

TEMA 35

La definición de datos.
Niveles de descripción.
Lenguajes. Diccionario
de datos.

Por:

informaticapreparacion@gmail.com

www.informaticapreparacion.es

TEMA 35

La definición de datos. Niveles de descripción. Lenguajes. Diccionario de datos.

ÍNDICE

1	INTRODUCCIÓN	3
2	DEFINICIONES Y CONCEPTOS	3
2.1	SISTEMAS DE INFORMACIÓN	3
2.2	SISTEMAS DE FICHEROS.	3
2.3	BASES DE DATOS.	4
2.4	MODELOS DE DATOS. ESQUEMAS. DISEÑO DE BASES DE DATOS.	4
3	DEFINICIÓN DE DATOS	5
4	NIVELES DE DESCRIPCIÓN	6
5	LENGUAJES DE DEFINICIÓN DE DATOS	8
5.1	SISTEMAS RELACIONALES.....	8
5.2	SISTEMAS NOSQL.....	13
6	DICCIONARIO DE DATOS	16
7	PLANTEAMIENTO DIDÁCTICO	17
8	CONCLUSIÓN	17
9	BIBLIOGRAFÍA	17

1 INTRODUCCIÓN

Con este tema continuamos con el bloque de contenidos dentro del temario oficial, dedicado al estudio en general de las Bases de Datos, en este tema en concretamos se presenta en concepto de Sistema Gestor de Base de Datos.

Las bases de datos nacen para solucionar los problemas presentados a los programadores a la hora de diseñar programas eficientes capaces de manipular grandes volúmenes de información[1].

Los pioneros en Ciencias de la Computación de finales de los 50 y principios de los 60 desarrollaron muchas técnicas nuevas para manipular los ficheros. Los ficheros se pueden manipular mediante lenguajes de tercera generación convencionales empleando los servicios del sistema operativo. La complejidad del trabajo de programación y la rigidez de las restricciones que imponía la tecnología de los ordenadores en aspectos como la inserción de datos o la gestión de casos especiales, hacía arduo el trabajo. De su evolución y necesidad de facilitar la tarea al programador surgen las bases de datos como conjunto, colección o depósito de datos almacenados en un soporte informático no volátil. A su vez alrededor de las bases de datos nacen los sistemas gestores de base de datos (SGBD) (DBMS, DataBase Management System) conjunto de utilidades que permiten la implantación, acceso y mantenimiento de una base de datos.

Nota: Marcamos [numero], las referencias a la Bibliografía.

2 DEFINICIONES Y CONCEPTOS

2.1 Sistemas de información.

Un sistema de información es un conjunto de elementos que gestiona la información de una determinada organización. Sus componentes son:

- *Datos.* Información relevante que almacena y gestiona el sistema de información.
- *Hardware.* Equipamiento físico que se utiliza para gestionar los datos.
- *Software.* Aplicaciones que permiten el funcionamiento adecuado del sistema.
- *Recursos humanos.* Personal que maneja el sistema de información.

Existen dos tipos fundamentales de sistemas de información:

- *Orientados al proceso.* En estos sistemas de información se crean diversas aplicaciones (software) para gestionar diferentes aspectos del sistema. Cada aplicación realiza unas determinadas operaciones, y almacena y utiliza sus propios datos.
- *Orientados a los datos.* En esos sistemas los datos se almacenan en una única estructura lógica que es utilizable por las aplicaciones. A través de esa estructura se accede a los datos que son comunes a todas las aplicaciones.

2.2 Sistemas de ficheros.

Los sistemas de ficheros son sistemas de información orientados al proceso que tienen las siguientes características:

- Los ficheros se diseñan para una determinada aplicación.
- Los datos se encuentran almacenados en soportes de almacenamiento secundario, mientras su descripción está separada de los mismos, formando parte de los programas.

- No hay control sobre el acceso y la manipulación de los datos más allá de lo impuesto por los programas de aplicación.

Los inconvenientes que se plantean son los siguientes:

- Hay una ocupación inútil de memoria secundaria.
- Suele aparecer un cierto grado de inconsistencia y duplicación en la información.
- Falta de flexibilidad del sistema de ficheros para adaptarse a las nuevas necesidades.
- Existe cierta dificultad para compartir información.

2.3 Bases de datos.

Una base de datos es un conjunto estructurado de datos relacionados entre sí que reside en soportes de almacenamiento secundario de acceso directo [1].

Las características que las diferencian de los sistemas de ficheros son las siguientes:

- Además de los datos, se almacenan las relaciones entre ellos y sus restricciones semánticas.
- No debe existir redundancia lógica, aunque se admite redundancia física por eficiencia.
- Las bases de datos han de atender a múltiples usuarios y diferentes aplicaciones.
- Existe independencia tanto física como lógica entre datos y tratamientos.
- La definición y la descripción de los datos están integradas con los mismos datos.
- Incorporan procedimientos de actualización y recuperación que mantienen la integridad, seguridad y confidencialidad de los datos.

2.4 Modelos de datos. Esquemas. Diseño de bases de datos.

Un modelo de datos es una colección de conceptos para la descripción de los datos, las relaciones entre ellos y las restricciones que deben cumplir. Un esquema es una descripción de una base de datos mediante un modelo de datos. Los modelos de datos se clasifican en: [2]

- *Modelos conceptuales.* Describen los datos con un alto nivel de abstracción, utilizando entidades (concepto del mundo real), atributos (propiedad de interés de una entidad) y relaciones (interacción entre dos o más entidades). Son independientes de la base de datos a utilizar. Por ejemplo, el modelo entidad/relación y el modelo orientado a objetos.
- *Modelos lógicos.* Representan los datos valiéndose de estructuras de registros de varios tipos, formados por un número determinado de campos. Son dependientes de la base de datos a utilizar. Por ejemplo, el modelo relacional, el modelo de red y el modelo jerárquico.
- *Modelos físicos.* Los modelos físicos describen cómo se almacenan los datos en cuanto al formato de los registros, la estructura de los ficheros y los métodos de acceso utilizados.

El diseño de bases de datos se estructura en tres pasos:

- *Diseño conceptual.* Recibe como entrada la especificación de requerimientos y su resultado es el esquema conceptual, que es una descripción de alto nivel de la estructura

de la base de datos mediante un modelo conceptual, y que es independiente del SGBD que se utilice.

- **Diseño lógico.** Recibe como entrada el esquema conceptual y da como resultado un esquema lógico, que es una descripción de la estructura de la base de datos mediante un modelo lógico, y que puede ser procesado por el SGBD que se utilice.
- **Diseño físico.** Recibe como entrada el esquema lógico y da como resultado un esquema físico, que es una descripción mediante un modelo físico de las estructuras de almacenamiento y de los métodos usados para tener un acceso efectivo a los datos.

3 DEFINICIÓN DE DATOS

La función de definición de datos debe permitir al diseñador de la base de datos especificar los elementos de datos que la integran, su estructura y las relaciones que existen entre ellos, las reglas de integridad semántica, etc., así como las características de tipo físico y las vistas lógicas de los usuarios.[2]

La arquitectura ANSI/X3/SPARC data de la segunda mitad de la década de los 70, y está considerada modelo estándar a la hora de implementar SGBD relacionales.

La arquitectura ANSI/X3/SPARC distingue dos partes, la superior para la definición de la base de datos, y la inferior para su manipulación. Ambas partes quedan separadas por la interfaz 14. En esta arquitectura se describen distintas funciones: humanas representadas en la figura por hexágonos; funciones de programa, que se presentan por medio de rectángulos; interfaces, que se representan mediante líneas y para cuya instrumentación el informe no dicta ninguna norma, pudiendo ser, por tanto, interfaces físicas, lógicas, microprogramadas, etc., y metadatos o diccionario de datos que desempeñan un papel fundamental en esta arquitectura y que se representan por medio de un triángulo.

A la hora de definir la base de datos, aunque el estándar establece tres administradores de base de datos (DBA) distintos, en la práctica la figura del DBA agrupa los papeles de administrador de sistema, administrador de empresa y administrador de aplicaciones en lo que se refiere a la base de datos:

El papel del administrador de empresa es definir el esquema conceptual usando el interfaz 1. El procesador de esquema conceptual compila este esquema y si es correcto se almacena en el diccionario de datos, que contiene todos los esquemas y reglas de proyección.

Los administradores de aplicaciones se encargan de definir los esquemas externos, usando lenguajes específicos de descripción de esquemas externos (interfaz 4), según las necesidades de los usuarios y las posibilidades del sistema. Para especificar las reglas de proyección entre un esquema externo y el esquema conceptual, el administrador de aplicaciones puede consultar el esquema conceptual mediante el interfaz 3. Cuando se define correctamente un esquema externo con sus reglas de proyección asociadas, es compilado por el procesador de esquema externo y almacenado en el diccionario de datos.[3]

Por último, el administrador del sistema, mediante un lenguaje de descripción interno (interfaz 6) completa la descripción de la base de datos definiendo su esquema interno y las reglas que lo proyectan sobre el esquema conceptual.

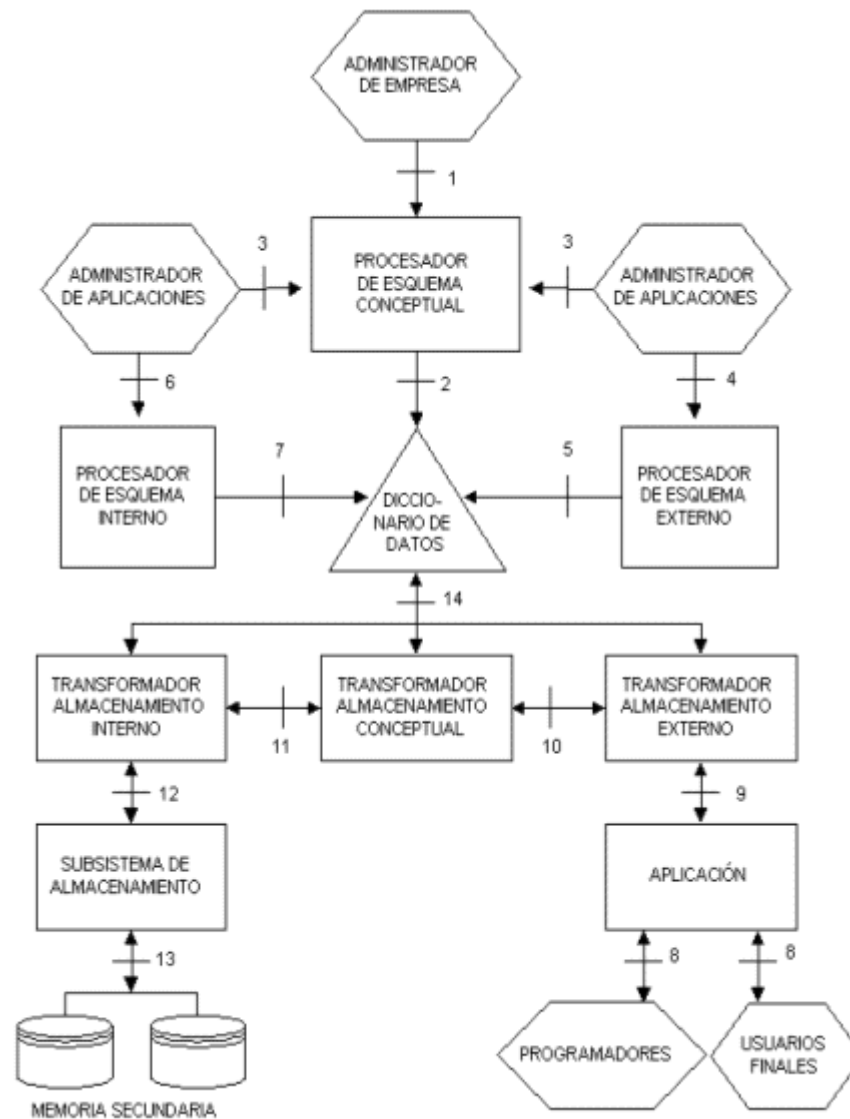


Figura. Arquitectura ANSI/X3/SPARC

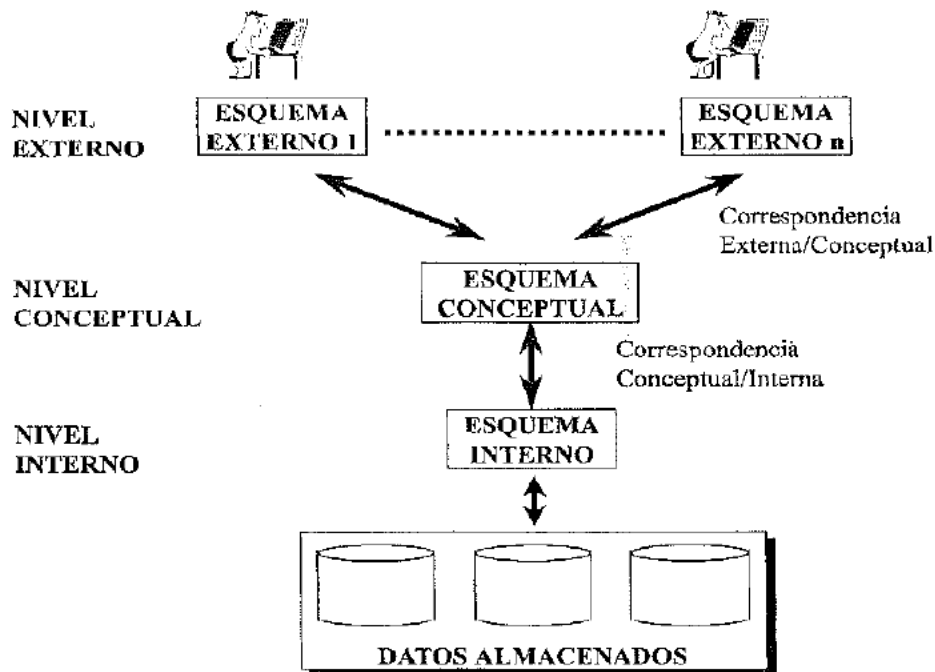
4 NIVELES DE DESCRIPCIÓN

Los estándares de base de datos emplean tres niveles de abstracción a la hora de la describir y utilizar las bases de datos, debido a que uno de los principales objetivos de las bases de datos es conseguir la independencia entre las estructuras lógica y física de los datos, que tiene como consecuencia la independencia entre datos y aplicaciones, de modo que los cambios en la estructura de datos tengan una repercusión mínima en los programas de aplicación y al contrario.

Existe 3 niveles:

- Nivel Interno o físico. En este nivel físico se describe cómo estarán almacenados los datos en la base de datos. Para ello se apoya en un modelo físico de datos.

- Nivel Conceptual. En este nivel se describe la estructura «conceptual» que tendrá la base de datos sin tener en cuenta los aspectos físicos de almacenamiento que se contemplan en la capa anterior. La distribución de la información en este nivel se fija en entidades, relaciones y atributos.
- Nivel Externo o de usuario. En este nivel se hace hincapié en que cada usuario tiene una visión de la base de datos distinta. Existen distintas vistas o esquemas externos de usuario.



El objetivo de esta estructuración a tres niveles es independizar los programas de la base de datos física. De esa manera el desarrollo de aplicaciones es más rápido y normalizado.

Hay que decir que los SGBD actuales respetan estas capas pero internamente no al 100%. Muchos de ellos intercambian elementos de un nivel en otro y en ocasiones no existe una independencia marcada entre el nivel externo o de usuario y el nivel conceptual. Obviamente hay que tener

ANSI/X3/SPARC	CODASYL	Club de Banco de Datos (INRIA)	GUIDE/SHARE
Externo	Subesquema	Lógico	Lógico
Conceptual	Esquema	Virtual	Entidad
Interno	Esquema de almacenamiento	Físico	Almacenado

Figura. Comparativa de Niveles en distintas Arquitecturas.

Este tipo de arquitectura en tres niveles tiene su sentido cuando se desea tener una independencia de datos tal, que se pueda modificar una capa sin tener que modificar las capas contiguas. Por lo tanto los dos tipos de independencia de datos que se pueden definir son los siguientes:

- Independencia lógica. Implica el poder cambiar el esquema conceptual sin tener que modificar las vistas de los usuarios pues siguen siendo válidas. Generalmente este tipo

de cambios consisten en ampliación o reducción de la base de datos (añadiendo o disminuyendo su número de entidades).

- Independencia física. Implica el poder cambiar el esquema físico de almacenamiento sin tener que modificar el esquema conceptual. Cualquier consulta tanto de selección como de borrado o inserción seguirá siendo válida.

5 LENGUAJES DE DEFINICIÓN DE DATOS

El administrador de la base de datos ha de disponer de instrumentos que le permitan describir los datos con facilidad y precisión, especificando sus distintas estructuras; es lo que se denomina lenguaje de definición de datos. Los lenguajes de definición de datos son autocontenidos, y no necesitan apoyarse en ningún lenguaje de programación. Sin embargo, pueden integrarse junto con los lenguajes de manipulación, como en SQL o en QUEL.

Puesto que la definición de datos existe a niveles externo, conceptual e interno, existen también lenguajes de definición de datos a cada nivel. En el caso del nivel conceptual, en realidad sólo hay lenguajes de definición de datos para los modelos lógicos, ya que los modelos conceptuales de alto nivel no están directamente reflejados en los SGBD.

Para los SGBD clásicos, con modelos lógicos jerárquicos y en red, cada producto disponía de su propio lenguaje de definición de datos, hasta la aparición de los primeros grupos de estandarización como el grupo Codasyl (modelo de datos en red), lo que abrió el camino a posteriores estándares como el lenguaje SQL en las bases de datos relacionales:

- *Bases de datos jerárquicas.* IMS, SYSTEM 2000.
- *Bases de datos en red.* IDS, IDMS, Codasyl.
- *Bases de datos relacionales: Lenguaje SQL*
- *Bases de datos NoSQL, Documentales: Lenguaje propio de manipulación.*

5.1 Sistemas Relacionales

Un lenguaje de definición de datos (Data Definition Language, DDL por sus siglas en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos, así como de los procedimientos o funciones que permitan consultarlos.

El lenguaje de programación SQL, el más difundido entre los gestores de bases de datos, admite las siguientes sentencias de definición: CREATE, DROP y ALTER, cada una de las cuales se puede aplicar a las tablas, vistas, procedimientos almacenados y triggers de la base de datos.

Otras que se incluyen dentro del DDL, pero que su existencia depende de la implementación del estándar SQL que lleve a cabo el gestor de base de datos son GRANT y REVOKE, los cuales sirven para otorgar permisos o quitarlos, ya sea a usuarios específicos o a un rol creado dentro de la base de datos.

Una vez finalizado el diseño de una base de datos y escogido un SGBD para su implementación, el primer paso consiste en especificar el esquema conceptual y el esquema interno de la base de datos, y la correspondencia entre ambos. En muchos SGBD no se mantiene una separación estricta de niveles, por lo que el administrador de la base de datos y los diseñadores utilizan el mismo lenguaje para definir ambos esquemas, es el lenguaje de definición de datos (LDD). El SGBD posee un compilador de LDD cuya función consiste en procesar las sentencias del lenguaje para identificar las descripciones de los distintos elementos de los esquemas y almacenar la

descripción del esquema en el catálogo o diccionario de datos. Se dice que el diccionario contiene metadatos: describe los objetos de la base de datos.

Cuando en un SGBD hay una clara separación entre los niveles conceptual e interno, el LDD sólo sirve para especificar el esquema conceptual. Para especificar el esquema interno se utiliza un lenguaje de definición de almacenamiento (LDA). Las correspondencias entre ambos esquemas se pueden especificar en cualquiera de los dos lenguajes. Para tener una verdadera arquitectura de tres niveles sería necesario disponer de un tercer lenguaje, el lenguaje de definición de vistas (LDV), que se utilizaría para especificar las vistas de los usuarios y su correspondencia con el esquema conceptual.

El DDL, Ha de permitir establecer nombres, longitudes y características de los campos de datos y agregados (tablas en el caso relacional), así como establecer las relaciones entre los mismos, y las restricciones semánticas a las que deben someterse.

En la descripción lógica no deberían incluirse informaciones con tendencia a determinar ningún tipo concreto de estructura física que obligue a cambiar dicha descripción ante alteraciones de las características del soporte físico o de los caminos de acceso. Sin embargo, aunque la independencia físico-lógica es deseable, en muchas ocasiones los SGBD no responden a una arquitectura a tres niveles y tienen un único lenguaje para todos ellos.

En algunos SGBD la estructura lógica se puede llevar a cabo mediante un interfaz gráfico, el caso de Access, o mediante la consola de administración como Enterprise Manager de Oracle o Mysql-Client.

5.1.1 Creación de una tabla

CREATE sirve para crear objetos de la base de datos, entre estos objetos tenemos tablas, vistas etc.

```
CREATE TABLE nombre tabla
(nombrecol1 tipocol1
 [CONSTRAINT nombre_restricción]
 [not NULL]
 [PRIMARY KEY]
 [UNIQUE]
 [DEFAULT valor]
 [check <condici>]
 [REFERENCES Nombre_tabla_ref (colref) [ON DELETE CASCADE]],...

 [Restricciones de la tabla]
)
[tablespace nombre-tablespace];
```

Donde:

Nombre tabla: Es el nombre de la nueva tabla. Debe ser único dentro de la BD y además debe identificar su contenido, el nombre de la tabla puede ser una cadena de 1 a 30 caracteres alfanuméricos (0-9, a-z, subrayado, \$, #) empezando siempre por un carácter alfabético.

nombreCol: Es el nombre de una columna de la tabla. Los nombres de columna deben cumplir las reglas de los identificadores y deben ser únicos en la tabla.

tipoCol: Especifica el tipo de datos de la columna. Se permiten los tipos de datos del sistema o definidos por el usuario, Especifica el tipo de datos de la columna. Se permiten los tipos de datos del sistema o definidos por el usuario.

Cuando almacenamos datos las tablas se ajustan a una serie de restricciones predefinidas, por ejemplo que una columna no pueda tomar valores negativos o que una columna tenga que almacenarse en mayúsculas.

La integridad hace referencia al hecho de que los datos de la BD han de ajustarse a restricciones antes de almacenarse en la BD y una restricción de integridad será una regla que restringe el rango de valores de una o más columnas de una tabla.

Existe otro tipo de integridad que es la integridad referencial, que garantiza que los valores de una o varias columnas de una tabla dependan de los valores de otro o otras columnas de otra tabla.

Las restricciones se definen dentro de la orden CREATE TABLE y para ello se utiliza la cláusula CONSTRAINT. Una vez creadas las restricciones se pueden añadir, modificar o borrar a través de la orden ALTER TABLE.

Una cláusula CONSTRAINT puede restringir una sola columna, se habla en este caso de restricción de columna o puede restringir un grupo de columnas de una tabla, en este caso se llama restricción de tabla.

CONSTRAINT. Es una palabra clave opcional que indica el principio de la definición de una restricción para la columna o tabla : PRIMARY KEY, NOT NULL, UNIQUE, FOREIGN KEY o CHECK. Las restricciones son propiedades especiales que exigen la integridad de los datos y pueden crear índices para la tabla y sus columnas.

nombre_restricción. Los nombres de restricción deben ser únicos en una base de datos.

NULL | NOT NULL. Son palabras clave que determinan si se permiten o no valores NULL en la columna. Si la restricción es NOT NULL significa que dicha columna no puede tener valores nulos, es decir, ha de tener un valor obligatoriamente; en caso contrario causa una excepción.

PRIMARY KEY. Es una restricción que indica qué columna o columnas formarán la clave primaria de la tabla. Sólo se puede crear una restricción PRIMARY KEY por cada tabla.

UNIQUE. Es una restricción que proporciona la integridad de entidad para la columna o columnas indicada a través de un índice único. Una tabla puede tener varias restricciones UNIQUE.

DEFAULT. Especifica el valor suministrado para la columna cuando no se ha especificado explícitamente un valor durante la inserción.

REFERENCES. Es una restricción que proporciona integridad referencial para los datos de una columna o columnas. Las restricciones REFERENCES requieren que cada valor de la columna o columnas existan en la columna o columnas de referencia correspondiente de la tabla a la que se hace referencia. Las restricciones REFERENCES pueden hacer referencia sólo a columnas que sean restricciones PRIMARY KEY en la tabla de referencia.

Nombre_tabla_ref. Es el nombre de la tabla a la que hace referencia la restricción REFERENCES.

(colref [,...n]). Es una columna o lista de columnas de la tabla a la que hace referencia la restricción REFERENCES.

ON DELETE CASCADE. Especifica qué acción tiene lugar en una fila de la tabla creada, si esa fila tiene una relación referencial y la fila a la que hace referencia se elimina en la tabla primaria. En nuestro caso si se elimina una fila de la tabla primaria, también se eliminan las filas de la tabla desde donde se hace referencia. Cuando la restricción de Integridad Referencial se

realiza sobre la definición de un campo en la sentencia CREATE TABLE solo se utiliza la clausula REFERENCES, no se utiliza la clausula FOREIGN KEY; esta última se utiliza cuando la restricción se crea a nivel de tabla.

CHECK. Es una restricción que exige la integridad del dominio al limitar los valores posibles que se pueden escribir en una o varias columnas.

Ejemplo:

```
CREATE TABLE Emp1
(empno      NUMBER          CONSTRAINT pk_emp1 PRIMARY KEY,
ename       VARCHAR2(10)    CONSTRAINT nn_ename1 NOT NULL
           CONSTRAINT upper_ename1 CHECK (ename = UPPER(ename)),
job         VARCHAR2(9),
mgr         NUMBER          CONSTRAINT fk_mgr1  REFERENCES scott.emp(empno),
hiredate    DATE            DEFAULT SYSDATE,
sal         NUMBER(10,2)    CONSTRAINT ck_sal1 CHECK (sal > 500),
comm        NUMBER(9,0)     DEFAULT NULL,
deptno      NUMBER(2)       CONSTRAINT nn_deptno1 NOT NULL
           CONSTRAINT fk_deptno1 REFERENCES scott.dept(deptno) );
```

5.1.2 Borrado de una tabla

```
Drop table nombre_tabla [CASCADE CONSTRAINT];
```

Al borrar una tabla, se borra tanto su estructura como sus datos, sus índices asociados y los privilegios concedidos sobre estas también se borran. Cada usuario puede borrar sus propias tablas, pero no puede borrar las de otro usuario al menos que tenga concedido un permiso adecuado. Si se hace referencia a la clave primaria de esta tabla mediante restricciones FOREIGN KEY o REFERENCES, la clausula CASCADE CONSTRAINT permite suprimir estas restricciones de integridad referencial en las tablas 'descendientes'.

5.1.3 Borrado de los registros de una tabla

Con la orden TRUNCATE se eliminan todas las filas de una tabla y se puede liberar espacio utilizado por esta tabla. Es una orden del lenguaje DDL y por tanto no se puede anular. Tampoco activa disparadores DELETE por lo que es más rápido que una orden DELETE. Su sintaxis es:

```
Truncate table nombre_table [{DROP | REUSE} STORAGE];
```

5.1.4 Modificar tabla

Para modificar la estructura de una tabla se utiliza el comando ALTER TABLE.

```
ALTER TABLE Tabla
{[ADD      ( Columna Tipodato [restricción de columna][...]) ]
[MODIFY   ( Columna Tipodato[restricción de columna][...]) ]
[ADD CONSTRAINTS restricción]
[DROP CONSTRAINTS restricción];
```

Añadir o modificar columnas (nombre, tipo, valor por defecto, restricción NOT NULL)

```
alter table nombre_table {ADD | MODIFY} ( columna tipo [restricción,...])
```

Eliminación de columnas

```
alter table nombre_table DROP COLUMN nombre_columna
```

Añadir restricción de tabla

```
alter table nombre_tabla ADD CONSTRAINT nombre_restriccion restriccion
```

Eliminar una restricción.

```
alter table nombre_table DROP CONSTRAINT nombre_restriccion
```

Activación y desactivación de una restricción.

```
alter table nombre_table [ENABLE VALIDATE|ENABLE NOVALIDATE|DISABLE]  
nombre_restriccion
```

5.1.5 Vistas

Las vistas son tablas virtuales ‘que contienen’ el resultado de una consulta SELECT, tienen la misma estructura que una tabla cualquiera, es decir, están organizadas por filas y columnas. Una de las principales ventajas de utilizar vistas procede del hecho de que la vista no almacena los datos, sino que hace referencia a una o varias tablas de origen mediante una consulta SELECT, consulta que se ejecuta cada vez que se hace referencia a la vista. De esta forma, cualquier modificación que se realice sobre los datos de las tablas de origen es inmediatamente visible en la vista, cuando ésta vuelva a ejecutarse. Su sintaxis es:

```
CREATE [OR REPLACE] VIEW Nombre_vista  
[(Lista de columnas)]  
AS SELECT[...]
```

La opción REPLACE, lo que hace es, reemplazar la vista en el caso de que esta ya exista. Las vistas se utilizan de forma análoga a las tablas, permitiendo realizar consultas sobre las vistas, también se pueden realizar sentencias DML sobre las vistas, sin embargo, las modificaciones, borrados e inserciones están restringidas a vistas que estén definidas sobre una única tabla.

La orden para borrar una vista es DROP VIEW. Su sintaxis es:

```
DROP VIEW Nombre_vista
```

5.1.6 Indices

Los índices sirven para mejorar el rendimiento de las consultas.

En general, los índices se crean sobre todas las claves externas y sobre los criterios de búsqueda actuales.

```
CREATE [unique] INDEX nombre_indice  
ON nombre_tabla (columnas [{asc | desc}] [,.....])  
[TABLESPACE Nombre_Tablespace]
```

5.1.7 Borrado de un índice

Cuando se borra una tabla, automáticamente se borran los índices asociados a ella. Los índices ocupan espacio dentro de la BD como si de una tabla se tratara y por esa razón se aconseja tener solo como índices aquellas columnas por las cuales se realizan consultas de forma periódica. Para borrar un índice se utiliza la orden:

```
drop index nombre_indice;
```

5.2 Sistemas NoSQL

Los sistemas de bases de datos NoSQL crecieron con las principales redes sociales, como Google, Amazon, Twitter y Facebook. Estas tenían que enfrentarse a desafíos con el tratamiento de datos que las tradicionales SGBDR no solucionaban {ver artículo}. Con el crecimiento de la web en tiempo real existía una necesidad de proporcionar información procesada a partir de grandes volúmenes de datos que tenían unas estructuras horizontales más o menos similares. Estas compañías se dieron cuenta de que el rendimiento y sus propiedades de tiempo real eran más importantes que la coherencia, en la que las bases de datos relacionales tradicionales dedicaban una gran cantidad de tiempo de proceso.[5]

En ese sentido, a menudo, las bases de datos NoSQL están altamente optimizadas para las operaciones recuperar y agregar, y normalmente no ofrecen mucho más que la funcionalidad de almacenar los registros (p.ej. almacenamiento clave-valor). La pérdida de flexibilidad en tiempo de ejecución, comparado con los sistemas SQL clásicos, se ve compensada por ganancias significativas en escalabilidad y rendimiento cuando se trata con ciertos modelos de datos.

Tipos de Bases de Datos NoSQL

- **Bases de datos clave - valor:** Es el modelo más popular y más sencilla. En este tipo de sistema, cada elemento está identificado por una llave única, lo que permite la recuperación de la información de forma muy rápida, información que habitualmente está almacenada como un objeto binario (BLOB).

Algunos ejemplos de este tipo son Cassandra, BigTable o HBase.

- **Bases de datos documentales:** Son las bases de datos NoSQL más versátiles. Se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales.

Algunos ejemplos de este tipo son MongoDB o CouchDB.

- **Bases de datos en grafo:** La información se representa como nodos de un grafo y sus relaciones con las aristas del mismo, de manera que se puede hacer uso de la teoría de grafos para recorrerla. Para sacar el máximo rendimiento a este tipo de bases de datos, su estructura debe estar totalmente normalizada, de forma que cada tabla tenga una sola columna y cada relación dos.

Algunos ejemplos de este tipo son Neo4j, InfoGrid o Virtuoso.

- **Bases de datos orientadas a objetos:** La información se representa mediante objetos, de la misma forma que son representados en los lenguajes de programación orientada a objetos (POO) como ocurre en JAVA, C# o Visual Basic .NET.

Algunos ejemplos de este tipo de bases de datos son Zope, Gemstone o Db4o.

Como ejemplo para este tema, y debido a la diversidad en que el Universo NoSQL se definen los datos, nos centramos en uno de los sistemas NoSQL más conocidos y utilizados en la actualidad: MongoDB.

5.2.1 Definición de Datos en MongoDB

MongoDB es un tipo de base de datos de esquema flexible. Esto significa que ha de ser el usuario el que determina y declara el esquema de cada una de las tablas justo en el momento de realizar las inserciones. Esta funcionalidad la comparten en cierta forma muchas bases de datos NoSQL.

Esto es bueno y malo a la vez. Por un lado se tiene la flexibilidad de añadir campos cuando son necesarios sin haberlos tenido que haberlos definido de antemano, o obviar algunos en los casos en los que no sean necesarios, pero a la vez se obliga al programador a ser mucho más metódico en su trabajo, puesto que el sistema no avisará de un posible error o despiste en el código.

Las bases de datos en MongoDB residen en un host que puede alojar varias bases de datos de forma simultánea almacenadas de forma independiente. Cada una de estas bases de datos puede contener una o más colecciones, a partir de las cuales se almacenarán los objetos o documentos [5].

5.2.2 BSON-JSON

BSON es un formato binario que se utiliza para almacenar la información en MongoDB. BSON es la codificación binaria del formato JSON. Esta codificación ha sido elegida porque presenta ciertas ventajas a la hora de almacenar los datos, como la eficiencia o la compresión.

Básicamente BSON y JSON son los formatos con los que trabaja MongoDB: JSON es el formato con el que se presenta la información a los usuarios y a las aplicaciones y BSON el formato que utiliza MongoDB de forma interna.

5.2.3 Estructura del documento

El reto principal que tienen los documentos como objetos en MongoDB es el de realizar un buen diseño de los mismos para que sean capaces de representar lo más ampliamente el mundo real. Para ello existen 2 herramientas que permiten a las aplicaciones representar las relaciones entre los datos: las referencias y los datos embebidos.

Una de las decisiones a tomar a la hora de implementar una aplicación que utilice MongoDB como capa de backend de aplicación es la de referenciar o embeber los documentos que se necesite.

5.2.4 Referencias

Las referencias almacenan relaciones entre los datos mediante enlaces entre documentos.

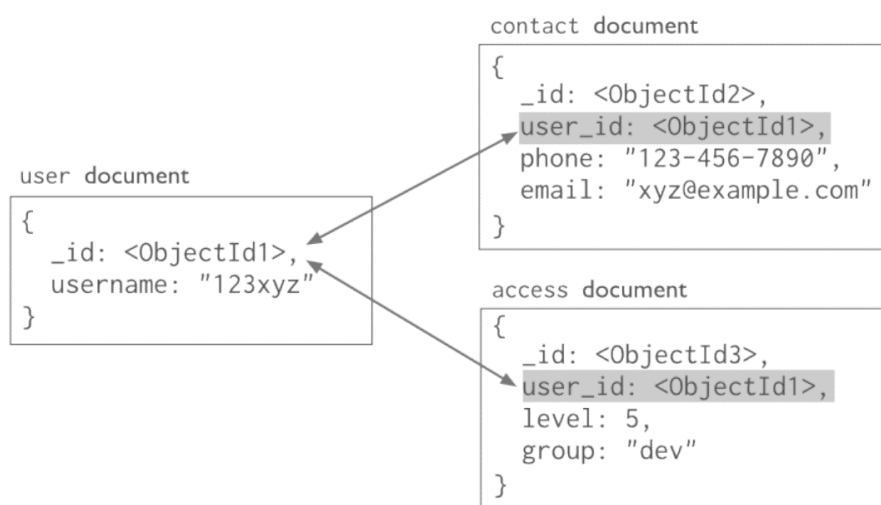


Figura. Modelo de Referencias

Los modelos de datos normalizados utilizan relaciones referenciales entre los documentos. Se usarán modelos de datos normalizados cuando:

- Embeber documentos produzca una duplicación de la información y los costes de mantenimiento sean mayores que las ganancias en lecturas.
- Sea necesario representar un modelo “varios a varios” con cierta complejidad.
- Para modelar conjuntos de datos de mucho tamaño.

5.2.5 Datos Embebidos

Los documentos embebidos representan relaciones entre los datos almacenando la información relacionada bajo el mismo documento. Los documentos permiten embeber estructuras documentales como subdocumentos dentro de un campo o de un array.

En general, se utilizarán modelos embebidos cuando:

- Existan relaciones contenidas entre dos entidades. Estos es, que dos entidades independientes sean habitualmente accedidas a través de sólo una de ellas. Por ejemplo, *Persona* y *Dirección*. Cuando únicamente se permitan búsquedas por *Persona* para obtener su dirección. En ese caso, es preferible tener el objeto *Dirección* embebido dentro del de *Persona*.
- Cuando exista una relación “uno a varios” entre entidades. En este caso, la parte de “varios” suele ir embebida dentro de la de “uno”. Los motivos son similares al anterior caso.



Figura. Datos Embebidos

- En general, en ambos se trata de minimizar en la medida de lo posible las operaciones de la base de datos. Embebiendo la información se consigue leer o escribir todos los datos simultáneamente y de forma secuencial.

Por otro lado, esta será una peor solución cuando los documentos crecen mucho después de su creación, puesto que se producirá fragmentación y las labores de mantenimiento del documento repercutirán directamente en el rendimiento a la hora de operar con él.

5.2.6 Índices

Las buenas decisiones en lo que a creación de índices se refiere será lo que determine, llegados a un volumen medio-alto de información, que una base de datos en MongoDB tenga unos tiempos de consulta razonables o los tenga muy elevados.

Los índices se utilizan para mejorar el rendimiento de las consultas. Es trabajo del administrador de base de datos construir índices en los campos que sean frecuentemente accedidos. Por defecto, MongoDB crea un índice sobre el campo *_id*, el identificador de cada documento.

A la hora de crear índices habrá que tener en cuenta una serie de consideraciones (10gen 2014):

- Cada índice requiere de al menos 8KB en disco. Cada índice activo ocupa espacio tanto en disco como en memoria. Hay que tenerlo en cuenta puesto que en algunos casos este espacio puede ser muy elevado.
- Los índices tienen un impacto negativo para las operaciones de escritura. Habrá que evaluar el uso que se le va a dar al sistema, puesto que los sistemas que reciba muchas escrituras y pocas lecturas, requerirán también tareas de actualización para cada índice que exista.
- Los índices no afectan para nada en las operaciones de lecturas sobre campos del documento que no han sido indexados.

6 DICCIONARIO DE DATOS

El directorio de datos ha sido desde siempre un componente del SGBD, encargado de describir dónde y cómo se almacenan los datos de la base, el modo de acceso y otras características físicas de los mismos, atendiendo de este modo las peticiones de los programas. Contiene, en definitiva, las especificaciones necesarias para pasar de la representación externa de los datos a la representación interna de los mismos. Su objetivo principal es transmitir al SGBD la información necesaria para poder acceder a los datos contenidos en la base.

Cuando los SGBD eran más limitados, los usuarios empleaban una lista "manual" donde anotaban la descripción narrativa y técnica (tamaño, tipo...) de los datos, así como los permisos de acceso a cada uno de ellos, el uso que las aplicaciones hacían de ellos, etc. Esta lista se denominaba diccionario de datos. Consistía en información sobre los datos, o, en otras palabras, datos sobre los datos, también conocidos como metadatos. Por eso a los diccionarios de datos reciben también la denominación de metabase. Posteriormente el diccionario de datos dejó de ser una simple lista en papel para poder tratarse de forma automatizada, como una especie de minibase de datos con programas habitualmente desarrollados a medida por el administrador. Sin embargo, la complejidad de estos programas era cada vez mayor, puesto que se añadían más y más funciones para los que originalmente no habían sido concebidos. Ello dio lugar a que durante los años 70 florecieran paquetes de software comercial de diccionario de datos, como por ejemplo DB/DC, DATADictionary, DATAMANAGER, DATADictionary SYSTEM, ADR/DATADictionary, LEXICON, etc. Actualmente se siguen comercializando diccionarios de datos como DICTIONARY MANAGER de Power-Components o DATA DICTIONARY MANAGER de International Business Developments.

Por tanto el diccionario de datos se considera como un conjunto de metadatos (datos sobre los datos) que contiene las características lógicas de los datos que se van a utilizar en el sistema que se programa, incluyendo nombre, descripción, alias, contenido y organización.

Si el diccionario de datos lo usan sólo los administradores, usuarios y diseñadores, y no el software del SGBD, se denomina diccionario de datos pasivo. De lo contrario, se denomina activo.

Por último, a partir de los 80, el diccionario de datos se incorporó a los SGBD como un componente más e integrándose junto al directorio. Al conjunto se le suele denominar catálogo o repositorio. El catálogo es el responsable de establecer la correspondencia entre los tres niveles de abstracción. Los módulos del SGBD usan y leen el catálogo con mucha frecuencia; por ello es importante implementar su acceso de la forma más eficiente posible.

Con la amplia difusión del modelo relacional el catálogo se almacena junto al resto de los datos de la base, en forma de tablas. La información almacenada en el catálogo de un SGBD relacional incluye, entre otros aspectos:

- Descripciones de los nombres de las relaciones (tablas).
- Nombres de los atributos.
- Tipos de datos de los atributos (dominios).

- Claves primarias.
- Claves ajenas.
- Obligación de establecer los datos como no nulos en algunas columnas.
- Vistas.
- índices.
- Estructuras de almacenamiento.
- Permisos.
- Información sobre el propietario de cada tabla.

Dado que los datos del catálogo se almacenan en forma de tablas, se puede acceder a los mismos mediante instrucciones de SQL

7 PLANTEAMIENTO DIDÁCTICO

Este tema y en general todo los incluidos dentro del bloque de Bases de Datos, de este temario oficial, tienen perfecta cabida dentro de los contenidos propios de la módulo de Base de Datos que se imparte en primer curso, en todos los Ciclos Superiores de la familia de Informática , donde tenemos atribución docente, igualmente dentro de Currículo de Educación Secundaria Obligatoria y Bachillerato, se pueden incluir en los tema de Ofimática de las asignaturas de Tecnología de la Información y la Comunicación, tanto en cuarto de la ESO como en primero de Bachillerato, independientemente del SGBD que usemos para impartir estos contenidos, ya que en gran medida, los saberes de este tema son comunes a muchos de ellos.

8 CONCLUSIÓN

Con este tema seguimos con el estudio mucho más en profundidad del mundo de las bases de datos y sus infinitas posibilidades dentro de las Ciencias de la Computación, hemos empezado por un conocimiento inicial de los Sistemas Gestores de Bases de Datos (SGBD) y sus posibilidades, continuamos conociendo como se definen los datos, como se describen y en que lenguajes en sistemas relacionales basados en SQL, como lo más modernos e innovadores orientados al tratamiento documental y de grandes volúmenes de información como son los sistemas en la Nube y en general los denominados NoSQL.

9 BIBLIOGRAFÍA

- [1] *Introducción a los Sistemas de Bases de Datos*; C.J. Date , Editorial Pearson.
- [2] *Fundamentos de Bases de Datos*; A.Silberschatz, HF. Korth, S. Sudarshan, Editorial McGraw Hill.
- [3] *Administración de Bases de Datos*; M.Mannino, Editorial McGraw Hill.
- [4] *Fundamentos de Sistemas de Bases de Datos*; R.Elmasri, S.Navathe , Editorial Pearson.
- [5] *Introducción a las bases de datos NOSQL usando MONGODD*; Saraza Cabezuelo, Editorial UOC.

Internet:

www.oracle.com

www.mysql.com

www.mongodb.com

[www. dbdb.io](http://www.dbdb.io)

www.cloud.google.com