usarse el operador IN o NOT IN para unir la consulta de primer nivel con la de la lista obtenida por la subconsulta.

El operador = es más eficiente cuando se conoce que la subconsulta devolverá una única fila. Ocurrirá al utilizar este operador con la clave primaria de una tabla.

10.4 Devolución de múltiples columnas

Puede darse el caso de que las subconsultas devuelvan más de una columna. En tales subconsultas, el orden de las columnas exoresadas en la cláusula WHERE de la consulta de primer nivel deberá corresponder al orden de las columnas seleccionadas por la consulta de nivel inferior. Las columnas del WHERE de nivel superior deber ir entre paréntesis.

10.5 Subconsultas correlacionadas

La consulta anidada no se evalúa entera devolviendo una tabla temporal, si no que para cada valor de la consulta superior realizará una evaluación de la consulta anidada. Un valor obtenido mediante una subconsulta depende de una variable que recibe un valor desde la consulta de nivel superior. Podemosvernos obligados a usar alias en las columnas FROM.

SELECT alias1.col1, alias1.col2...alias1.colN
FROM tabla1 alias1
WHERE alias.colN in (SELECT alias2.col1...alias2.colM
FROM tabla2 alias2
WHERE alias1.colM=alias2.colN...)

La cláusula WHERE de la anidada comprueba que el valor iene de la consulta externa sea igual al de la interna, y así evaluarla y devolver el o los valores a la WHERE externa.

10.6 Operadores ANY y ALL

La lista obtenida por una subconsulta puede tratarse mediante los operadores ANY o ALL. Debe colocarse en una cláusula WHERE seguido de un operador de comparación y seguido de nuevo por ANY O ALL, y este de la consulta anidada.

- ANY comprueba que al menos una columna de la tabla a consultar cumple la condición de la anidada evaluando el WHERE como verdadero.
- ALL comprueba que la condición de la anidada se emplea para todas las tuplas de la anidada evaluando como verdadero el WHERE.

10.7 Operador EXISTS

Se emplea para realizar consultas de existencia, consultas en las que necesitamos utilizar el cuantificador universal, o la operación de dicisión del Álgebra relacional.

La subconsulta debe colocarse en una cláusula WHERE y después EXISTS seguido de la subconsulta. Devolverá al menos una fila, y la consulta anidada será cierta si encuentra una o más filas. En caso contrario, el WHERE se evaluará como falso y la consulta superior no se ejecutará.

10.8 Anidadas en HAVING

Una cláusula HAVING permite especificar un criterio de selección que deben verificar grupos. Puede a su vez depender de una anidada.

SELECT col1 aliascol1
FROM tabla
GROUP BY col 1
HAVING col1 operador (SELECT colN FROM tabla...)

11

CONSULTAS A MULTIPLES TABLAS

11.1 Producto cartesiano

Une cada fila de una tabla con cada una de las filas de otra tabla.

SELECT tab1.col1, ..., tab1.colN, tab2.col1, ..., tab2.colN
FROM tab1,tab2

Si la cláusula FROM incluye varias tablas se comienza realizando el producto cartesiano de las mismas. Dicho producto es una concatenación de las filas de una tabla original. Si la primera tabla contiene M filas y la segunda N, la tabla resultante contendrá MxN filas. Posteriormente se proyectarán las columnas seleccionadas en la cláusula SELECT, y estas aparecerán en el mismo orden especificado. La tabla resultante contiene información que no es cierta.

Si combinamos este producto natural con un criterio de selección usando WHERE, se denomina producto theta ($\bowtie_\theta$)

11.2 Producto natural

Se realiza sobre columnas que contienen la misma información en tablas distintas. Reunimos información entre tablas cuyo resultado, ahora si, es correcto, utilizando producto natural o join ($\bowtie$)

Debemos tener al menos una columna común entre las tablas que relacionamos, correspondiendose estas columnas con las claves foraneas.

Para realizar un producto natural basta con especificar en la cláusula SELECT las columnas buscadas y escribir una condición de igualdad en la cláusula WHERE entre las dos columnas comunes. La cláusula FROM debe contener el nombre de ambas.

SELECT tab1.col1,...,tab1.colN,tab2.col1,...,tab2.colN,...
FROM tab1,tab2
WHERE tab1.colN=tab2.colM

El procedimiento es igual que el producto cartesiano, pero se descartan finalmente todas las filas que no cumplan el criterio de unión de la cláusula WHERE. Podemos ordenar los resultados con ORDER BY.

11.3 Union externa

Permite solucionar problemas tales como obtener una lista de filas en funcion de un criterio que es función de la unión con otra tabla, y si la condición no se cumple, la fila normalmente no aparece en el resultado. Para viasualizar estas filas es necesario hacer uso de la unión externa.

Se identifica mediante el uso de (+) en la cláusula WHERE tras la columna donde pueden no darse ciertos valores. Devuelve todas las filas recuperadas por el producto natural, y aparte añade las tuplas de una taba que no están emparejadas con las de la otra tabla. Una unión externa hace que SQL suministre columnas nulas en una tabla para las filas que no cumplan la condición.

11.4 Autounión

Aveces hay que realizar la union de una tabla consigo misma. Se produce cuando el criterio de unión afecta al valor de una columna en relación al valor de esta misma columna en otra fila de la misma tabla. Hay que asignar alias por seguridad.

11.5 Operadores conjuntistas

Son cuatro: unión, intersección, diferencia, y producto cartesiano, ya visto anteriormente. Sólo union y producto cartesiano son comunes a todos los SGBD. Para realizar combinaciones de listas, $Oracle^{TM}$ usa los operadores UNION, INTERSECT, y MINUS. Una consulta SELECT afectada por estas operaciones no puede contener cláusulas DISTINCT ni ORDER BY

11.5.1 Operador UNION

Efectua la union de los resultados de dos consultas SELECT. Partiendo de dos tablas temporales, crea una tercera con el cto. de filas de las dos anteriores, eliminando o no las filas repetidas. El tipo de columna debe ser compatible.

SELECT col1...colN
FROM tabla1
[WHERE criterio de selección]
UNION
SELECT col1...colN
FROM tabla2
[WHERE criterio de selección]

11.5.2 Operador INTERSECT

Encuentra las tuplas comunes en las dos consultas.

SELECT col1...colN
FROM tabla1
[WHERE criterio de selección]
INTERSECT
SELECT col1...colN
FROM tabla2
[WHERE criterio de selección]

11.5.3 Operador MINUS

Obtiene todas las filas de la primera tabla menos las de la segunda

SELECT col1...colN
FROM tabla1
[WHERE criterio de selección]
MINUS
SELECT col1...colN
FROM tabla2
[WHERE criterio de selección]

12

TRATAMIENTO DE FECHAS

12.1 Aritmética de fechas

Para almacenar y operar con fechas, $Oracle^{TM}$ proporciona el tipo de datos *date* que se almacena en un formato interno especial que incluye años, meses, dias, horas, minutos, y segundos, pudiendo así saber el momento exacto de cualquier suceso.

Tenemos que tener en cuenta que si sumamos una fecha a un entero, obtenemos una fecham y si restamos una fecha a otra, obtenemos un periodo de tiempo (normalmente expresado en días). No ha de ser un valor exacto ya que $Oracle^{TM}$ guarda hasta segundos, y puede ser un periodo de tiempo decimal. Podemos cambiar dicho formato para una sesión concreta, o para dar formato permanente en el fichero de arranque.

12.2 Función SYSDATE

Esta función es bastante importante ya que $Oracle^{TM}$ la usa para extraer del SO la fecha y la hora actual. No necesita argumentos.

Por ejemplo, para saber la fecha de hoy, debemos usar esta función junto a la tabla dual:

SELECT SYSDATE
FROM dual;

12.3 Fuciones de fechas

Operan con valores de tipo *date* y devuelven los valores tipo date (exceptuando months_between que devuelve un dato numérico). Son funciones de fechas add_months, last_day, next_day...

12.4 Formatos y conversión de fechas

Las fechas pueden convertirse en cadenas de caracteres y viceversa. TO_CHAR y TO_DATE tienen poderosas capacidades para formatear.

TO_CHAR(fecha[,'formato']

TO_DATE(cadena[,'formato'])

13

GLOSARIO DE PRÁCTICAS

13.1 Órdenes

SELECT Se utiliza para realizar cualquier consulta a una o varias tablas o vistas de nuestro esquema, o de otros. Es la orden que más se emplea en SQL y la más versatil. Cualquier consulta, tanto a una como varias tablas deberá constar de como mínimo una sentencia formada por la orden SELECT y la cláusula FROM.

INSERT Se usa para añadir nuevas filas a una tabla ya existente. Su formulación más sencilla sirve para añadir una fila cada vez. Para ello se citará el nombre de la tabla, el nombre de las columnas que hay que inicializar, y el valor que se les desea dar.

insert into nombretabla values(valor1,valor2...valorN)

UPDATE Cambia o actualiza los datos existentes de una tabla, especificando sus nuevos valores en las columnas seleccionadas. Pueden cambiarse todos los valores de una columna para todos los registros, o bien únicamente algunos valores para registros específicos. La cláusula WHERE es usada para seleccionar las actualizaciones.

update nombretabla set nombrecolumna = nuevovalor[nombrecolumna = nuevo valor] [where criteriodeselección]

DELETE Se usa para borrar registros de una tabla. Se pueden borrar todos los registros o bien algunos según criterio de selección.

delete [from] nombretabla [where criteriodeselección]

Si una orden DELETE no contiene la cláusula WHERE, todas las filas de esa tabla serán borradas. Sin embargo, la tabla continuaría existiendo en la BD. Los registros no pueden borrarse parcialmente, ya que la orden DELETE borra el registro completo.

DROP Elimina todos los registros de una tabla así como su definición en el banco de datos, liberando espacio ocupado en el banco de datos.

drop table nombretabla

TRUNCATE Borra todas las tuplas de una tabla pero no borra la definición de la tabla en el diccionario de datos, además de liberar el espacio ocupado por ella para poner dicho espacio a disposición de Oracle.

truncate table nombretabla

13.2 Cláusulas

FROM Se usa para indicar los nombre de las tablas cuyas columnas queremos seleccionar, separadas por comas.

WHERE cláusula donde se especifican las condiciones de selección. Es opcional y se coloca a continuación de la cláusula FROM. El criterio está basado en expresiones lógicas y compuesto por condiciones.

ORDER BY Clasifica las filas seleccionadas en función el valor de las columnas.

GROUP BY Reordena la tabla que resulta de una consulta SELECT en un número mínimo de grupos tales que, en el interior de cada uno, la columna especificada tenga el mismo valor para cada fila. No afecta a la organización física de la tabla. Permite generalmente la visualización del valor común de la columna, seguido de valores de las funciones aplicadas a grupos. Se utiliza para definir múltiples grupos de filas dentro de una consulta con la orden SELECT.

HAVING Equivale a una cláusula WHERE pero aplicada a grupos. Generalmente no puede ser usada si previamente no se ha empleado una cláusula GROUP BY. Afecta al valor de una función calculada sobre un grupo. Se utiliza para restringir los grupos seleccionados con el resultado de una consulta una vez ejecutada esta con la cláusula GROUP BY. Los grupos que no cumplan el criterioe specificado en la cláusula HAVING no se incluyen en el resultado.

DISTINCT Se usa para eliminar ocurrencias repetidas devueltas en una consulta.

13.3 Operadores

BETWEEN Permite seleccionar las filas cuya columna específica contenga un valor que se encuantre dentro de un intervalo determinado, incluyendo a los propios valores que determina el intervalo.

IN/NOT IN Permite seleccionar las filas para las cuales se cumple que una de sus columnas contiene un valor incluido en una lista de valores dada. NOT IN, al contrario, un valor distinto al existente en la lista.

LIKE Permite seleccionar filas que contienen en la columna indicada un valor (de tipo alfanumérico) coincidente con el patrón dado. Útil cuando no se conoce el valor exacto.

IS NULL/IS NOT NULL Usados para comprobar si se ha inicializado un valor o no.

EXISTS/NOT EXISTS Es un predicado que vale verdadero si la consulta tiene resultado no vacío. Una fila de la consulta exterior forma parte del resultado si existe en la tabla interior al menos una fila que satisface las condiciones de la cláusula WHERE de la subconsulta. Cuando la expresión está precedida de NOT el predicado se evalúa como verdadero si no existe ninguna fila de la tabla interior que satisface las condiciones exigidas.

ALL La condición se cumple si la comparación se verifica para todos los valores invocados por la consulta anidada. un NOT IN equivale a un !=ALL

ANY La condición se cumple si la comparación se verifica para uno al menos de los valores invocados por la consulta anidada. IN equivale a =ANY

UNION Efectúa la unión de los resultados de dos consultas SELECT.

INTERSECT Encuentra las tuplas comunes en las consultas.

MINUS Obtiene todas las filas seleccionadas por la primera consulta eliminando las de la segunda.

función SYSDATE Oracle la usa para extraer del SO la fecha y hora actual. No necesita argumentos.