Diseño de Bases de datos II

1. Fundamentos SQL

Una base de datos es una colección estructurada de datos. Puede ser cualquier cosa, desde una simple lista de compras hasta una galería de imágenes o la gran cantidad de información en una red corporativa. Bajo la estructura de las bases de datos se encuentra el modelo de datos: una colección de herramientas conceptuales para describir los datos, sus relaciones, su semántica y las restricciones de consistencia. Los modelos de datos ofrecen un modo de describir el diseño de las bases de datos en los niveles físico, lógico y de vistas

El modelo relacional usa una colección de tablas para representar tanto los datos como sus relaciones. Cada tabla tiene varias columnas, y cada columna tiene un nombre único. Cada tabla contiene registros de un tipo dado. Cada tipo de registro define un número fijo de campos, o atributos. Las columnas de la tabla se corresponden con los atributos del tipo de registro. El modelo de datos relacionales es el modelo de datos más ampliamente usado, y una gran mayoría de sistemas de bases de datos actuales se basan en el modelo relacional.

Para agregar, acceder y procesar datos almacenados en una base de datos de computadora, se necesita un sistema de administración de base de datos (como MySQL Server). El DBMS es un conjunto de programas que se encargan de controlar el almacenamiento, organización y despliegue de los datos. Así como la creación y manejo de todos los accesos a las bases de datos.

Se debe acreditar a IBM no solo por inventar la base de datos relacional, sino también por desarrollar el lenguaje que todavía se usa para interactuar con dichas bases de datos, SQL (Structured Query Language, lenguaje estructurado de consultas). Las operaciones de SQL reciben el nombre de sentencias y están formadas por diferentes partes que denominamos cláusulas. Una sentencia SQL es un programa de computadora o instrucción que consiste en identificadores, parámetros, variables, nombres, tipos de datos y palabras reservadas de SQL.

SQL unifica tareas como crear, reemplazar, alterar y soltar objetos. Insertar, actualizar y eliminar filas de la tabla. Consultar datos, controlar el acceso a la base de datos y sus objetos y garantizar la coherencia e integridad de la base de datos. Estos grupos de tareas se realizan por medio de diferentes tipos de instrucciones que se clasifican de la siguiente manera: lenguaje de definición de datos (DDL: Data Definition Language), Lenguaje de manipulación de datos (DML: Data Manipulation Language) y lenguaje de consulta (SQL: Structured Query Language).

1.1 Sentencias DDL y DML

1.1.1 Data Definition Language

El lenguaje de definición de datos (DDL) es utilizado para describir todas las estructuras de información y los programas que se usan para construir, actualizar e introducir la información que contiene una base de datos. El DDL contiene un diccionario de datos que se utiliza para almacenar y crear las definiciones de los datos, incluyendo localización, forma en que se almacenan y algunas otras características. Este lenguaje de datos debe permitir describir los datos y las estructuras de los archivos del sistema, especificando la forma en que serán agrupados en registros o divididos en campos. Una vez que se ha elaborado la definición de la base de datos, el DBMS se encarga de construir y generar las estructuras de información de manera automática.

Sentencia	Descripción
CREATE	Se utiliza para modificar los atributos o configuraciones de los objetos creados en el esquema y el esquema.
ALTER	Se utiliza para modificar los atributos o configuraciones de los objetos creados en el esquema y el esquema.
DROP	Se utiliza para eliminar los objetos de un esquema, y el esquema como tal.
TRUNCATE	Eliminar todos los datos en objetos de esquema sin eliminar la estructura de estos objetos

1.1.2 Data Manipulation Language

Las declaraciones del lenguaje de manipulación de datos (DML) consultan o manipulan datos en objetos de esquema existentes. Mientras que las declaraciones DDL le permiten cambiar la estructura de la base de datos, las declaraciones DML le permiten consultar o cambiar el contenido. Por ejemplo, ALTER TABLE cambia la estructura de una tabla, mientras que INSERT agrega una o más filas.

Sentencia	Descripción
INSERT INTO	Se utiliza para hacer inserciones de datos en las tablas.
UPDATE	Se utiliza para modificar valores en uno o varios registros de una tabla.
DELETE FROM	Es la instrucción que se utiliza para borrar registros de una tabla.
SELECT FROM	Es la sentencia para consultar datos de las tablas y otros objetos de un esquema.

1.1.3 Consultas SQL

El lenguaje de consulta (SQL) es empleado por el usuario para extraer información de la base de datos. Una consulta (Query) es una operación que recupera datos de una tabla o vista. SELECT es la única instrucción SQL que se puede usar para consultar datos. El conjunto de datos recuperados de la ejecución de una instrucción SELECT se conoce como conjunto de resultados (Result set).

SELECT se usa para recuperar filas seleccionadas de una o más tablas, y puede incluir sentencias UNION y subconsultas. La estructura básica de una expresión SQL consta de tres cláusulas: select, from y where. Sin embargo, pueden existir otras instrucciones complementarias.

El resultado de una consulta SELECT nos devuelve una tabla lógica. Es decir, los resultados son una relación de datos, que tiene filas/registros, con una serie de campos/columnas. Igual que cualquier tabla de la base de datos. Sin embargo esta tabla está en memoria mientras la utilicemos, y luego se descarta. Cada vez que ejecutamos la consulta se vuelve a calcular el resultado.

Sentencia	Descripción
SELECT	Especifica qué columnas se deben mostrar en el resultado. La proyección produce un subconjunto de las columnas en la tabla. Cada select_expr indica una columna que desea recuperar. Debe haber al menos uno select_expr. Esta cláusula es obligatoria.
FROM	Especifica las tablas o vistas desde las que se deben recuperar los datos. Esta Cláusula es obligatoria.
WHERE	Especifica una condición para filtrar filas, produciendo un subconjunto de las filas en la tabla. Una condición especifica una combinación de una o más expresiones y operadores lógicos (booleanos) y devuelve un valor de VERDADERO, FALSO o DESCONOCIDO. Esta sentencia no es obligatoria.
ORDER BY	Especifica el orden en que se deben mostrar las filas.

1.1.4 Subconsultas

Una subconsulta es una instrucción SELECT anidada dentro de otra instrucción SQL. Las subconsultas son útiles cuando se deben ejecutar varias consultas para resolver un único problema. Cada parte de consulta de una instrucción se denomina bloque de consulta.

Decimos que la subconsulta está anidada dentro de la consulta externa, y de hecho es posible anidar subconsultas dentro de otras subconsultas, a una profundidad considerable. Una subconsulta siempre debe aparecer entre paréntesis. La subconsulta entre paréntesis es el bloque de consulta interno.

```
Select campo1, campo2 from (

  Select * from (

      Select campo1, campo2,
      campo3
      From nombre_tabla
      Where condicion

  )

)
```

Las principales ventajas de las subconsultas son:

- Permiten consultas que están estructuradas para que sea posible aislar cada parte de una instrucción.
- Proporcionan formas alternativas de realizar operaciones que de otro modo requerirían uniones y uniones complejas.
- Muchas personas encuentran subconsultas más legibles que las uniones complejas. De hecho, fue la innovación de las subconsultas lo que dio a la gente la idea original de llamar al sql temprano "Lenguaje de consulta estructurado".

Una subconsulta puede devolver un escalar (un valor único), una sola fila, una sola columna o una tabla (una o varias filas de una o varias columnas). Se denominan subconsultas escalares, de columna, de fila y de tabla. Las subconsultas que devuelven un tipo determinado de resultado a menudo se pueden usar solo en determinados contextos.

Subconsulta como un operando escalar

Una subconsulta escalar es un operando simple y puede usarlo casi en cualquier lugar donde un valor de columna o literal sea legal, y puede esperar que tenga esas características que tienen todos los operandos: un tipo de datos, una longitud, una indicación de que puede ser NULL, etc.

```
SELECT (SELECT s2 FROM t1);
SELECT (SELECT s1 FROM t2) FROM t1;
```

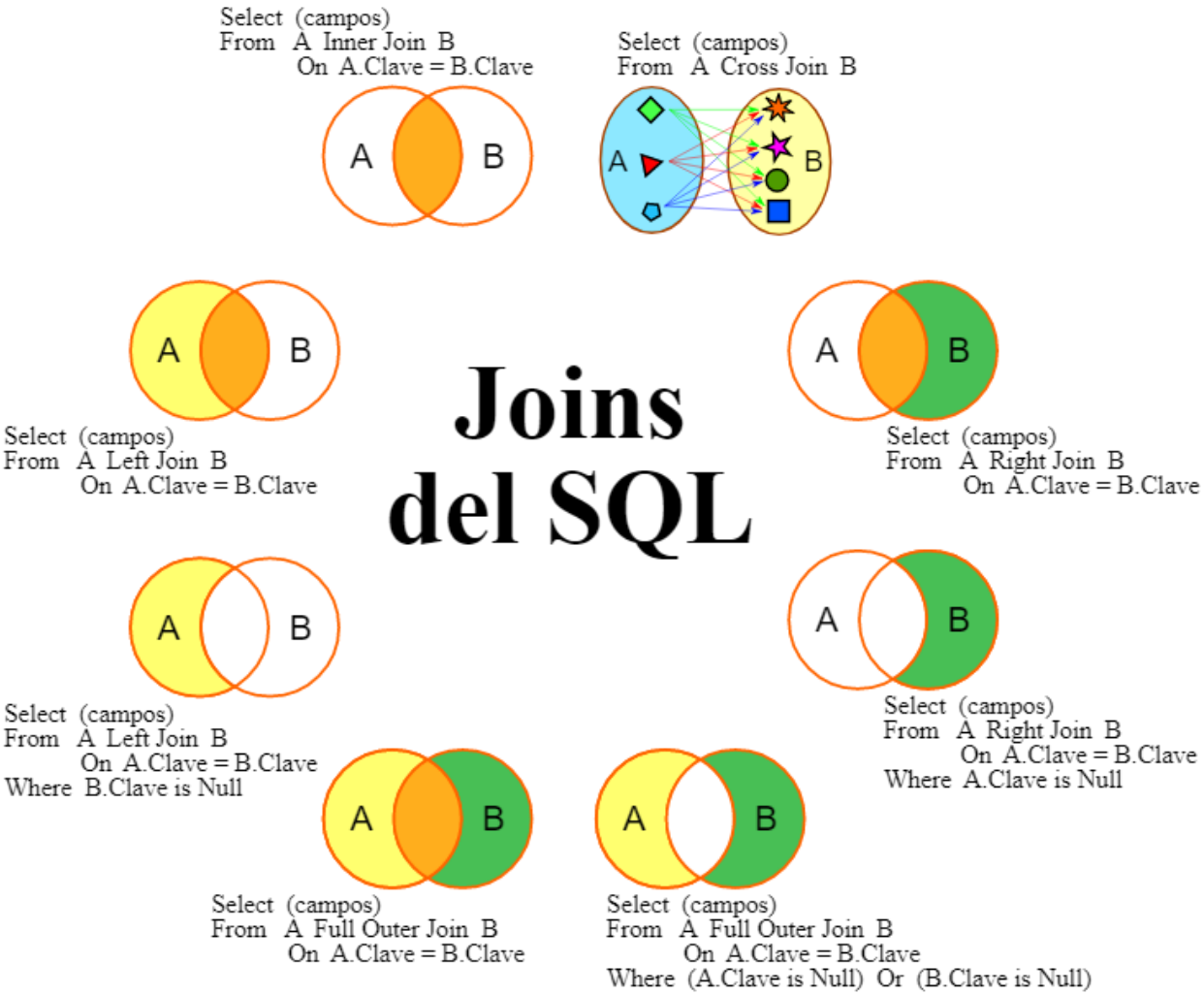
Una subconsulta escalar puede formar parte de una expresión, pero recuerde los paréntesis, incluso si la subconsulta es un operando que proporciona un argumento para una función.

```
SELECT UPPER((SELECT s1 FROM t1)) FROM t2;
```

1.2 Operaciones con conjuntos (JOINS)

En bases de datos las operaciones binarias de algebra relacional son denominadas JOIN. Un JOIN es una consulta que combina filas de dos o más tablas, vistas.

La Cláusula UNION combina el resultado de varias instrucciones SELECT en un solo conjunto de resultados. Los nombres de las columnas del conjunto resultado se toman de los nombres de columna de la primera instrucción SELECT. Las columnas seleccionadas enumeradas en las posiciones correspondientes de cada instrucción SELECT deben tener los mismos datos tipo.



Una INNER JOIN (unión interna) es una unión de dos o más tablas que devuelve solo filas que satisfacen la condición de unión. Es decir, únicamente aquellos registros cuyos campos evaluados en la condición de unión se encuentren en ambas tablas. Es importante tener en cuenta que el uso del INNER JOIN puede ocasionar pérdida de datos de los conjuntos de resultados relacionados.

Con la cláusula LEFT JOIN se hace una combinación externa en la que obtenemos todos las tuplas del primer conjunto y aquellos que son coincidentes en el segundo conjunto de resultados. De este modo no tenemos pérdida de información en ese primer conjunto. El resultado recuperado mostrará campos con valores nulos para aquellas tuplas que no hayan coincidido en el segundo conjunto.

La cláusula RIGHT JOIN es una combinación externa, en la que se obtienen las tuplas del primer conjunto que son coincidentes con las tuplas del segundo conjunto, conservando aquellas tuplas del segundo conjunto que no están en el primero. De esta forma se evita la pérdida de tuplas en el segundo conjunto, mostrando valores nulos para aquellas tuplas que no hayan coincidido con el primer conjunto.

1.3 Funciones agregadas (Group by y Having)

Las funciones agregadas devuelven una única fila de resultados basada en grupos de filas en lugar de en filas individuales. La agregación es fundamental para el almacenamiento de datos. Las funciones agregadas también se llaman funciones de agrupación dado que realiza cálculos con grupos de datos. Van acompañadas por la cláusula de agrupación GROUP BY. A menos que se indique lo contrario, las funciones de grupo omiten los valores NULL.

La cláusula GROUP BY agrupa un conjunto de filas en un conjunto de filas de resumen por valores de columnas o expresiones. La cláusula GROUP BY devuelve una fila para cada grupo. En otras palabras, reduce el número de filas en el conjunto de resultados. Si utiliza una función de grupo en una instrucción que no contiene ninguna cláusula GROUP BY, equivale a agrupar en todas las filas. Para eliminar valores duplicados, utilice la cláusula DISTINCT.

La claùsula WHERE se utiliza para filtrar filas en un conjuntos de resultados de una consulta. La cláusula HAVING se utiliza en la instrucción SELECT para especificar condiciones de filtro para un grupo de filas o agregados. Si se omite la cláusula GROUP BY, la cláusula HAVING se comporta como la cláusula WHERE. Tenga en cuenta que la cláusula HAVING aplica una condición de filtro a cada grupo de filas, mientras que la cláusula WHERE aplica la condición de filtro a cada fila individual.

Función	Descripción
AVG()	Retorna promedio de un conjunto de valores en una fila.
COUNT()	Devuelve un recuento del número de filas devueltas
MAX()	Retorna el valor máximo de un conjunto de valores de una fila.
MIN()	Retorna el valor mínimo de un conjunto de valores de una fila.
SUM()	Retorna la suma de un conjunto de valores en una fila.

1.4 Transacciones y variables de usuario

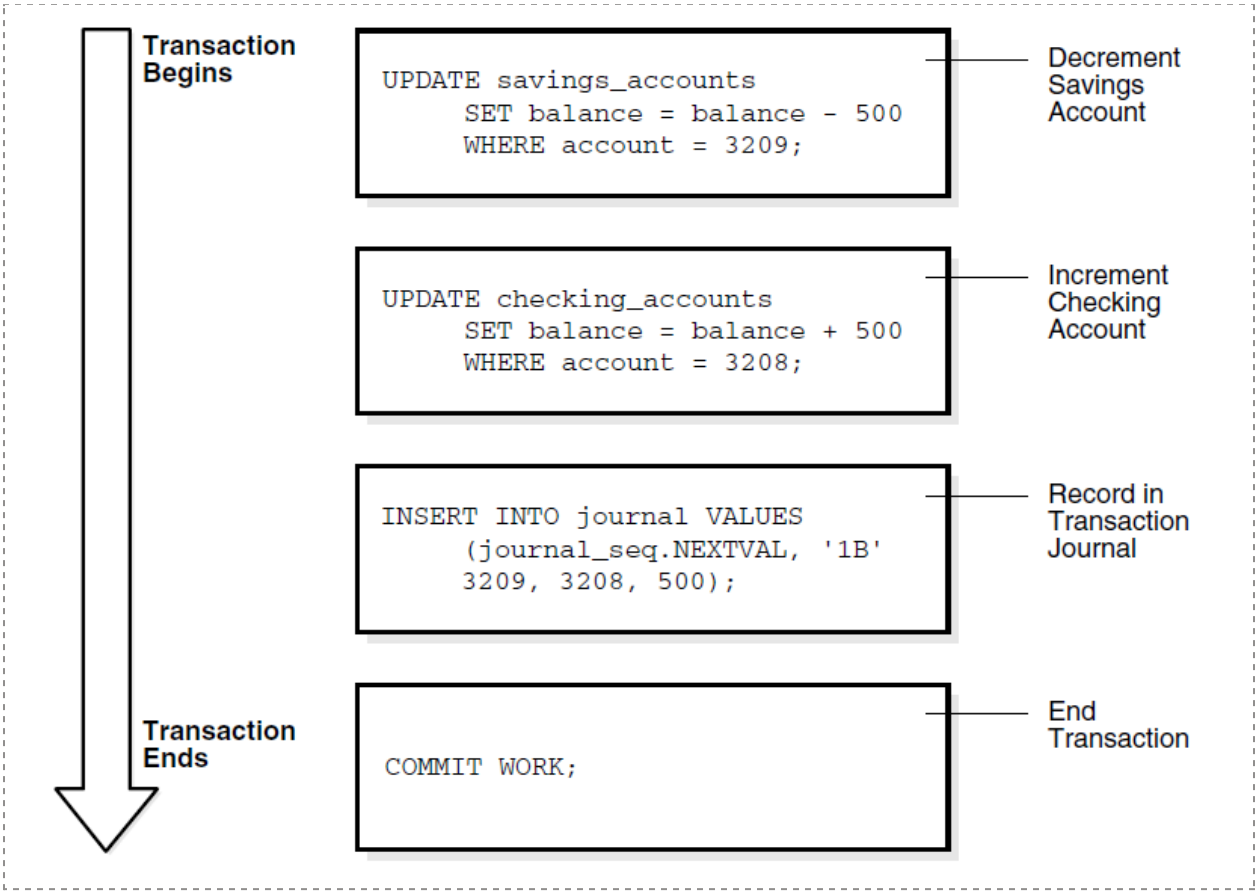
1.4.1 Transacciones

Una colección de sentencias DML que forma una unidad lógica de trabajo se denomina transacción. Una transacción agrupa las instrucciones SQL para que estén confirmadas, lo que significa que se aplican a la base de datos, o todas se revierten, lo que significa que se deshacen de la base de datos.

Por ejemplo, una transacción para transferir dinero podría involucrar tres operaciones discretas: disminuir el saldo de la cuenta de ahorros, aumentar el saldo de la cuenta corriente y registrar la transferencia en una tabla de historial de cuenta. A diferencia de las declaraciones DDL, las declaraciones DML no comprometen implícitamente la transacción actual.

Para ilustrar el concepto de una transacción, considere una base de datos bancarios. Cuando un cliente transfiere dinero de una cuenta de ahorros a una cuenta corriente, la transacción debe consistir en tres operaciones separadas:

- Disminuir la cuenta de ahorro
- Incrementar la cuenta corriente
- Registrar la transacción en el diario de transacciones

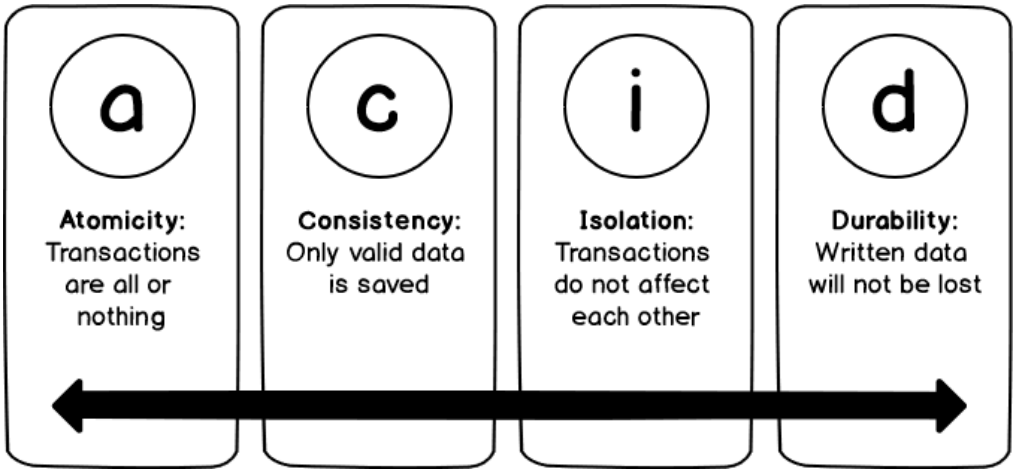


El primer extracto resta $ 500 de la cuenta de ahorros 3209. El segundo extracto agrega $ 500 a la cuenta corriente 3208. El tercer extracto inserta un registro de la transferencia en la tabla del diario. La declaración final confirma la transacción.

Si las tres declaraciones SQL mantienen las cuentas en el equilibrio adecuado, los efectos de la transacción se pueden aplicar a la base de datos. Sin embargo, si un problema como fondos insuficientes, número de cuenta inválido o una falla de hardware impide que uno o dos de los estados de cuenta de la transacción se completen, entonces la base de datos debe revertir toda la transacción para que el saldo de todas las cuentas sea correcto.

Propiedades de una transacción ACID

Un acrónimo de atomicidad, consistencia, aislamiento y durabilidad. Estas propiedades son todas deseables en un sistema de base de datos, y todos están estrechamente vinculados a la noción de una transacción. Las características transaccionales de InnoDB adherirse a los principios de ACID.



- **Atomicidad**: Se realizan todas las tareas de una transacción o ninguna de ellas. No hay transacciones parciales. Por ejemplo, si una transacción comienza a actualizar 100 filas, pero el sistema falla después de 20 actualizaciones, la base de datos revierte los cambios a estas 20 filas.
- **Consistencia**: La transacción lleva la base de datos de un estado consistente a otro estado consistente. Por ejemplo, en una transacción bancaria que debita una cuenta de ahorros y acredita una cuenta corriente, una falla no debe causar que la base de datos acredite solo una cuenta, lo que conduciría a datos inconsistentes.
- **Aislamiento** (Isolation): El efecto de una transacción no es visible para otras transacciones hasta que se confirma la transacción. Por ejemplo, un usuario que actualiza la tabla hr.employees no ve los cambios no confirmados en los empleados realizados simultáneamente por otro usuario. Por lo tanto, a los usuarios les parece que las transacciones se ejecutan en serie.

- **Durabilidad**: Los cambios realizados por transacciones comprometidas son permanentes. Una vez que se completa una transacción, la base de datos asegura a través de sus mecanismos de recuperación que no se pierden los cambios de la transacción.

Control de transacciones

Las declaraciones de control de transacciones gestionan los cambios realizados por las declaraciones DML y agrupan las declaraciones DML en transacciones. Estas declaraciones le permiten:

Sentencia	Descripción
COMMIT	Confirma la transacción actual, haciendo que sus cambios sean permanentes.
ROLLBACK	Revierte la transacción actual, cancelando sus cambios.
START TRANSACTION	Marca el inicio de una transacción
SET AUTOCOMMIT	Deshabilita o habilita el modo de confirmación automática predeterminado para la sesión actual.

Por defecto, MySQL se ejecuta con el modo de confirmación automática (autocommit) habilitado. Esto significa que, de lo contrario, dentro de un transacción, cada declaración es atómica, como si estuviera rodeada por un START TRANSACTION and COMMIT. No se puede utilizar ROLLBACK para deshacer el efecto; sin embargo, si se produce un error durante la ejecución de la declaración, la declaración es revertida.

Algunas declaraciones no pueden revertirse. En general, estos incluyen el lenguaje de definición de datos (DDL) declaraciones, como las que crean o eliminan bases de datos, las que crean, eliminan o alteran tablas o almacenan rutinas.

1.4.2 Variables y sentencia SET

La instrucción SET tiene varias formas de uso, dependiendo de la función para la cual va ser utilizado en el servidor de MySQL. Puede ser utilizado para la administración de la base de datos, también para el manejo de transacciones y para asignación de valores a variables.

Con MySQL se puede almacenar un valor en una variable definida por el usuario en una sentencia y consultarlo más adelante en otra sentencia. Las variables de usuario se escriben como @var_name, donde el nombre de la variable consta de caracteres alfanuméricos. El nombre de una variable de usuario puede contener otros caracteres si lo cita como una cadena o un identificador (ejemplo, @'my-var').

Las variables definidas por el usuario son específicas de la sesión. Una variable de usuario definida por un cliente no puede ser vista o utilizada por otros clientes. Los nombres de las variables de usuario no distinguen entre mayúsculas y minúsculas. Los nombres tienen una longitud máxima de 64 caracteres.

A las variables de usuario se les puede asignar un valor de un conjunto limitado de tipos de datos: integer, decimal, floating-point, binary or nonbinary string, también se les puede asignar un valor NULL. Si se selecciona el valor de una variable de usuario en un conjunto de resultados, se devuelve al cliente como una cadena. Si se refiere a una variable que no se ha inicializado, tiene un valor NULL y un tipo string.

Sintaxis

```
SET @var_name = expr [, @var_name = expr] ...
```

1.5 Vistas

Una vista es una representación lógica de una o más tablas. En esencia, una vista es una consulta almacenada. Una vista deriva sus datos de las tablas en las que se basa, llamadas tablas base. Las tablas base pueden ser tablas u otras vistas. Todas las operaciones realizadas en una vista en realidad afectan las tablas base. Puede usar vistas en la mayoría de los lugares donde se usan tablas.

Las vistas le permiten adaptar la presentación de datos a diferentes tipos de usuarios. Las vistas se usan a menudo para:

- Proporcione un nivel adicional de seguridad de tabla al restringir el acceso a un conjunto predeterminado de filas o columnas de una tabla
- Ocultar la complejidad de los datos.
- Presente los datos en una perspectiva diferente de la de la tabla base
- Aísle las aplicaciones de los cambios en las definiciones de las tablas base.

Para crear una vista utilizamos sentencias DDL. Así, para crear una vista utilizamos la sentencia CREATE.
La instrucción CREATE VIEW crea una nueva vista o reemplaza una vista existente si la cláusula OR REPLACE es dado. Si la vista no existe, CREATE O REPLACE VIEW es lo mismo que CREATE VIEW. Si la vista existe, CREATE O REPLACE VIEW lo reemplaza.

```
CREATE [OR REPLACE]  VIEW view_name [(column_list)]  AS
select_statement
```

```
CREATE VIEW staff AS
SELECT employee_id, last_name, job_id, manager_id, department_id FROM
employees;
```

La definición de vista está "congelada" en el momento de la creación y no se ve afectada por cambios posteriores en las definiciones de las tablas subyacentes. Por ejemplo, si una vista se define como SELECT * en una tabla, las nuevas columnas agregadas a la tabla más adelante no se convierten en parte de la vista, y las columnas eliminadas de la tabla darán lugar a un error al seleccionar desde la vista.

Dentro de una base de datos, las tablas y vistas base comparten el mismo espacio de nombres, por lo que una tabla base y una vista no pueden tener el mismo nombre. Las columnas recuperadas por la instrucción SELECT pueden ser simples referencias a columnas de tablas o expresiones que usan funciones, valores constantes, operadores, etc.

Se puede crear una vista a partir de muchos tipos de sentencias SELECT. Puede referirse a tablas base u otras vistas. Eso puede usar combinaciones, UNION y subconsultas.

1.6 Índices

Un índice es una estructura opcional, asociada con una tabla o grupo de tablas, que a veces puede acelerar el acceso a los datos. Los índices son uno de los muchos medios para reducir la E / S de disco. Los índices se utilizan para buscar filas con valores de columna específicos rápidamente. Sin un índice, MySQL debe comenzar con la primera fila y luego leer toda la tabla para encontrar las filas relevantes. Cuanto más grande es la tabla, más alto es el costo. Si la tabla tiene un índice para las columnas en cuestión, MySQL puede determinar rápidamente la posición a buscar en el medio del archivo de datos sin tener que mirar todos los datos. Esto es mucho mas rápido que leer cada fila secuencialmente.

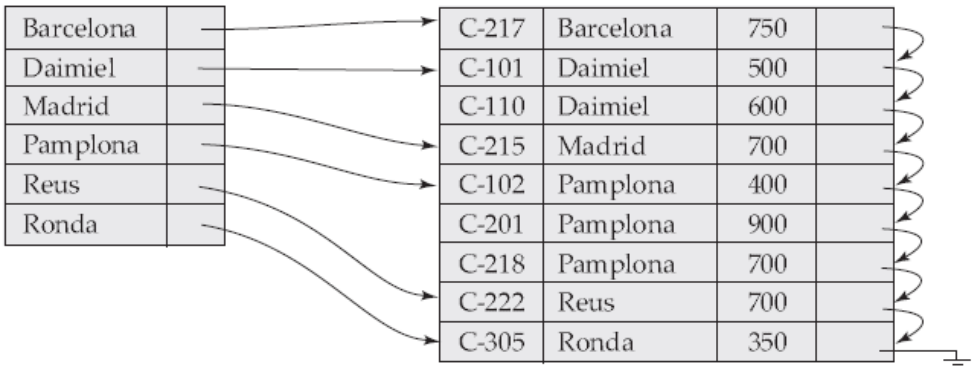


Figura 12.2 Índice denso.

Un índice para un archivo del sistema funciona como el índice de un libro. Si se va a buscar un tema (especificado por una palabra o una frase) se puede buscar en el índice al final del libro, encontrar las páginas en las que aparece y después leerlas para encontrar la información buscada. Las palabras del índice están ordenadas alfabéticamente, lo cual facilita la búsqueda. Además, el índice es mucho más pequeño que el libro, con lo que se reduce aún más el esfuerzo necesario para encontrar las palabras en cuestión.

Los índices de los sistemas de bases de datos juegan el mismo papel que los índices de los libros en las bibliotecas. Por ejemplo, para recuperar un registro cuenta dado su número de cuenta, el sistema de bases de datos buscaría en un índice para encontrar el bloque de disco en que se localice el registro correspondiente, y entonces extraería ese bloque de disco para obtener el registro cuenta.

En general, considere crear un índice en una columna en cualquiera de las siguientes situaciones:

- Las columnas indexadas se consultan con frecuencia y devuelven un pequeño porcentaje del número total de filas en la tabla.
- Se colocará una restricción de clave única en la tabla y desea especificar manualmente el índice y todas las opciones de índice.