

4-1-2024

## Capítulo 14 Aplicaciones de bases de datos

Shambhavi Roy

Clinton Daniel  
*University of South Florida*

Manish Agrawal  
*University of South Florida*

Pablo Brescia  
*University of South Florida*

Clara Olivia Ocampo

*See next page for additional authors*

Follow this and additional works at: [https://digitalcommons.usf.edu/dit\\_tb\\_spa](https://digitalcommons.usf.edu/dit_tb_spa)

---

### Scholar Commons Citation

Roy, Shambhavi; Daniel, Clinton; Agrawal, Manish; Brescia, Pablo; Ocampo, Clara Olivia; and Labrador, Sonia, "Capítulo 14 Aplicaciones de bases de datos" (2024). *FUNDAMENTALS OF INFORMATION TECHNOLOGY: Textbook – Spanish*. 14.  
[https://digitalcommons.usf.edu/dit\\_tb\\_spa/14](https://digitalcommons.usf.edu/dit_tb_spa/14)

This Book Chapter is brought to you for free and open access by the The Modernization of Digital Information Technology at Digital Commons @ University of South Florida. It has been accepted for inclusion in FUNDAMENTALS OF INFORMATION TECHNOLOGY: Textbook – Spanish by an authorized administrator of Digital Commons @ University of South Florida. For more information, please contact [digitalcommons@usf.edu](mailto:digitalcommons@usf.edu).

---

**Authors**

Shambhavi Roy, Clinton Daniel, Manish Agrawal, Pablo Brescia, Clara Olivia Ocampo, and Sonia Labrador

## CONTENIDOS DEL CAPÍTULO

|  |            |
|--|------------|
| <b>Panorama</b>  | <b>328</b> |
| Evolución de las tecnologías de bases de datos                                     | 328        |
| Por qué estudiar las bases de datos  | 331        |
| <b>Bases de datos relacionales</b>   | <b>331</b> |
| <b>Introducción a SQL</b>  | <b>333</b> |
| <b>Tipos de comandos SQL</b>   | <b>333</b> |
| <b>Consultar una base de datos mediante comandos DML</b>                           | <b>334</b> |
| Consultar desde varias tablas  | 335        |
| Actualizar datos   | 336        |
| Eliminar datos   | 337        |
| Opciones de consulta avanzada  | 338        |
| Ordenar  | 338        |
| Agrupar  | 339        |
| Campos calculados  | 340        |
| <b>Crear y administrar tablas usando comandos DDL</b>                              | <b>341</b> |
| Insertar datos   | 343        |
| Importar datos   | 344        |
| <b>Transacciones usando comandos TCL</b>   | <b>344</b> |
| <b>Control de privilegios de las bases de datos usando comandos DCL</b>            | <b>346</b> |
| <b>Conclusión</b>  | <b>347</b> |
| <b>Términos y definiciones del capítulo</b>  | <b>348</b> |
| <b>Caso del capítulo: base de datos de seguimiento de calificaciones de Dionne</b> | <b>349</b> |

*Una vez centralizados e integrados los datos empresariales, el valor de la base de datos es mayor que la suma de las partes preexistentes.*

—Larry Ellison, cofundador de Oracle.

## Panorama

Las bases de datos son colecciones estructuradas de datos que las computadoras pueden buscar y manipular eficientemente.<sup>223</sup> Las bases de datos son los almacenes de información subyacente que impulsan nuestro mundo moderno fomentado por la tecnología. Estas **bases de datos** son como bóvedas digitales donde se guarda información de forma segura y los usuarios autorizados pueden leerla o actualizarla según sea necesario. Si bien es posible que aún no sepas qué son, afectan tu vida a diario. Tus expedientes estudiantiles, tu historial de puntajes en videojuegos en línea, tus registros de préstamos en la biblioteca están todos almacenados en bases de datos en algún lugar. Con toda probabilidad, tu información personal, incluida la dirección de tu casa, tu número de teléfono y tus calificaciones escolares, se almacenan en múltiples bases de datos.

Las aplicaciones de bases de datos son programas de software que ayudan a los usuarios a interactuar con la información almacenada en las bases de datos. Se dividen en categorías como Gestión de relaciones con el cliente (CRM, siglas en inglés de *Customer Relationship Management*), Planificación de recursos empresariales (ERP, siglas en inglés de *Enterprise Resource Planning*), Sistema de gestión de aprendizaje (LMS, siglas en inglés de *Learning Management System*), etc. Por ejemplo, las aplicaciones de Planificación de recursos empresariales ayudan a los minoristas a actualizar la información de su inventario cuando se compran artículos en la tienda. Los Sistemas de gestión del aprendizaje asisten a las instituciones escolares a monitorear el desempeño de todos los estudiantes de la escuela. Cuando compras algo en línea, las aplicaciones de Gestión de inventario actualizan la información de las existencias en el sitio y crean la etiqueta de envío para los artículos que se entregarán en tu domicilio. Hemos analizado estas aplicaciones a lo largo de este libro.

En este capítulo, nos centramos en las bases de datos e introducimos las tecnologías y conceptos utilizados para crear bases de datos, de las que existen varios tipos, incluidas las orientadas a objetos, las jerárquicas y las relacionales; la mayoría de las aplicaciones modernas utilizan las **bases de datos relacionales**. Por lo tanto, en este capítulo hacemos énfasis en la tecnología de esta última.

## Evolución de las tecnologías de bases de datos<sup>224</sup>

Las bases de datos comerciales tienen sus orígenes en el programa lunar Apolo. El cohete *Saturn V* utilizado para llevar a Neil Armstrong a la luna tenía más de tres millones de piezas.<sup>225</sup> *Rockwell*

---

223 Esta definición está adaptada del sitio de Oracle en <https://www.oracle.com/database/what-is-database/> (consultado en abril del 2024).

224 Esta sección se basa en gran medida en la publicación del blog de Sinclair Target en <https://twobithistory.org/2017/12/29/codd-relational-model.html>. Todas las publicaciones de este blog son relevantes para las Tecnologías de la Información Digital (DIT por sus siglas en inglés); se puede ver en <https://twobithistory.org/> (consultado en abril del 2024).

225 Una descripción general del *Saturn V* está en <https://airandspace.si.edu/exhibitions/space-race/online/sec300/sec384.htm> (consultado en abril del 2024).

*International*, la empresa que construyó el cohete, le pidió a IBM construir un sistema informático para rastrear el estado de todas estas piezas. En 1968 un equipo de unos 25 ingenieros de IBM, Rockwell y Caterpillar, liderados por el ingeniero de IBM Vern Watts, crearon IMS (siglas en inglés de *Information Management System*), la primera base de datos comercial del mundo. Desde entonces, IMS se ha convertido en la columna vertebral de la banca, hoteles, y aplicaciones de líneas aéreas en todo el mundo. Cada vez que se utiliza una tarjeta de crédito o se hace una reservación aérea, es probable que la transacción involucre a IMS en algún momento.<sup>226</sup>

IMS se considera una base de datos jerárquica, muy eficiente, pero también muy inflexible. Las búsquedas en bases de datos jerárquicas requieren atravesar el almacén de datos.<sup>227</sup> Están diseñadas para responder a consultas específicas y pueden ser muy lentas a la hora de responder otras consultas.<sup>228</sup> Las bases de datos relacionales mejoran esta limitación.<sup>229</sup>

El modelo relacional para almacenar información fue presentado por Edgar F. Codd de IBM en 1970.<sup>230</sup> Codd utilizó conceptos de la teoría de conjuntos para organizar los datos. Usó el término “relación” para referirse a una **tabla** en la base de datos basada en la teoría de conjuntos, y esta referencia a las relaciones le da nombre a la tecnología.

Tras la publicación de su histórico artículo en 1970 que presentó el modelo relacional, Codd<sup>231</sup> describió cómo se podía optimizar el almacenamiento de datos en el modelo relacional (llamado

---

226 Esto proviene de la página de IBM sobre IMS en <https://www.ibm.com/ibm/history/ibm100/us/en/icons/ibmims/> (consultado en junio del 2023).

227 Charles Bachman, uno de los líderes de las bases de datos jerárquicas, ofrece una visión general en su conferencia del premio Turing de 1973, “Programmer as Navigator”, en <http://people.csail.mit.edu/tanford/6830papers/bachman-programmer-as-navigator.pdf>. Wikipedia tiene una página sobre Bachman en [https://en.wikipedia.org/wiki/Charles\\_Bachman](https://en.wikipedia.org/wiki/Charles_Bachman) (consultado en abril del 2024). En español encontramos información en [https://es.wikipedia.org/wiki/Charles\\_Bachman](https://es.wikipedia.org/wiki/Charles_Bachman) (consultado en abril del 2024).

228 Una descripción general de las bases de datos jerárquicas se encuentra en: <https://dataintegrationinfo.com/hierarchical-vs-relational-database/> y [https://en.wikipedia.org/wiki/Hierarchical\\_database\\_model](https://en.wikipedia.org/wiki/Hierarchical_database_model) (consultado en abril del 2024). En español encontramos información en [https://es.wikipedia.org/wiki/Base\\_de\\_datos\\_jer%C3%A1rquica](https://es.wikipedia.org/wiki/Base_de_datos_jer%C3%A1rquica) (consultado en abril del 2024).

229 La página de Wikipedia sobre bases de datos relacionales proporciona una buena descripción general de la tecnología en [https://en.wikipedia.org/wiki/Relational\\_database](https://en.wikipedia.org/wiki/Relational_database) (consultado en abril del 2024). En español encontramos información en [https://es.wikipedia.org/wiki/Base\\_de\\_datos\\_relacional](https://es.wikipedia.org/wiki/Base_de_datos_relacional) (consultado en abril del 2024).

230 Este artículo se considera un hito en la comercialización de tecnologías informáticas. Se puede descargar en <https://www.seas.upenn.edu/~zives/03f/cis550/codd.pdf> (consultado en abril del 2024).

231 E. F. Codd, Wikipedia [https://en.wikipedia.org/wiki/Edgar\\_F.\\_Codd](https://en.wikipedia.org/wiki/Edgar_F._Codd) (consultado en abril del 2024). En español encontramos información en [https://es.wikipedia.org/wiki/Edgar\\_Frank\\_Codd](https://es.wikipedia.org/wiki/Edgar_Frank_Codd) (consultado en abril del 2024).

normalización).<sup>232</sup> También demostró que con un pequeño conjunto de operaciones se puede extraer cualquier información de la base de datos.<sup>233;234</sup> Esto ahora se llama **Lenguaje de consulta estructurado (SQL, siglas en inglés de *Structured Query Language*)**.<sup>235</sup>

Si bien el modelo relacional era conceptualmente sólido, el éxito de IMS impidió que las inversiones comerciales crearan con anterioridad una tecnología funcional basada en el modelo relacional. IBM tenía un pequeño equipo trabajando en un proyecto llamado *System R* para construir una base de datos relacional, y el profesor Michael Stonebraker<sup>236</sup> de la UC Berkeley obtuvo financiación de la NSF, así como del Ejército, la Armada y la Fuerza Aérea para iniciar un proyecto llamado INGRES para crear una base de datos utilizando el modelo relacional.<sup>237</sup> La viabilidad de INGRES<sup>238</sup> y la amplia distribución de su código abierto facilitaron la comercialización del modelo relacional. Hoy en día, el modelo relacional se enseña como el estándar en todos los programas académicos relevantes.

Algunas bases de datos relacionales populares incluyen:

- Oracle, ofrecida por la corporación Oracle, es probablemente el sistema de base de datos relacional más popular del mundo.
- MySQL es un sistema de gestión de bases de datos relacionales de código abierto ampliamente utilizado. MySQL también es propiedad de Oracle desde la compra de Sun Microsystems.

---

232 E. F. Codd, Normalized Database Structure: A Brief Tutorial, disponible en <https://dl.acm.org/doi/pdf/10.1145/1734714.1734716> (consultado en abril del 2024). *Actas del ACM SIGFIDET* de 1971 (ahora SIGMOD). Taller sobre descripción, acceso y control de datos, noviembre de 1971, pp. 1 a 17.

233 E. F. Codd, A Database Sublanguage Based on Relational Calculus <https://dl.acm.org/doi/pdf/10.1145/1734714.1734718> (consultado en abril del 2024). *Actas del ACM SIGFIDET* de 1971 (ahora SIGMOD). Taller sobre descripción, acceso y control de datos, noviembre de 1971, pp. 35-68.

234 E. F. Codd, "Relational Completeness of Data Base Sublanguages", Computer Science Research Report, 1972, disponible en <http://www.geology.cz/personal/j/jan.sedlacek/codd2.pdf> (consultado en junio del 2023).

235 Para los interesados en obtener más detalles sobre el modelo relacional de Codd, su colega C. J. Date publicó un análisis del modelo, "Codd's First Relational Papers: A Critical Analysis", disponible en <https://www.dcs.warwick.ac.uk/~hugh/TTM/CJD-on-EFC%27s-First-Two-Papers.pdf> (consultado en abril del 2024).

236 Se puede ver un resumen del trabajo de Stonebraker en la presentación para recibir la medalla John von Neumann <https://par.nsf.gov/servlets/purl/10219488> (consultado en abril del 2024).

237 Este párrafo se basa en la excelente historia de "The Rise of Relational Databases", capítulo 6 de Financing a Revolution: Government Support for Computing Research, disponible en <http://worrydream.com/refs/National%20Research%20Council%20-%20Funding%20a%20Revolution.pdf> pp. 159–168 (consultado en abril del 2024).

238 Para obtener un informe sobre INGRES, consulte Michael Stonebraker, Eugene Wong, Peter Kreps y Gerald Held, "The Design and Implementation of INGRES Tandem Computers, Inc.", disponible en <https://dl.acm.org/doi/pdf/10.1145/320473.320476>. Entra en los detalles del funcionamiento de INGRES. (consultado en abril del 2024).

- Microsoft SQL Server es proporcionado por Microsoft y es común en los ecosistemas basados en Microsoft.
- PostgreSQL es otro popular sistema de gestión de bases de datos relacionales de código abierto conocido por sus funciones avanzadas y su confiabilidad. Es el sucesor del proyecto INGRES.
- IBM DB2 lo proporciona IBM y se utiliza ampliamente en aplicaciones empresariales.
- SQLite es un sistema de gestión de bases de datos relacionales ligero empleado a menudo en sistemas integrados y aplicaciones móviles. Por ejemplo, tu navegador utiliza SQLite para administrar marcadores.

## Por qué estudiar las bases de datos

En comparación con las tecnologías tratadas en otros capítulos de este libro, las tecnologías de bases de datos tienen una característica única: son una tecnología central para el mundo empresarial; hay toda una clase de profesionales altamente remunerados (administradores de bases de datos) que mantienen las bases de datos. En este libro presentamos la tecnología de bases de datos relacionales debido a su importancia y también porque los conceptos básicos de las bases de datos relacionales deberían ser muy interesantes para los estudiantes de secundaria. Si encuentras atractivo el contenido de este capítulo, puedes aprender más en línea a tu propio ritmo en sitios como Khan Academy,<sup>239</sup> W3Schools,<sup>240</sup> Coursera,<sup>241</sup> EdX,<sup>242</sup>, etc.

## Bases de datos relacionales

Las bases de datos relacionales organizan los datos como tablas vinculadas entre sí. Cada fila de una tabla representa un elemento de la base de datos. Una única base de datos puede tener un sinnúmero de tablas, y cada tabla representa todos los elementos almacenados de un tipo.

Cada tabla (o relación) en una base de datos relacional representa una entidad o un concepto. Cada tabla consta de filas, también conocidas como registros o tuplas, y columnas, también conocidas como atributos o campos. Cada fila de una tabla representa una instancia específica de la entidad y cada columna representa una característica o datos específicos relacionados con esa entidad.

En una base de datos relacional, las tablas se relacionan entre sí mediante **claves**. Según la teoría de conjuntos, cada fila de la base de datos es única. Las claves identifican de forma única los datos de cada fila y también permiten unir datos de diferentes tablas. Repasemos estas ideas con un ejemplo de estudiantes matriculados en cursos.

---

239 “Intro to SQL: Querying and Managing Data”, <https://www.khanacademy.org/computing/computer-programming/sql> (consultado en abril del 2024).

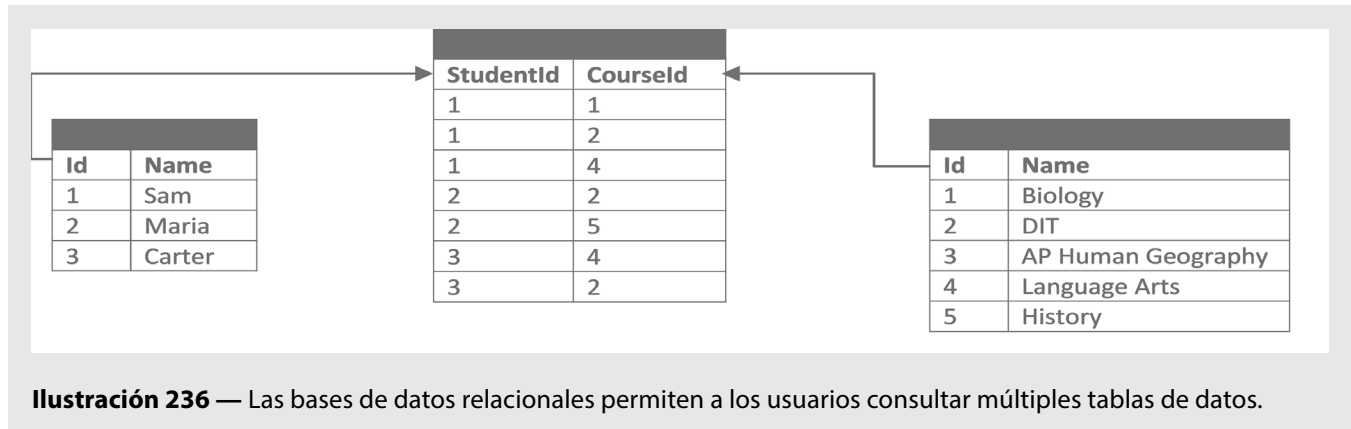
240 “SQL Tutorial”, <https://www.w3schools.com/sql/> (consultado en abril del 2024).

241 “SQL for Data Science”, <https://www.coursera.org/learn/sql-for-data-science> (consultado en abril del 2024).

242 “Databases: Relational Databases and SQL”, <https://www.edx.org/course/databases-5-sql> (consultado en abril del 2024).



La [Ilustración 236](#) muestra un modelo de base de datos relacional que almacena información sobre los cursos tomados por los estudiantes. Hay una tabla llamada Students, que almacena información sobre los estudiantes como los nombres (también se podría agregar otra información). Otra tabla separada llamada Courses tiene información sobre los cursos, como el título del curso. Finalmente, una tabla exclusiva del modelo de base de datos relacional, llamada StudentCourses, identifica todos los cursos tomados por cada estudiante.



Las tablas para estudiantes y cursos son intuitivas, pero la forma en que las tablas están vinculadas entre sí para enumerar todos los cursos tomados por todos los estudiantes le da poder al modelo de base de datos relacional. Primero, se debe tener en cuenta que las tablas Students y Courses tienen una columna denominada Id, que asigna una etiqueta única (llamada identificador o Id para abreviar) a cada fila. En la tabla StudentCourses, tanto los estudiantes como los cursos están representados por los respectivos identificadores únicos (Ids). Podemos ver en la tabla StudentCourses que solo María (studentId 2) tomó Historia (courseId 5), pero Sam, María y Carter tomaron DIT (course Id 2).

En el modelo relacional, una columna (o combinación de columnas) garantizada como única en una tabla puede servir como clave principal de la tabla. Por ejemplo, tu número de licencia es una clave principal que te identifica de manera única en la base de datos del Departamento de Vehículos Motorizados (DMV, por sus siglas en inglés). Tu identificación de estudiante es una clave principal que te reconoce como un ente singular en la base de datos de tu escuela. Por lo general, los identificadores únicos sirven como columna de clave principal de las tablas. Por ejemplo, en la tabla Students, la columna "Id" es la clave principal de esa tabla. De modo similar, en la tabla Courses, la columna "Id" es la clave principal. Normalmente, el software de base de datos genera un valor único para la clave principal de una fila.

Cuando se define una clave principal en una tabla, el sistema de gestión de bases de datos relacionales garantiza que no haya dos filas en la tabla que tengan los mismos valores de clave principal. Además, las columnas de clave principal no pueden tener valores nulos (vacíos) ya que un valor nulo no puede identificar de forma única una fila.

La organización de datos como se muestra en la [Ilustración 236](#) permite el almacenamiento y la recuperación eficiente de información mediante diversas operaciones relacionales y comandos del Lenguaje de consulta estructurado (SQL). Por ejemplo, si compras productos de Amazon con regularidad, pueden mantener un registro de todas las compras y devoluciones realizadas recientemente usando tu clave principal y sacando todos los registros de compras asociados con esa clave. Cada compra es un registro de la base de datos, al igual que cada devolución. Cada registro de compra tiene muchos detalles, incluido el nombre del artículo, el nombre del vendedor, el momento



de la transacción, el monto y la ubicación de la transacción. Una vez que tengas una manera de ingresar tus datos en una base de datos, podrás rastrear, actualizar, eliminar e informar sobre esos datos fácilmente. En la base de datos de Amazon la tabla de transacciones está compuesta por una colección de compras y devoluciones de todos sus clientes. ¿Te imaginas lo grande que debe ser esa tabla, y cuántos bytes de datos debe tener?

En el modelo de base de datos relacional de la [Ilustración 236](#), exclusivamente se necesita interactuar con la tabla `Students` si sólo se desea tener la información básica relacionada con los estudiantes. De la misma manera, si se quiere obtener información sobre los cursos ofrecidos, únicamente se requiere [consultar](#) la tabla `Courses`. Sin embargo, para tener la información del curso de cada estudiante, se usa la tabla `StudentCourses`, que une todas las tablas de `Students` y `Courses` por `Ids`. La recuperación de registros por `Ids` es mucho más rápida que por nombres, lo que hace que las bases de datos relacionales bien diseñadas sean muy eficientes.

En las bases de datos relacionales, el proceso de mantener la información distribuida en varias tablas se denomina normalización y garantiza que ninguna tabla tenga un tamaño demasiado inmanejable. No es sorprendente que este modelo se utilice ampliamente en todas las industrias por su flexibilidad, escalabilidad y solidez.

## Introducción a SQL

Almacenar la información en una base de datos es sólo el primer paso. Una vez que la información está en una base de datos, se debe interactuar con el Sistema de administración de la base de datos (DBMS, siglas en inglés de *Database Management System*), el software que la administra, para ver o cambiar la información. El Lenguaje de consulta estructurado (SQL) es un lenguaje de programación que se utiliza para interactuar con las bases de datos relacionales.

Los comandos SQL se utilizan para crear nuevas bases de datos y tablas. Además, hay comandos SQL para insertar, actualizar y eliminar información en tablas existentes. SQL también se utiliza para buscar y recuperar datos específicos mediante consultas, que son sentencias que detallan los criterios para seleccionar datos. La mayoría de los programas universitarios en informática tienen un curso sobre bases de datos, que aborda todos los detalles de SQL. En este capítulo, sólo proporcionamos una descripción general de SQL.<sup>243</sup>

## Tipos de comandos SQL

Hay cuatro tipos principales de comandos SQL que debes conocer:

[Comandos del lenguaje de manipulación de datos \(DML, siglas en inglés de \*Data Manipulation Language\*\)](#) se utilizan para manipular la información almacenada en las bases de datos. Los comandos DML más utilizados son `SELECT` (seleccionar), `INSERT` (insertar), `UPDATE` (actualizar) y `DELETE` (eliminar). `SELECT` se utiliza para recuperar la información; `INSERT` para agregar nuevos datos; `UPDATE` es para modificar los datos existentes; y `DELETE` se emplea para eliminar información.

[Comandos del lenguaje de definición de datos \(DDL, siglas en inglés de \*Data Definition Language\*\)](#)

---

<sup>243</sup> Se puede practicar SQL en varios sitios en línea. Un sitio popular para aprender códigos es [jdoodle](https://www.jdoodle.com/execute-sql-online/), <https://www.jdoodle.com/execute-sql-online/> (consultado en abril del 2024).

se usa para definir la estructura (esquema) de las bases de datos. Cuando se crea una nueva base de datos, se necesitan tablas, especificar la relación entre ellas y agregar restricciones a los datos para asegurar la integridad de la información. CREATE (crear), ALTER (alterar) y DROP (retirar) son los comandos DDL más comunes y necesarios. CREATE se utiliza para crear nuevas tablas y otros objetos; ALTER es para modificar la estructura de tablas existentes y DROP se emplea para eliminar tablas y otros objetos.

Comandos del lenguaje de control de transacciones (TCL siglas en inglés de *Transaction Control Language*) sirven como guardianes de los cambios. COMMIT (comprometer), ROLLBACK (revertir) y SAVEPOINT (punto de retorno) son los comandos TCL más comunes. Con COMMIT los cambios serán permanentes; ROLLBACK deshace los cambios realizados anteriormente y SAVEPOINT crea un marcador en una transacción al que se puede retroceder más tarde.

Comandos del lenguaje de control de datos (DCL, siglas en inglés de *Data Control Language*) se utilizan para controlar quién puede acceder a la base de datos. GRANT (conceder) y REVOKE (revocar) son los comandos más comunes. GRANT se emplea para otorgar permiso a los usuarios para acceder y modificar la base de datos y REVOKE es para revocar permisos.

## Consultar una base de datos mediante comandos DML

Si eres un programador de bases de datos, utilizarás los comandos DML con mucha frecuencia para cambiar información agregada con anterioridad, eliminar registros existentes o insertar datos nuevos. Afortunadamente, SQL es fácil de aprender porque los comandos son las mismas palabras del inglés para dar órdenes para seleccionar, insertar, actualizar y eliminar. SELECT (seleccionar) permite encontrar datos; INSERT (insertar) se utiliza para agregar datos; UPDATE (actualizar) se usa para cambiar datos agregados previamente y DELETE (eliminar) se emplea para eliminar registros.

Consulta de datos: la primera operación que probablemente desees realizar es ver la información almacenada en tu base de datos. Aquí hay un ejemplo de una sentencia SELECT básica:

```
SELECT *  
FROM (de) Students (estudiantes)  
WHERE (donde) Id= 56745
```

Esta sentencia le dice al RDBS que obtenga todos los detalles (\*) de la tabla Students sobre el estudiante cuyo Id es 56745. Si se desea obtener sólo algunas columnas específicas sobre el estudiante, se puede modificar la consulta para poner los nombres de las columnas. Por ejemplo:

```
SELECT StudentName (nombre del estudiante), StudentAge (edad del estudiante),  
StudentAddress (dirección del estudiante)  
FROM Students  
WHERE Id= 56745;
```

El comando WHERE es poderoso ya que ofrece muchas posibilidades y se utiliza para seleccionar con exactitud lo que se desea encontrar.

Supongamos que se quiere obtener el dato de todos los estudiantes que son de California o Florida; en ese caso, se cambia la cláusula WHERE de la siguiente manera:

```
SELECT *
FROM Students
WHERE State (estado) = 'California' OR State = 'Florida';
```

Esta sentencia sólo mostrará aquellos estudiantes que tengan los valores California o Florida en la columna "State". Al igual que las operaciones igual (=) y OR (o), también puedes emplear > (mayor que), < (menor que), BETWEEN (entre), LIKE (como) y IN (en).

```
SELECT *
FROM Students
WHERE State = 'California' AND GPA >= 3.5
```

Este comando sólo muestra las filas donde la columna "State" es igual a "California" y la columna "GPA" es mayor o igual a "3,5".

También se puede utilizar los operadores "AND" (y) y "OR" (o) en la misma consulta e incluso agrupar condiciones utilizando paréntesis. Por ejemplo:

```
SELECT *
FROM Students
WHERE (State = 'California' AND GPA >= 3.5)
OR (State = 'Florida' AND GPA >=3.5);
```

## Consultar desde varias tablas

En las bases de datos relacionales, la información se almacena en muchas tablas para eliminar la duplicación de datos. Sin embargo, con frecuencia se necesita unir tablas para obtener la información deseada. En SQL, JOIN (unir) es un comando especial que vincula tablas en función de columnas compartidas.

Consideremos dos tablas "Customers (clientes)" y "Orders (pedidos)" (Ilustración 237) para ver cómo funciona JOIN:

El esquema se refiere a la forma en que una base de datos está estructurada. Para relacionar los pedidos con los clientes los realizaron, la tabla Orders tiene el Id del cliente como columna customer id. Esto nos permite vincular las dos tablas (Customers y Orders). Por ejemplo, para obtener los pedidos realizados por los clientes, podemos vincular las dos tablas, como se muestra a continuación, y obtener todos los pedidos de todos los clientes:

Table 1: customers

| id | first_name | last_name |
|----|------------|-----------|
| 1  | John       | Smith     |
| 2  | Jane       | Doe       |
| 3  | Bob        | Johnson   |

Table 2: orders

| id | order_date | total | customer_id |
|----|------------|-------|-------------|
| 1  | 2022-01-01 | 10.00 | 1           |
| 2  | 2022-02-01 | 20.00 | 2           |
| 3  | 2022-03-01 | 30.00 | 1           |

**Ilustración 237** — Los datos se estructuran en tablas con filas y columnas, similares a los archivos de hojas de cálculo.

```
SELECT customers.first_name, customers.last_name, orders.order_date, orders.total
FROM customers, orders
WHERE customers.id = orders.customer_id;
```

Con esta consulta, se obtienen los nombres y apellidos de los clientes de la tabla Customers y order\_date (fecha del pedido) y el total de la tabla Order. Podemos unir las dos tablas porque la columna id de la tabla Customers y la columna customer\_id de la tabla Order contienen el mismo dato: el id de los clientes. El modelo relacional nos brinda este método conveniente para recuperar la información en las bases de datos. Las consultas SQL se leen como oraciones en inglés simple (Ilustración 238).

| first_name | last_name | order_date | total |
|------------|-----------|------------|-------|
| John       | Smith     | 2022-01-01 | 10.00 |
| Jane       | Doe       | 2022-02-01 | 20.00 |
| John       | Smith     | 2022-03-01 | 30.00 |

**Ilustración 238** — Se pueden unir tablas separadas utilizando columnas o “claves” coincidentes.

## Actualizar datos

Las bases de datos deben actualizarse con la información más reciente todo el tiempo. Por ejemplo, en el DMV los conductores van a registrar automóviles nuevos, actualizar sus placas o cambiar sus direcciones residenciales. Con la ayuda de SQL, se identifica el registro en la base de datos que necesita actualizarse y se establecen los nuevos valores.

El comando UPDATE (actualizar) permite cambiar los valores de una o más columnas en una o más filas de una tabla. La sintaxis básica del comando SQL UPDATE es la siguiente:

```
UPDATE table_name
SET column1 = new_value1, column2 = new_value2, ...
WHERE some_condition;
```

En esta sintaxis:

- table\_name (nombre de la tabla) es el nombre de la tabla que se desea actualizar.
- column1, column2, etc. son los nombres de las columnas que se quieren actualizar.
- new\_value1, new\_value2, etc. son los nuevos valores que se desean establecer para las columnas correspondientes.
- some\_condition es una condición que especifica las filas a actualizar. Sólo se actualizarán las filas que cumplan la condición.

A continuación, se muestra un ejemplo de cómo utilizar el comando UPDATE para cambiar la dirección de correo electrónico de un cliente:

```
UPDATE customers
SET email = 'newemail@example.com'
WHERE customer_id = 123
```

Este comando actualizará la dirección de correo electrónico del cliente con el Id 123 a “newemail@example.com”.

Se debe tener mucho cuidado al utilizar el comando UPDATE y no cambiar datos que no se desean alterar. Es importante siempre verificar con detenimiento la cláusula WHERE para asegurarse de que están especificadas correctamente las filas a actualizar. Por ejemplo, el siguiente comando actualizará las direcciones de correo electrónico de TODOS los clientes en la base de datos a “newemail@example.com”. Si la base de datos tiene información sobre un millón de usuarios, las direcciones de correo electrónico del millón de usuarios se actualizarán instantáneamente. A menos que todos sus clientes tengan la misma dirección de correo electrónico, nunca se debe ejecutar un comando como el siguiente:

```
Update customers
SET email = 'newemail@example.com';
```

## Eliminar datos

Si un estudiante abandona una clase, debe removerse de la tabla Class (clase). Si un cliente solicita no recibir anuncios publicitarios mensuales, se deberá eliminar al cliente de la tabla de Subscriptions (suscripciones).

La sentencia DELETE elimina una o más filas de una tabla según los criterios de selección de la cláusula WHERE. La sintaxis básica de la sentencia DELETE es la siguiente:

```
DELETE FROM table_name
WHERE condition;
```

En esta sintaxis:

- table\_name es el nombre de la tabla de la que se desea eliminar datos;
- condition (condición) especifica qué filas eliminar y sólo se descartan las filas que cumplan la condición.

A continuación, se muestra un ejemplo que muestra cómo utilizar la sentencia DELETE para eliminar un registro de cliente en particular:

```
DELETE FROM customers
WHERE customer_id = 123;
```

Este comando eliminará el registro de cliente con Id. 123 de la tabla clientes.

Hay que tener cuidado al emplear la sentencia DELETE para evitar eliminar datos que no se desean descartar. Es conveniente siempre tener una copia de seguridad de la información antes de realizar eliminaciones. Además, como se mencionó anteriormente para el comando UPDATE, se debe verificar minuciosamente la cláusula WHERE para asegurarse que se descartarán sólo los datos que se desean eliminar.

Si se olvida incluir una cláusula WHERE, como en el siguiente ejemplo, se puede terminar eliminando TODAS las filas de la tabla de los clientes:

```
Delete FROM customers;
```

## Actualizaciones y eliminaciones defensivas

Antes de ejecutar un comando DELETE, es mejor verificar la cláusula WHERE en la instrucción SELECT para confirmar que se está eliminando el conjunto correcto.

Por ejemplo, para DELETE todos los clientes que viven en un código postal en particular, primero se ejecuta el comando SELECT.

```
Select * from CUSTOMERS where zipcode= '95070';
```

Si la salida tiene todas las filas que se desea eliminar, se utiliza la misma cláusula WHERE en el comando de eliminación para descartar los registros.

```
Delete FROM customers WHERE zipcode= '95070';
```

## Opciones de consulta avanzada

Las bases de datos pueden crecer rápidamente y necesitar de informes, clasificación y agrupación especiales para que los datos almacenados tengan sentido. Si tienes una base de datos para llevar el control de los resultados de los exámenes de tu escuela, puede que la tabla "Students" tenga miles de hileras. Si asumes que cada estudiante toma 50 cursos a lo largo de sus años en la secundaria, la tabla "Students courses" sería 50 veces más grande que la de "Students". Ahora bien, si cada curso tiene 100 actividades para nota, ya te puedes imaginar lo grande que sería la tabla "Grading".

Para hacer informes con estos datos, es necesario saber cómo clasificar, agrupar y limitar la cantidad de resultados para obtener rápidamente la información que se necesita. Por ejemplo, puede ser que a tu maestro le interese identificar a dos estudiantes que no hayan entregado su proyecto o a los cuatro o cinco que sobresalieron en un examen. En estos casos no sería útil imprimir los 5 millones de filas de todos los trabajos asignados en la escuela. Es mucho más práctico usar las capacidades de la base de datos para enumerar solamente las cuatro o cinco filas con la información necesaria.

## Ordenar

El comando ORDER BY (ordenar por) clasifica el conjunto de resultados de forma ascendente o descendiente según una o más columnas en la sentencia SELECT. La sintaxis básica de la cláusula ORDER BY es como sigue:

```
SELECT column1, column2, ...
```

```
FROM table_name
```

```
ORDER BY column1 [ASC|DESC], column2 [ASC|DESC];
```

Se puede especificar la columna o columnas que se quieren clasificar, seguido por "ASC" para un orden ascendente o "DESC" para un orden descendiente. Se puede clasificar también usando múltiples columnas separadas por comas.

Veamos, por ejemplo, una tabla de empleados con los siguientes datos: id, nombre, edad, y salario. (Ilustración 239)



Para ordenar esta tabla por salario en orden descendiente, se puede usar la siguiente consulta SQL:

```
SELECT * FROM employees  
ORDER BY salary DESC;
```

El resultado será el siguiente: (Ilustración 240)

Para ordenar la misma tabla por edad y luego por salario, se puede intentar la siguiente consulta SQL:

```
SELECT * FROM employees  
ORDER BY age ASC, salary DESC;
```

El resultado se muestra en la Ilustración 241.

## Agrupar

El comando GROUP BY (agrupar por) logra un resumen de los datos. Si hay una base de datos de estudiantes, pueden agruparse por sus cursos para ver cómo les va en las diferentes asignaturas. O, también puede agruparse a los estudiantes por año (primero, segundo, tercero, cuarto) para ver el promedio según la promoción.

El comando GROUP BY en SQL se usa para agrupar filas que tienen el mismo valor en una o más columnas. Normalmente se usa con funciones agregadas como AVERAGE (promedio), MAX (máximo), MIN (mínimo) y COUNT (contar). El conjunto de resultados va a contener solo una fila por cada valor de la columna en la cláusula GROUP BY. Por ejemplo, si se usa el comando GROUP BY para el campo de departamento en una tabla que tiene el nombre del empleado, el salario, y el departamento, el conjunto de resultados tendrá una fila por cada departamento.

Otro ejemplo: para contar el número de empleados y el salario promedio en cada departamento, se pueden usar los comandos GROUP BY, COUNT, y AVG.

```
SELECT department, COUNT(*) as num_employees, AVG(salary) as avg_salary  
FROM employees  
GROUP BY department;
```

| id | name  | age | salary |
|----|-------|-----|--------|
| 1  | John  | 25  | 50000  |
| 2  | Jane  | 30  | 60000  |
| 3  | Bob   | 28  | 55000  |
| 4  | Alice | 35  | 65000  |

**Ilustración 239** — El comando ORDER BY les permite a los usuarios clasificar u ordenar eficientemente datos en una tabla.

| id | name  | age | salary |
|----|-------|-----|--------|
| 4  | Alice | 35  | 65000  |
| 2  | Jane  | 30  | 60000  |
| 3  | Bob   | 28  | 55000  |
| 1  | John  | 25  | 50000  |

**Ilustración 240** — Se puede ordenar una tabla según el nombre de la columna de manera ascendente o descendente. Esta tabla está ordenada por salario en orden ascendente.

| id | name  | age | salary |
|----|-------|-----|--------|
| 1  | John  | 25  | 50000  |
| 3  | Bob   | 28  | 55000  |
| 2  | Jane  | 30  | 60000  |
| 4  | Alice | 35  | 65000  |

**Ilustración 241** — El comando ORDER BY no se limita a una sola condición. El usuario puede combinar condiciones para refinar aún más la solicitud.



En este ejemplo, la consulta cuenta el número de empleados en cada departamento (COUNT(\*)) y da el salario promedio de esos empleados. En la tabla original, no teníamos el recuento de empleados o el salario promedio de los empleados. Por lo tanto, en la salida, tenemos dos nuevos campos que no existían antes. Se puede usar el operador "as" para darle nombres significativos a estos campos en la salida como se muestra en la [Ilustración 242](#).

La sintaxis general para usar el comando GROUP BY es como sigue:

```
SELECT column1, column2, ..., aggregate_function(column)
FROM table_name
GROUP BY column1, column2, ...;
```

En la sintaxis de arriba la columna 1 y la columna 2 son las columnas que se desean agrupar con GROUP BY. La función aggregate\_function(column) es la que se quiere aplicar a los datos agrupados, tales como COUNT, AVG, SUM, MAX, MIN, etc.

Se puede utilizar múltiples columnas en la cláusula GROUP BY si se desea agrupar datos por más de una columna.

Como puede apreciarse, el comando GROUP BY es una poderosa herramienta en SQL que permite resumir datos basado en criterios específicos, lo que facilita extraer información útil de grandes conjuntos de datos.

| department  | num_employees | avg_salary |
|-------------|---------------|------------|
| Sales       | 10            | 50000      |
| Marketing   | 8             | 45000      |
| Engineering | 15            | 70000      |
| Finance     | 5             | 55000      |

**Ilustración 242** — Hay una variedad de comandos, incluyendo GROUP BY, que le permite al usuario contar conjuntos de datos.

### \* in SQL

El símbolo "\*" tal como se usa en el ejemplo de GROUP BY anterior significa "todos". En el ejemplo, este símbolo le dice a la base de datos que cuente todas las filas en cada grupo.<sup>244</sup>

## Campos calculados

Hay ocasiones cuando no se quiere simplemente presentar los datos en la base de datos, sino también aplicar algunos cálculos antes de imprimir el resultado. Por ejemplo, si se mira una base de datos de estudiantes con sus calificaciones, quizás también sea bueno calcular e imprimir las calificaciones (A,B,C). O si hay una base de datos de empleados, quizás se desee calcular su bonificación como un porcentaje de su salario (qué porcentaje de su salario es la bonificación). SQL permite hacer estos cálculos fácilmente sin afectar los datos subyacentes.

Un campo calculado es una columna en un conjunto de resultados de una consulta que se crea aplicando una expresión o fórmula a una o más columnas en una tabla. La expresión puede ser una

244 Para información adicional, se puede consultar el sitio de Stackoverflow, <https://stackoverflow.com/questions/38662318/what-does-mean-in-sql> (consultado en abril del 2024).

combinación de operadores aritméticos, funciones y constantes, lo que es especialmente útil cuando se necesita realizar cálculos complejos en columnas de una tabla o crear métricas personalizadas que no están disponibles en el conjunto de datos original. Consideremos la siguiente consulta:

```
SELECT first_name, last_name, salary, salary * 0.1 AS bonus
FROM employees;
```

En este ejemplo, la consulta calcula una bonificación del 10% para todos los empleados.

El conjunto de resultados de esta consulta tiene cuatro columnas: `first_name`, `last_name`, `salary` y `bonus` (nombre, apellido, salario y bonificación). (Ilustración 243) La columna bonificación es un campo calculado que contiene la bonificación de los empleados basado en su salario. Como el campo no existe en la tabla original, se le puede dar un nombre significativo con la ayuda del operador "AS".

| first_name | last_name | salary | bonus   |
|------------|-----------|--------|---------|
| John       | Smith     | 50000  | 5000.00 |
| Jane       | Doe       | 60000  | 6000.00 |
| Bob        | Johnson   | 70000  | 7000.00 |
| Alice      | Williams  | 80000  | 8000.00 |

**Ilustración 243** — Las consultas SQL también pueden realizar cálculos y generar nuevos campos, lo cual es crucial en el análisis de datos.

## Crear y administrar tablas usando comandos DDL

Para crear una tabla en SQL, se puede usar la sentencia CREATE.

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
    ...
);
```

En el comando DDL anterior, *datatype* (tipo de datos) especifica el tipo de datos que se puede guardar en la columna. *Datatypes* comunes son VARCHAR para texto, INT para números enteros, DECIMAL para números decimales y DATE para fechas. Una característica interesante de las bases de datos relacionales es que la mayoría de los datos en las bases de datos son números o textos.

He aquí un ejemplo de una sentencia CREATE TABLE para crear una tabla Clientes con columnas para tres tipos de información (nombre, correo electrónico y alquiler) y una identificación:

```
CREATE TABLE boarders (
    id INT PRIMARY KEY,
    name VARCHAR(50),
    email VARCHAR(100),
    rent INT
);
```

Si se ejecuta este comando SQL, se va a crear una tabla llamada inquilinos con cuatro columnas: id, nombre, correo electrónico y alquiler. La columna del id. es un entero y se establece como la clave principal de la tabla.

Hay ocasiones cuando ya no se necesita una tabla, sea porque se quiere crear una nueva o porque se tienen los mismos datos en otra tabla. Con el comando DROP se elimina permanentemente una tabla.

```
DROP TABLE table_name;
```

Por ejemplo, para borrar la tabla de los inquilinos, ejecuta el siguiente comando DDL:

```
DROP TABLE boarders;
```

A veces se necesita añadir nuevas columnas, cambiar el nombre o el tipo de datos de una columna. Con el comando ALTER TABLE se puede añadir, remover y cambiar el nombre de una columna. Inclusive se puede cambiar el tipo o tamaño de los datos almacenados en las columnas. He aquí ejemplos para cada uno de estos comandos:

```
ALTER TABLE table_name ADD COLUMN column_name data_type;
```

```
ALTER TABLE table_name MODIFY COLUMN column_name new_data_type;
```

```
ALTER TABLE table_name
```

```
RENAME COLUMN old_column_name TO new_column_name;
```

```
ALTER TABLE table_name DROP COLUMN column_name;
```

El comando DROP COLUMN remueve cualquier columna de la tabla.

### **Almacenamiento de imágenes, documentos y otra información en bases de datos relacionales**

Las bases de datos relacionales normalmente almacenan tipos de datos simples como números y textos. Sin embargo, a menudo necesitamos almacenar información como imágenes y documentos. ¿Cómo se puede almacenar dicha información en bases de datos relacionales?

La mayoría de las bases de datos relacionales modernas tienen un tipo de datos llamado blob, (*binary large object*, objeto binario grande, blob, por sus siglas en inglés,) con el que se puede almacenar imágenes, documentos, vídeos y otra información. Sin embargo, una mejor alternativa es almacenar las imágenes, documentos, vídeos, etc., como archivos en el sistema de archivos y solo guardar el nombre y la ubicación del archivo en la base de datos. De este modo, la base de datos se mantiene compacta y se aceleran las búsquedas.

## Insertar datos

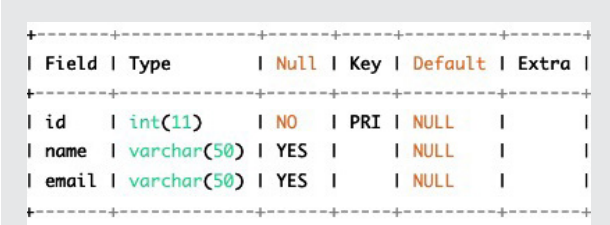
Para añadir datos a las tablas en una base de datos, se puede usar el comando `INSERT INTO` (insertar en).

El comando `INSERT INTO` espera que sepas los nombres de las columnas y los tipos de datos que las columnas pueden contener; por lo tanto, se necesita conocer la estructura de la tabla antes de insertar datos en ella. Para chequear la estructura de una tabla, usa el comando `DESCRIBE` (describir).

```
DESCRIBE Users;
```

Cuando se ejecuta el comando anterior, se ve la estructura de `Users`, un conjunto de resultados que enumera todas las columnas en `Users`, junto con los tipos de datos y cualquier restricción que se haya aplicado.

Como se puede apreciar en la [Ilustración 244](#), la tabla `Users` tiene tres columnas: `id.`, nombre, y correo electrónico. La columna de `id.` es un número entero con una longitud máxima de 11 dígitos y está marcado como la clave principal. El tipo de columna del nombre y del correo electrónico es `varchar`, con una longitud máxima de 50 caracteres. Además, están marcados como anulables (*nullable*), lo que significa que puede contener valores nulos. En una base de datos relacionales, un valor nulo significa que los datos son desconocidos o que faltan. Aunque en teoría la tabla se ve bien, un registro que solo tiene un `id.`, pero sin nombre o correo electrónico no va a tener sentido por lo que, idealmente, se debe cambiar la tabla para asegurarse de que el nombre o el correo electrónico contengan un valor —no puede ser nulo.



| Field | Type        | Null | Key | Default | Extra |
|-------|-------------|------|-----|---------|-------|
| id    | int(11)     | NO   | PRI | NULL    |       |
| name  | varchar(50) | YES  |     | NULL    |       |
| email | varchar(50) | YES  |     | NULL    |       |

**Ilustración 244** — Utilizar el comando `DESCRIBE` es útil para determinar la estructura de los datos.

Una vez que se sabe lo que se espera de una tabla, se puede usar la sentencia `INSERT INTO` para comenzar a añadir datos. He aquí un ejemplo:

```
INSERT INTO Users (id, name, email) VALUES (1, 'Cecilia Flores', 'cecilia.flores@example.com');
```

```
INSERT INTO Users (id, name, email) VALUES (2, 'Ratna Devi', 'ratna.devi@writers.com');
```

### Variables del comando `INSERT INTO`

Los ejemplos mostrados son las maneras típicas en que se usa el comando `INSERT`. Enumeramos tres nombres de columnas junto al nombre de la tabla (`Users (id, nombre, correo electrónico)`) y le dimos valores para insertarse para cada una de las columnas. Hay otro modo de usar el comando `INSERT`, omitiendo los nombres de las columnas. Estos nombres solo se pueden omitir si la base de datos a la que se pasan tiene las mismas columnas en el mismo orden que las columnas existentes en la tabla. Por esta razón, es más típico el uso mostrado en estos ejemplos.

## Importar datos

Hay ocasiones en que ya se tienen en un archivo los datos que se quieren insertar en una tabla, por ejemplo, al mover los datos de un archivo en Excel a una base de datos. El concepto general para importar datos es similar en todos los sistemas de bases de datos relacionales y utiliza algunos métodos comunes, incluyendo los siguientes:

- Ya hemos usado el comando `INSERT INTO` para añadir datos a una tabla en una base de datos. También se puede usar el comando `LOAD DATA INFILE` para insertar datos de un archivo en la tabla de una base de datos. Idealmente, el archivo debe ser solamente un archivo de texto con datos de entrada bien formateados para que se pueda asignar los datos a las columnas de su tabla de destino.
- Muchos sistemas de gestión de bases de datos ofrecen utilidades de línea de comandos para importar datos de diferentes fuentes. Por ejemplo, MySQL ofrece la utilidad `mysqlimport` que se puede usar para importar datos de archivos CSV o TSV.
- Algunos sistemas de gestión de bases de datos ofrecen herramientas GUI para la importación de datos. Por ejemplo, el Microsoft SQL Server Management Studio tiene un asistente de importación, Import Wizard, para importar datos con diferentes formatos de archivos como Excel, CSV o TSV.
- Se puede escribir un *script* (programa de software) en un lenguaje de programación como Python o Java para leer datos de un archivo e insertarlo en una tabla de una base de datos usando el comando `INSERT INTO`.

En general, estos son los pasos que siguen los programadores para importar datos a una base de datos SQL:

- Conectarse a la base de datos usando un cliente de base de datos o un lenguaje de programación.
- Crear una tabla o seleccionar una existente a la cual importar los datos.
- Preparar los datos limpiándolos y asegurándose de que estén en el formato correcto.
- Usar comandos SQL, utilidades de línea de comandos o lenguajes de programación para importar los datos a una tabla.
- Verificar que los datos importados estén correctos consultando la tabla.

## Transacciones usando comandos TCL

Como almacenes de información, las bases de datos tienen requisitos estrictos de precisión. Por lo tanto, si por cualquier motivo la actividad de una base de datos falla, normalmente se quiere restaurar los datos a su estado original. Veamos un ejemplo.

Supongamos que administras una base de datos de Expedia.com y que se necesita transferir algunos pasajeros a un nuevo vuelo porque el original no es conveniente. Como los pasajeros van a querer ser transferidos solo si hay asientos disponibles, hay que asegurarse de que ambas transacciones (reservar asientos en el nuevo vuelo y cancelar los del vuelo original) se realicen exitosamente. Si remueves pasajeros de su vuelo original, debes de poder asignarles asientos en el nuevo vuelo. Si tu

operación logra el primer paso pero fracasa en el segundo, vas a tener algunos pasajeros enojados y sin vuelo. Sin la garantía de que ambas transacciones se realicen como un conjunto o que, de lo contrario, no se hagan, las bases de datos no van a ser muy útiles.

De igual modo, las transacciones bancarias necesitan poder realizarse como un conjunto. Si se saca dinero de una cuenta, pero no se deposita en la cuenta destino (digamos que falla la conexión de internet), vas a terminar en una inconsistencia en lo que a la base de datos se refiere. Los desafortunados dueños de la cuenta se van a preguntar qué pasó con su dinero y el banco hasta quizás reciba algunas llamadas frenéticas.

Para evitar estos problemas, las bases de datos sirven para realizar transacciones. Puedes asegurarte de que las operaciones de la base de datos de una transacción se ejecuten como una sola unidad, o que sino no, no se realicen. Los comandos de Lenguaje de control transaccional, (siglas en inglés de Transaction Control Language) que incluyen BEGIN TRANSACTION (comenzar la transacción), SAVEPOINT, ROLLBACK y COMMIT son útiles para garantizar de que un grupo de transacciones se realicen exitosamente o fracasen, pero en conjunto.

La sentencia BEGIN TRANSACTION comienza la transacción y la sentencia COMMIT consolida o guarda la transacción en la base de datos permanentemente. Si cualquier parte de la transacción fracasa, la sentencia ROLLBACK se utiliza para deshacer los cambios hechos en la transacción y regresar la base de datos a su estado original. Consideremos este ejemplo:

```
BEGIN TRANSACTION;  
  
UPDATE accounts SET balance = balance - 250 WHERE account_id =786543;  
  
UPDATE accounts SET balance = balance + 250 WHERE account_id = 90643;  
  
COMMIT;
```

En este ejemplo, la transacción tiene dos sentencias UPDATE. Una resta \$250 de la cuenta account\_id 786543 y la otra añade \$250 en otra cuenta. Si cualquiera de las dos sentencias UPDATE fracasara, otra sentencia, ROLLBACK, deshacería los cambios de la transacción y devolvería la base de datos a su estado original. En cualquier momento que estés inseguro de los cambios que estén a punto de hacerse, es recomendable crear una sentencia SAVEPOINT para poder revertir los cambios de la base de datos al SAVEPOINT. Consideremos el siguiente conjunto de transacciones:

```
SAVEPOINT Mark1;  
  
UPDATE accounts SET balance = balance - 250 WHERE account_id =786543;  
  
SAVEPOINT Mark2;  
  
UPDATE accounts SET balance = balance + 250 WHERE account_id = 90643;  
  
ROLLBACK TO Mark2;
```

Al final de este conjunto de transacciones, la base de datos solo tendrá un cambio (la primera actualización) ya que revertimos el segundo cambio.

Las transacciones implementan ciertos conceptos únicos para las bases de datos: atomicidad, consistencia, aislamiento y durabilidad, (ACID, por sus siglas en inglés); son populares en la industria y se les conoce como las propiedades ACID de las transacciones.

- **Atomicidad:** una transacción es atómica, es decir, que se ejecuta como una sola unidad de trabajo. O todas las operaciones en la transacción se completan exitosamente o ninguna. Si cualquier parte de la transacción fracasa, la transacción completa se revierte y la base de datos vuelve a su estado original. En nuestro ejemplo, cuando se transfiere dinero entre cuentas, ambas actividades, (retirar de una cuenta y depositar en otra) se completan como una sola transacción.
- **Consistencia:** una transacción debe mantener la base de datos en un estado válido antes y después de la transacción. La transacción no puede violar ninguna restricción o regla que se aplique a los datos. En nuestro ejemplo, la cantidad total de dinero en las cuentas debe ser la misma antes y después de la transferencia de dinero entre las cuentas.
- **Aislamiento:** los datos a los que tiene acceso una transacción deben de estar aislados y no afectar otras transacciones realizadas paralelamente. Una transferencia de dinero de una cuenta a otra no debe tener ningún impacto en otras actividades bancarias entre otros usuarios y sus cuentas.
- **Durabilidad:** una vez que se confirma o se guarda una transacción en la base de datos, los cambios son permanentes y deben sobrevivir a cualquier malfuncionamiento o crash. Incluso si la falla ocurriera después que la transacción se haya completado, los datos deben de ser recuperables.

## Control de privilegios de las bases de datos usando comandos DCL

Los administradores de bases de datos gestionan bases de datos mediante comandos DCL, los cuales los ayudan a crear roles y usuarios, a conceder privilegios a roles, a asignar roles a los usuarios y a modificar y revocar privilegios. Otorgar privilegios específicos previene errores accidentales, mientras que eliminar usuarios cuando salen o cambian de organización evita accesos no autorizados.

El comando `CREATE USER` crea una nueva cuenta para un usuario en la base de datos.

Ejemplo: `CREATE USER jose_robelo IDENTIFIED BY 'password';`

El comando `ALTER USER` modifica las propiedades o privilegios de un usuario actual.

Ejemplo: `ALTER USER jose_robelo PASSWORD 'new_password';`

El comando `CREATE ROLE` se usa para crear un nuevo rol. Una vez que tienes un rol, puedes asignarle privilegios a ese rol. Es posible que tengas roles como programador, marketing, ventas y administrador, cada uno con un conjunto específico de privilegios. Cuando José Robelo se una a la organización, asignarle el rol adecuado va a asegurar que tenga todos los privilegios necesarios.

`CREATE ROLE marketing;`

Una vez que se tiene el rol, se le puede otorgar (`GRANT`) privilegios; y luego, asignar los roles a los usuarios.

`GRANT select, update ON customers TO marketing;`

`GRANT marketing TO jose_robelo;`



Más adelante, si José Robelo abandona la organización de marketing se le puede revocar (REVOKE) todos sus privilegios con el siguiente comando:

```
REVOKE marketing FROM jose_robelo;
```

## Conclusión

No hay ninguna organización moderna ni gobierno que no posea y dependa de su enorme base de datos con información sobre su gente, industrias, educación, sistemas bancarios y hospitales, entre otros. Las bases de datos modernas normalmente se construyen usando el modelo relacional. Un lenguaje sencillo, parecido al inglés, permite trabajar con información en una base de datos.

Aunque las bases de datos relacionales son populares y se usan ampliamente en las industrias y gobiernos, tecnologías más nuevas como las bases de datos NoSQL y bases de datos de gráficos son también útiles en ciertas situaciones. Cada base de datos tiene sus propios puntos fuertes y débiles. De modo que debes conocerlas antes de escoger una para almacenar tus datos. La mayoría de los programas informáticos de las universidades estudian el desarrollo y mantenimientos de las bases de datos más detalladamente.



## Términos y definiciones del capítulo

**Base de datos (*Database*):** Colección estructurada de datos que puede ser buscada y manipulada de manera eficiente por computadoras.

**Base de datos relacional (*Relational Database*):** Base de datos que organiza los datos en tablas interrelacionadas.

**Clave (*Key*):** Datos identificados de manera única en cada fila que se pueden utilizar para unir diferentes tablas entre sí.

**Consultar (*Query*):** SQL escrita para buscar y recuperar información puntual de una base de datos utilizando criterios específicos.

**Lenguaje de consulta estructurado (SQL) (*Structured Query Language*):** Lenguaje de programación diseñado para ejecutar un conjunto de operaciones que permiten extraer información de las bases de datos.

**Lenguaje de control de datos (DCL) (*Data Control Language*):** Conjunto de comandos SQL que funcionan como guardianes de los cambios de la información dentro de la base de datos. Algunos ejemplos incluyen COMMIT (comprometer), ROLLBACK (revertir) y SAVEPOINT (punto de retorno).

**Lenguaje de definición de datos (DDL) (*Data Definition Language*):** Conjunto de comandos SQL utilizados para definir la estructura de las bases de datos. Algunos ejemplos incluyen CREATE (crear), ALTER (alterar) y DROP (retirar).

**Lenguaje de manipulación de datos (DML) (*Data Manipulation Language*):** Conjunto de comandos SQL usados para manipular la información almacenada en las bases de datos. Algunos ejemplos incluyen SELECT (seleccionar), INSERT (insertar), UPDATE (actualizar) y DELETE (eliminar).

**Sistema de administración de bases de datos (DBMS) (*Database Management System*):** El software que gestiona la base de datos.

**Tabla (*Table*):** Una entidad dentro de una base de datos relacional que consiste en filas y columnas.



### Base de datos de seguimiento de calificaciones de Dionne

Dionne piensa ir a la Universidad del Sur de la Florida (USF) cuando se gradúe de la secundaria. USF tiene como requisito de admisión un promedio de notas (GPA) y por eso Dionne quiere estar muy pendiente de sus calificaciones. Ahora que ha aprendido más sobre el poder de las bases de datos, quiere diseñar una base de datos que lo ayude a mantenerse al tanto de sus notas. Pero no está seguro de qué base de datos escoger y qué opciones ofrecen. Tampoco está seguro si las bases de datos son gratuitas o si va a tener que pagar por usar una. Como no tiene mucho dinero, una gratuita o barata sería lo mejor. Dionne no está seguro si escoger una base de datos que tenga que instalar en su computadora o seleccionar una que opere en línea. Hay muchas preguntas que Dionne tiene que responder antes de diseñar su base de datos.



.....

**Pregunta 1:** Usa la Internet para buscar información en la World Wide Web de al menos un sistema de gestión de bases de datos que Dionne pueda usar para diseñar su base de datos. Si tienes dificultades encontrando esta información por favor repasa la sección “Evolución de las tecnologías de bases de datos” en este capítulo, donde puedes encontrar una lista de los sistemas de gestión de bases de datos relacionales más populares. Haz una lista de los sistemas de gestión de bases de datos que hayas encontrado para Dionne y describe lo siguiente: ¿Cuáles son algunas de las características más importantes del sistema de gestión de la base de datos? ¿Cuánto cuesta?



## Caso del capítulo (continuado)

**Pregunta 2:** Describe detalladamente qué tiene que hacer Dionne para usar el sistema de gestión de bases de datos. En otras palabras, ¿tendrá que instalarlo en su computadora, entrar a un sitio en la red, o utilizar algún otro método?