

Modelo de datos relacional.
Estructuras. Operaciones.
Álgebra relacional.

TEMA 38 (37 SAI)

ABACUS NT

Oposiciones 2021

Índice

1. Introducción

1. Arquitectura de una base de datos relacional

- 1.1. El modelo de datos relacional
- 1.2. Las doce reglas de Codd
- 1.3. Estructura de datos relacional**
 - 1.3.1. Relaciones
 - 1.3.2. Dominios, atributos y claves
 - 1.3.3. Idea de normalización
 - 1.3.4. Restricciones de integridad

2. Formas normales

- 2.1. A. Primera forma normal
- 2.2. B. Segunda forma normal
- 2.3. C. Tercera forma normal
- 2.4. D. Forma normal de Boyce/Codd (BCNF)
- 2.5. E. Cuarta forma normal
- 2.6. F. Quinta forma normal

3. Lenguaje de definición de datos (DDL).

4. Lenguaje de manipulación de datos (DML)

- 4.1. SQL
 - 4.1.1. Instrucciones de actualización
 - 4.1.2. Consulta de Datos - Cláusula Select

5. Álgebra relacional

6. Conclusión

- 6.1. Relación del tema con el sistema educativo actual

7. Bibliografía

1. Introducción

El modelo relacional es posterior a los modelos jerárquicos y de red. Nació como consecuencia de los trabajos publicados en 1970 por E. F. Codd, aunque los primeros SGBD relacionales no aparecen en el mercado hasta principios de los años ochenta.

La estructura lógica y el modo de realizar las operaciones de E/S en el enfoque relacional son distintos respecto a los enfoques jerárquico y en red.

El modelo relacional representa la Base de Datos por medio de tablas llamadas relaciones, cada una de las cuales se implementa como un fichero. Se caracteriza porque:

- A la hora de operar con los datos, en lugar de hacerlo sobre registros actúa sobre la relación.
- Peticiones complejas de información que pueden necesitar varios ficheros se pueden especificar de forma muy simple y sin ninguna dificultad.

Tiene el inconveniente de que su implementación puede ser más compleja. En el mercado hay diversos sistemas relacionales que pertenecen a una de las siguientes categorías: SYSTEM R, QUEL y QBE.

1. Arquitectura de una base de datos relacional

En la arquitectura relacional de SYSTEM R, bajo las recomendaciones ANSI/SPARC, se pueden apreciar las siguientes partes:

1. El **nivel conceptual**. Definido por medio de una colección de relaciones, también llamadas tabla base, cada una de las cuales se representa por medio de un fichero almacenado distinto.
2. El **esquema externo**, llamado **submodelo relacional de datos o vista**. Es una tabla que no tiene existencia por sí misma y se deriva de una o más tablas base, a diferencia de un subesquema CODASYL, que sólo puede extraer datos de un fichero. Cada programa utiliza un buffer, llamado área de trabajo del usuario (UWA), para depositar o recuperar los datos antes de guardarlos en la Base de Datos o antes de ser procesados.
3. El **nivel interno**. Cada tabla base se representa por un fichero distinto que puede tener cualquier número de índices asociados.
4. Un **sublenguaje de datos**. Para el manejo de datos del sistema relacional. Con estos lenguajes, llamados de especificación, es posible procesar una tabla entera cada vez que se ejecuta una operación de Entrada/Salida en lugar de un registro. Los lenguajes que operan registro a registro se llaman lenguajes de navegación (jerárquicos y en red). Los sublenguajes de datos se pueden clasificar en dos grupos:
 - a) **Lenguaje de procedimiento**. Basado en el **álgebra relacional**, permite al usuario manipular relaciones por medio de operadores algebraicos relacionales para obtener los resultados requeridos.
 - b) **Lenguaje sin procedimiento**. Basado en el **cálculo relacional**, permite al usuario especificar exactamente qué datos desea, sin tener que detallar ni

codificar la forma de obtenerlos a partir de las relaciones disponibles en la Base de Datos.

1.1. El modelo de datos relacional

Las desventajas de los modelos jerárquico y en red condujeron a un intenso interés por el nuevo modelo de datos relacional. El modelo relacional fue descrito por primera vez por el Dr. Codd en 1970, y era un intento de simplificar la estructura de las Bases de Datos. Eliminaba las estructuras explícitas Padre/Hijo de la Base de Datos y en su lugar representaba todos los datos de la Base de Datos como sencillas tablas Fila/Columna de valores de datos.

Los primeros SGBD relacionales fallaron en implementar algunas partes clave del modelo de Codd, sólo ahora están incorporándose en productos comerciales. Conforme el concepto relacional crecía en popularidad, muchas Bases de Datos que se llamaban a sí mismas "relacionales" no lo eran de hecho. En respuesta a la corrupción del término "relacional" el Dr. Codd, escribió un artículo en 1985 estableciendo 12 reglas a seguir por cualquier Base de Datos Relacional. Las 12 reglas de Codd han sido aceptadas desde entonces como la definición de un SGBD verdaderamente relacional, sin embargo, es más fácil la siguiente definición informal:

Una Base de Datos Relacional es una Base de Datos en donde todos los datos visibles al usuario están organizados estrictamente como tablas de valores, y en donde todas las operaciones de la Base de Datos operan sobre estas tablas.

Esta definición está destinada específicamente a eliminar estructuras tales como los punteros incorporados de una Base de Datos jerárquica o en red. Un SGBD Relacional puede representar relaciones Padre/Hijo, pero éstas se representan estrictamente por los valores contenidos en las tablas de la Base de Datos.

1.2. Las doce reglas de Codd

1^a. Regla de Información. Toda información de una Base de Datos Relacional está representada explícitamente a nivel lógico y exactamente de un modo (mediante valores en tablas).

2^a. Regla de Acceso Garantizado. Todos y cada uno de los datos (valor atómico) de una Base de Datos Relacional se garantiza que sean lógicamente accesibles recurriendo a una combinación de nombre de tabla, valor de clave primaria y nombre de columna.

3^a. Tratamiento sistemático de valores nulos. Los valores nulos (distintos de la cadena de caracteres vacía o en blanco y distinta de cero o de cualquier otro número) se soportan en los SGBD completamente relacionales para representar la falta de información y la información inaplicable de un modo sistemático e independiente del tipo de datos.

4^a. Catálogo en línea dinámico basado en el modelo relacional. La descripción de la Base de Datos se representa a nivel lógico del mismo modo que los datos ordinarios, de modo que los usuarios autorizados puedan aplicar a su interrogación el mismo lenguaje relacional que aplican a los datos regulares.

5^a. Regla de Sublenguaje completo de datos. Un sistema relacional puede soportar varios modos de uso terminal (Ej. el modo de llenar con blancos). Sin embargo, debe haber al menos un lenguaje cuyas sentencias sean expresables, mediante alguna sintaxis bien definida, como cadenas de caracteres, y que sea completo en cuanto al soporte de los puntos siguientes:

- Definición de datos.
- Definición de vistas.
- Manipulación de datos (interactivo y por programa).
- Restricciones de integridad.
- Autorización.
- Fronteras de transacciones (comienzo, cumplimentación y vuelta atrás).

6^a. Regla de actualización de vista. Todas las vistas que sean teóricas actualizables son también actualizables por el sistema.

7^a. Inserción, actualización y supresión de alto nivel. La capacidad de manejar una relación de Base de Datos o una relación derivada como un único operando se aplica no solamente a la recuperación de datos, sino también a la inserción, actualización y supresión de los datos,

8^a. Independencia física de los datos. Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cualesquiera que sean los cambios efectuados, ya sea a las representaciones de almacenamiento o a los métodos de acceso.

9^a. Independencia lógica de los datos. Los programas de aplicación y las actividades terminales permanecen lógicamente inalterados cuando se efectúen sobre las tablas de base, cambios preservadores de la información de cualquier tipo que teóricamente permita alteraciones.

10^a. Independencia de integridad. Las restricciones de integridad específicas para una Base de Datos Relacional particular deben ser definibles en el sublenguaje de datos relacional y almacenadas en el catálogo, no en los programas de aplicación.

11^a. Independencia de distribución. Un SGBD Relacional tiene independencia de distribución.

12^a. Regla de no subversión. Si un sistema relacional tiene un lenguaje de bajo nivel (un registro cada vez), ese bajo nivel no puede ser utilizado para subvertir o suprimir las reglas de integridad y las restricciones expresadas en el lenguaje relacional de nivel superior (múltiples registros a la vez).

En su artículo de 1985 en Computer World, Codd presentó las 12 reglas que una Base de Datos debe obedecer para que sea considerada verdaderamente relacionar. Desde entonces se han convertido en una definición "semioficial" de una Base de Datos Relacional.

Ningún SGBD Relacional actualmente disponible satisface totalmente las 12 reglas de Codd. De hecho, se está convirtiendo en una práctica popular elaborar "tarjetas de tanteo" para productos SGBD comerciales, que muestran lo bien o mal que éstos satisfacen cada una de las reglas.

Desgraciadamente, las reglas son subjetivas de modo que los calificadores están generalmente llenos de notas a pie de página y no revelan demasiado acerca de los productos.

1.3. Estructura de datos relacional

1.3.1. Relaciones

Definición: Dada una serie de conjuntos D_1, D_2, \dots, D_n (no necesariamente distintos) se dice que R es una relación sobre estos n conjuntos si es un conjunto de n tuplas ordenadas $\langle d_1, d_2, \dots, d_n \rangle$ tales que d_1 pertenece a D_1 , d_2 pertenece a D_2 d_n pertenece a D_n . Los conjuntos D_1, D_2, \dots, D_n son los dominios de R . El valor de n es el grado de R . (Un **dominio** es el conjunto de valores que puede tener un atributo).

Es conveniente representar una relación en forma de tabla bidimensional. Cada renglón de la tabla representa una tupla de la relación. El número de tuplas de la relación se llama cardinalidad de la relación.

También se puede definir como producto cartesiano: Dada una serie de conjuntos D_1, D_2, \dots, D_n (no por fuerza distintos), el producto cartesiano de estos n conjuntos, denotado por $D_1 \times D_2 \times \dots \times D_n$ es el conjunto de todas las n tuplas ordenadas posibles $\langle d_1, d_2, \dots, d_n \rangle$ tales que d_1 pertenece a D_1 , d_2 pertenece a D_2 d_n pertenece a D_n .

Ahora se define que R es una relación sobre los conjuntos D_1, D_2, \dots, D_n si es un subconjunto del producto cartesiano $D_1 \times D_2 \times \dots \times D_n$.

1.3.2. Dominios, atributos y claves

Es importante apreciar la diferencia entre un dominio, por una parte, y las columnas - atributos- que se sacan de ese dominio, por otra. Un atributo representa el uso del dominio dentro de una relación. Para acentuar la distinción se pueden asignar nombres a los atributos que sean distintos de los dominios subyacentes.

La **clave primaria** es un identificador único de la tabla. Esto es, una columna (o combinación de columnas) con la propiedad de que, en un momento dado, no existen dos filas en la tabla conteniendo el mismo valor en la columna (o combinación de columnas).

1.3.3. Idea de normalización

En una Base de Datos relacional se requiere que todas las relaciones satisfagan la condición siguiente: *Que todo valor en la relación, es decir, cada valor de atributo en cada tupla sea atómico.*

En otras palabras, en cada intersección de un renglón y una columna de la tabla siempre hay exactamente un valor, nunca un conjunto de valores. (Se permite la posibilidad de valores nulos, es decir, valores especiales que representan algo desconocido o inapelable, como el caso de horas trabajadas de un empleado en vacaciones). De una relación que satisface la condición anterior se dice que está **normalizada** (O en primera forma normal).

Es elemental convertir una relación no normalizada en otra equivalente normalizada.

1.3.4. Restricciones de integridad

Las reglas de integridad son dos: **integridad de entidad** e **integridad referencial**. Estas reglas son generales, en el sentido de que deben aplicarse a toda Base de Datos definida de acuerdo a las prescripciones del modelo relacional. Aparte de estas dos, puede contener tantas reglas como se quiera, definidas de forma particular para ella.

Antes de enunciarlas, vamos a definir conceptos previos:

- **Claves candidatas:** Las claves primarias definidas anteriormente no son más que un caso particular de lo que se denominan claves candidatas. Estas son precisamente un identificador único para la relación. Por definición, toda relación tiene al menos una clave candidata, pero en general, puede tener muchas. Cada una será una combinación de atributos que identifican exactamente a una tupla. Para una relación dada, nosotros elegimos una de ellas y la convertimos en clave primaria. Una clave candidata tiene dos propiedades fundamentales:
 - **Unicidad:** En un momento dado, no puede haber dos tuplas en la relación en que todos los valores de los atributos que forman la clave sean exactamente iguales.
 - **Minimalidad:** Ninguno de estos atributos puede ser extraído de la clave sin que se pierda la primera propiedad.
- **Claves externas** (ajenas, foreign): Una clave externa es un atributo (o conjunto de atributos) en una relación A cuyos valores nos van a permitir buscar en la clave primaria de otra relación B.

Sobre estas definiciones podemos dar las dos restricciones de integridad:

- **Integridad de entidad:** No se permite ningún atributo que participe en la clave primaria con valores nulos.
- **Integridad referencial:** Si una relación A incluye una clave externa CE que corresponde con la clave primaria CP en una relación B, entonces todo valor de CE en A debe:

Ser igual al valor de CP en B.

o bien

Ser nulo.

Las dos relaciones A y B no tienen que ser necesariamente distintas.

En las dos restricciones, es fundamental el papel de los valores nulos. El valor nulo en el modelo relacional significa "propriamente inaplicable" o "información desconocida".

La justificación de las dos reglas de integridad es la siguiente:

La base de relaciones corresponde a entidades del mundo real.

Por definición, las entidades del mundo real son distinguibles. Esto es, tienen un identificador único para cada clave.

Las claves primarias llevan a cabo la función de identificación en el modelo relacional.

Así una clave primaria con valor nulo entra en contradicción con estos términos. No se pueden enunciar propiedades de objetos no definidos o parcialmente definidos.

Por otra parte, si un atributo existente en A refiere a una tupla en B, entonces esa tupla debe existir para que se pueda encontrar, o no tendría sentido la clave. No obstante, a veces es necesario admitir valores nulos en una clave externa; corresponderían al caso de "información desconocida".

2. Formas normales

Las formas normales son las líneas maestras para el diseño de los registros, estas reglas están pensadas para evitar anomalías en la puesta al día e inconsistencias en los datos. A pesar de ello, no hay obligación de seguir todas las normalizaciones hasta el final.

Antes de comenzar a enunciarlas, también hay que dar unos conceptos previos:

Dependencia funcional. Dada una relación R, se dice que el atributo Y de R es funcionalmente dependiente del atributo X de R (Se escribe $R.X \rightarrow R.Y$ y se dice "R.X determina funcionalmente a R.Y") si y solo si para cada valor de X hay asociado precisamente un único valor a la vez de Y. Tanto X como Y pueden ser compuestos, es decir, grupos de atributos. Informalmente, se puede decir que Y depende de X si no puede existir dos tuplas con el mismo valor en X y distinto en Y. *Ejemplo:* El atributo *nombre* depende funcionalmente del atributo *DNI*.

Determinante es un atributo de una relación del que otro atributo cualquiera de ella es totalmente dependiente.

Hecho multivaluado es una relación muchos a muchos, como la relación hombre-empleo, en que un hombre puede tener varios empleos y un empleo (p.e. Electricista) lo pueden tener muchos hombres.

2.1. A. Primera forma normal

La primera forma normal simplemente define las tablas que forman el modelo relacional y excluye campos repetidos y grupos.

Una relación está en la primera forma normal si y sólo si los dominios de sus atributos toman sólo valores atómicos. Es decir, si no contiene atributos multivaluados.

Esta definición es equivalente a la anterior de relación normalizada e indica que una relación está en primera forma normal si de hecho es correcta.

2.2. B. Segunda forma normal

La segunda forma normal trata de las relaciones entre los atributos clave y los no clave. Esta forma es violada cuando un campo no clave es de hecho un subconjunto de una clave.

Una relación R está en segunda forma normal si y sólo si está en primera forma normal y todo atributo no clave es completamente dependiente de la clave primaria. Es decir, si todos sus atributos dependen funcionalmente de toda la clave primaria. (Si la clave de una relación es simple ya se tiene la segunda forma normal).

La relación anterior RI, viola esta regla, ya que, el atributo direcc-almacén no depende de la clave (pieza, almacén), sino que es función sólo de almacén. Para que se cumpla la segunda forma normal en este caso habrá que descomponer la relación en dos:

2.3. C. Tercera forma normal

La tercera forma normal no se cumple cuando un campo no clave es de hecho clave de otro campo.

La tercera forma normal se puede enunciar como:

Una relación está en la tercera forma normal si y sólo si está en la segunda y todo atributo no-clave es dependiente de la clave de forma no transitiva. Es decir, si está en la segunda forma normal y ningún atributo depende transitivamente de la clave primaria.

Este enunciado significa que la dependencia de Y con respecto a la clave X no se hace a través de un tercer atributo Z sino directamente:

$$R.X \rightarrow R.Y \text{ en vez de } R.X \rightarrow R.Z \rightarrow R.Y$$

2.4. D. Forma normal de Boyce/Codd (BCNF)

La anterior definición de Codd de la tercera forma normal, tiene ciertas inadecuaciones, en concreto no trata los casos en que hay varias claves candidatas que se componen de varios atributos y están solapadas (tienen atributos comunes). Para complementarla se enuncia la forma normal de Boyce/Codd que mejora la primera definición.

Una relación está en la forma normal de Boyce/Codd si y sólo si todo determinante es clave candidata. Es decir, si todas las partes izquierdas de las dependencias son claves candidatas.

Toda relación en forma normal de Boyce/Codd está en la tercera forma normal por definición. La interpretación es que todo atributo que determine únicamente otro en una relación debe poder ser clave de la misma.

2.5. E. Cuarta forma normal

Tanto la cuarta como la quinta forma normal trabajan con hechos multivaluados.

Bajo la cuarta forma normal, dos o más hechos independientes multivaluados no pueden coexistir en la misma relación.

Una relación está en cuarta forma normal, si y sólo si siempre que existe un hecho multivaluado, $A \rightarrow B$, entonces todos los atributos de la relación son funcionalmente dependientes de A. Es decir, si está en BCNF y no existe entre sus atributos ninguna dependencia multivaluada que no sea combinación de dependencias funcionales.

Por supuesto, A y B pueden ser compuestos.

En otras palabras, las únicas dependencias (simples o multivaluadas) dentro de la relación son las del tipo $K \rightarrow X$ donde K es una clave candidata completa y X cualquier otro atributo.

2.6. F. Quinta forma normal

La quinta forma normal trata los casos en que la información está en distintas piezas que hay que mantener separadamente y sin redundancia.

Informalmente, podemos decir que una relación está en la quinta forma normal cuando su información no puede ser reconstruida si la dividimos en pequeños trozos.

Una relación está en la quinta forma normal si y sólo si toda dependencia en ella es una consecuencia de sus claves candidatas.

3. Lenguaje de definición de datos (DDL).

Cualquier lenguaje de bases de datos está formado por un DDL y por un DML. Existe un lenguaje considerado como estándar para manipular bases de datos relacionales: SQL (Structured Query Language). Estudiaremos el DDL y el DML que utiliza el SQL. Las características principales de SQL son su sencillez, su carácter estándar y ser un lenguaje declarativo o no procedural (para que SQL realice una acción concreta no se le dice cómo tiene que hacerla, sino lo que queremos obtener).

El lenguaje de definición de datos (DDL) proporciona órdenes para definir, eliminar y modificar tablas, así como para crear índices y vistas. Las órdenes SQL más importantes del DDL son:

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
(nombrecoll tipocoll
[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 elimina 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 cláusula REFERENCES, no se utiliza la cláusula 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.

Borrado de una tabla (estructura y datos)

```
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, las vistas creadas directa o indirectamente sobre esta tabla son desactivadas de forma automática por ORACLE, pero no borradas.

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 cláusula CASCADE CONSTRAINT permite suprimir estas restricciones de integridad referencial en las tablas 'descendientes'.

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];
```

Con DROP STORAGE se desasigna todo el espacio. Con DROP REUSE mantendrá reservado el espacio para nuevas filas.

Modificar la estructura de una 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_restricción
```

Donde:

- ENABLE VALIDATE activa la restricción si el conjunto de filas ya presentes en la tabla cumplen dicha restricción.
- ENABLE NOVALIDATE activa la restricción para las siguientes instrucciones de manipulación de datos.
- DISABLE desactiva la restricción.

Hay que tener en cuenta que si la tabla está vacía, al añadir una columna con la restricción NOT NULL no habrá ningún error, pero si tiene filas no permitirá añadir una columna con esta opción.

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.

Borrado de una vista.

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

```
DROP VIEW Nombre_vista
```

Creación de un índice

Los índices sirven para mejorar el rendimiento de las consultas. El optimizador de Oracle los utiliza implícitamente y se actualizan de forma automática al actualizar las filas.

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

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

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_indexe;
```

4. Lenguaje de manipulación de datos (DML)

4.1. SQL

4.1.1. Instrucciones de actualización

Las instrucciones de actualización son aquellas que no devuelven ningún registro, sino que son las encargadas de acciones como añadir, borrar y modificar registros. En este post veremos las órdenes INSERT, DELETE y UPDATE.

- **INSERT**: sirve para insertar registros en una tabla
- **DELETE**: permite eliminar registros de una tabla.
- **UPDATE**: permite modificar registros de una tabla.

Inserción de registros

La sentencia INSERT nos permite introducir nuevas filas en una tabla de la base de datos. Su sintaxis más simple es:

```
Insert into tabla ([<lista_campos>]) Values ([<lista de="de" valores="valores">])
```

donde tabla representa la tabla a la que queremos añadir el registro y los valores que siguen a la cláusula VALUES son los valores que damos a los distintos campos del registro. Si no se especifica la lista de campos, la lista de valores debe seguir el orden de todos los campos de la tabla.

La lista de campos a llenar se indica si no queremos llenar todos los campos. Los campos no llenados explícitamente con la orden INSERT, se llenan con su valor por defecto (DEFAULT) o bien con NULL si no se indicó valor alguno.

Insert into tabla ([<lista_campos>])

Select

En esta segunda sintaxis se permite añadir registros a una tabla obteniéndolos mediante una consulta SELECT. Por supuesto el tipo de los campos y el orden de estos debe coincidir con los de la lista de campos o con los de la tabla destino si estos últimos no se indican.

Borrado de registros

La sentencia DELETE nos permite borrar filas de una tabla de la base de datos. Su sintaxis más simple es:

Delete [from] tabla [Where <condición>]

La sentencia DELETE es de tipo DML mientras que la sentencia TRUNCATE es de tipo DDL; la diferencia está en dos aspectos:

- DELETE puede borrar 0, 1 o más registros de una tabla, mientras que TRUNCATE los borra todos.
- DELETE puede disparar un trigger de tipo DELETE asociado a la tabla con la que estemos trabajando, mientras que TRUNCATE no disparará ningún trigger.

Una vez que se han eliminado los registros utilizando la sentencia DELETE, no puede deshacer la operación. Si desea saber qué registros se eliminarán, primero examine los resultados de una consulta de selección que utilice el mismo criterio y después ejecute la consulta de borrado. Mantenga copias de seguridad de sus datos en todo momento. Si elimina los registros equivocados podrá recuperarlos desde las copias de seguridad. Hay que tener en mucho cuidado con la restricción de ON DELETE CASCADE.

Modificación de registros

La sentencia UPDATE nos permite modificar filas de una tabla de la base de datos. Su sintaxis más simple es:

Update tabla Set columna1= valor1 [, columna2= valor2,] [Where <condición>]

Se modifican las columnas indicadas en el apartado SET con los valores indicados. La cláusula WHERE permite especificar qué registros serán modificados.

Su segundo tipo de sintaxis es:

Update tabla Set columna1=(Sentencia SELECT) [Where <condición>]

Este tipo de actualizaciones sólo son válidas si la Sentencia SELECT devuelve un único valor, que además debe de ser compatible con la columna que se actualiza.

Sentencia MERGE

Este comando sirve para actualizar los valores de los registros de una tabla a partir de valores de registros de otra tabla o consulta. Permite pues combinar los datos de dos tablas a fin de actualizar la primera. Su sintaxis es:

```
MERGE INTO tabla alias
USING (instrucción SELECT) alias
ON (condición de unión)
WHEN MATCHED THEN
    UPDATE SET col1=valor1 [col2=valor2]
WHEN NOT MATCHED THEN
    INSERT (listaDeColumnas)
VALUES (listaDeValores)
```

MERGE compara los registros de ambas tablas según la condición indicada en el apartado ON. Compara cada registro de la tabla con cada registro del SELECT. Los apartados de la sintaxis significan lo siguiente:

- **tabla** es el nombre de la tabla que queremos modificar.
- **USING**. En esa cláusula se indica una instrucción SELECT que muestre una tabla que contenga los datos a partir de los cuales se modifica la tabla.
- **ON**. permite indicar la condición que permite relacionar los registros de la tabla con los registros de la consulta SELECT.
- **WHEN MATCHED THEN**. El UPDATE que sigue a esta parte se ejecuta cuando la condición indicada en el apartado ON sea cierta para los dos registros actuales.

- **WHEN NOT MATCHED THEN.** El INSERT que sigue a esta parte se ejecuta para cada registro de la consulta SELECT que no pudo ser relacionado con ningún registro de la tabla.

4.1.2. Consulta de Datos - Cláusula Select

La instrucción DML más utilizada es la de consulta de datos SELECT. Su función principal es la de recuperar filas de la tabla o tablas. Además, esta sentencia es capaz de realizar las siguientes funciones:

- Obtener datos para la creación de una tabla.
- Realizar operaciones estadísticas.
- Definir cursos.
- Realizar operaciones totalizadoras sobre grupos de valores que tienen los mismos valores en ciertas columnas.
- Se puede utilizar como sub-consulta para formar parte de una condición.

La sintaxis básica es:

```
SELECT select_list
      FROM table_source
      [WHERE search_condition]
      [GROUP BY group_by_expression]
      [HAVING search_condition]
      [ORDER BY order_expression [ASC | DESC] ]
```

Vamos a ir viendo las diferentes cláusulas que componen la sentencia SELECT:

- Cláusula **SELECT** : Especifica qué columnas o expresiones han de ser devueltas por la consulta
- Cláusula **FROM**: En esta cláusula se indican la tabla o tablas a las que vamos a tener acceso.
- Cláusula **WHERE**: Selecciona aquellas filas que cumplen la condición especificada
- Cláusula **GROUP BY**: Agrupa las filas seleccionadas por la cláusula WHERE
- Cláusula **HAVING**: Especifica una condición de selección para un grupo
- Cláusula **ORDER BY**: Ordena las filas seleccionadas por la cláusula WHERE

5. Álgebra relacional

El lenguaje DML está basado en el álgebra relacional. Las *operaciones básicas* del álgebra relacional son:

Selección: Extraer las tuplas especificadas de una relación (filas).

Proyección: Extraer atributos especificados de una relación (columnas).

Producto: Construye una relación de otras dos combinando de todas las formas posibles pares de tuplas, una de cada relación.

Unión: Construye una relación uniendo las tuplas que aparecen en una o ambas relaciones.

Intersección: Construye una relación con las tuplas que aparecen en ambas relaciones a la vez.

Diferencia: Construye una relación con las tuplas que aparecen en la primera y no en la segunda.

Enlace: Construye una relación de otras dos enlazando todas los posibles pares de tuplas, de forma que se cumpla una condición específica.

División: Tomando dos relaciones, una binaria (dos atributos) y otra unaria (uno sólo), construye una relación consistente en todos los valores de la relación binaria que tienen un enlace en la otra relación.

Mediante este conjunto de ocho operaciones, se puede construir cualquier relación a partir de las ya dadas. El lenguaje resultante es altamente amigable y fácil de aprender. Plantea, sin embargo, el inconveniente de que es difícil el determinar la manera óptima de crear una relación con estas primitivas, ya que suelen existir varias opciones posibles, con costes distintos de utilización del ordenador.

6. Conclusión

A modo de síntesis, se podría destacar el éxito en el sector productivo de los SGBD, cuya utilidad hoy en día, ya nadie cuestiona. Estos ofrecen, en su mayoría, lenguajes de manipulación de datos declarativos, con los que los programadores de aplicaciones pueden abstraerse de cómo se realizan las consultas, e incluso, de cómo se optimizan. En este sentido cabe destacar SQL, como el lenguaje más utilizado, aunque también merece fijarse en el surgimiento de nuevas propuestas, a nivel comercial, de nuevos lenguajes y la existencia de herramientas de mapeo, como los ORM, para mapear SQL con lenguajes orientados a objetos.

6.1. Relación del tema con el sistema educativo actual

Este tema es aplicado en el aula en los módulos profesionales siguientes, con las atribuciones docentes indicadas (PES/SAI):

Formación profesional básica

- Operaciones auxiliares para la configuración y la explotación(TPB en Informática de Oficina/ TPB en informática y Comunicaciones) (PES/SAI)
- Ofimática y archivo de documentos (TPB en Informática de Oficina) (PES/SAI)

Grado Medio

- Aplicaciones ofimáticas (GM de SMR) (PES/SAI)

Grado Superior

- Gestión de bases de datos (ASIR) (PES)
- Bases de Datos (DAW/DAM) (PES)

Bachillerato:

- 4º ESO – Tecnología de la Información y la comunicación (PES)
- Bachillerato – Tecnologías de la Información y la Comunicación (PES)

7. Bibliografía

- C.J. Date: **Introducción a los sistemas de bases de datos** Pearson, 2001.
- Elmasri, R.A. y Navathe S.B: "**Fundamentos de Sistemas de Bases de Datos**". Addison-Wesley, 3^a Edic, 2002.
- Olga Pons, J M Medina, M.A. Vila. **Introducción a los Sistemas de Bases de Datos** Edt Paraninfo (2005)
- Korth, H.F. y Silberschatz: "**Fundamentos de Bases de Datos**". McGraw -Hill, 4^a Edic., 2002.
- Garcia-Molina, H.; Ullman, J.D.; Widom, J. **Database systems: the complete book** - Pearson Education Limited, 2013.
- Abraham Silberschatz, Henry F. Korth, y S. Sudarshan, **Fundamentos de bases de datos** Edt. Mc Graw-Hill (2014)
- <https://elbauldelprogramador.com/> (2020)
- www.Unir.net (2020)