

El rol del lenguaje SQL en los SGBDR y en la implementación del Modelo Relacional.

Claudia Jiménez, Thomas Armstrong
cjimenez@inf.udec.cl, tamrstro@inf.udec.cl

Resumen

El presente artículo muestra el papel de los lenguajes en los SGBD, indicando los objetivos que deben cumplir, y cómo deben hacerlo. Se muestra cómo SQL cumple estos objetivos, adoptando una postura crítica y objetiva, en especial con respecto a su implementación del Modelo Relacional

Introducción

Los sistemas informáticos que permiten a las personas utilizar computadores para su beneficio, deben satisfacer demandas de almacenamiento y procesamiento de datos. Es el conjunto de estas dos capacidades de los computadores, junto con la posibilidad de presentar los datos en un formato familiar al usuario (textos, esquemas, gráficos, etc.), lo que hace que el computador sea una herramienta útil y necesaria.

En los sistemas de información se tienen los siguientes componentes:

- Sistema de Gestión de Base de Datos;
- Subsistema de Recuperación de Datos;
- Subsistema de Almacenamiento de Datos;
- Sistema de Comunicaciones:
- Redes de Computadores;
- Protocolos de Comunicación;
- Lenguaje de Comunicación;
- Interfaz con el Usuario.

Como es bien sabido, el objetivo de los Sistemas de Gestión de Bases de Datos (SGBD) es almacenar los datos en un computador, de forma segura y consistente, para permitir las solicitudes de acceso a los datos formuladas por los usuarios. Estos sistemas permiten la implementación de bases de datos utilizando algún modelo, con el fin de satisfacer demandas particulares de los usuarios.

Para la definición de la estructura de la base de datos (BD), realizada por el diseñador según un modelo de datos (MD), así como para la utilización de ella por los usuarios, son necesarios lenguajes de comunicación con los SGBD.

Estos lenguajes deben así permitir las siguientes operaciones:

- Definir de Estructura de la BD;
- Crear las bases de datos necesarias;
- Eliminar bases de datos existentes;
- Reestructurar las bases de datos existentes;
- Consultar acerca de la estructura de las bases de datos existentes;
- Manipular los Datos Almacenados;
- Insertar de datos en las bases de datos;

- Eliminar datos de las bases de datos;
- Modificar datos de las bases de datos;
- Consultar datos de las bases de datos.

Mediante las operaciones mencionadas es posible utilizar los SGBD para almacenar bases de datos, las cuales pueden ser parte de sistemas informáticos completos, solucionándose así la necesidad de los usuarios por almacenamiento persistente de datos.

El Rol de los Lenguajes de SGBD

Generalmente, los SGBD utilizan algún lenguaje específico, de forma que sólo a través de él se puede interactuar con el sistema para realizar alguna de las operaciones expuestas. La estructura del lenguaje depende directamente del modelo de datos que implementa el SGBD.

Cuando se habla de un lenguaje de SGBD, se tiende a pensar que el lenguaje es el SGBD en sí, lo cual es un error, ya que el lenguaje como tal sólo es parte de un conjunto de herramientas que permiten la implementación de sistemas de información computacionales.

El papel que juegan los lenguajes de SGBD es simplemente proveer un protocolo de comunicaciones con los SGBD, de forma que cualquier usuario tenga acceso al SGBD (a través de una red, por ejemplo) y pueda realizar transacciones con él. Se debe notar que los usuarios pueden ser personas, aplicaciones cliente, o incluso otros SGBD.

De esta manera, cuando un usuario desea comunicarse con un SGBD, puede hacerlo sólo a través del lenguaje del SGBD, ya sea utilizando una interfaz de línea de comandos, que permite realizar cualquier operación con el lenguaje; o una aplicación cliente específica, desarrollada para satisfacer necesidades particulares de información, y que utiliza el mismo lenguaje.

En el caso de los SGBD Relacionales (basados en el Modelo Relacional de datos), el lenguaje utilizado es SQL (Structured Query Language), cuyos aspectos se discutirán posteriormente.

Elementos de un Lenguaje de SGBD

Los lenguajes de SGBD deben permitir realizar todas las operaciones antes mencionadas sobre las bases de datos y los datos que ellas contienen. Así, surge de manera natural la siguiente clasificación de estos lenguajes:

- Lenguaje de Definición de Datos (LDD):
- Permite definir y manipular la estructura de bases de datos almacenadas en el SGBD;
- Lenguaje de Manipulación de Datos (LMD)
- Permite ingresar, modificar y eliminar datos de las bases de datos almacenadas en el SGBD;
- Permite realizar y controlar las transacciones con el SGBD.

Es una práctica generalizada el definir un solo lenguaje que permite realizar todas las operaciones descritas, de manera que exista la posibilidad de utilizar operaciones de diferentes tipos sin restricción.

Lenguaje de Definición de Datos

Los LDD permiten definir la estructura de la base de datos, y generalmente es el diseñador de ésta quien lo utiliza. La estructura de la base de datos se define a tres niveles:

Estructura Externa o de Usuario

Esta parte del LDD permite definir qué visión deben tener los usuarios sobre la base de datos, la cual puede ser mucho más simple que la estructura lógica global.

En el caso de los SGBDR y el lenguaje SQL, se tienen los siguientes elementos de definición de estructura externa:

- Vistas (CREATE VIEW);
- Procedimientos Almacenados (CREATE PROCEDURE / FUNCTION);

Estructura Lógica Global

Esta otra parte del lenguaje permite definir la estructura de los datos, esto es, qué datos se deben almacenar, basándose en el modelo de datos (MD) que utiliza el SGBD.

En el caso de los SGBD Relacionales (SGBDR) y el lenguaje SQL, se tienen los siguientes elementos de definición de estructura lógica:

- Dominios (CREATE DOMAIN);
- Relaciones (CREATE TABLE);
- Atributos (COLUMN);
- Claves (PRIMARY / FOREIGN KEY);
- Restricciones (CHECK)

Estructura interna o Física

Debido a la estructura lógica de los datos y a la semántica en la cual están inmersos, no es posible utilizar de forma óptima y eficiente los datos almacenados en el SGDB. Así, se hace necesario que exista una parte del LDD dedicada a especificar características externas a la estructura global, que tienen que ver generalmente con la forma en que los datos se almacenan y cómo se utiliza la base de datos (datos más frecuentemente usados, forma en que se buscan, etc.).

En el caso de los SGBDR y el lenguaje SQL, se tienen, por ejemplo, los siguientes elementos de definición de estructura lógica:

- Índices (CREATE INDEX);
- Secuencias (CREATE SEQUENCE).

Lenguaje de Manipulación de Datos

Este lenguaje permite la utilización de los datos almacenados en la base de datos, y es usado generalmente por usuarios finales o por aplicaciones cliente utilizadas por ellos.

La forma más común de utilización del LMD es mediante el uso de lenguajes de programación convencionales que poseen al LMD del SGBD como lenguaje embebido. Así, se puede programar la aplicación cliente en el lenguaje que se elija, pudiendo realizar llamadas a sentencias del LMD cuando sea necesario.

Se diferencian dos tipos de sentencias de manipulación de datos

Actualización y Consulta

Son sentencias que permitan manipular los datos, esto es, insertar nuevos datos, consultar la base de

datos, etc.

En el caso de los SGBDR y el lenguaje SQL, se tienen las siguientes sentencias de manipulación de datos:

- Inserción de Datos (INSERT);
- Eliminación de Datos (DELETE);
- Modificación de Datos (UPDATE);
- Consulta de Datos (SELECT).

Control de Transacciones

Son sentencias que permiten controlar el desarrollo de una transacción con el SGBD, de manera que el usuario pueda, a medida que se va realizando la transacción, confirmarla o retractarse de ella.

En el caso de los SGBDR y el lenguaje SQL, se tienen las siguientes sentencias de control de transacciones:

- Confirmación de transacción (COMMIT);
- Retracción de Transacción (ROLLBACK);
- Almacenamiento de puntos intermedios de transacciones (SAVEPOINT).

Formas en que se Deben Incorporar los Elementos al Lenguaje

En lo que sigue se describen las características que un lenguaje de SGBD debe cumplir desde diferentes puntos de vista:

Características de un Buen Lenguaje

Todo lenguaje, ya sea de programación, de modelamiento, o de cualquier tipo, debe cumplir con las siguientes características fundamentales:

Claridad, Simplicidad y Unidad

Los lenguajes deben ser conceptualmente integrales, esto es, proveer de un conjunto de conceptos y de reglas para relacionarlos, de manera simple, clara y unificada. Además, deben ser legibles, para facilidad de comprensión.

Ortogonalidad

Las características del lenguaje, como funciones y procedimientos, por ejemplo, deben ser combinables en todas sus alternativas. Así, todas las posibles expresiones de un cierto tipo se pueden utilizar en todos los lugares en que se puede utilizar una expresión de ese tipo.

Adaptación Natural al Problema

Los lenguajes deben ser aplicables con naturalidad a los problemas. De aquí que surjan diferentes lenguajes para diferentes dominios de aplicación.

Soporte de Abstracción

Los lenguajes deben proveer medios de abstracción, ya sea de datos y/o procedural. Así se permite al

programador el representar el dominio del problema, por medio del lenguaje, sin preocuparse por los detalles de implementación más que una vez.

Verificación Fácil de Programas

Los lenguajes debe permitir una verificación fácil de los programas construidos, ya sea mediante verificación formal, lectura de ellos, o test.

Ambiente Apropiado de Codificación

Hoy en día no sólo el lenguaje en sí es importante, sino que también el ambiente en que se realiza la programación en él. Parte de este ambiente es una implementación completa, correcta y documentada del lenguaje, así como herramientas de creación, de verificación, o de prueba de código.

Portabilidad de Programas

Los lenguajes deben estar definidos de manera independiente de los sistemas computacionales en que se utilizan, de forma estandarizada. Así es posible la existencia de implementaciones del mismo lenguaje en diferentes sistemas, para poder utilizar un mismo código en todos ellos sin modificaciones.

Costo de Uso Bajo

El costo de uso de un lenguaje esta asociado al costo en tiempo y recursos de hardware, software y otros que es necesario para crear, compilar, ejecutar, probar, utilizar y mantener los programas.

Objetivos de los Lenguajes de los SGBD

Los SGBD pretenden ofrecer servicios de almacenamiento de datos de una forma abstracta, pudiendo el usuario de esos datos utilizarlos sin preocuparse por detalles de representación o de implementación de la base de datos. En general, los SGBD ofrecen un servicio que cumple con las características que se mencionan a continuación.

Éstas deben estar presentes en los lenguajes de SGBD, ya que es a través de él que el usuario del SGBD aprecia las características.

Soporte para Modelo de Datos

Los SGBD permiten almacenar datos de manera abstracta, en base a un modelo de datos determinado. El lenguaje del SGBD debe ser capaz de definir una estructura lógica de los datos de acuerdo al modelo de datos elegido, soportándolo en su totalidad.

Independencia Física

La forma en que se almacenan los datos en la BD no influye en su manipulación lógica (ya sea por LDD o por LMD), de manera que cambios en el almacenamiento físico no influyen en los programas de usuario.

Independencia Lógica

La ejecución de operaciones del LMD por parte de un usuario no repercute en las operaciones que realizan otros usuarios sobre la misma BD. Así, los problemas de concurrencia deben ser transparentes al lenguaje del SGBD.

Flexibilidad

El LDD del lenguaje del SGBD debe permitir definir diferentes formas de ver los datos, para satisfacer requerimientos diferentes por parte de usuarios diferentes.

Uniformidad

La estructura lógica de la BD definida mediante el LDD debe ser uniforme y acorde al modelo de datos del SGBD, para facilitar la manipulación de esta estructura.

Sencillez

El lenguaje del SGBD debe ser sencillo, para facilitar a los usuarios la comprensión de la estructura lógica de los datos.

LDD y LMD

Como ya se vio anteriormente, los lenguajes de SGBD necesitan de dos partes: un lenguaje de definición de datos, y uno de manipulación de datos.

Al momento de decidir qué lenguaje utilizar, una posibilidad es usar dos lenguajes diferentes para cada parte del lenguaje. Sin embargo, generalmente se define un solo lenguaje, el cual incluye en una misma sintaxis el LDD y el LMD.

Cualquiera sea la elección, el lenguaje debe poseer sentencias para cada funcionalidad requerida, ya sea del LDD o del LMD.

LDD

En el caso específico del LDD, el lenguaje del SGBD debe ser capaz de definir la estructura lógica de la BD, sin entrar en detalles de implementación ni mecanismos en que se accede a los datos de la BD.

La forma idónea de realizar lo anterior es mediante un lenguaje declarativo, el cual permite declarar la estructura del modelo de acuerdo al modelo de datos que utiliza el SGBD.

LMD

Para el caso del LMD, el lenguaje del SGBD debe incluir formas de especificar qué se desea hacer con los datos (insertar, recuperar, modificar o borrar datos), sin entrar en detalles acerca de cómo se realizan estas operaciones.

Igual que en el caso anterior, la mejor manera de realizar esto es mediante un lenguaje declarativo que permita especificar la estructura de la operación a realizar, de acuerdo siempre con el modelo de datos utilizado por el SGBD.

El Lenguaje SQL y el Modelo Relacional

Evolución del SQL

Las propuestas de Codd en 1970 relativas al Modelo Relacional (MR) incentivaron a las universidades y centros de investigación a diseñar lenguajes que soporten este modelo.

El lenguaje relacional SQL fue desarrollado originalmente por IBM en un sistema prototipo llamado System R, con el nombre de SEQUEL, en los años 1974-1975. Posteriormente por razones legales, pasó a llamarse SQL (Structured Query Language). En 1979 surgieron los primeros productos comerciales que lo implementaron (SQL Oracle). Después de casi 13 años, en 1992, el lenguaje SQL fue finalmente estandarizado por los grupos ANSI e ISO, llamándose SQL-92 o SQL2.

Características de SQL

La función del lenguaje SQL es la de soportar la definición, manipulación y control de los datos en una base de datos relacional. Desde el punto de vista de este lenguaje una base de datos relacional es un conjunto de tablas, donde cada tabla es un conjunto de filas y columnas.

SQL como Lenguaje

Primero que nada debe quedar claro que SQL no es un lenguaje de programación como C o Pascal, ya que no posee sentencias de selección o iteración (IF ... THEN, FOR, WHILE, etc.). Por lo tanto, cuando se desean utilizar los datos de la BD para realizar algún proceso, se debe programar en otro lenguaje (lenguaje anfitrión), desde el cual se hacen llamadas a sentencias SQL (SQL embebido) cuando se desea interactuar con la BD.

Además, como lenguaje es bastante natural y fácil de leer, y se aplica de forma natural al problema de implementar BD Relacionales, pero como consecuencia de esto es poco estructurado y utiliza muchas palabras superfluas (que podrían omitirse quitando claridad a las sentencias).

SQL permite abstraer al usuario de la estructura lógica de la BD mediante las vistas, aunque éstas imponen bastantes restricciones a los procesos de insertar y modificar datos en ellas. Además, habiendo sido estandarizado ya hace mucho tiempo, las BD Relacionales en SQL son muy portables, ya que existen muchos SGBDR que lo soportan.

Al ser SQL un lenguaje declarativo, es muy fácil verificar los programas, ya que simplemente se declara lo que se desea como resultado de la ejecución.

SQL y los SGBD

SQL es mucho más que un lenguaje de consulta de BD, ya que permite:

- Definir la estructura de los datos;
- Recuperar y manipular datos;
- Administrar y controlar el acceso a los datos;
- Compartir datos de forma concurrente;
- Asegurar su integridad.

Lo anterior se hace basándose siempre en el Modelo Relacional de datos. En consecuencia, SQL permite hacer efectivos los objetivos de los SGBDR, permitiendo a los usuarios utilizar todo el potencial de los SGBDR.

SQL: LDD y LMD

SQL posee sentencias tanto para definición de la estructura de los datos, como para la manipulación de ellos, tal como se indicó en el capítulo III (página *). Así, SQL provee sentencias para ambos tipos de tareas.

SQL y el Modelo Relacional

El Modelo Relacional se compone, formalmente, de las siguientes partes:

- Parte Estática:
- Dominios;
- Atributos;
- Relaciones;
- Reglas de Integridad;
- Parte Dinámica:
- Álgebra Relacional.

El lenguaje SQL permite representar casi todos los elementos del MR, salvo los conceptos de dominio (que no se soporta) y de relación (que se soporta de forma restringida).

Cabe destacar que la manipulación de datos proveída por SQL, es más poderosa que la definida originalmente en el MR, mediante el álgebra relacional. Esto ha motivado que ésta haya sido extendida para representar las posibilidades de SQL.

Se ha considerado importante analizar en detalle qué aspectos del MR no son soportados por el lenguaje SQL2, de forma que se ha dedicado el apartado siguiente a este efecto.

Aspectos del Modelo Relacional no soportados por SQL

Dominios Relacionales

Los dominios SQL y los dominios del MR son desafortunadamente conceptos diferentes. Un dominio relacional es un tipo de dato con las siguientes características:

- Los dominios pueden ser definidos por el usuario;
- La definición de un dominio especifica un conjunto de valores permitidos para éste;
- Los valores del dominio pueden ser de una complejidad arbitraria;
- La representación interna de dichos valores no es visible para el usuario;
- Los valores son manipulados solamente por operadores definidos por el dominio;
- Los dominios pueden ser subtipos de otros dominios;
- Se puede definir un dominio en términos de otro dominio (herencia): Si B es un subtipo de A, entonces todos los operadores que son aplicables a A son también aplicables a B, pero B puede tener operadores propios que no son aplicables a A.

Los dominios relaciones ofrecen, además, la propiedad de "strong typing", lo que significa que cuando un usuario escribe una expresión, el sistema chequea que los operandos de cada operador sean los permitidos por el dominio.

En SQL, sin embargo, el propósito de los dominios es permitir la especificación de un tipo de dato, tal como CHAR(5). Los tipos de datos se definen sólo una vez, permitiendo de este modo que dicha especificación sea usada por columnas pertenecientes a muchas tablas.

Las diferencias más importantes entre los dominios SQL y dominios relacionales son:

- Los dominios SQL son especificaciones sintácticas; no son tipos de datos definidos por el usuario;
- Los dominios SQL no siempre se usan, ya que las columnas de las tablas pueden definirse directamente en términos de tipos de datos soportados por el sistema;
- Los dominios SQL no soportan definiciones de dominios en términos de otros dominios, ya que se definen siempre en términos de tipos de datos del sistema (no hay herencia);

- Los dominios SQL no permiten chequeos, ya que los únicos requerimientos para hacer comparaciones es que los comparandos sean del mismo tipo de datos;
- SQL no permite que los usuarios definan operaciones que se apliquen a los dominios;
- SQL no hace una clara distinción entre un dominio SQL y la representación de ese dominio en términos de tipos de datos soportados por el sistema.

En resumen, los dominios SQL no soportan el concepto fundamental de dominio relacional: el dominio de valores verdaderos definidos por el usuario.

La sintaxis de definición de dominios en SQL es la siguiente:

```
CREATE DOMAIN domain-data-type
```

```
[ default-definition]
```

```
[ dom-constraint-deflist] ;
```

Donde:

- *domain-data-type* es el tipo de dato SQL (escalar y definido por el sistema) para el dominio;
- *default-definition* especifica un valor por defecto que se aplica a todas las columnas que están definidas sobre el dominio. y no permite la definición de un valor por defecto explícito perteneciente a los propios del dominio;
- *dom-constraint-deflist* especifica un conjunto de restricciones de integridad que se aplican a todas las columnas definidas sobre ese dominio. Las reglas de integridad sólo permiten enumeraciones de valores.

Relaciones y Tablas Base

Las propiedades de las relaciones en el MR son:

- No existen tuplas repetidas, dado que una relación es un conjunto matemático (conjunto de tuplas), y por definición éstos no incluyen elementos repetidos;
- Las tuplas no están ordenadas (de arriba a abajo), dado que el cuerpo de una relación es un conjunto matemático, y por definición éstos no son ordenados. Esto significa que no existe en el MR los conceptos de direccionamiento por dirección ni adyacencia;
- Los atributos no están ordenados (de izquierda a derecha), dado que la cabecera de una relación se define también como un conjunto (de pares atributo- dominio).

Las tablas de SQL no son relaciones, ya que no cumplen con las propiedades recién mencionadas. La forma en que se representa una relación en SQL es mediante un tabla base, las cuales tienen las siguientes características anómalas:

- Las tablas base de SQL permiten filas repetidas, a no ser que se defina un mecanismo que lo evite;
- Las tablas SQL tienen un orden de columnas, de izquierda a derecha, y de filas, de arriba a abajo.

Claves Candidatas

Las claves candidatas se declaran en SQL como:

UNIQUE (*column-commalist*)

ó

PRIMARY KEY (*column-commalist*)

Donde:

- *column-commalist* no debe ser vacío.

El MR asegura que toda relación tiene al menos una clave candidata. La importancia de éstas es que constituyen el mecanismo de direccionamiento a nivel de tuplas en el sistema relacional. Anteriormente mencionamos que las tablas SQL pueden contener tuplas repetidas, lo que contradice el concepto de clave. Aún más, aquellas tablas que contengan tuplas repetidas presentarán un comportamiento extraño y errado en muchas circunstancias.

Otra consecuencia de lo anterior es que si un sistema no maneja correctamente las claves primarias, presentará problemas en la actualización de vistas.

Reglas de integridad

El término integridad en las bases de datos se refiere a asegurar que los datos sean válidos. La especificación de reglas de integridad (llamadas también reglas del negocio) es relativamente compleja, y actualmente no es posible definir las como elementos de la estructura de la base de datos. Estas reglas se definen actualmente mediante procedimientos almacenados y/o triggers (programación, lo cual no es parte del estándar SQL).

Es deseable que los SGBD almacenen las reglas de integridad en el catálogo, de forma que el subsistema de integridad esté constantemente monitoreando las transacciones de que realizan los usuarios, asegurando que las reglas se cumplan.

Es necesario, entonces especificar las reglas de integridad, mediante un comando SQL, tal como CREATE INTEGRITY RULE. Este comando hipotético de SQL representa la estructura de una regla de integridad, consistente del nombre de la regla, la restricción (que especifica la evaluación de una expresión como verdadera o falsa) y la respuesta a la violación de la regla (que define qué acciones se tomarán cuando la regla no se cumpla).

Álgebra Relacional y SQL

La propiedad fundamental del álgebra relacional es la de cierre. Esto significa que el resultado de cualquier expresión algebraica es una relación.

SQL maneja un operador llamado GROUP BY (agrupar por). El resultado de utilizar la cláusula GROUP BY, en alguna consulta, no es otra tabla, sino que un conjunto de tablas, no cumpliéndose el principio de clausura.

Ortogonalidad y SQL

Un lenguaje es ortogonal cuando en todo lugar en que se espera un valor de cierto dominio, se puede incluir una expresión cualquiera que retorne un valor de ese dominio.

En particular, las funciones de agregados de SQL carecen de ortogonalidad, como por ejemplo realizar una consulta que implique sumar ciertos valores numéricos de un cierto atributo (con la función SUM) y

luego consultar cuáles de aquellos valores (ya sumados) cumplen una cierta condición, por ejemplo mayores que un cierto número.

Facilidades de Implementación de BD Relacionales con SQL

Para ofrecer funcionalidades que permitan a los diseñadores de sistemas de bases de datos, los SGBDR ofrecen extensiones del SQL, basadas en triggers, procedimientos almacenados y cursores.

Además el SQL con su característica de ser un lenguaje dual, le entrega a los usuarios finales la posibilidad de que sean ellos mismos los que realicen consultas a la base de datos.

Estas características se describen a continuación.

Dualidad del Lenguaje

Los comandos del lenguaje SQL pueden ser ejecutados interactivamente o bien formando parte de un programa de aplicación. En este último caso, las sentencias de SQL están insertas dentro del código fuente, mezcladas con las sentencias del lenguaje anfitrión.

Esta característica de SQL se conoce con el nombre de principio de modo dual.

Existen diferencias entre las sentencias SQL declaradas interactivamente y su contraparte embebida. Además, algunas sentencias de SQL embebido no pueden usarse interactivamente y viceversa.

Para realizar eficientemente programas de aplicación, los SGBDR comerciales ofrecen una extensión de SQL, que consiste en agregar sentencias procedurales tradicionales, tales como IF-THEN-ELSE, FOR, WHILE, etc.

Triggers

Los triggers, no están explícitamente incluidos en el SQL2, a pesar de ser muy populares en los SGBD comerciales.

SQL2 soporta la validación a través de la cláusula CHECK CONSTRAINT, dentro del comando CREATE TABLE, la cual especifica una expresión lógica puede ser evaluada como *verdadero* o *falso*.

De esta manera el SGBD evalúa automáticamente dicha cláusula cada vez que se ejecuta un comando SQL de UPDATE, DELETE e INSERT.

Los triggers, además de cumplir con la funcionalidad anterior, pueden causar una acción en la base de datos, tal como agregar un fila o cambiar un dato en otra tabla.

Procedimientos Almacenados

Los procedimientos almacenados son un conjunto de transacciones SQL, a las que se les asigna un nombre, son compiladas y almacenadas en un servidor de base de datos. Después de que se define un procedimiento almacenado en la base de datos, éste puede ser llamado desde un programa de aplicación, mediante su nombre.

Esta funcionalidad está presente en la mayoría de los SGBDR comerciales, dentro de lo que se conoce como API (Application Program Interface).

Cursores

Los cursores soportan el desarrollo de programas de base de datos, permitiendo al usuario moverse libremente a través de un conjunto de resultados de consultas. Este tipo de interfaz de usuario es muy valorada en aplicaciones de soporte de decisiones y en aplicaciones de bases de datos basadas en consultas.

Esta extensión de SQL, es entregada por muchos SGBDR comerciales, proporcionando funcionalidad a los programadores.

Conclusiones

Como ya se mencionó en el capítulo V.V.3, los aspectos del MR no soportados por SQL, constituyen el gran problema de este lenguaje. Principalmente el concepto de dominio relacional difiere del de dominio SQL, lo que acarrea problemas al implementar restricciones sobre los valores de los atributos.

Desde otro punto de vista, SQL es un lenguaje declarativo fácil de entender y de aprender, lo cual explica en parte la aceptación que ha tenido como lenguaje de SGBDR. Debido a lo anterior, el lenguaje SQL posee muchas palabras superfluas, que podrían omitirse sin cambiar el significado de las sentencias, pero que lo aclaran a los ojos del programador.

SQL cumple con todos los requerimientos de los SGBDR. Sin embargo, como no permite representar el MR de forma completa, los SGBDR que utilizan SQL tampoco lo hacen. Como resultado, se tienen muchos SGBDR comercialmente difundidos que no soportan completamente el Modelo Relacional.

Los SGBDR implementan en la mayoría de los casos algún superconjunto de SQL, incluyendo una o más de las extensiones a SQL mencionadas anteriormente. Esto se debe a la necesidad que ha surgido por realizar aplicaciones cliente / servidor, con procesos centralizados en la parte servidora, que en este caso es el SGBDR.

La estandarización de los lenguajes permite que los productos desarrollados puedan perdurar en el tiempo, protegiendo la inversión de desarrolladores y usuarios. SQL2 es un estándar que ya tiene 5 años en vigencia y que está siendo revisado, a la luz de las nuevas necesidades que han surgido en estos años. Es probable entonces, que dentro de poco surja un nuevo estándar que soporte de manera más robusta el Modelo Relacional e introduzca nuevos conceptos asociados a otros modelos de datos, necesarios para algunas aplicaciones.

Bibliografía

C. J. Date: "An Introduction To Database Systems", Addison- Wesley, Sexta Edición, 1996

C. J. Date: "A Guide To The SQL Standard", Addison- Wesley, Segunda Edición, 1989

T. W. Pratt, M. V. Zelkowitz: "Programming Languages", Prentice- Hall, Tercera Edición, 1996

A. de Miguel, M. Piottini: "Concepción y Diseño de bases de Datos: del modelo E/R al modelo Relacional", Addison- Wesley, 1993

J. R. Graff, P. N. Weinberg: "LAN Times: A Guide To SQL", McGraw- Hill, 1994

Abreviaciones Usadas en este Documento:

SQL: Structured Query Language;

SGBD: Sistema de Gestión de Bases de Datos;

BD: Base de Datos;

SGBDR: Sistema de Gestión de Bases de Datos Relacionales;

BDR: Base de Datos Relacional;

MR: Modelo Relacional;

LDD: Lenguaje de Definición de Datos;

LMD: Lenguaje de Manipulación de Datos;