# UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

## SEDE VICTOR LEVI SASSO

## SISTEMA DE BASE DE DATOS I

## INCLUYE PRUEBAS SUMATIVAS Y PRESENTACIONES DEL CONTENIDO

**ELABORADO POR:**

**DR. CARLOS A. ROVETTO**

**JULIO 2020**

# Contenido

# Índice de figuras

# Introducción

Las bases de datos son un conjunto organizado de datos almacenados en una computadora. Considerando esto, es evidente la importancia que tiene esta área de estudio de la computación, dado que desde la perspectiva de las organizaciones, las bases de datos conforman uno de los recursos más valiosos: la información. La información, que no es más que datos con significado, necesita ser almacenada para que pueda agregar valor dentro de una organización y gracias a la evolución de los sistemas para el almacenamiento de la información, actualmente se cuentan con aplicaciones que facilitan este proceso y brindan una gran serie de ventajas.

Para una organización, no sólo es vital el almacenamiento de datos, sino que estos sean consistentes, que guarden relación con la realidad, que todos los individuos pueden acceder a ellos y principalmente, que sólo puedan ser obtenidos por un conjunto de individuos. En este sentido, se puede decir que la consistencia, la integridad, la distribución y la seguridad son factores primordiales para toda organización en cuanto a la información se refiere y las bases de datos han logrado convertirse en el elemento que todas las empresas necesitan para posicionarse dentro del mercado y competir con otras.

Lo expuesto anteriormente demuestra que para los profesionales de la computación es primordial conocer, comprender y desarrollar los fundamentos de las bases de datos, de modo que estos puedan aportar sus conocimientos no sólo para el progreso de las organizaciones, sino también para el avance en otras áreas de la computación que requieren de las bases de datos. El presente folleto tiene como objetivo ser una herramienta para transmitir los conceptos fundamentales a los estudiantes de la materia de Base de Datos I, de modo que a través de la teoría y ejemplos prácticos, se facilite no sólo la comprensión sino también la aplicación de estos.

# Capítulo I: Conceptos sobre base de datos

## *Introducción a las bases de datos*

Las bases de datos son una parte esencial de todo sistema de información y le han permitido a organizaciones grandes y pequeñas automatizar sus procedimientos. En la actualidad, casi todos los sistemas que utilizamos almacenan datos que pueden ser consultados o recuperados por un usuario y la manera en que se almacenan estos datos es a través de los sistemas de bases de datos.

Las bases de datos son uno de los avances más importantes dentro del campo de la Ingeniería de Software, ya que además del almacenamiento de datos, permiten su modificación y actualización. Por lo tanto, pueden adaptarse a las necesidades del individuo u organización y orientarlos en la toma de decisiones que permitan aumentar su competitividad y productividad.

## *Enfoque tradicional vs enfoque de base de datos para el tratamiento de datos e información*

El enfoque tradicional es el sistema basado en archivos y su estudio es el fundamento para el desarrollo del enfoque de base de datos. (Connolly & Begg, 2015) define al sistema basado en archivos como un conjunto de programas de aplicación, en el que cada programa define y gestiona sus propios datos y le brinda servicios con esos datos al usuario final. Este tipo de enfoque permitió digitalizar al obsoleto sistema en el que se almacenaban datos de forma manual, el cual contaba con muchas limitaciones para el procesamiento de la información.

Aunque el enfoque tradicional basado en archivos es más eficiente que el sistema manual, este cuenta con muchas limitaciones en comparación con el enfoque de base de datos. A continuación se definen estas limitaciones, basado en las descripciones de (Connolly & Begg, 2015):

- **Dependencia de los datos:** Dentro del código en el que se programa la aplicación están definidas la estructura física y la manera en la que se almacenan los archivos con los datos. Esto conlleva a que la modificación de esos datos sea difícil, debido a que se requiere la alteración de la estructura del código de la aplicación o la creación de un programa exclusivo para este propósito.
- **Duplicación de datos:** Debido a la descentralización de los datos en el enfoque tradicional, es necesario que los datos se repiten en diferentes archivos. Sin embargo, esto implica desperdicio de recursos, mayor espacio de almacenamiento y pérdida de integridad de los datos.
- **Consultas fijas y proliferación de programas de aplicación:** Para obtener información sobre los datos almacenados se requiere de consultas, las cuales forman parte del código de la aplicación y por lo tanto, es dependiente de su

desarrollador. En algunos casos, el desarrollador agregaba las consultas que se requerían, pero no se podían realizar consultas que no estuvieran dentro del código. En otros casos, se creaban aplicaciones que permitieran realizar las consultas que no estuviesen incluidas en el código, pero esto resultaba en una proliferación de programas de aplicación que generalmente eran poco eficientes, debido a que: *a)* se requería de una gran cantidad de recursos para su mantenimiento, *b)* la integridad y la seguridad de los datos era poco controlada, *c)* no era posible consultar datos de forma simultánea, debía ser una usuario a la vez y *d)* era casi imposible la recuperación de información en el caso del fallo de software o hardware.

- **Incompatibilidad de formatos de archivos:** Los datos se asocian al lenguaje de programación utilizado para escribir el código de la aplicación porque la estructura de los archivos está ligada a la aplicación. Esto dificulta la consulta de archivos entre dos o más aplicaciones y requiere que se conviertan los archivos a un formato en común para ambas aplicaciones, lo cual implica desperdicio de recursos.
- **Separación y aislamiento de datos:** Esto dificulta el acceso a los datos, pues estos están en archivos separados. Esto conlleva a que sea más complicado obtener datos de archivos diferentes.

El enfoque de base de datos resuelve estas limitaciones porque permite el almacenamiento independiente de los datos y el acceso y la manipulación de estos puede realizarse de forma simultánea.

**Ventajas de un ambiente de bases de datos**

El uso de los sistemas de bases de datos supone muchas ventajas, las cuales se describen a continuación considerando las descripciones de (Ricardo, 2009):

- **Control de redundancia:** La integración de los datos permite el control de la redundancia, esto significa que sólo se almacenan varias copias de un mismo dato si es necesario. Este tipo de redundancia se limita para la existencia de enlaces lógicos entre las distintas propiedades de los datos. En las bases de datos, a diferencia del sistema basado en archivos, se elimina la duplicación innecesaria de datos.
- **Compartición de datos:** La información contenida en la base de datos está distribuida, lo que quiere decir que esta le pertenece a toda la organización. Esto no era posible cuando se utilizaba el sistema basado en archivos porque diferentes datos pertenecían a diferentes departamentos dentro de una misma organización, a pesar de que estos datos tuvieran algún tipo de relación lógica. Sin embargo, aunque la base de datos pertenece a toda la organización, es posible autorizar el acceso a ciertas porciones de datos a múltiples usuarios, dependiendo de las necesidades o requisitos de los departamentos de la organización.

- **Consistencia de los datos:** Este es el resultado del control de la redundancia, ya que existe coherencia entre los datos. La existencia de conexiones lógicas asegura que la actualización del valor de un dato se haga en toda la base de datos y que sea visible para todos los usuarios con acceso a estos. A su vez, cuando hay redundancia controlada, si se actualiza un valor, todas la ocurrencias de este dato se actualizarán en la base de datos.
- **Mejor integridad:** Se logra a través de la definición de restricciones o *constraints*, las cuales no son más que reglas que la base de datos debe cumplir y estas pueden ser aplicadas a las propiedades dentro de un registro, a las relaciones entre distintos registros o bien, pueden ser restricciones relacionadas a la organización. Estas características evitan que se hagan modificaciones que infrinjan contra alguna de las restricciones establecidas.
- **Mayor seguridad:** La base de datos cuenta con propiedades que permiten la protección de los datos, ya sea de accesos no autorizados por parte de los usuarios o de programas que puedan hacer mal uso de los datos. Cada usuario de la base de datos tiene diferentes permisos de acceso, de esta manera un usuario tendrá autorización para ver una determinada vista que corresponde a un fragmento de los datos y estos permisos de acceso pueden incluir la actualización, borrado o inserción de registros. Además, la base de datos puede cifrar los datos o permitir el acceso de los datos a través de contraseñas. Dentro de las organizaciones, diferentes departamentos tienen acceso a la base de datos, pero no todos tienen las mismas vistas ni permisos. Esto garantiza la seguridad de los datos, los cuales son un recurso de gran importancia para toda organización.
- **Equilibrio de requisitos en conflicto:** Los usuarios o los departamentos dentro de una organización, pueden tener necesidades distintas en cuanto a la información contenida en la base de datos y algunas veces estas necesidades entran conflicto con las de otros usuarios. Para ello, el programa que manipula a la base de datos es capaz de brindar soluciones que resulten en el mayor beneficio para la organización, a través del diseño, uso y mantenimiento de la base de datos.
- **Economía en escala:** El costo del diseño, creación y mantenimiento de una base de datos puede resultar más bajo, que si se asignarán porciones de un presupuesto a todos los departamentos de una organización, si estos utilizaran el sistema basado en archivos. Como la base de datos pertenece a toda la organización, cada departamento no tiene que financiar de forma independiente el procesamiento de los datos a través de los archivos y este presupuesto puede invertirse en el desarrollo de la base de datos.
- **Control sobre la concurrencia:** Los sistemas de bases de datos permiten que varios usuarios ingresen datos de forma simultánea. Puede suceder que alguno de los usuarios actualiza datos y para evitar la pérdida de información o la pérdida de

integridad de los datos, los sistemas de bases de datos son capaces de controlar la concurrencia, de modo que se mantenga el correcto desempeño.

- **Mejor accesibilidad a los datos:** Los sistemas de bases de datos les permiten a los usuarios la interacción con los datos almacenados, a través de un lenguaje de consulta que le brinda al usuario la información necesaria para tomar decisiones.
- **Asignación de estándares:** Corresponde a las reglas de la organización con relación a la representación de los datos almacenados en la base de datos. Estas reglas pueden referirse al formato de las propiedades de los datos, la constancia con la que se debe actualizar o realizar respaldos de la información y los procedimientos para realizar esos respaldos.
- **Rapidez en el desarrollo de nuevas aplicaciones:** La base de datos de una organización es un patrón para el desarrollo de nuevas aplicaciones, ya que estas almacenan datos que generalmente se requieren para realizar este proceso. De esta manera se reduce el tiempo y el costo que podría generarse si se utilizara el sistema basado en archivos.
- **Mayor capacidad de respaldo y recuperación:** Los sistemas de bases de datos cuentan con un mecanismo que permite respaldar regularmente la información almacenada. Al ocurrir una falla, el sistema de base de datos cuenta con métodos que le permiten al sistema autorrecuperarse al estado en el que estaba antes de este ocurriese la falla.

A pesar de las numerosas ventajas que supone el uso de un sistema de base de datos, es conveniente mencionar que existen algunas desventajas. Estas se describen a continuación tomando en consideración lo explicado por (Ricardo, 2009):

- **Alto costo de software:** Los sistemas de bases de datos son paquetes de software completos y complejos, por lo tanto, son costosos.
- **Alto costo de hardware:** Es necesario invertir dinero en hardware que cuente con las capacidades de almacenamiento y procesamiento necesarias para ejecutar el sistema de base de datos.
- **Alto costo de programación:** De igual forma, es necesario invertir dinero en la capacitación o contratación de programadores que puedan sacar el mejor provecho al sistema de base de datos. Esto debido a la complejidad de este tipo de software.
- **Aumento de vulnerabilidad en la seguridad:** El resto de las aplicaciones que dependan del sistema de base de datos, pueden verse afectadas si alguna de estas falla y dar lugar a datos erróneos dentro de la base de datos.
- **Aumento en la dificultad de recuperación:** Cuando la base de datos falla, muchos procesos denominados *transacciones*, pueden estar ocurriendo en ese momento, además de que los usuarios pueden actualizar los datos de forma simultánea. Esto dificulta la recuperación de la información porque el sistema debe

verificar cuáles transacciones se completaron y verificar la existencia de inconsistencia en los datos.

**Evolución de las bases de datos y nuevas tendencias**

El uso de computadoras dentro de grandes y pequeñas organizaciones permitió automatizar el procesamiento de la información y por ello, es importante conocer cómo se originó lo que hoy se conoce como base de datos. A continuación se describen de forma cronológica los acontecimientos y avances que permitieron su desarrollo, basado en lo expuesto por (Silberschatz, Korth, & Sudarshan, 2011):

| Rango de tiempo | Características |
|---|---|
| **1950 a inicios de 1960** | • Se utilizaban cintas magnéticas y tarjetas perforadas para el almacenamiento de datos.<br>• Para procesar los datos estos se leían a partir de una o más cintas y se escribían en una cinta nueva. En el caso de las tarjetas perforadas, la salida de los datos era a través de la impresión.<br>• Las cintas magnéticas y las tarjetas perforadas sólo se podían leer de forma secuencial y los volúmenes de datos eran mucho mayores que los de la memoria principal. |
| **Finales de 1960 a 1970** | • El procesamiento de datos dio un gran giro gracias al incremento en el uso de los discos duros, ya que estos permitían acceder de forma directa a los datos.<br>• Disminuyó el tiempo de acceso a los datos, debido a que estos ya no debían ser leídos de forma secuencial. Era posible crear redes y bases de datos jerárquicas que permitían que programadores manipularan los datos, a través de estructuras de datos como listas y árboles.<br>• Edgar F. Codd definió en un artículo las bases del modelo relacional y las técnicas para la consulta de datos en este modelo, dando así origen a las bases de datos relacionales. |
| **Década de 1980** | • IBM desarrolló un sistema de base de datos denominado System R, como un proyecto para la aplicación de técnicas para la construcción de una base de datos relacional eficiente. Esto surgió tras la publicación del artículo de Codd, el cual aunque era de gran interés académico, no pudo ser aplicado de forma práctica al inicio debido a varias limitaciones por el uso de las bases de datos de red y jerárquicas.<br>• El proyecto System R dio origen al uso de SQL como lenguaje de consulta y a partir de este acontecimiento, las bases de datos relacionales empezaron a ser más utilizadas. |

| | |
|---|---|
| | • Las bases de datos relacionales eventualmente empezaron a reemplazar a las bases de datos de red y jerárquicas, debido a su fácil uso.<br>• Empezaron a desarrollarse investigaciones relacionadas a las bases de datos paralelas, distribuidas y orientadas a objetos. |
| **Inicios de 1990** | • El uso del lenguaje SQL, el cual está diseñado para realizar consultas, comenzó a convertirse en una importante aplicación dentro del área de las bases de datos.<br>• Surgió el amplio uso de herramientas para el análisis de grandes cantidades de datos.<br>• Los proveedores de sistemas de bases de datos comenzaron a comercializar productos basados en bases de datos paralelas y con soporte relacional de objetos. |
| **Década de 1990** | • La implementación y el uso de las bases de datos aumentó extensivamente gracias al crecimiento en el uso del Internet.<br>• Los sistemas de bases de datos ahora debían ser capaces de soportar altas cantidades de transacciones y estar disponibles para funcionar 24/7. |
| **Década del 2000 y el presente** | • En la primera mitad de la década del 2000 surgió el uso de XML y su correspondiente lenguaje de consulta XQuery como una nueva tecnología en las bases de datos. Sin embargo, las bases de datos relacionales tienen un uso más extendido.<br>• Aumentó el uso de sistemas de bases de datos de código abierto, principalmente PostgreSQL y MySQL.<br>• Durante la segunda mitad de la década, debido al incremento en el análisis de grandes volúmenes de datos, surgió el uso de sistemas de bases de datos especializados para estos procesos.<br>• Los sistemas de bases de datos han empezado a agregar funcionalidades que permitan manipular los requerimientos de almacenamiento de grandes sitios web.<br>• Ha surgido la importancia de las técnicas para los procesos relacionados a la minería de datos. |

## *Conceptos básicos*

Las bases de datos son una parte esencial de toda organización que utilice computadoras para el almacenamiento y procesamiento de información. En la actualidad, esta tecnología se aplica en una gran variedad de áreas como negocios, medicina, ingeniería, educación, investigación científica, entre otras; incluyendo a individuos que

utilicen las bases de datos para uso personal. Considerando estos hechos es importante conocer diferentes conceptos relacionados a las bases de datos con el fin de comprender su funcionamiento.

**Bases de datos**

(Elmasri & Navathe, 2016) define a una base de datos, desde un punto de vista general, como un conjunto de datos relacionados. Sin embargo, una definición más específica y de acuerdo con (Connolly & Begg, 2015), una base de datos es un conjunto compartido de datos que están relacionados de forma lógica y cuentan con descripciones que le permiten a una organización tomar decisiones relacionadas con esa información.

Una base de datos es un repositorio en el que se registran datos que representan diversos aspectos del mundo real. Por ello, una base de datos debe ser capaz de almacenar una gran cantidad de información y debe ser mantenida y actualizada periódicamente para que sea precisa con lo que representa. El mantenimiento de la base de datos puede ser hecha por uno o más programadores o por un conjunto de programas de aplicación que están diseñados para ejecutar esta tarea.

Por otro lado, los datos se obtienen a partir de diversas fuentes y luego se registran en la base de datos que se diseña y se crea con un fin específico. Por ello, estos datos no son escogidos al azar, mantienen coherencia entre ellos. (Connolly & Begg, 2015) explica que al analizar la información necesaria para crear la base de datos de una organización, se pueden identificar tres elementos:

- **Entidad:** Es un objeto concreto que se representa en la base de datos. Estos pueden ser una persona, un lugar, una cosa, un evento o un concepto.
- **Atributo:** Es una propiedad que describe al objeto que está registrado en la base de datos.
- **Relación:** Es la correspondencia o vínculo entre entidades.

Estos tres elementos son los que permiten que los datos tengan descripciones dentro de la base de datos. Las descripciones de los datos se conocen como *metadatos* y estos son datos acerca de los datos.

**DBMS**

El Sistema de Gestión de Bases de Datos (SGBD), conocido en inglés como Database Management System (DBMS), es un software que cuenta con herramientas que permiten definir, crear, mantener y manipular a una base de datos (Millán, 2012). El DBMS es el programa que funciona como interfaz entre el usuario y la base de datos, a través de accesos controlados y por consiguiente, dependiendo de los permisos que los usuarios tengan, estos obtienen diferentes vistas para la manipulación de los datos.

Como explica (Kroenke & Auer, 2012), el DBMS es un programa complejo y que por lo general, se comercializa bajo licencia. Además, es de propósito general, lo cual lo hace

ideal para uso en diferentes áreas. Algunos de los DBMS que se pueden mencionar son MySQL de Oracle Corporation y Microsoft Access y SQL Server de Microsoft Corporation.

Uno de los principales elementos que componen al DBMS es la misma base de datos, que como se ha explicado anteriormente, contiene una colección de datos interrelacionados. Sin embargo, el DBMS cuenta con otros elementos y características importantes que se describirán a continuación:

- **Arquitectura general:** La arquitectura del DBMS se compone de tres niveles que se denominan *arquitectura ANSI/SPARC*, porque fue inicialmente propuesta por estas dos organizaciones: el American National Standards Institute (ANSI) y el Standards Planning And Requirements Committe (SPARC). Esta arquitectura es una herramienta que permite la visualización de los niveles en el sistema de base de datos y cada nivel no es más que una descripción de los datos (Elmasri & Navathe, 2016). Como indica (Ricardo, 2009), los motivos principales que justifican el uso de tres niveles en la arquitectura son:
  - La base de datos es utilizada por usuarios que realizan tareas diferentes y por consiguiente, cada usuario requiere vistas diferentes de los datos.
  - Las vistas pueden variar con el tiempo, dependiendo de las necesidades del usuario.
  - El proceso de mantener la estructura de la base de datos no es tarea de los usuarios.
  - El administrador de la base de datos debe poder realizar cambios a la estructura de la base de datos, esto incluye al modelo lógico, el modelo conceptual y la estructura de los datos. Estos cambios no deben afectar a los usuarios y sus correspondientes vistas.
  - Los cambios que realicen a los elementos físicos que permiten el almacenamiento de los datos, no deben afectar a la estructura de la base de datos.

  En la Figura 1 se muestra la arquitectura general de un DBMS, el cual consta de tres niveles que se describen a continuación:

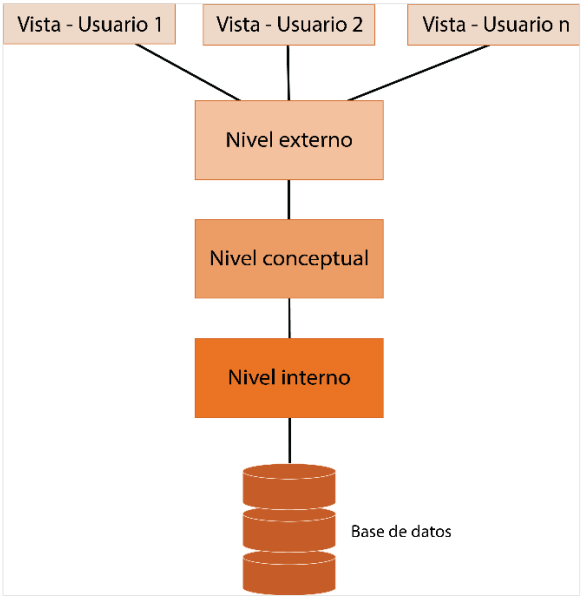



*Figura 1. Arquitectura general de un DBMS. - Adaptado de* (Millán, 2012).

A continuación se describen los tres niveles que componen a la arquitectura ANSI/SPARC:

- **Nivel interno:** Corresponde a la representación y el almacenamiento físico de la base de datos, tanto a nivel de software (sistema de gestión de la base de datos) como a nivel de hardware (equipo informático para el almacenamiento de la base de datos). Este nivel se representa a través de un modelo de datos físico que define cómo se almacenan los datos y las rutas de acceso para la base de datos (Elmasri & Navathe, 2016).
- **Nivel conceptual:** Corresponde a una descripción de la base de datos completa, exceptuando detalles sobre el almacenamiento físico de los datos. Se describen elementos como las entidades, tipos de datos, relaciones y restricciones (Elmasri & Navathe, 2016).
- **Nivel externo:** Corresponde a la representación de las vistas de los distintos usuarios de la base de datos. Cada vista muestra una descripción de sólo una parte de la base de datos la cual es de interés para un usuario en particular, esto quiere decir que la vista no incluye aquellos datos a los que el usuario no tiene permiso de acceso (Cardona et al., 2014). Los lenguajes de manipulación de datos y las interfaces de las aplicaciones que utilizan los usuarios también forman parte de este nivel.

A continuación en la Figura 2, se muestran las diferencias entre cada uno de los niveles de la arquitectura de un DBMS.

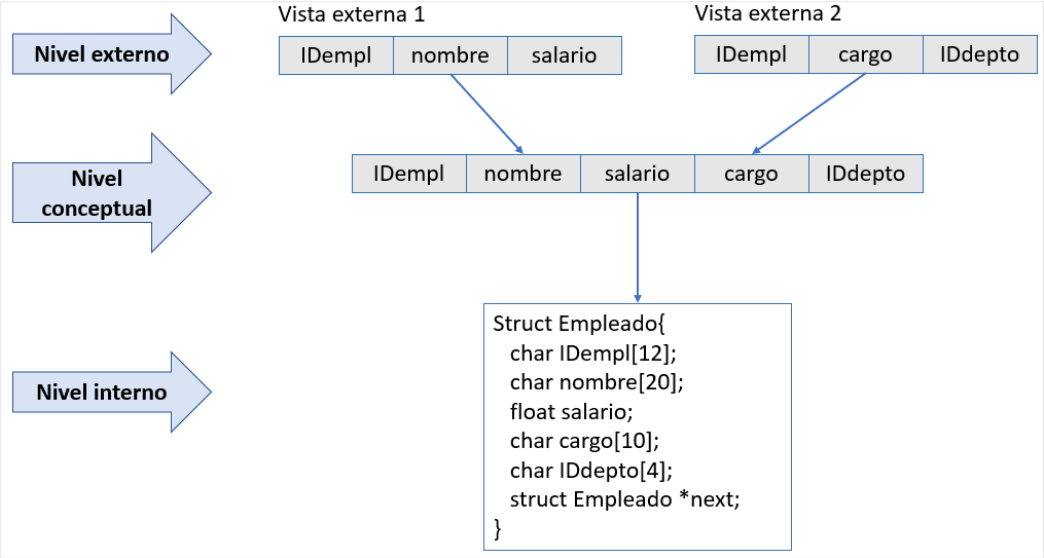



*Figura 2. Diferencias entre los niveles de la arquitectura de un DBMS. - Adaptado de* (Connolly & Begg, 2015).

- **Lenguaje de definición y manipulación:** Ambos lenguajes le proveen al usuario la facilidad de realizar tareas con los datos almacenados en la base de datos. El lenguaje de definición de datos o *Data Definition Language (DDL)*, permite definir el tipo, la estructura y las restricciones de los datos, mientras que el lenguaje de manipulación de datos o *Data Manipulation Language (DML)*, permite la inserción, eliminación, actualización y recuperación de datos (Connolly & Begg, 2015). El DML más utilizado es *Structured Query Language (SQL)*.
- **Funciones del DBMS:** Tras un vistazo general sobre las diversas capacidades del DBMS, conviene profundizar en sus funcionalidades. Estas se describen a continuación, basado en lo explicado por (Kroenke & Auer, 2012):
  - Creación de la base de datos y de las tablas que contendrán los registros con los datos.
  - Creación y mantenimiento de estructuras de soporte dentro de la base de datos. Las tablas son un ejemplo de estas estructuras.
  - Lectura y modificación de datos. Para ello el DBMS recibe consultas en lenguaje SQL y otros tipos de solicitudes, y estas son transformadas en acciones que se ejecutan dentro de la base de datos.
  - Declaración y aplicación de reglas para los datos. Estas reglas se denominan *restricciones de integridad referencial*.
  - Administración de la base de datos a través de:
    - Control de la concurrencia, con el cual se asegura que los usuarios puedan compartir el acceso a la base de datos y que ningún usuario interfiera con otro.

- Control de seguridad, a través del cual se permite el acceso a la base de datos sólo a los usuarios autorizados y que estos realicen las tareas para las que tienen permiso.
- Procesos de respaldo y recuperación, para reestablecer la base de datos en los casos en los que haya falla de software o hardware.

- **Interfaces, utilitarios, herramientas de aplicación y recursos de comunicación:** A continuación se describen estos elementos, considerando las descripciones de (Elmasri & Navathe, 2016):
  - **Interfaces:** Permiten la interacción entre el usuario y las diversas funcionalidades del DBMS. A continuación se describen los tipos de interfaces más utilizados:

| Tipos de interfaces | Características |
|---|---|
| **Interfaces basadas en menús para clientes o navegación web** | • A través de listas de opciones, denominadas *menús*, el usuario es guiado para que haga una solicitud.<br>• El menú funciona como una guía paso a paso para que el usuario escoja los comandos que se muestran. No es necesario que el usuario memorice los comandos o la sintaxis del lenguaje de consulta.<br>• En las interfaces de clientes suelen utilizarse menús desplegables, mientras que en las de navegación los usuarios pueden explorar entre el contenido de la base de datos. |
| **Aplicaciones para dispositivos móviles** | • El usuario acceda a sus datos a través de una interfaz incorporada en el código de la aplicación.<br>• Por lo general, el usuario debe iniciar sesión en una cuenta creada en la aplicación para que esta le proporcione un menú con opciones para acceder a sus datos. |
| **Interfaces basadas en formularios** | • Se basa en el uso de formularios que el usuario llena. Las entradas del formulario son los datos que serán insertados o bien, el usuario llena algunas de las entradas y el DBMS se encarga de recuperar datos para llenar los entradas restantes.<br>• Muchos DBMS cuentan con lenguajes específicos para que los programadores definan estos formularios. |
| **Interfaces basadas en búsqueda de palabras clave** | • El usuario inserta palabras en lenguaje natural y el DBMS las asocia con documentos en sitios o páginas web desde un motor de búsqueda. |

| | |
|---|---|
| | • El DBMS utiliza índices predeterminados de palabras y a través de funciones de clasificación, recupera y le presenta los resultados al usuario en orden decreciente. |
| **Interfaces para usuarios paramétricos** | • Son interfaces específicas para usuarios inexpertos, es decir, aquellos que no tienen conocimiento sobre base de datos y sólo la utilizan a través de la aplicación.<br>• El diseño e implementación de este tipo de interfaces incluye una interfaz específica para cada tipo de usuario paramétrico que se conoce utilizará la aplicación. |
| **Interfaces para el Administrador de la Base de Datos (DBA)** | • Son de uso exclusivo para aquellos que tienen la tarea de administrar la base de datos, ya que estas interfaces incluyen comandos únicos.<br>• Los comandos le permiten al administrador realizar diferentes tareas como crear cuentas, determinar los permisos de estas, definir parámetros del sistema, entre otros. |

- **Utilitarios:** Son elementos que asisten al administrado de la base de datos en el mantenimiento del sistema. A continuación se detallan los utilitarios más utilizados:
  - **Inserción:** Carga de archivos de datos a la base de datos, conversión del archivo al formato especificado por el utilitario y transferencia de datos de un DBMS a otro.
  - **Monitoreo de rendimiento:** Monitoreo del uso de la base de datos, generación de estadísticas que orientan al administrado en la toma de decisiones relacionadas a la base de datos.
  - **Reorganización del almacenamiento de la base de datos:** Reordenamiento de archivos dentro de la base de datos y creación de nuevos accesos para mejorar el rendimiento de la base de datos.
  - **Respaldo:** Creación de copia de seguridad de la base de datos, la cual luego se almacena en un medio de almacenamiento masivo. Suele utilizarse el respaldo de tipo incremental porque ahorra espacio de almacenamiento.
- **Herramientas de aplicación y recursos de comunicación:** Son funcionalidades adicionales que pueden utilizar los usuarios, el diseñador de la base de datos y el propio DBMS. A continuación se especifican estas funcionalidades:
  - **Sistema de diccionario de datos:** Almacenamiento de un catálogo con información sobre esquemas, restricciones, usuarios y estándares.

- **Ambiente para el desarrollo de aplicaciones:** Entorno para desarrollar aplicaciones para la base de datos, tales como el diseño de la base datos, desarrollo de la interfaz gráfica, consultas y actualización de los datos.
- **Recursos de comunicación:** Acceso remoto a la base de datos desde los puestos de trabajo y computadoras personales. Esto requiere del hardware adecuado para establecer la comunicación tales como enrutadores, líneas telefónicas y redes locales.

- **Arquitectura cliente/servidor:** Como explica (Elmasri & Navathe, 2016), la arquitectura cliente/servidor se utiliza en entornos en los que hay un gran número de elementos de software y hardware que están conectados a través de una red. El cliente provee al usuario la interfaz adecuada para utilizar el servidor así como la capacidad de ejecutar aplicaciones de forma local, mientras que el servidor puede ser accedido por más un de cliente y tiene funciones específicas. Considerando esto, desde la perspectiva de las bases de datos, el cliente es la máquina que utiliza el usuario para acceder a la base de datos que se encuentra dentro de una máquina servidor y así realizar tareas que no pueden ser ejecutadas de forma local en la máquina cliente; la máquina servidor es aquella en la que se ejecuta el sistema de la base de datos. Se pueden identificar dos tipos de arquitecturas cliente/servidor, las cuales se describen a continuación con base en lo presentado por (Silberschatz et al., 2011):
  - **Arquitectura de dos niveles:** La aplicación de la base de datos se ejecuta en la máquina cliente desde la cual se solicita alguna funcionalidad del sistema de base de datos en la máquina servidor. En la Figura 3 se muestra la representación de la arquitectura cliente/servidor de dos niveles.

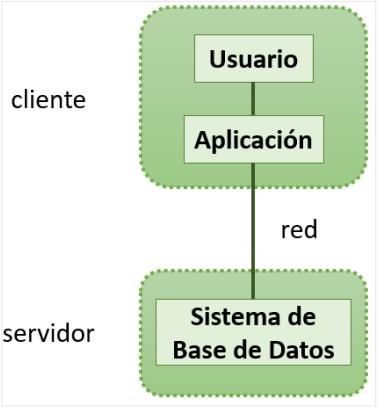



*Figura 3. Arquitectura cliente/servidor de dos niveles. - Tomado y adaptado de* (Silberschatz et al., 2011).

  - **Arquitectura de tres niveles:** La máquina cliente es simplemente una interfaz y no se pueden realizar solicitudes a la base de datos de forma directa. Para ello, requiere de una aplicación del servidor con el cual se comunica y este a su vez, se comunica con el sistema de base de datos para obtener acceso a

los datos. En la Figura 4 se muestra la representación de la arquitectura cliente/servidor de tres niveles.

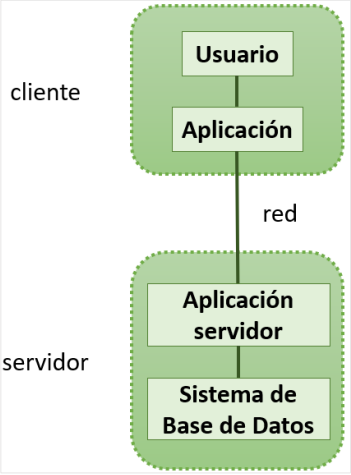



*Figura 4. Arquitectura cliente/servidor de tres niveles. - Tomado y adaptado de* (Silberschatz et al., 2011).

**Esquemas, instancias y estado de una base de datos**

Como indica (Silberschatz et al., 2011), es importante diferenciar los conceptos *esquema* e *instancia*, pues el esquema corresponde al diseño lógico de la base de datos, mientras que la instancia es una imagen del estado de los datos. A continuación, con base en lo explicado por (Elmasri & Navathe, 2016), se describirán con mayor detalle estos conceptos:

- **Esquemas:** Describe a la base de datos y se define durante el diseño de la base de datos. Dependiendo de los requerimientos de las aplicaciones de la base de datos, el esquema puede cambiar; sin embargo, no se espera que estos cambios sean frecuentes. La representación conceptual de un esquema se denomina *diagrama de esquema* y de acuerdo con (Connolly & Begg, 2015) se pueden distinguir tres tipos de esquemas:
  1. **Esquema a nivel interno:** Descripción completa de la representación interna de la base de datos y especifica las definiciones de los registros almacenados, los campos de los datos y las estructuras de almacenamiento utilizadas.
  2. **Esquema a nivel conceptual:** Describe todas las entidades, atributos y relaciones de los datos, así como las restricciones que aseguran su integridad.
  3. **Esquema a nivel externo:** Corresponde a las distintas vistas de los datos que se le presentan a los usuarios.

  Toda base de datos cuenta con un esquema a nivel interno y uno a nivel conceptual, sin embargo puede tener varios esquemas a nivel externo.
- **Instancias:** El estado de los datos en un instante determinado se denomina *estado de la base de datos* y se pueden distinguir tres tipos de estados, de acuerdo con (Elmasri & Navathe, 2016):
  1. **Estado vacío:** Ocurre cuando la base de datos no tiene datos almacenados.

2. **Estado inicial:** Ocurre cuando la base de datos se carga con datos por primera vez.
3. **Estado actual:** Ocurre en cualquier instante en el tiempo, es decir, cada vez que se actualice la base de datos.

**Independencia lógica y física de datos**

Considerando la arquitectura de tres niveles que se describió anteriormente, uno de los principales motivos de su aplicación en un DBMS es la de proveer la independencia de los datos. El término *independencia de datos* es definida por (Elmasri & Navathe, 2016) como la capacidad de modificar un esquema de un determinado nivel sin que se afecte un esquema de un nivel superior.

Se pueden distinguir dos tipos de independencia de datos que se definen a continuación, de acuerdo con las descripciones de (Elmasri & Navathe, 2016):

1. **Independencia lógica:** Permite que se realicen modificaciones en el esquema conceptual sin que esto afecte a los esquemas externos. Como ejemplos de las modificaciones que se le pueden hacer al esquema conceptual se mencionan la expansión o reducción de la base de datos a través de un registro o el atributo de un dato y el cambio de restricciones.
2. **Independencia física:** Permite que se realicen modificaciones en el esquema interno sin esto afecte al esquema conceptual y por consiguiente, a los esquemas externos. Como ejemplo de modificación que se puede realizar en el esquema interno es cuando se requiere la reorganización de archivos físicos, de modo que se mejoren las tareas de recuperación y actualización de los datos.

**Modelo de datos**

Anteriormente se mencionó que las bases de datos son una representación de la vida real y para lograr esto, es necesario describir la estructura de la base de datos, lo cual corresponde a los datos, sus relaciones y sus restricciones. Estos conceptos pueden representarse de una manera más sencilla y comprensible a través de un modelo de datos. (Connolly & Begg, 2015) indica que un modelo de datos cuenta con tres componentes: 1) una *parte estructural* que corresponde al conjunto de reglas sobre las que se basa el diseño de la base de datos, 2) una *parte manipulativa* que corresponde a la definición de los procedimientos que pueden realizarse sobre los datos y 3) un *conjunto de restricciones de integridad* que aseguran la exactitud de los datos.

- **Definición:** Una definición formal del concepto de *modelo de datos* es brindada por (Elmasri & Navathe, 2016): es un conjunto de conceptos que permiten la descripción de la estructura de una base de datos y brinda los recursos requeridos para lograr esta abstracción.
- **Abstracción:** Este término se refiere a que un modelo de datos detalla las características esenciales sobre los datos de modo que se mejore la comprensión

de estos, pero omitiendo aspectos relacionados a la organización y el almacenamiento de esos datos (Elmasri & Navathe, 2016).

- **Clasificación:** A continuación se describe la clasificación de modelos de datos propuesta por (Connolly & Begg, 2015):
  - **Modelo de datos basado en objetos:** Describe a la base de datos a través de los conceptos de entidad, atributo y relación. (Millán, 2012) agrega que en este tipo de modelo, a las entidades se les asocia una conducta y los atributos contienen valores, donde las conductas son acciones a los que responde la entidad y los valores representan el estado de la entidad. Se pueden distinguir diferentes tipos de modelos de datos basados en objetos, sin embargo se describirá el más utilizado:
    - **Entidad-relación (ER):** (Millán, 2012) explica que este tipo de modelo se compone de entidades que pertenecen a un conjunto y que están asociadas a través de relaciones. Cada entidad tiene una función en la relación y se puede obtener información sobre la entidad-relación a través de un par atributo-valor. En la Figura 5 se muestran dos ejemplos de modelos ER, siendo el primero mostrando a las entidades *estudiante* y *curso* y el segundo mostrando a las entidades *empleado* y *departamento*.

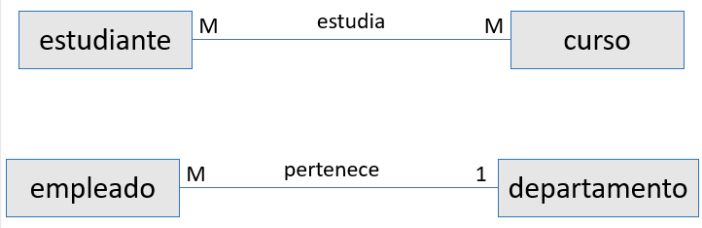



*Figura 5. Ejemplos de modelos entidad-relación.*

  - **Modelo de datos basado en registros:** Describe a la base de datos a través de un conjunto de registros de formato fijo y con una cantidad fija de campos. Estos registros pueden ser de diferentes tipos. Se pueden distinguir tres tipos de modelos de datos basados en registros:
    - **Modelo de datos relacional:** Representa los datos y sus relaciones a través de tablas, donde cada una de ellas tiene columnas que permiten la relación entre las tablas. Cada columna de la tabla cuenta con un nombre único y contiene un registro que corresponde a un valor que se asocia al valor de otra tabla. En la Figura 6 se muestra un ejemplo de un modelo de datos relacional.

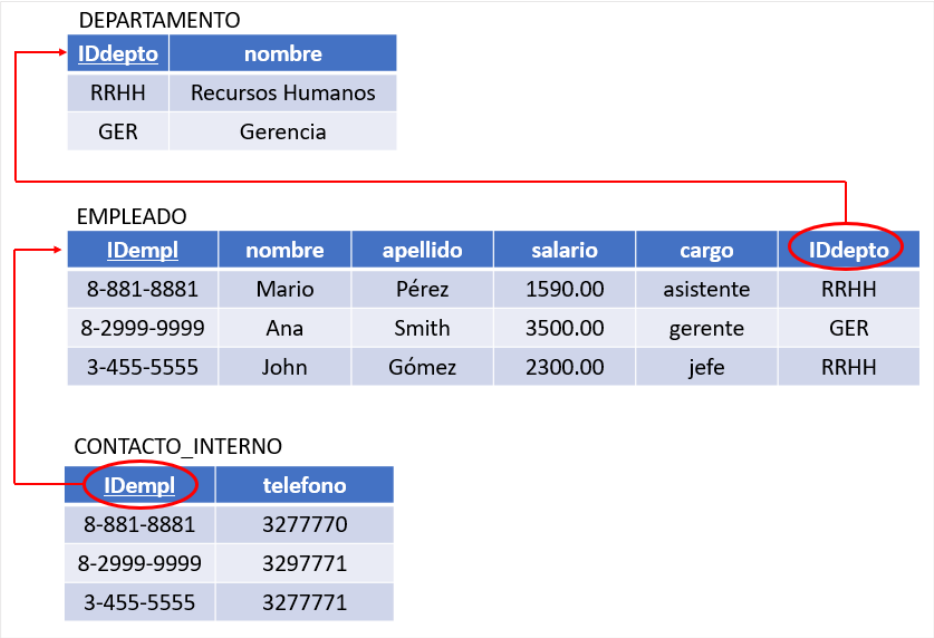



*Figura 6. Ejemplo de un modelo de datos relacional. – Adaptado de* (Connolly & Begg, 2015).

- **Modelo de datos en red:** Representa los datos como un grupo de registros organizados en un grafo, donde los registros corresponden a los nodos y sus relaciones se representan a través de conjuntos que corresponden a las aristas del grafo. En la Figura 7 se muestra un ejemplo de un modelo de datos en red.

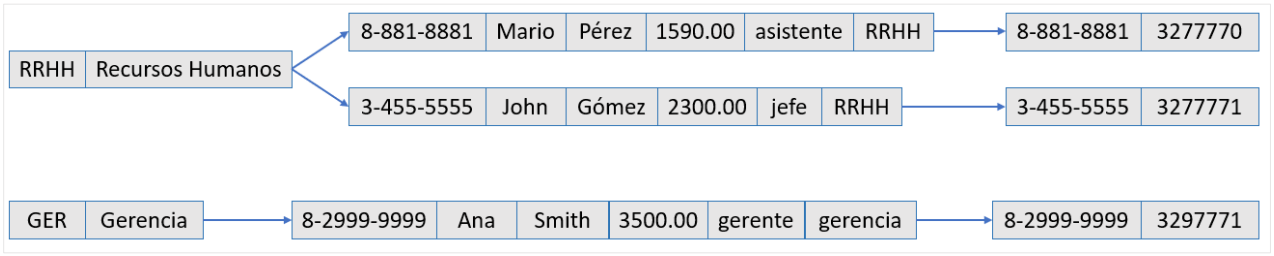



*Figura 7. Ejemplo de un modelo de datos en red. – Adaptado de* (Connolly & Begg, 2015).

- **Modelo de datos jerárquico:** Al igual que el tipo de modelo anterior, representa los datos como un grupo de registros y a las relaciones como conjuntos. Sin embargo, la diferencia radica en que cada nodo (registro) sólo puede tener un padre y se representa a través de un árbol, en el que los conjuntos (relaciones) son los enlaces del árbol. En la Figura 8 se muestra un ejemplo de un modelo de datos jerárquico.

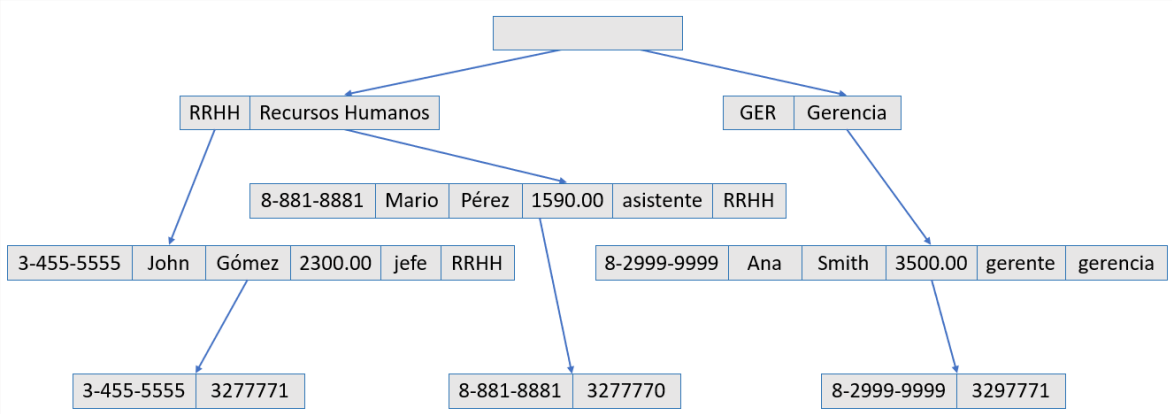



*Figura 8. Ejemplo de un modelo de datos jerárquico. – Adaptado de* (Connolly & Begg, 2015).

- **Modelo de datos físico:** Representa la forma en que se almacenan los datos, describiendo información tal como la estructura de los registros y las rutas de acceso de estos. Como indica (Elmasri & Navathe, 2016), este tipo de modelo está principalmente orientado para el uso por parte de especialistas en computación y no por usuarios finales.

## *Usuarios en un ambiente de base de datos*

Las personas son una parte esencial en el entorno de base de datos, pues son ellos quienes toman las decisiones sobre los datos y la propia base de datos. A continuación se describen los tipos de usuarios o roles que se pueden distinguir, considerando lo especificado por (Connolly & Begg, 2015):

| Rol | Funciones | |
|---|---|---|
| **Administradores de datos y de la base de datos** | **Administrador de datos (DA)** | • Planificar la base de datos.<br>• Diseñar el modelo conceptual y lógico de la base de datos.<br>• Desarrollar y mantener estándares, políticas y procedimientos.<br>• Consultar con sus superiores y orientarlos, de modo que el desarrollo de la base de datos vaya de acuerdo con los objetivos de la organización. |
| | **Administrador de la base de datos (DBA)** | • Desarrollar el diseño y la implementación de la base de datos.<br>• Controlar la seguridad e integridad de la base de datos.<br>• Asegurar el correcto funcionamiento de las aplicaciones para los usuarios. |

| | | |
|---|---|---|
| | | • Autorizar los permisos de acceso a la base de datos (Elmasri & Navathe, 2016).<br>• Tener un conocimiento detallado del DBMS que utilizará la organización, ya que su función es más técnica que la del DA. |
| **Diseñadores de la base de datos** | **Diseñador lógico** | • Responder al *qué* debe contener la base de datos.<br>• Identificar los datos, en términos de entidades y atributos, las relaciones entre esos datos y sus correspondientes restricciones.<br>• Comprender detalladamente sobre los datos y las restricciones que deben aplicarse, de acuerdo con las reglas de la organización.<br>• Desarrollar el modelo de datos considerando a los futuros usuarios de la base de datos. |
| | **Diseñador físico** | • Responder al *cómo* representar la base de datos.<br>• Desarrollar el diseño físico de la base de datos, considerando el diseño lógico.<br>• Diseñar las tablas y las restricciones de integridad correspondientes.<br>• Seleccionar las estructuras de almacenamiento y los métodos de acceso a los datos.<br>• Diseñar las medidas de seguridad que requieran los datos. |
| **Desarrolladores de aplicaciones** | • Desarrollar e implementar aplicaciones con funcionalidades que cumplan los requerimientos de los usuarios finales.<br>• Fundamentar el desarrollo de las aplicaciones en las especificaciones dadas por los analistas de sistemas. | |
| **Usuarios finales** | **Usuarios paramétricos** | • Por lo general, no tiene conocimiento sobre el DBMS (usuarios inexpertos).<br>• Acceder a los datos a través de aplicaciones que facilitan en la mayor medida de lo posible las tareas a realizar.<br>• Realizar solicitudes a la base de datos a través del uso de comandos simples o |

| | | |
|---|---|---|
| | | escogiendo opciones presentadas en un menú. |
| | **Usuarios avanzados** | • Conocer la estructura de la base de datos.<br>• Conocer las funcionalidades con las que cuenta el DBMS.<br>• Utilizar un lenguaje de consulta específico para realizar las tareas requeridas. |

### *Componentes de un ambiente de base de datos*

El entorno de base de datos se compone de diferentes elementos. Cada uno de ellos se describe a continuación, de acuerdo con las definiciones de (Connolly & Begg, 2015):

| | Componente | Descripción |
|---|---|---|
| Máquina | **Hardware** | • Partes físicas del sistemas informático que se requieren para ejecutar el DBMS y las aplicaciones.<br>• Puede variar desde una computadora personal hasta una red de computadoras.<br>• Depende de las necesidades de la organización y del DBMS que se utilizará. |
| Máquina | **Software** | • Comprende el DBMS, las aplicaciones, el sistema operativo y el programa de red, si el DBMS se utiliza a través de una red.<br>• Las aplicaciones suelen escribirse en un lenguaje de programación de tercera generación (Java, Visual Basic, C, etc.) o en un lenguaje de cuarta generación (SQL) dentro de un lenguaje de tercera generación.<br>• Algunos DBMS incluyen su propio lenguaje de cuarta generación, lo que facilita el desarrollo de aplicaciones. |
| Puente | **Datos** | • Es el componente más importante dentro del ambiente de base de datos.<br>• Funciona con un puente que crea el vínculo máquina-humano.<br>• Se compone de los datos contenidos en la base de datos, incluyendo los metadatos (datos sobre los datos). |
| Humano | **Procedimientos** | • Son las instrucciones y pautas que encaminan el diseño y el uso de la base de datos.<br>• Algunas de estas instrucciones pueden ser el inicio de sesión en el DBMS y la creación copias de respaldo. |

| | | |
|---|---|---|
| | **Personas** | • Corresponde a todas aquellas personas que forman parte del entorno de la base de datos, quienes desempeñan diferentes roles en este. |

En la Figura 9 se muestra la representación de los componentes de un ambiente de base de datos.

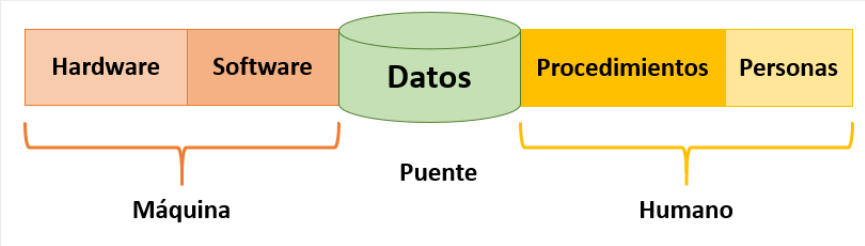



*Figura 9. Componentes de un ambiente de base de datos. - Tomado y adaptado de* (Connolly & Begg, 2015).

# Capítulo II: Fases en el desarrollo y construcción de una base de datos

## *Ciclo de vida del desarrollo de sistemas de bases de datos*

Al ser las bases de datos una parte fundamental del sistema de información de una organización, su ciclo de vida debe enfocarse en los requerimientos más imprescindibles; por ello, el ciclo de vida de un sistema de base de datos está vinculado al ciclo de vida del sistema de información de la organización (Connolly & Begg, 2015).

Los sistemas de bases de datos son un tipo de sistema de software que requieren de etapas específicas para su desarrollo, aunque muchas de estas son similares a las utilizadas en otros sistemas de software. Por ello, como indica (Gupta, Mata-Toledo, & Monger, 2011), es importante aplicar con un determinado orden las etapas para el desarrollo de un sistema de base de datos, ya que de esta manera es posible transformar los requisitos del usuario de forma eficiente y adecuada.

### Etapas en el desarrollo de un sistema de base de datos

Estas etapas serán descritas a continuación, basado en las explicaciones de (Connolly & Begg, 2015):

1. **Planificación de la base de datos:** Administración de las actividades que permiten que las etapas del ciclo de vida del desarrollo de la base de datos se realicen de manera eficiente y eficaz. Para llevar a cabo esta etapa se requiere identificar y definir lo siguiente:
   - ✓ **Misión del sistema:** Permite precisar los objetivos del sistema de base de datos, lo cual orienta en el establecimiento de su propósito y los procedimientos adecuados y necesarios para su creación.
   - ✓ **Objetivos del sistema:** Cada uno de los objetivos corresponde a una tarea específica que el sistema de base de datos debe ejecutar.
   - ✓ **Estándares:** Señalan cómo deben recolectarse los datos, su formato, qué documentación se necesitará y cómo se realizarán los procesos de diseño e implementación. El desarrollo de estándares es un proceso que toma tiempo, pero su correcto diseño puede proveer las bases para el control de la calidad y el entrenamiento de personal.
2. **Definición del sistema:** Establecimiento del alcance y los límites del sistema de base de datos y de las vistas de los usuarios. Esto incluye:
   - ✓ Especificar la actual y futura interacción del sistema de base de datos con el resto del sistema de información de la organización.
   - ✓ Las vistas de usuario se definen de acuerdo con los requisitos del sistema de base de datos. Se debe determinar quiénes son los usuarios o las áreas de aplicación empresarial y los datos que estos requieren de las vistas.

3. **Recolección y análisis de requisitos:** Recolección y análisis de la información sobre la organización, que debe ser incorporada en el sistema de base de datos y que permitirá identificar los requerimientos para el desarrollo de dicho sistema.
   - ✓ Se utilizan técnicas de determinación de hechos para la recolección de la información para cada vista de usuario. La información debe describir los datos que se utilizan y se generan, cómo son utilizados y cualquier otro requisito adicional para el sistema que se va a desarrollar.
   - ✓ Los requisitos identificados para el sistema de base de datos se describen en una documentación denominado *especificación de requisitos*. Es conveniente utilizar *técnicas de especificación de requisitos* que pueden ser: *a)* Análisis y diseño estructurado (SAD), *b)* Diagramas de flujo de datos (DFD) o *c)* Modelo jerárquico de entrada, proceso y salida (HIPO).
   - ✓ En el caso en el que el sistema de base de datos requiera de más de una vista de usuario, se pueden gestionar los requisitos del sistema aplicando alguno de los enfoques que se describen a continuación:
     - ➢ **Enfoque centralizado:** Los requisitos de cada vista de usuario se combinan en un solo conjunto de requisitos y se representan en un modelo de datos global durante el diseño de la base de datos. Este enfoque se suele utilizar cuando hay coincidencia de requisitos en cada vista de usuario y el sistema de base de datos no es muy complejo.
     - ➢ **Enfoque de integración de vistas:** Los requisitos para cada vista de usuario se mantienen separados, luego cada vista se representa en un modelo de datos local y cada modelo de datos local se combina en un modelo de datos global durante el diseño de la base de datos. Este último modelo debe representar todos los requisitos de los usuarios de la base de datos. Este enfoque se utiliza cuando hay diferencias significativas entre las vistas de usuario y la complejidad del sistema de base datos lo justifica.
     - ➢ **Combinación de ambos enfoques:** Se utiliza cuando el sistema a desarrollar es muy complejo y se requieren múltiples vistas de usuario.
4. **Diseño de la base de datos:** Creación de un diseño que incorpore la misión y objetivos del sistema, así como los estándares especificados como requisitos para el sistema de base de datos. Una tarea importante dentro de esta etapa es el *modelado de datos* y este tiene como objetivos: *a)* facilitar la comprensión del significado de los datos y *b)* facilitar la comunicación de la información sobre los requisitos. Además de esto, dentro de esta etapa se pueden distinguir tres enfoques de diseño y tres fases de diseño de la base de datos, los cuales se describen en el cuadro a continuación:

| | | Descripción |
|---|---|---|
| **Enfoques de diseño** | **De abajo a arriba** | Se inicia el diseño desde los atributos, los cuales son las propiedades de las entidades y las relaciones, y luego se agrupan en relaciones que representen los tipos de entidades y las relaciones entre las entidades. Este proceso de agrupación se denomina *normalización* y se describirá más detalladamente en capítulos posteriores. Este enfoque es conveniente cuando la base de datos a diseñar es simple y tiene una pequeña cantidad de atributos. |
| | **De arriba abajo** | Se inicia con la creación de un modelo de datos que muestre las entidades y relaciones de alto nivel, el cual se va depurando hasta obtener las entidades, relaciones y atributos de bajo nivel. Este enfoque es conveniente utilizarlo para el diseño de bases de datos complejas. |
| | **Mixta** | Se utilizan ambos enfoques y al final del proceso se combinan todas las partes en las que se utilizaron estos enfoques. |
| **Fases de diseño** | **Conceptual** | • Se crea un modelo de datos que represente las partes de la organización que están bajo estudio. El modelo es independiente de los detalles físicos o de implementación.<br>• Se utiliza la información que se documentó en la especificación de requisitos de los usuarios y estos requisitos son validados con el modelo.<br>• Brinda información para la próxima fase de diseño (lógico). |
| | **Lógico** | • Se depura el modelo de la fase anterior para diseñar un modelo lógico de datos detallado. Es independiente de los detalles físicos y de implementación, pero se basa en el modelo de datos que utiliza el DBMS en el que se implementará la base de datos.<br>• Durante su desarrollo se valida el modelo con los requisitos de los usuarios.<br>• Es un modelo importante durante la etapa de mantenimiento que forma parte del ciclo de vida del sistema. |

| | | |
|---|---|---|
| | | • Brinda información para la siguiente fase de diseño (físico). |
| | **Físico** | • Se describen detalles sobre la implementación física de la base de datos y cómo se almacenará la información que contenga.<br>• Se describen la organización de los archivos, los índices de acceso a los datos y las restricciones de integridad.<br>• Aunque la estructura es independiente del DBMS que se utilizará, el diseño es específico para este. |

En la Figura 10 se muestra la arquitectura general de un DBMS y las correspondientes fases de diseño descritas anteriormente:

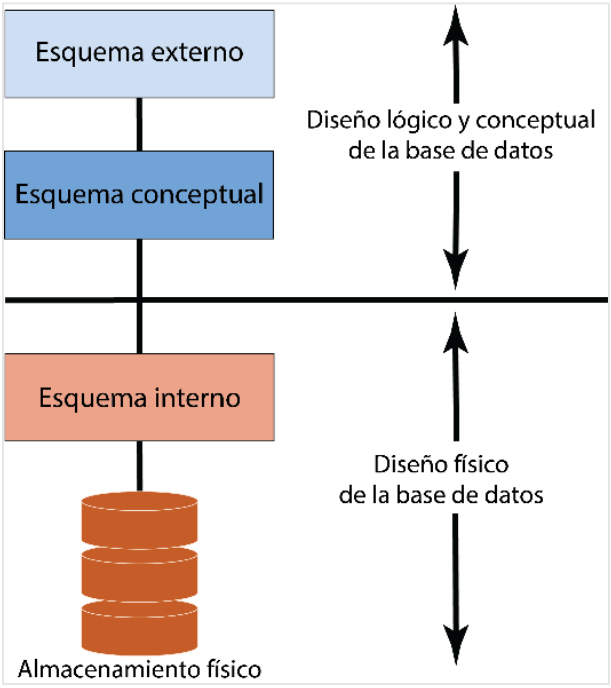



*Figura 10. Arquitectura general de un DBMS y sus correspondientes fases de diseño. - Tomado y adaptado de* (Connolly & Begg, 2015).

5. **Selección del DBMS:** Corresponde a la selección de un DBMS que cumpla con los requisitos del sistema de base de datos. El DBMS puede ser escogido antes del diseño lógico, si se conoce información suficiente acerca de los requisitos del sistema y esta selección puede hacer a través de la comparación de las características de distintos DBMS con los requisitos. Este proceso se compone de cuatro pasos que se describen a continuación:

a) **Definir Términos de Referencia del estudio:** Es un documento que indica los criterios de evaluación de los productos (DBMS) que se evaluarán. Los criterios van de la mano con la especificación de requisitos de los usuarios. Se describen los objetivos y el alcance del estudio y se provee una lista inicial de posibles productos para evaluar.
b) **Listar dos o tres productos:** La lista inicial de productos a evaluar puede hacerse basado en aquellos criterios que se consideren decisivos. Algunos de estos criterios pueden ser el presupuesto, la compatibilidad del DBMS con otros softwares y la calidad del servicio ofrecido por el vendedor del DMBS.
c) **Evaluar productos:** La evaluación de los productos pueden hacerse con base en las características organizadas en grupos, de modo que cada grupo sea ponderado para al final obtener un valor final sobre el DBMS. Los grupos de características para la evaluación pueden ser la definición de datos, la definición física, la accesibilidad y el manejo de transacciones.
d) **Recomendar selección y generar reporte:** El proceso de selección se documenta, se indican los resultados de la evaluación y se provee una recomendación sobre alguno de los productos de DBMS.

6. **Diseño de la aplicación:** Corresponde al diseño de la interfaz de usuario y de los programas de aplicación que permitirán el uso de la base de datos. Durante el desarrollo de esta etapa es importante asegurar que el diseño se fundamente en la especificación de requisitos de los usuarios. Esta etapa se compone de dos partes:
   - ✓ **Diseño de transacciones:** Las transacciones son las acciones que el usuario o el programa de aplicación ejecutarán para acceder o modificar lo que contiene la base de datos. Considerando esto, esta parte del diseño requiere que se definan aspectos como los datos que utilizará la transacción, los aspectos funcionales de la transacción, qué generará la transacción y el tipo de transacción (recuperación o actualización de datos, o combinación de ambos).
   - ✓ **Diseño de la interfaz de usuario:** Corresponde al diseño de los aspectos físicos que los usuarios verán en la aplicación. Algunos de estos aspectos físicos son: título, instrucciones claras, abreviaciones y términos consistentes, colores consistentes, espacio visible para la entrada de datos, mensajes de error, entre otros.
7. **Prototipado:** Es una etapa opcional que consiste en construir un modelo de trabajo del sistema en desarrollo. El prototipo no cuenta con todas las funcionalidades del sistema de base de datos final, los usuarios pueden utilizarlo y describir qué aspectos funcionan de manera adecuada y cuáles no, así como brindar sugerencias de mejoras o funcionalidades que podrían agregarse al sistema. El prototipado es esencial cuando el sistema que se desarrolla es de alta complejidad o es muy costoso.

8. **Implementación:** Implica utilizar las sentencias DDL del DBMS seleccionado o una GUI (Interfaz Gráfica de Usuario), para definir el diseño físico de la base de datos y de los diseños de las aplicaciones. Los programas de aplicación se implementan a través de lenguajes de tercera (3GL) o cuarta (4GL) generación, tales como Java, C, Python y Visual Basic. También se crean los controles de seguridad e integridad para el sistema de base de datos.
9. **Conversión y carga de datos:** Consiste en transferir los datos al nuevo sistema de base de datos y esta etapa es imprescindible cuando el nuevo sistema reemplaza a un sistema antiguo. Es común que los DBMS actuales brinden la funcionalidad de cargar los archivos existentes a la nueva base de datos, el programador únicamente debe especificar el archivo fuente y la base de datos destino y el DBMS convierte al formato requerido de forma automática. En ciertos casos, es posible utilizar programas de aplicación del sistema antiguo. Esta etapa requiere de planificación, ya que de esta manera se aseguran una transferencia y ejecución del sistema fluidos.
10. **Prueba:** Consiste en ejecutar el nuevo sistema de base de datos con el propósito de encontrar errores y validar que se cumplan los requisitos de los usuarios. Es importante involucrar a los usuarios en este proceso y utilizar respaldo de los datos si los que se utilizarán durante las pruebas son los datos reales.
11. **Mantenimiento:** Consiste en monitorear y darle mantenimiento al sistema de base de datos, una vez ha sido instalado y se ha iniciado su ejecución dentro de la organización. Como parte de esta etapa: *a)* se monitorea el rendimiento del sistema para asegurar que este se encuentre siempre dentro de niveles aceptables y *b)* se actualiza el sistema cuando se necesite agregar un nuevo requisito. Esta etapa es una de las más importantes y debe ejecutarse constante y permanentemente, mientras el sistema de base de datos esté en operación.

A continuación en la Figura 11 se muestran las etapas del ciclo de vida del desarrollo de una base de datos:

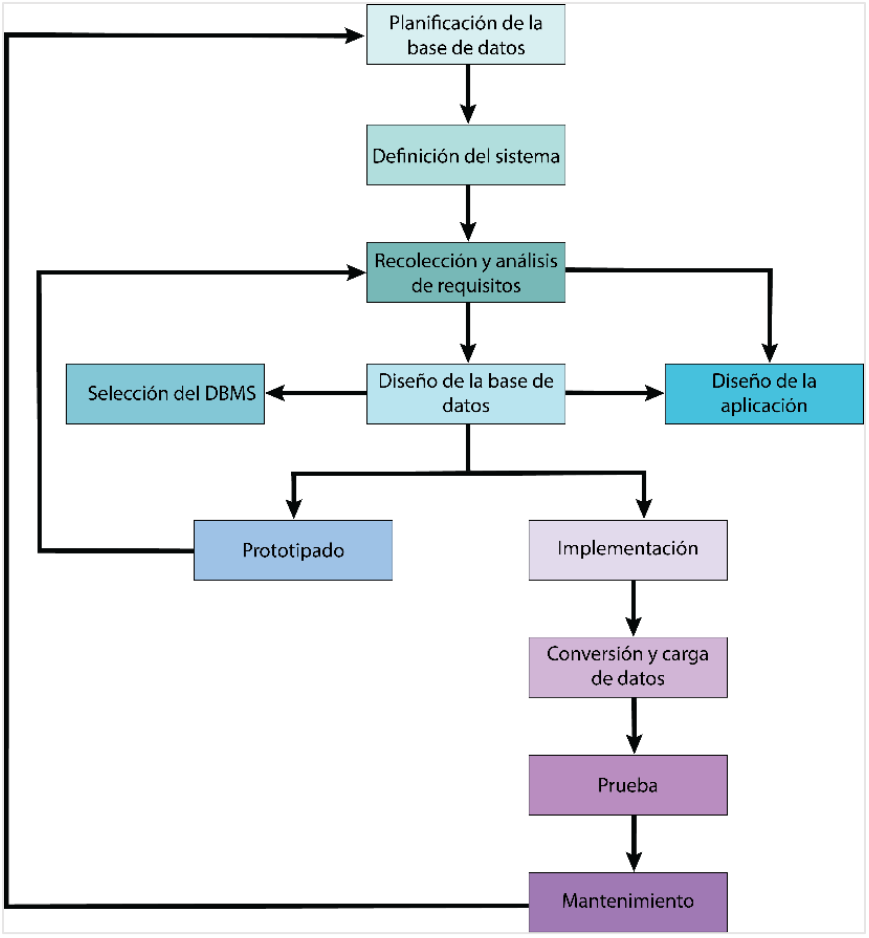



*Figura 11. Etapas del ciclo de vida del desarrollo de un sistema de base de datos. - Tomado y adaptado de* (Connolly & Begg, 2015).

## *Técnicas de determinación de hechos*

### Introducción

El desarrollo de un sistema de base de datos requiere de información sobre la organización, por lo que es necesario determinar hechos que permitirán darle estructura al sistema. (Mittal, 2012) explica que la determinación de hechos implica hacer una revisión detallada para obtener la mayor cantidad posible de información acerca del sistema actual, por lo que resume a la determinación de hechos como: *a)* la búsqueda de una estructura jerárquica y los procedimientos utilizados dentro de la organización y *b)* el desarrollo de diferentes perspectivas dado que las de muchos usuarios sobre su trabajo puede no ser la correcta. Considerando esto, es conveniente utilizar las técnicas apropiadas para construir un sistema de base de datos capaz de brindarle soluciones a la organización.

### Definición

La determinación de hechos se define como un procedimiento formal en el que se aplican técnicas para recolectar acontecimientos sobre las terminologías utilizadas dentro de la

organización, problemas que presenta el sistema actual, oportunidades que generaría el nuevo sistema, restricciones sobre los datos y los usuarios y los requisitos para el nuevo sistema de base de datos (Connolly & Begg, 2015).

Este proceso es lo que fundamenta el desarrollo de un sistema de base de datos, por lo que es particularmente importante aplicarlo durante las primeras etapas de planificación, definición del sistema y la recolección y análisis de requisitos. Sin embargo, como indica (Connolly & Begg, 2015), la determinación de hechos se aplica durante todo el ciclo de vida del sistema.

**Descripción de las técnicas**

Existen diferentes técnicas de determinación de hechos y el desarrollador de la base de datos puede elegir utilizar más de una de ellas. De hecho, esto lo más conveniente debido a que, como explica (Mittal, 2012), se asegura la obtención de información sobre las fortalezas y debilidades del sistema actual. A continuación se describirán las diferentes técnicas de determinación de hechos:

- **Revisión de documentación:** Consiste en analizar todos aquellos documentos de la organización que podrían aportar información valiosa, tales como manuales, descripción de puestos de trabajo, reportes, formularios y cualquier tipo de documento relacionado al sistema actual. Esta técnica requiere de un poco más de tiempo debido al gran volumen de documentos con el que generalmente cuentan las organizaciones, sin embargo (Sharmila & Umarani, 2011) explica que el analista debe revisar minuciosamente todos los documentos con el fin de discernir cuáles de ellos son necesarios para el desarrollo del sistema. Se considera la mejor fuente de información cuantitativa.
- **Entrevistas:** Son reuniones formales con los usuarios actuales del sistema actual, así como con usuarios potenciales del sistema nuevo y se considera la mejor fuente de información cualitativa (Mittal, 2012). Los usuarios pueden ser cualquier persona dentro de la jerarquía de la organización, dado que cada uno de ellos puede proveer información sobre el funcionamiento del sistema en uso, sin embargo, es conveniente planificar las entrevistas así como definir el propósito de esta. Una importante ventaja de esta técnica, como explica (Connolly & Begg, 2015), es que los entrevistados pueden sentir que forman parte del proceso de desarrollo del nuevo sistema; sin embargo, consume mucho tiempo y depende tanto de las habilidades de comunicación del entrevistador como de la disposición de las personas para ser entrevistados. Se pueden distinguir dos tipos de entrevistas:
  - ✓ **Entrevistas estructuradas:** El entrevistador tiene una serie de preguntas específicas para el entrevistado. Las respuestas a estas preguntas pueden ser de dos tipos: abiertas si el entrevistado contesta de la manera que considere

más apropiada, o cerradas si al entrevistado se le provee un conjunto de opciones para limitar sus respuestas.
    - ✓ **Entrevistas no estructuradas:** El entrevistador hace cualquier pregunta que considere relevante para obtener información sobre el sistema.
- **Observación:** Se analizan a las personas mientras realizan sus labores dentro de la organización, con el fin obtener una perspectiva sobre el funcionamiento del sistema actual. Una de las principales ventajas es que se pueden validar hechos y datos obtenidos con otras técnicas, sin embargo, en muchos casos las personas inconscientemente se desempeñan de forma diferente al saber que son observados (Connolly & Begg, 2015), lo cual representa una gran desventaja.
- **Investigación:** Consiste en investigar problemas similares que permitan obtener información sobre posibles soluciones para el sistema de la organización. Las principales fuentes de información para esta técnica son el Internet y libros de referencia, de esta manera es posible ahorrar tiempo dado que la solución al problema ya puede existir; sin embargo, se requieren de fuentes de información apropiadas y en el caso de que no exista una solución se desperdiciaría tiempo valioso (Connolly & Begg, 2015).
- **Cuestionarios:** Consiste en documentos con preguntas ordenadas y formales para recolectar datos y es útil en casos en los que no es posible realizar entrevistas o cuando el número de usuarios del sistema actual es demasiado grande (Mittal, 2012). Al igual que sucede con las entrevistas, las preguntas de los cuestionarios pueden ser de tres tipos:
    - ✓ **Abiertas:** El usuario responde las preguntas de la forma que considere más apropiada dentro de un determinado espacio del documento.
    - ✓ **Cerradas:** Al usuario se le presentan una serie de opciones para cada pregunta de las cuales debe escoger la respuesta correcta.
    - ✓ **Mixtas:** Se combinan preguntas tanto abiertas como cerradas en un solo cuestionario. (Connolly & Begg, 2015) explica que entre las principales ventajas de esta técnica están que no es costosa y las respuestas pueden ser fácilmente tabuladas para su análisis, sin embargo, se corre el riesgo de que algunas preguntas no sean respondidas y que no es posible reformular las preguntas, como sucedería con una entrevista.

**Ejemplos**

A continuación se presentará un caso para el cual se ejemplificarán diferentes técnicas de determinación de hechos que se pueden aplicar para el desarrollo de un nuevo sistema de base de datos.

El caso es el siguiente: Se quiere crear un sistema de base de datos para la administración de una pequeña escuela. El analista decide utilizar las siguientes técnicas:

- ✓ **Cuestionarios:** El analista decide entregarles cuestionarios al personal encargado de la administración para así obtener la mayor información posible sobre el sistema que utilizan actualmente. El cuestionario tiene una combinación de preguntas abiertas y cerradas. Algunas de las preguntas que contiene el cuestionario son:
  - ¿Cuántos estudiantes están matriculados actualmente?
  - ¿Qué datos solicitan sobre los estudiantes? ¿Qué datos de contacto solicitan a los padres de los estudiantes?
  - ¿Cuántos estudiantes nuevos se matriculan aproximadamente por año?
  - ¿Qué materias imparten en la escuela?
  - ¿Cuántos niveles de estudio ofrecen?
    a) Preescolar
    b) Primaria
    c) Secundaria
- ✓ **Observación:** El analista decide que para comprender mejor cómo funciona la administración de la escuela, es necesario observar el trabajo que realiza el personal de esta área. Aplica esta técnica durante cinco días y convenientemente, la escuela está realizando el proceso de cobro de las matrículas. Se da cuenta que la información de los pagos realizados por los padres de los estudiantes se archiva en carpetas, por lo que descubre una gran deficiencia en el sistema utilizado. De igual manera, empieza a descubrir otras deficiencias que considera podrían resolverse con un sistema de base de datos.
- ✓ **Entrevistas:** Dado que el analista confirmó la poca eficiencia del sistema utilizado por la administración de la escuela, decide hacerle entrevistas no estructuradas al personal de esta área para conocer las diferentes perspectivas de cada individuo. Algunas de las preguntas son:
  - ¿Cuál el rol que desempeña en su puesto de trabajo?
  - ¿Cuáles son los requisitos necesarios para que un estudiante pueda matricularse? ¿Qué documentos solicitan?
  - ¿Cuántos profesores laboran en la escuela y cuántas materias imparte cada uno? ¿Cuál es el mínimo y el máximo de materias que puede impartir un solo profesor?
  - ¿Cómo almacenan la información de los pagos de las matrículas?
  - ¿Qué deficiencias considera que presenta el sistema que utilizan actualmente? ¿Cuáles serían sus sugerencias para que el sistema actual sea más eficiente?

Con este caso y los ejemplos de las técnicas de determinación de hechos, se puede confirmar que estas técnicas son una herramienta imprescindible para el desarrollo de los sistemas de base de datos.

# Capítulo III: Modelaje conceptual de base de datos

## *Importancia de la modelización conceptual*

Dentro de la etapa de diseño de la base de datos, la modelización conceptual de la base de datos es un proceso indispensable para la identificación de los datos que se requieren en el sistema en desarrollo. Como explica (Connolly & Begg, 2015), este proceso es importante porque:

a) Los desarrolladores y diseñadores de la base de datos obtienen información acerca de la naturaleza de los datos y cómo estos son utilizados por la organización.
b) Se asegura el cumplimiento de los requisitos de los usuarios de la base de datos.
c) Permite la comunicación de información dado que se evitan ambigüedades sobre los datos.

La modelización conceptual de la base de datos se realiza a través de una técnica gráfica o diagrama que permite la representación de los datos, siendo el *modelo entidad-relación (ER)* la técnica más utilizada. El modelo ER fue propuesto en 1976 por Peter Chen, precisamente con el objetivo de brindar una perspectiva del mundo real de forma gráfica y fácil de comprender. A continuación se especifican las características básicas del modelo entidad-relación:

- ✓ Los datos se representan en términos de *entidades*, que son objetos del mundo real, las *relaciones* entre esos objetos y *atributos*, que son propiedades de los objetos. Todo este conjunto muestra un panorama de la realidad de la organización o de una parte de esta.
- ✓ Cuenta con capacidad de expresividad, porque representa la semántica del problema en estudio, y con capacidad de abstracción porque es independiente de los aspectos físicos del sistema vinculados al almacenamiento de la base de datos, así como de la manipulación de los datos, el sistema operativo y demás aplicaciones del sistema.

## *Componentes básicos de un modelo entidad-relación*

A continuación se describirán los componentes básicos que forman parte del modelo ER:

### Entidad

Una entidad corresponde a un objeto o cosa en el mundo real con existencia propia, ya sea física o abstracta (Cardona et al., 2014). Dicho esto, una entidad puede ser una persona, animal, lugar, concepto, actividad, acontecimiento o cualquier elemento que puede diferenciarse en la realidad; por este motivo, las entidades pueden identificarse fácilmente porque, por lo general, corresponden a sustantivos.

Otros aspectos importantes sobre las entidades es que estas tienen propiedades que las describen y que toman valores específicos. Se representan en el modelo ER a través de un rectángulo que dentro tiene un nombre único asociado al objeto que representa. En

la Figura 12 se muestra la representación gráfica de tres entidades diferentes, cada rectángulo corresponde a una entidad.


*Figura 12. Representación gráfica de diferentes entidades.*

**Tipos de entidades**

Las entidades pueden clasificarse en dos categorías y estas se describen a continuación:

- **Entidades fuertes:** Son entidades cuya existencia no depende de la existencia de otra entidad (Cardona et al., 2014). Este tipo de entidades cuentan con un atributo que es una llave primaria que permite la identificación de cada una de las ocurrencias de la entidad. Su representación en el modelo ER es la mostrada en la Figura 12.
- **Entidades débiles:** Son entidades cuya existencia depende de la existencia de otra entidad (Cardona et al., 2014). Como no cuentan con un atributo como llave primaria, la identificación de las ocurrencias de la entidad se hace a través de la relación que tiene con una entidad fuerte. Su representación en el modelo ER es a través de una rectángulo dentro de otro que contiene el nombre de la entidad, como se muestra en la Figura 13.


*Figura 13. Representación gráfica de una entidad débil.*

**Relación**

Una relación es el vínculo que existe entidades y estas relaciones corresponden a la realidad (Cardona et al., 2014). Se representa en el modelo ER a través de un rombo que contiene el nombre que describe a la relación o escribiendo el nombre de la relación sobre la línea que asocia a las entidades y generalmente, el nombre de la relación es un verbo en voz pasiva o activa (Coronel, Morris, & Rob, 2011). Cabe mencionar que entre dos entidades puede existir más de una relación. En la Figura 14 se muestran dos entidades y la relación que las asocia, utilizando las dos formas en que se puede representar gráficamente.

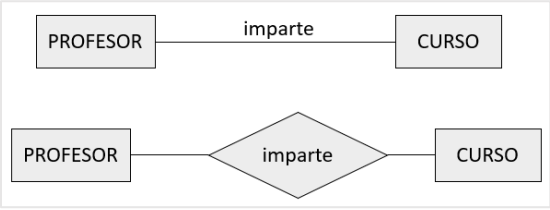



*Figura 14. Representación gráfica de una relación entre entidades.*

**Tipos de relación**

Los tipos de relación entre entidades corresponden al conjunto de asociaciones que existen entre los distintos tipos de entidades que se representan (Connolly & Begg, 2015). En este sentido, se habla de ocurrencia de una relación, la cual corresponde a las ocurrencias de la entidad específica con la cual se relaciona.

**Grado**

El grado de una relación se refiere a la cantidad de entidades asociadas a una determinada relación. Se pueden identificar los siguientes grados de relación, según lo explicado por (Coronel et al., 2011):

- **Unitaria o recursiva:** Es aquella relación en la que hay asociación con una sola entidad. En la Figura 15 se muestra un ejemplo de una relación recursiva.

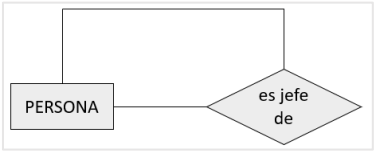



*Figura 15. Ejemplo de una relación recursiva.*

- **Binaria:** Es aquella relación en la que se asocian dos entidades. Es el grado de relación más común. En la Figura 16 se observa un ejemplo de una relación binaria.

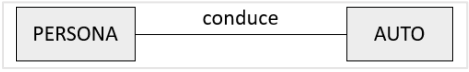



*Figura 16. Ejemplo de una relación binaria.*

- **Ternaria:** Es aquella relación en la que hay asociación entre tres entidades. En la Figura 17 se observa un ejemplo de una relación ternaria.

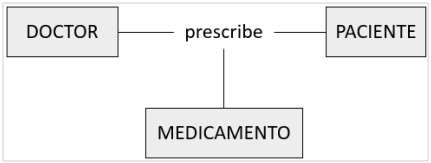



*Figura 17. Ejemplo de una relación ternaria. – Tomado y adaptado de* (Coronel et al., 2011).

- **Múltiple:** Es cuando existe más de una relación entre dos entidades. En la Figura 18 se muestra un ejemplo de una relación múltiple.

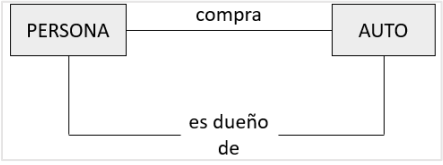



*Figura 18. Ejemplo de una relación múltiple.*

**Atributos**

Los atributos son las propiedades que describen a una entidad o relación y toman valores que describen cada ocurrencia de una determinada entidad (Connolly & Begg, 2015). Considerando esto, se habla del dominio de un atributo como el conjunto de valores que un determinado atributo puede tomar. En este sentido, (Coronel et al., 2011) indica que los atributos pueden ser *requeridos* u *opcionales*: si es un atributo requerido quiere decir que el atributo **debe** tomar un valor y no puede quedar vacío, mientras que si es un atributo opcional quiere decir que el atributo no requiere expresamente de un valor y por consiguiente, puede quedar vacío y tomar el valor nulo (null).

Los atributos pueden clasificarse de la siguiente manera indicada en el cuadro a continuación, según las explicaciones de (Coronel et al., 2011):

- **Simples o compuestos:** Los atributos simples son aquellos que no se subdividen, mientras que los atributos compuestos son aquellos que puede subdividirse en otros atributos. La *edad* es un ejemplo de un atributo simple, mientras que una *dirección* es un atributo compuesto porque puede subdividirse en *ciudad*, *barriada*, *calle* y *número de casa*.
- **Univaluados o multivaluados:** Un atributo univaluado es aquel que sólo puede tomar un solo valor, mientras que un atributo multivaluado es aquel que puede tomar múltiples valores. La *cédula* es un ejemplo de atributo univaluado porque una persona sólo puede tener un solo número de cédula posible, mientras que un *número de contacto* es un ejemplo de un atributo multivaluado dado que puede ser un número de casa o número de celular. Es importante mencionar que un atributo univaluado puede ser un atributo compuesto.
- **Derivados:** Se refiere a un atributo cuyo valor es el resultado del cálculo de otros atributos, es decir, su valor depende de otros atributos. Un ejemplo de atributo derivado puede ser la *edad*, ya que puede calcularse a partir de la fecha actual menos la fecha de nacimiento de una persona.

Como los atributos describen a las entidades, estos son la parte más importante de los datos almacenados en la base de datos. Su importancia también radica en que los atributos también pueden ser identificadores. Un *identificador o llave primaria* es uno o más atributos que identifican de forma única a la instancia de una entidad (Coronel et al., 2011), siendo la instancia una ocurrencia de la entidad.

Cuando dos o más atributos funcionan como identificador de una instancia se denomina *llave compuesta*, mientras que se denomina *llave foránea* al atributo que es llave primaria de otra entidad. (Coronel et al., 2011) indica que para que un atributo pueda considerarse como identificador debe cumplir con las siguientes condiciones: *a)* No deben existir dos instancias de la misma entidad con el mismo valor en el identificador y *b)* No deben tomar valores nulos. Por otro lado, cuando una entidad tiene atributos que pueden ser llaves,

estas se denominan *llaves candidatas* y entre esas se escoge la que será la llave primaria.

En el modelo ER, un atributo se representa con un óvalo que dentro contiene el nombre de este atributo y cuando se trata de una llave primaria, el nombre del atributo se subraya. En la Figura 19 se muestra un ejemplo de una relación entre dos entidades y sus atributos, los atributos subrayados corresponden a una llaves primarias.

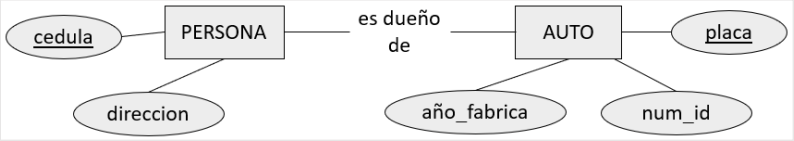



*Figura 19. Representación gráfica de los atributos de una entidad.*

**Atributos de las relaciones**

Como indica (Connolly & Begg, 2015), las relaciones entre entidades también pueden tener atributos. En este caso, se representa con el mismo símbolo utilizado para representar el atributo de una entidad, pero se utiliza una línea punteada asociada a la relación. En la Figura 20 se muestra un ejemplo de una relación entre dos entidades y el atributo asociado a esta relación.

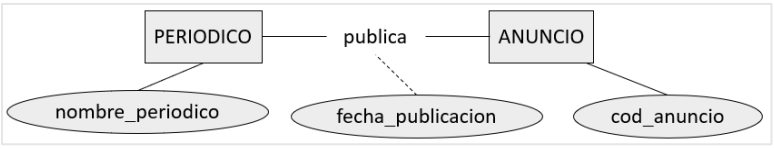



*Figura 20. Representación gráfica del atributo de una relación. – Adaptado de (Connolly & Begg, 2015).*

**Restricciones estructurales**

Las restricciones estructurales son aquellas que toman las entidades que forman parte de una relación y estas corresponden a las políticas o términos de la organización o del usuario (Connolly & Begg, 2015). Esto no es más que la cantidad de ocurrencias posibles de un tipo de entidad, asociadas a la relación. A continuación se describen los tres tipos de restricciones estructurales, basado en lo explicado por (Universidad Privada TELESUP, 2018):

- **Relaciones uno a uno (1:1):** La ocurrencia de una entidad se asocia con una sola ocurrencia de otra entidad, es decir, una sola entidad A se relaciona con una sola entidad B y viceversa. En la Figura 21 se muestra un ejemplo de relación uno a uno: un alumno realiza un proceso de matrícula, un proceso de matrícula es realizado por un alumno.

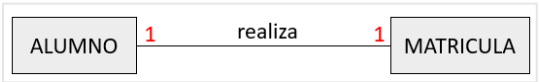



*Figura 21. Ejemplo de relación uno a uno.*

- **Relaciones uno a muchos (1:M):** La ocurrencia de una entidad se asocia con más de una ocurrencia de otra entidad, es decir, una sola entidad A se relaciona con muchas entidades B y una entidad B se relaciona con una sola entidad A. En la Figura 22 se muestra un ejemplo de relación uno a muchos: una persona puede poseer muchos autos, un auto es poseído por una persona.

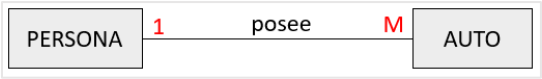



*Figura 22. Ejemplo de relación uno a muchos.*

- **Relaciones muchos a muchos (M:M):** Más de una ocurrencia de una entidad se asocia con más de una ocurrencia de otra entidad, es decir, una entidad A se relaciona con muchas entidades B y viceversa. En la Figura 23 se muestra un ejemplo de relación muchos a muchos: un archivo almacena muchos documentos, un documento es almacenado en muchos archivos.

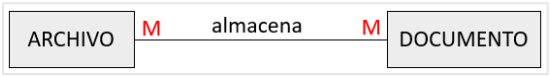



*Figura 23. Ejemplo de relación muchos a muchos.*

**Restricciones de cardinalidad**

La cardinalidad se define como el número máximo posible de ocurrencias que puede tener una entidad que pertenece a una determinada relación (Connolly & Begg, 2015). La cardinalidad puede ser de cuatro tipos, de acuerdo con lo descrito por (Cardona et al., 2014): cero a uno (0,1), uno a uno (1,1), cero a muchos (0,M) y uno a muchos (1,M). Este mismo autor agrega que se denomina *cardinalidad mínima* al número mínimo de relaciones en las que puede estar cada ocurrencia de la entidad y en este caso puede tomar el valor de cero (0) o uno (1); mientras que se denomina *cardinalidad máxima* al número máximo de relaciones en las que puede estar cada ocurrencia de la entidad y en este caso puede tomar el valor de uno (1) o muchos (M).

En la Figura 24 se muestra un ejemplo en donde la cardinalidad indica lo siguiente: una persona vive en mínimo una casa y máximo una casa, en una casa vive mínimo una persona y máximo muchas personas.

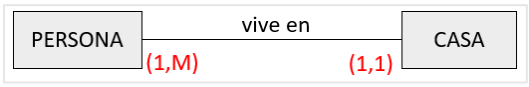



*Figura 24. Ejemplo de restricciones de cardinalidad.*

En la Figura 25 se muestra otro ejemplo en donde la cardinalidad indica lo siguiente: una persona posee mínimo cero autos y máximo muchos autos, un auto es poseído por mínimo una persona y máximo una persona.

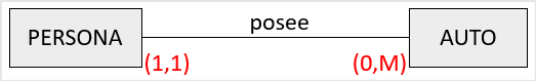



*Figura 25. Otro ejemplo de restricciones de cardinalidad.*

**Especialización/Generalización**

Para comprender estos conceptos es conveniente definir dos tipos especiales de entidades denominados superclases y subclases. Una *superclase* es una entidad tiene uno o más subgrupos de ocurrencias, mientras que una *subclase* es una entidad que se considera un subgrupo de ocurrencias de otra entidad (Connolly & Begg, 2015). Estos conceptos crean una jerarquía que da lugar a la *herencia de atributos*, dado que una subclase hereda los atributos de la superclase a la que pertenece, además de tener los atributos asociados a ese subgrupo en específico. Así mismo, una subclase puede contener una o más subclases que también heredan los atributos de la subclase y de la superclase.

Considerando lo anterior, se habla de dos procesos de abstracción, los cuales se describen a continuación de acuerdo con lo definido por (Connolly & Begg, 2015):

- **Especialización:** Consiste en identificar las características que no son comunes entre los miembros o subgrupos de una entidad, con el fin de maximizar las diferencias.
- **Generalización:** Consiste en identificar las características comunes entre entidades con el fin de minimizar las diferencias ellas.

Como explica (Camps Paré et al., 2005), la especialización/generalización permite demostrar que existe una entidad general (superclase) que puede especializarse en otras entidades (subclases). En este sentido, la superclase permite modelar las propiedades comunes de la entidad de una perspectiva general, mientras que las subclases permiten modelar las propiedades de las especializaciones. Es importante recordar que toda ocurrencia de una subclase también será una ocurrencia de la superclase.

En la Figura 26 se muestra el concepto general de especialización/generalización y en la Figura 27 se muestra un ejemplo en donde se aplica este concepto.

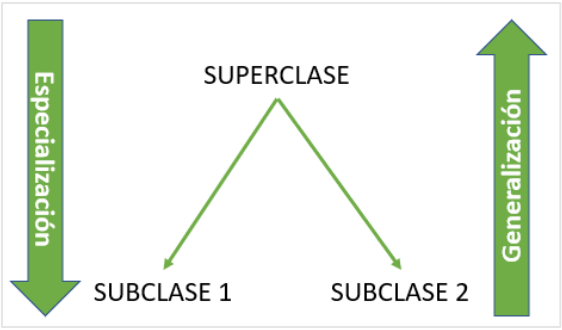



*Figura 26. Concepto general de especialización/ generalización. – Adaptado de (Cardona et al., 2014).*

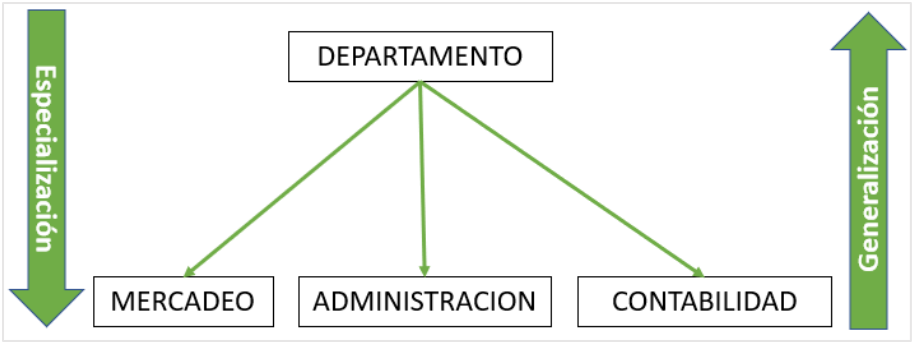
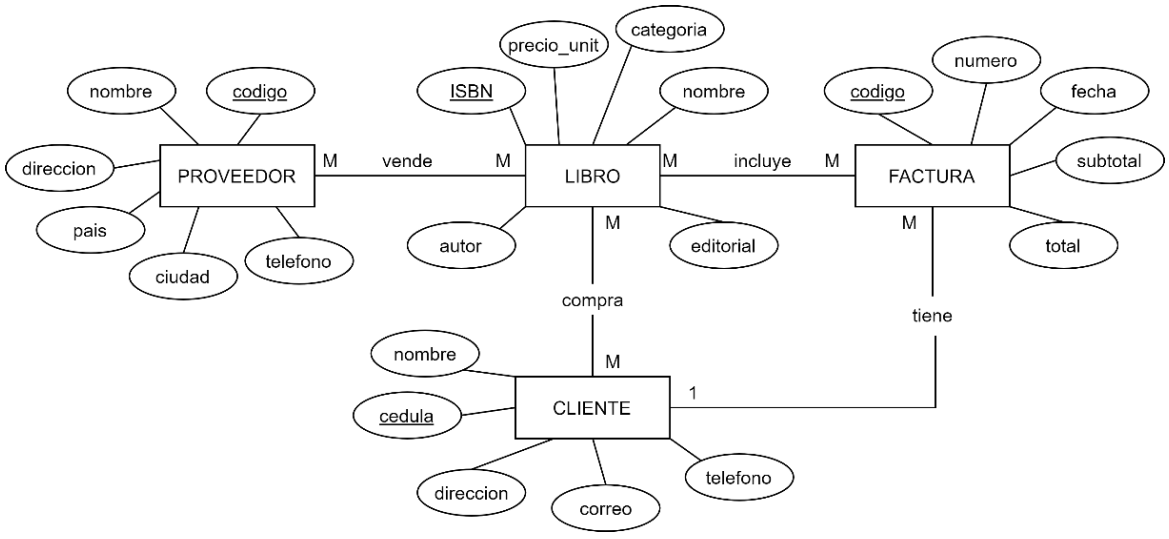



*Figura 27. Ejemplo donde se aplica el concepto de especialización/generalización. Elaboración propia.*

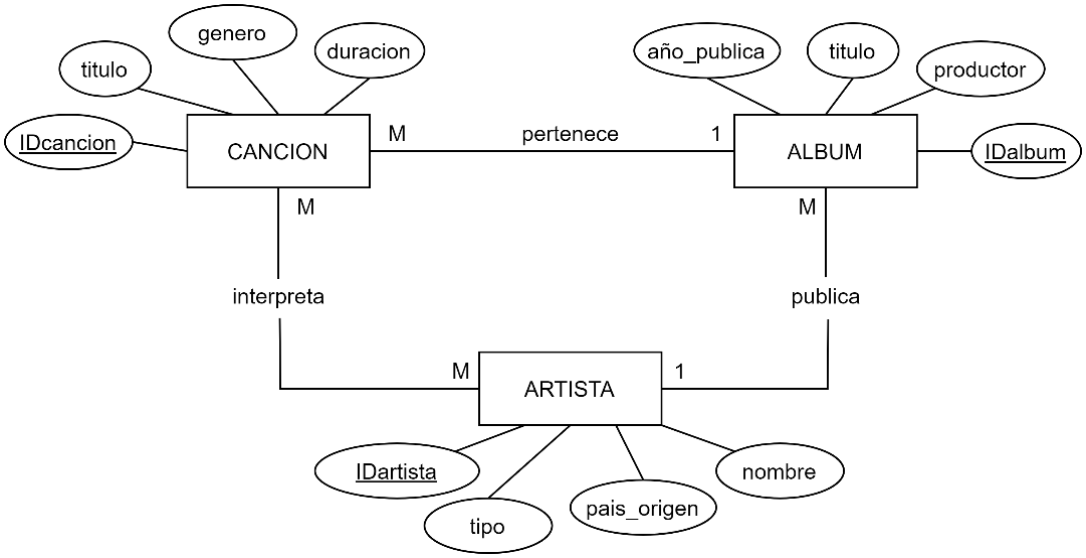
**Ejemplos**

- **Ejemplo #1 (adaptado de un problema propuesto por (Universidad Privada TELESUP, 2018)):** Una librería requiere la creación de una base de datos para un manejo más eficiente de la información. El diseñador de la base de datos cuenta con la siguiente información para el diseño del modelo ER del sistema solicitado:
  - Información sobre los proveedores: código, nombre, dirección, teléfono, ciudad, país.
  - Información sobre los clientes: cédula, nombre, dirección, teléfono.
  - Información sobre los libros: ISBN, nombre, precio unitario, autor, editorial, tipo de contenido.
  - Información sobre las facturas: código, número, fecha, subtotal, total.




- **Ejemplo #2:** Un individuo, con conocimientos de base de datos, desea crear una base de datos de las canciones que tiene almacenadas en su computadora. Para ello, cuenta con la siguiente información para creación del modelo ER:
  - Una canción se caracteriza por un título, género y duración.
  - Cada canción pertenece a un álbum y cada álbum cuenta con un título, artista, año de publicación y uno o varios productores.

- Cada artista cuenta con un país de origen, nombre y el tipo (si es una banda o un solista).
- Una canción puede ser interpretada por varios artistas, pero sólo aparece en un álbum, el cual es publicado por sólo un artista.
- Cada canción, álbum y artista deben tener un código.




**Conversión de un esquema ER a tablas**

Un modelo ER puede convertirse a un conjunto de tablas que representen el problema en estudio. De hecho, como explica (Sumathi & Esakkirajan, 2007), la implementación de una base de datos se hace a través del modelo relacional que se compone de tablas mapeadas a través de las llaves de las entidades, de allí que a este proceso se le denomine *mapeo*.

Para ello, se debe considerar lo siguiente, basado en lo descrito por (Lumbreras, n.d.):

a) Cada entidad corresponde a una tabla.
b) Los atributos de una entidad corresponden a las columnas de la tabla, cada columna lleva el nombre de un atributo.
c) Las filas (denominadas tuplas) de una tabla son las ocurrencias de cada entidad y estos toman valores para cada atributo.
d) El grado de la relación corresponde a la cantidad de columnas (atributos) de la tabla, mientras que la cardinalidad corresponde a la cantidad de tuplas de la tabla. (Sharma et al., 2010)

Sobre las llaves de las entidades, también hay algunas reglas que se deben considerar:

a) Si hay una relación 1:1, entonces la llave primaria de una de las entidades pasa a la otra como llave foránea.

b) Si hay una relación 1:M, entonces la llave primaria de la entidad en 1 pasa como llave foránea a la entidad en M.
c) Si hay una relación M:M, entonces se crea una nueva entidad y las llaves primarias pasan como llaves foráneas a la nueva entidad.

A continuación se considerará el Ejemplo #2, descrito anteriormente, para convertir el modelo ER al modelo relacional.

- Primero se escriben todas las entidades y sus correspondientes atributos. Se subrayan las llaves y se les colocan las letras PK (Primary Key) para señalar que son llaves primarias. A continuación se muestra este paso:

CANCION (IDcancion, titulo, duracion, genero)
PK

ALBUM (IDalbum, titulo, año_publica, productor)
PK

ARTISTA (IDartista, nombre, tipo, pais_origen)
PK

- Luego se siguen las reglas de mapeo para las llaves. Para el ejemplo en cuestión:
  - ✓ Hay una relación de 1:M entre las entidades CANCION y ALBUM, por lo que la llave primaria de la entidad que está en 1 (ALBUM) pasa como llave foránea a la entidad que está en M (CANCION). Sucede lo mismo con la relación entre las entidades ARTISTA y ALBUM.
  - ✓ Hay una relación de M:M entre las entidades CANCION y ARTISTA, por lo que se crea una nueva tabla que denominaremos CANCION_ARTISTA. Las llaves primarias de ambas entidades pasan como llaves foráneas a la nueva tabla.

  A las llaves foráneas se les colocan las letras FK (Foreign Key). A continuación se muestra este paso:

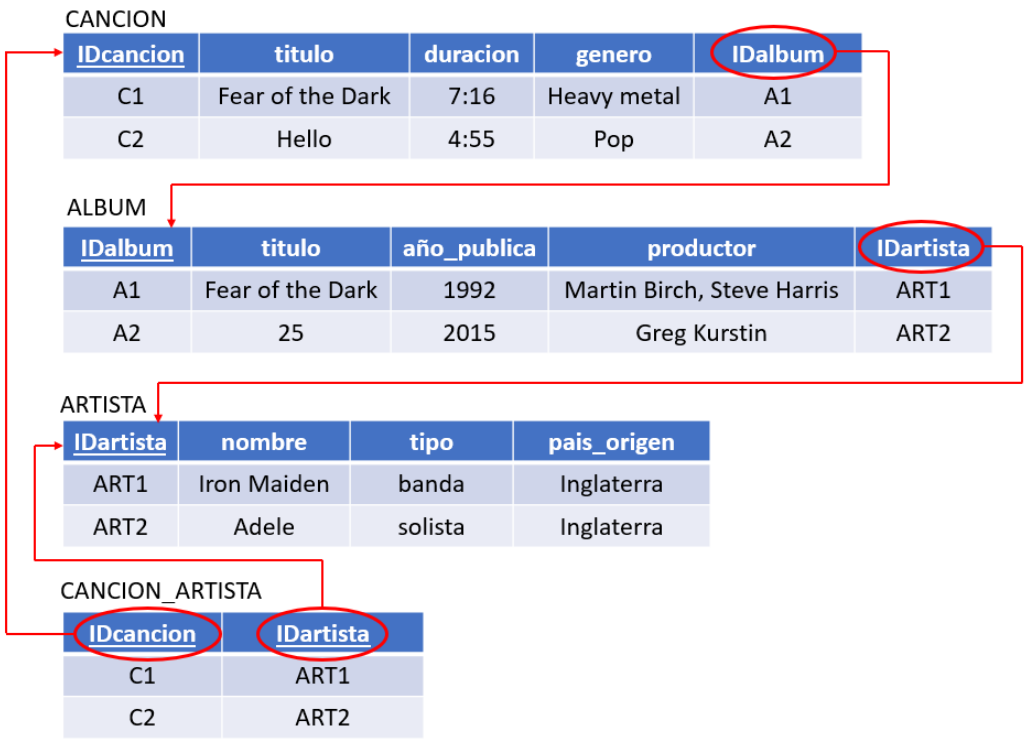
CANCION (IDcancion, titulo, duracion, genero, IDalbum)
PK FK

ALBUM (IDalbum, titulo, año_publica, productor, IDartista)
PK FK

ARTISTA (IDartista, nombre, tipo, pais_origen)
PK

CANCION_ARTISTA (IDcancion, IDartista)
FK FK

- Finalmente, se crean la tablas con sus correspondientes atributos. Para el ejemplo en cuestión, a continuación se muestran cómo quedarían las tablas y se han colocado valores a los atributos para fines demostrativos:




Como se observa, las tablas son mapeadas o relacionadas a través de las llaves y esto es lo que le da estructura a la base de datos.

# Capítulo IV: Fundamentos del modelo relacional

## *Definición del modelo de datos relacionales*

El modelo de datos relacionales fue introducido por Edgar F. Codd en el año 1970. Este modelo se compone de un conjunto de estructuras básicas denominadas *tablas*, que forman relaciones entre los datos almacenados en ellas. El modelo de datos relacional cuenta con diferentes objetivos, los cuales se especifican a continuación basado en lo descrito por (Camps Paré et al., 2005):

a) **Independencia lógica de los datos:** El usuario interpreta a las relaciones como la estructura lógica que conforma a la base de datos.
b) **Independencia física de los datos:** El usuario desconoce la estructura de datos utilizada para almacenar los datos en la base de datos, pues la implementación física está oculta. De igual manera, esto no influye en la estructura lógica de los datos.
c) **Simplicidad y uniformidad:** Los datos se representan de una sola forma a través de valores explícitos que permiten formar las relaciones de la base de datos. Para esto se requiere que los valores sean atómicos, es decir que no puedan descomponerse en partes más pequeñas. Como indica (Garcia-Molina, Ullman, & Widom, 2009), los valores deben ser de un tipo de dato elemental como un entero o una cadena, no estructuras como listas o arreglos.

### Definición de conceptos básicos del modelo relacional

Para estudiar el modelo relacional es primordial definir conceptos asociados a este. (Marqués, 2011) define estos conceptos de la siguiente manera:

- **Relación:** Es una tabla compuesta por filas y columnas. Las filas corresponden a registros individuales, mientras que las columnas son los campos de esos registros.
- **Atributo:** Es el nombre de una columna en una relación. Son los elementos de una relación.
- **Dominio:** Es el conjunto de valores permitidos para un atributo determinado. Cada atributo se define sobre un dominio, pero sobre un dominio se pueden definir varios atributos.
- **Tupla:** Es una fila en una relación y un conjunto de tuplas conforman el elemento de una relación.
- **Grado de una relación:** Cantidad de atributos que contiene la relación. El grado de una relación por lo general es constante.
- **Cardinalidad de una relación:** Cantidad de tuplas que contiene la relación. La cardinalidad de una relación puede variar frecuentemente, debido a la actualización de tuplas.

En la Figura 28 se muestra un ejemplo, la relación ALBUM, para demostrar los conceptos definidos anteriormente. Además, se puede observar que el grado de la relación es de cinco porque tiene cinco atributos. En cuanto al dominio, se puede tomar como ejemplo al atributo *año_publica*, el cual corresponde al año de publicación del álbum y debe tener 4 caracteres de valor numérico.


*Figura 28. Demostración de los conceptos básicos del modelo relacional. Elaboración propia.*

Es importante mencionar que las relaciones cuentan con las siguientes propiedades, descritas por (Marqués, 2011):

a) Cada relación tiene un nombre único.
b) Para cada tupla, cada atributo toma un solo valor.
c) Cada atributo tiene un nombre único, no hay dos atributos con el mismo nombre.
d) Cada tupla es única, no hay duplicados.
e) Los atributos y las tuplas no están ordenados, su orden no importa.

**Restricciones de integridad del modelo**

Para que una base de datos represente a la realidad a través de los datos que almacena, es primordial aplicar restricciones de integridad. La integridad es precisamente aquella característica de los datos que permite que estos representen adecuadamente al mundo real.

- **Definiciones de claves:** Una *clave* o *llave* es uno o más atributos que identifican a una tupla en una relación. A continuación se describen las restricciones para los diferentes tipos de llaves, basado en lo explicado por (Mannino, 2007):
  - **Llave candidata:** Es un atributo o combinación de atributos que pueden servir como llave primaria en una relación. Para que un atributo sea una llave candidata, el valor de este para cada tupla debe ser único.
  - **Llave primaria (PK):** Es uno o más atributos que se escogen a partir de las llaves candidatas e identifican de forma única a cada tupla de la relación. Cada relación debe tener una llave primaria única y esta no puede contener valores nulos.
  - **Llave foránea (FK):** Es uno o más atributos que hacen referencia a una tupla en otra relación. Las llaves foráneas permiten unir tablas, por lo que el valor y el tipo de dato de la llave foránea debe coincidir con el valor de la llave primaria con la cual está asociada.

- **Valores nulos:** Cuando un atributo en una tupla tiene un valor *nulo*, significa que el valor es desconocido. Un valor nulo es diferente al valor cero o a la cadena vacía, pues estos tienen significado. Como explica (Rosas Hernandez, 2018), el valor nulo indica ausencia de información para una determinada tupla, ya sea porque no se conocía el valor del atributo cuando la tupla se insertó o porque el atributo no tiene sentido para dicha tupla. Por otro lado, como explica (Marqués, 2011), algunas veces el uso de valores nulos puede causar problemas de implementación, así como algunos DBMS relacionales no soportan el uso de valores nulos.
- **Dominios:** El dominio permite definir las características de los atributos de una relación. Un dominio especifica los valores válidos para un atributo definido sobre ese dominio, además de que cada dominio cuenta con su propio nombre, tipo de dato y un tamaño lógico que el usuario determina (no es el tamaño de almacenamiento interno) (Sumathi & Esakkirajan, 2007). Como ejemplos, para el atributo *edad* el dominio es un número entero entre 18 y 62, mientras que para el atributo *nombre_empleado* el dominio es un cadena de caracteres.
- **Otros:** Existen otras reglas de integridad que son importantes y por lo tanto, es necesario describirlas. A continuación se definen estas reglas, basado en (Camps Paré et al., 2005):
  - **Regla de integridad de entidad:** La llave primaria de una relación no puede contener valores nulos, dado que si una llave primaria tuviera un valor nulo entonces algunas tuplas no podrían identificarse al no contener un valor único.
  - **Regla de integridad referencial:** La llave foránea debe contener un valor que coincida con el valor que contiene la llave primaria con la cual se asocia o un valor nulo. De otro modo, no existiría un vínculo entre las relaciones.

**Operaciones en el modelo relacional: álgebra relacional**

El álgebra relacional es un lenguaje procedimental teórico que consiste en un conjunto de operaciones que toman como entrada a una o más relaciones y produce una nueva relación como salida (Silberschatz et al., 2011). Al utilizarse para realizar consultas en la base de datos, la salida de la operación no genera cambios en la/las relaciones que se utilizan como entrada. A continuación se describen las operaciones que forman parte del álgebra relacional:

- **Selección (σ):** Utiliza una sola relación R para tomar aquellas tuplas que satisfagan la condición especificada (predicado).

$$\sigma_{predicado}(R)$$

  - **Ejemplo:** Mostrar el ID, nombre y salario de todos los empleados que tengan un salario mayor a 1500.

empleado

| IDempleado | nombre | salario |
|---|---|---|
| 001 | John Smith | 1600 |
| 002 | Ana Rodríguez | 500 |
| 003 | José Pérez | 700 |
| 004 | Camila Gómez | 2300 |
| 005 | Mario Brown | 550 |

La consulta utilizando el álgebra relacional sería:

$$\sigma_{salario>1500}(empleado)$$

| IDempleado | nombre | salario |
|---|---|---|
| 001 | John Smith | 1600 |
| 004 | Camila Gómez | 2300 |

- **Proyección (Π):** Utiliza una sola relación R para tomar algunos atributos y definir otra relación. La nueva relación contiene los valores de los atributos especificados y no muestra tuplas duplicadas.

$$\Pi_{a_1 \ldots, a_n}(R)$$

  - **Ejemplo:** Mostrar el ID y el nombre de todos los empleados.

empleado

| IDempleado | nombre | salario |
|---|---|---|
| 001 | John Smith | 1600 |
| 002 | Ana Rodríguez | 500 |
| 003 | José Pérez | 700 |
| 004 | Camila Gómez | 2300 |
| 005 | Mario Brown | 550 |

La consulta utilizando el álgebra relacional sería:

$$\Pi_{IDempleado,nombre}(empleado)$$

| IDempleado | nombre |
|---|---|
| 001 | John Smith |
| 002 | Ana Rodríguez |
| 003 | José Pérez |
| 004 | Camila Gómez |
| 005 | Mario Brown |

- **Unión (∪):** Utiliza dos relaciones R y S para definir otra relación que contiene todas las tuplas de ambas relaciones, sin incluir duplicados. Para aplicar esta operación ambas relaciones deben tener la misma cantidad de atributos y estos deben tener el mismo dominio.

$$R \cup S$$

  - **Ejemplo:** Mostrar la unión entre la relación *depto_finanzas* y la relación *depto_rrhh*, considerando el ID y el nombre de todos los empleados.

depto_finanzas

| IDempleado | nombre | salario |
|---|---|---|
| 001 | John Smith | 1600 |
| 002 | Ana Rodríguez | 500 |
| 003 | José Pérez | 700 |
| 004 | Camila Gómez | 2300 |
| 005 | Mario Brown | 550 |

depto_rrhh

| IDempleado | nombre | salario |
|---|---|---|
| 006 | Ricardo Pérez | 2300 |
| 007 | María López | 1600 |
| 008 | Astrid Jiménez | 500 |
| 009 | Elena Gutiérrez | 750 |
| 010 | Mario Johnson | 250 |

La consulta utilizando el álgebra relacional sería:

$$(\Pi_{IDempleado,nombre}(depto\_finanzas)) \cup (\Pi_{IDempleado,nombre}(depto\_rrhh))$$

| IDempleado | nombre |
|---|---|
| 001 | John Smith |
| 002 | Ana Rodríguez |
| 003 | José Pérez |
| 004 | Camila Gómez |
| 005 | Mario Brown |
| 006 | Ricardo Pérez |
| 007 | María López |
| 008 | Astrid Jiménez |
| 009 | Elena Gutiérrez |
| 010 | Mario Johnson |

- **Diferencia de conjuntos (−):** Utiliza dos relaciones R y S para producir una nueva relación que muestra las tuplas que están en una relación pero no en la otra.

$$R - S$$

  - **Ejemplo:** Obtener la diferencia entre la relación *fac_sistemas* y la relación *fac_electrica*.

fac_sistemas

| IDprof | nombre | curso |
|---|---|---|
| 001 | John Smith | BDI |
| 002 | Ana Rodríguez | Robótica |
| 003 | José Pérez | Cálculo |

fac_electrica

| IDprof | nombre | curso |
|---|---|---|
| 004 | Camila Gómez | Circuitos |
| 005 | Mario Brown | Termodinámica |
| 003 | José Pérez | Cálculo |

La consulta utilizando el álgebra relacional sería:

$$(\Pi_{nombre,curso}(fac\_sistemas)) - (\Pi_{nombre,curso}(fac\_electrica))$$

| nombre | curso |
|---|---|
| John Smith | BDI |
| Ana Rodríguez | Robótica |

- **Producto cartesiano (×):** Utiliza dos relaciones R y S para generar una nueva relación que combina las tuplas de ambas relaciones. Además, la nueva relación contiene todos los atributos de ambas relaciones.

$$R \times S$$

  - **Ejemplo:** Mostrar el producto cartesiano de la relación *cliente* y la relación *factura*.

cliente

| IDcliente | nombre |
|---|---|
| 001 | John Smith |
| 002 | Ana Rodríguez |
| 003 | José Pérez |
| 004 | Camila Gómez |

factura

| IDfactura | total | cliente |
|---|---|---|
| 123-456 | 600.90 | 001 |
| 789-101 | 35.50 | 003 |
| 987-654 | 44.00 | 004 |
| 654-123 | 135.50 | 001 |
| 321-897 | 200.50 | 003 |
| 101-789 | 120.35 | 002 |

La consulta utilizando el álgebra relacional sería:
*cliente* × *factura*

| IDcliente | nombre | IDfactura | total | cliente |
|---|---|---|---|---|
| 001 | John Smith | 123-456 | 600.90 | 001 |
| 002 | Ana Rodríguez | 101-789 | 120.35 | 002 |
| 003 | José Pérez | 321-897 | 200.50 | 003 |
| 004 | Camila Gómez | 987-654 | 44.00 | 004 |
| 001 | John Smith | 654-123 | 135.50 | 001 |
| 003 | José Pérez | 789-101 | 35.50 | 003 |

- **Renombramiento (ρ):** Permite cambiar el nombre de una relación o de los atributos de una relación resultante.

$$\rho_s(E) \quad \text{o} \quad \rho_{s(a_1,a_2,\ldots,a_n)}(E)$$

  - **Ejemplo:** Renombrar las siguientes columnas de la relación *fac_electrica*, la columna *IDprof* renombrarla como *codigo* y la columna *nombre* renombrarla como *profesor*.

fac_electrica

| IDprof | nombre | curso |
|---|---|---|
| 004 | Camila Gómez | Circuitos |
| 005 | Mario Brown | Termodinámica |
| 003 | José Pérez | Cálculo |

La consulta utilizando el álgebra relacional sería:
$\rho_{s(codigo,profesor,curso)}(fac\_electrica)$

| codigo | profesor | curso |
|---|---|---|
| 004 | Camila Gómez | Circuitos |
| 005 | Mario Brown | Termodinámica |
| 003 | José Pérez | Cálculo |

- **Intersección (∩):** Utiliza dos relaciones R y S para generar una nueva relación que contiene las tuplas que pertenecen a ambas relaciones.

$$R \cap S$$

  - **Ejemplo:** Mostrar la intersección entre la relación *fac_sistemas* y la relación *fac_electrica*.

fac_sistemas

| IDprof | nombre | curso |
|---|---|---|
| 001 | John Smith | BDI |
| 002 | Ana Rodríguez | Robótica |
| 003 | José Pérez | Cálculo |

fac_electrica

| IDprof | nombre | curso |
|---|---|---|
| 004 | Camila Gómez | Circuitos |
| 005 | Mario Brown | Termodinámica |
| 003 | José Pérez | Cálculo |

La consulta utilizando el álgebra relacional sería:

*fac_sistemas* ∩ *fac_electrica*

| IDprof | nombre | curso |
|---|---|---|
| 003 | José Pérez | Cálculo |

- **Reunión natural (⋈):** Utiliza dos relaciones R y S para generar una nueva relación que contiene las tuplas que coinciden con los atributos en común de ambas relaciones.

$$R \bowtie S$$

  - **Ejemplo:** Mostrar el resultado entre la relación *cliente* y *factura*, considerando que tienen en común el ID del cliente.

cliente

| IDcliente | nombre |
|---|---|
| 001 | John Smith |
| 002 | Ana Rodríguez |
| 003 | Camila Gómez |

factura

| IDfactura | total | IDcliente |
|---|---|---|
| 123-456 | 600.90 | 001 |
| 789-101 | 35.50 | 003 |
| 987-654 | 44.00 | 004 |
| 654-123 | 135.50 | 002 |
| 321-897 | 200.50 | 003 |

La consulta utilizando el álgebra relacional sería:

*cliente* ⋈ *factura*

| nombre | IDcliente | IDfactura | total |
|---|---|---|---|
| John Smith | 001 | 123-456 | 600.90 |
| Ana Rodríguez | 002 | 654-123 | 135.50 |
| Camila Gómez | 003 | 789-101 | 35.50 |
| Camila Gómez | 003 | 321-897 | 200.50 |

- **División (/):** Utiliza dos relaciones R y S para generar una nueva relación que contiene todas las tuplas de R que coinciden con las tuplas de S. Para definir la nueva relación, la operación se basa en los atributos por lo que la relación R debe ser de grado *n*+*m* y la relación S de grado *m*, el resultado es una relación de grado *n*.

$$R \,/\, S$$

➢ **Ejemplo:** Aplicar la operación división a la relación *curso* entre la relación *estudiante*.

curso

| IDcurso | IDestudiante |
|---|---|
| CALC1 | 8-123-456 |
| BD1 | 3-210-2110 |
| ROBOT | 8-789-1011 |
| CALC1 | 3-210-2110 |
| BD1 | 8-123-456 |
| BD1 | 8-789-1011 |

estudiante

| IDestudiante |
|---|
| 8-123-456 |
| 3-210-2110 |
| 8-789-1011 |

La consulta utilizando el álgebra relacional sería:

*curso / estudiante*

| IDcurso |
|---|
| BD1 |

**Álgebra relacional extendida**

El álgebra relacional extendida permite que se realicen operaciones aritméticas en combinación con las operaciones básicas del álgebra relacional. A continuación se describen las operaciones que forman parte del álgebra relacional extendida:

- **Proyección generalizada:** Funciona igual que la operación proyección, con la diferencia de que se pueden utilizar funciones aritméticas. En la siguiente función $F_n$ son expresiones aritméticas y $E$ es una expresión del álgebra relacional.

$$\Pi_{F_1,F_2\ldots,F_n}(E)$$

➢ **Ejemplo:** Mostrar cuánto es el salario total de cada empleado y renombrar la columna del resultado como *salario_total*.

La consulta utilizando el álgebra relacional sería:

$$\Pi_{nombre,(salario\_neto+bono)as\ salario\_total}(planilla)$$

planilla

| nombre | salario_neto | bono |
|---|---|---|
| John Smith | 1600.00 | 45.00 |
| Ana Rodríguez | 500.00 | 20.00 |
| José Pérez | 700.00 | 20.00 |
| Camila Gómez | 2300.00 | 15.00 |

| nombre | salario_total |
|---|---|
| John Smith | 1645.00 |
| Ana Rodríguez | 520.00 |
| José Pérez | 720.00 |
| Camila Gómez | 2450.00 |

- **Funciones de agregación:** Son funciones que toman como entrada un conjunto de valores y da como resultado un solo valor. Las funciones de agregación más utilizadas son:
  - **a) sum:** Devuelve la suma de los valores asociados a un atributo.
  - **b) avg:** Devuelve el promedio de los valores asociados a un atributo.
  - **c) count:** Devuelve la cantidad de valores que hay.
  - **d) min:** Devuelve el valor más pequeño en el atributo.
  - **e) max:** Devuelve el valor más grande en el atributo.

En la siguiente función, $G_n$ corresponde al conjunto de atributos o relación sobre los que se realiza la operación, $F_n$ corresponde a alguna de las funciones de agregación y $A_n$ es el nombre del atributo.

$$_{G_1,G_2,\ldots,G_n}g_{F_1(A_1),\ldots F_n(A_n)}(E)$$

- **Ejemplo:** Mostrar la cantidad que debe desembolsar cada departamento para pagarle el salario a sus empleados.

planilla

| depto | nombre | salario |
|---|---|---|
| FIN | John Smith | 1600 |
| FIN | Ana Rodríguez | 500 |
| IT | José Pérez | 700 |
| RRHH | Camila Gómez | 2300 |
| RRHH | Mario Brown | 550 |

La consulta utilizando el álgebra relacional sería:

$$depto\ g_{sum(salario)as\ desembolso}(depto\_finanzas)$$

| depto | desembolso |
|---|---|
| FIN | 2100 |
| RRHH | 2850 |
| IT | 700 |

- **Reunión externa:** Es parecido a la operación unión, pero además el resultado es una relación que incluye las tuplas de una relación R que no coinciden con los atributos de una relación S. Aquellos valores que no coinciden los coloca como *null*.
  - **Ejemplo:** Aplicar la operación de reunión externa a las relaciones *cliente* y *factura*.

cliente

| IDcliente | nombre |
|---|---|
| 001 | John Smith |
| 002 | Ana Rodríguez |
| 003 | Camila Gómez |

factura

| IDfactura | total | IDcliente |
|---|---|---|
| 123-456 | 600.90 | 001 |
| 789-101 | 35.50 | 002 |
| 987-654 | 44.00 | 004 |
| 654-123 | 135.50 | 001 |
| 321-897 | 200.50 | 002 |

El resultado al aplicar esta operación sería:

| nombre | IDcliente | IDfactura | total |
|---|---|---|---|
| John Smith | 001 | 123-456 | 600.90 |
| John Smith | 001 | 654-123 | 135.50 |
| Ana Rodríguez | 002 | 789-101 | 35.50 |
| Ana Rodríguez | 002 | 321-897 | 200.50 |
| Camila Gómez | 003 | null | null |
| null | 004 | 987-654 | 44.00 |

## *Prácticas*

# Capítulo V: Normalización

## *Justificación*

La normalización es una técnica de diseño de base de datos, que permite identificar las relaciones entre los atributos que son óptimas para la representación de los datos de la organización. Las reglas de normalización se aplican al modelo relacional con el fin de eliminar la redundancia y asegurar la integridad de los datos.

El conocimiento adquirido hasta este punto, conforma los fundamentos para la creación de una base de datos y es posible crear un modelo para su representación. Sin embargo, (Bender, Deco, González Sanabria, & Ponce Gallegos, 2014) indica que es posible que el modelo diseñado genere problemas, por lo que a través de la normalización se aplica un procedimiento formal para identificar y corregir anomalías en la estructura que se ha diseñado.

La redundancia de datos significa que se puede encontrar un mismo dato en más de una ubicación dentro de las tablas de la base de datos. Esto a su vez, conlleva problemas que se describirán a continuación, basado en lo explicado por (Mannino, 2007):

- **Anomalía de inserción:** Ocurre cuando al insertar un registro, se requiere de la adición de datos no relacionados al dicho registro.
- **Anomalía de eliminación:** Ocurre cuando al eliminar un registro, inadvertidamente se eliminan otros datos no relacionados.
- **Anomalía de actualización:** Ocurre cuando al modificar un registro, es necesario modificar otros registros.

Una base de datos bien diseñada cuenta con relaciones en su estructura que eliminan la redundancia y por consiguiente, las anomalías descritas anteriormente.

## *Concepto de dependencias funcionales*

Una dependencia funcional (DF) es un tipo de restricción de integridad que describe la relación entre los atributos de una tabla. La representación de una dependencia funcional entre dos atributos es la siguiente, como indica (Elmasri & Navathe, 2011): $A \rightarrow B$, donde $A$ (lado izquierdo o LHS) y $B$ (lado derecho o RHS) son conjuntos de atributos que son subconjuntos de una relación $R$. $A \rightarrow B$ significa que $B$ es funcionalmente dependiente de $A$, $A$ determina a $B$ si existe al menos un valor de $B$ para cada valor de $A$. El lado izquierdo de la dependencia funcional se denomina **determinante**.

### Formas normales y axiomas

Las formas normales permiten la eliminación de las redundancias cuando se hace el proceso de normalización. Una forma normal (NF) es definida por (Mannino, 2007) como "*una regla sobre las dependencias permisibles*". Cada una de las formas normales se enfoca en resolver redundancias específicas y estas formas normales son: las tres primeras denominadas primera forma normal (1NF), segunda forma normal (2NF) y

tercera forma normal (3NF), la forma normal de Boyce-Codd (BCNF) que es una versión más estricta de la 3NF y las dos últimas denominadas cuarta forma normal (4NF) y quinta forma normal (5NF).

Los axiomas, formalmente denominados *axiomas de Armstrong*, son reglas de inferencia que facilitan la deducción de las dependencias funcionales dentro de un conjunto de estas (Sharma et al., 2010). A continuación se describen estos axiomas, los tres últimos axiomas derivan de los tres primeros y son considerados reglas adicionales:

| Axioma | Definición |
|---|---|
| Reflexividad | Si $B$ es un subconjunto de $A$, entonces $A \rightarrow B$. |
| Aumentatividad | Si $A \rightarrow B$ y $C$ es otro atributo, entonces $AC \rightarrow BC$. |
| Transitividad | Si $A \rightarrow B$ y $B \rightarrow C$, entonces $A \rightarrow C$. |
| Auto-determinación | $A \rightarrow A$ |
| Descomposición | Si $A \rightarrow BC$, entonces $A \rightarrow B$ y $A \rightarrow C$. |
| Unión | Si $A \rightarrow B$ y $A \rightarrow C$, entonces $A \rightarrow BC$. |
| Composición | Si $A \rightarrow B$ y $C \rightarrow D$, entonces $AC \rightarrow BD$. |

**Cálculo de la clausura de atributos**

Para el cálculo del cierre de atributos se tomará una relación $R$ con un conjunto de atributos, se calculará el cierre $(A)$. (Sharma et al., 2010) explica el proceso de la siguiente manera:

1. Primero establecer el cierre como $(A) = A$.
2. Para cada DF, si $A \rightarrow B$, entonces agregar $B$ al cierre $(A)$. De esta manera queda el cierre $(A) \cup B$.
3. Para cada subconjunto de $A$, hacer $C$ un subconjunto de $A$ para obtener una DF trivial $A \rightarrow C$. Si $C \rightarrow D$ y $D$ no es subconjunto de $A$, entonces agregar $D$ al cierre $(A)$.
4. Repetir el paso 3 hasta que no haya más conjuntos de atributos que agregar al cierre $(A)$.

**Cierre de un conjunto de dependencias funcionales**

El cierre de un conjunto de DFs se refiere a todas las dependencias funcionales que están implícitas para un conjunto dado de una DF determinada (Sharma et al., 2010), es decir, todas las dependencias funcionales que pueden derivar de esa DF. Para un conjunto de dependencias funcionales denominado $F$, su cierre de conjunto de dependencias funcionales se denota como $F^+$.

Es importante recordar que las dependencias funcionales están implícitas *lógicamente*, para lo cual se utilizan los axiomas de Armstrong como una técnica sencilla de inferencia.

**Recubrimiento minimal**

El recubrimiento minimal, denotado como $F_c$ se refiere a un conjunto de dependencias funcionales de modo que el cierre de dependencias funcionales y el conjunto $F$ de DFs dado para una relación $R$ son equivalentes (Sharma et al., 2010). Esto quiere decir que $F^+ = F_c^+$.

Se debe considerar lo siguiente:

- Cada dependencia funcional en el recubrimiento minimal tiene un solo atributo en el lado derecho.
- Reducir los atributos en el lado izquierdo de cada dependencia funcional, genera cambios en el cierre del recubrimiento minimal.
- Las dependencias funcionales no son redundantes, por lo que la eliminación de alguna de las dependencias en el recubrimiento minimal hará que el cierre de este cambie.

**Determinación de las claves**

Las dependencias funcionales también pueden utilizarse para la determinación de claves, o bien para determinar posibles claves. De acuerdo con (Mannino, 2007), se debe considerar lo siguiente para determinar las claves:

a) Un determinante es una llave candidata si al establecer la dependencia funcional $A \rightarrow B$, $A$ y $B$ se colocan juntos en una tabla sin otras columnas o atributos.
b) Un determinante es una llave candidata cuando determina otras columnas o atributos de una tabla.
c) Un determinante es una llave primaria si no existen otras llaves candidatas y si dicho determinante no permite valores nulos.

## *Normalización*

El proceso de normalización puede considerarse como una serie de pasos que se aplican para asegurar que se cumple una determinada forma normal. Cada paso corresponde a una forma normal y como explica (Connolly & Begg, 2015), las restricciones de las relaciones se vuelven más estrictas y por consiguiente, se disminuye la posibilidad de que la futura base de datos presente anomalías de actualización.

**Formas normales basadas en DFs**

En esta sección se describirán las principales formas normales que generalmente se aplican a las relaciones de un modelo relacional. (Connolly & Begg, 2015) explica que sólo la 1NF es decisiva en la creación de una base de datos y aplicar al menos hasta la 3NF evita que haya anomalías de actualización.

En la Figura 29 se muestra un diagrama que representa la relación entre las diferentes formas normales, empezando desde la *forma no normalizada* que corresponde a las relaciones a las que no se les aplicado el proceso de normalización.

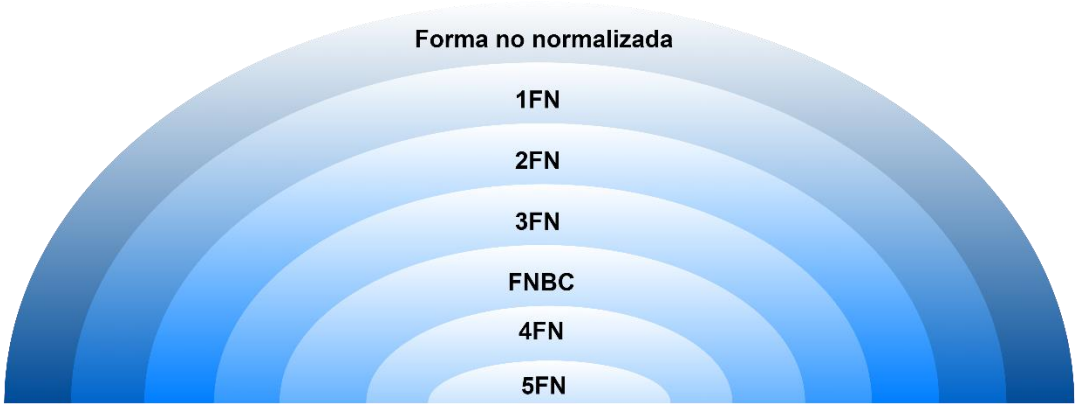



*Figura 29. Representación de la relación entre las formas normales. Elaboración propia.*

A continuación se describirán las diferentes formas normales, basado en las explicaciones de (Churcher, 2007):

- **Primera forma normal (1FN):** Este es el nivel de normalización más básico. Una relación está en 1FN si y sólo si:
  a) Todos los atributos son valores atómicos, es decir, cada columna de la tabla sólo puede tener un valor para cada registro.
  b) La relación no tiene grupos repetitivos.
  c) La relación tiene una llave primaria.
  d) Ninguna clave candidata acepta valores nulos.

  Considerar lo siguiente:

  ✓ Si la relación tiene atributos multivaluados, entonces se debe crear una tabla nueva con esa información y debe contener la llave primaria de la tabla original.
  ✓ Si la llave es simple, pero los valores se repiten en varios registros de la tabla, entonces se puede escoger otra llave de entre las llaves candidatas. Puede ser otra llave simple o una llave compuesta, lo importante es que permita identificar de forma única cada registro.

  **Ejemplo #1:** Observe la siguiente relación:

PROFESOR

| **cod_prof** | cedula | nombre | nom_curso | facultad |
|---|---|---|---|---|
| P030 | 8-881-8881 | John Smith | Cálculo I, Cálculo II, Ecuaciones Diferenciales | FCT |
| P015 | 8-299-9999 | Ana Rodríguez | Química I, Química II | FCT |
| P033 | 3-455-5111 | José Pérez | Sistemas Operativos, Base de Datos I, Base de Datos II | FISC |

Valores no atómicos

- Se observa que en la columna *cursos*, los valores no son atómicos y por lo tanto la tabla no está en 1FN.
- Para que esté en 1FN se crea una nueva tabla con los valores de la columna *nom_curso* y con la columna *cod_prof* que es la llave primaria de la tabla PROFESOR. Además, cada uno de los valores no atómicos corresponderá a un registro en la nueva tabla, para un mismo atributo. Se obtienen las siguientes tablas:

PROFESOR

| **cod_prof** | cedula | nombre | facultad |
|---|---|---|---|
| P030 | 8-881-8881 | John Smith | FCT |
| P015 | 8-299-9999 | Ana Rodríguez | FCT |
| P033 | 3-455-5111 | José Pérez | FISC |

CURSO

| cod_prof | nom_curso |
|---|---|
| P030 | Cálculo I |
| P030 | Cálculo II |
| P030 | Ecuaciones Diferenciales |
| P015 | Química I |
| P015 | Química II |
| P033 | Sistemas Operativos |
| P033 | Base de Datos I |
| P033 | Base de Datos II |

**Ejemplo #2:** Observe la siguiente relación:

LIBRO

| cod_libro | nom_libro | autor_1 | autor_2 | editor | cod_editor |
|---|---|---|---|---|---|
| 0123 | Base de datos no relacionales | Jacob Martínez | John Pérez | Elle Watson | E0005 |
| 0135 | Sistemas dinámicos y sus aplicaciones | Olivia Morgan | Henry Brown | Carlos Parker | E0129 |
| 0120 | Modelado y simulación de sistemas | Michael Lee | Sara López | Helen Carter | E0301 |
| 0105 | Aplicaciones de las ecuaciones diferenciales | Jane Rivera | Grace Torres | José Phillips | E0401 |

Grupos repetitivos

- Se observa que la tabla contiene grupos repetitivos, por lo tanto no está en 1FN.
- Para que esté en 1FN se crea una nueva tabla con la llave primaria de la tabla original y se separan los grupos repetitivos en valores atómicos de registros diferentes para un solo atributo. Se obtendrían las siguientes tablas:

LIBRO

| cod_libro | nom_libro | editor | cod_editor |
|---|---|---|---|
| 0123 | Base de datos no relacionales | Elle Watson | E0005 |
| 0135 | Sistemas dinámicos y sus aplicaciones | Carlos Parker | E0129 |
| 0120 | Modelado y simulación de sistemas | Helen Carter | E0301 |
| 0105 | Aplicaciones de las ecuaciones diferenciales | José Phillips | E0401 |

AUTOR

| cod_libro | autor |
|---|---|
| 0123 | Jacob Martínez |
| 0123 | John Pérez |
| 0135 | Olivia Morgan |
| 0135 | Henry Brown |
| 0120 | Michael Lee |
| 0120 | Sara López |
| 0105 | Jane Rivera |
| 0105 | Grace Torres |

- **Segunda forma normal (2FN):** Una relación está en 2FN si y sólo si:
  a) Está en la primera forma normal.
  b) Cada atributo-no-llave es completamente dependiente de la llave primaria completa, no sólo de una parte de la llave.

  Considerar lo siguiente:

  ✓ Si la relación no está en 1FN, se deben remover los atributos-no-llave que no dependen de la llave primaria y crear otra tabla que contenga esos atributos y la parte de la llave primaria de la que dependen.
  ✓ Si la relación está en 1FN y tiene una llave simple, automáticamente está en 2FN. Sólo se aplica la normalización de la segunda forma normal cuando la relación tiene una llave compuesta.

  **Ejemplo:** Observe la siguiente relación:

CURSO

| **cod_curso** | nom_profesor | **cedula** | nom_curso | facultad | año_estudio |
|---|---|---|---|---|---|
| 0001 | John Smith | 8-881-8881 | Cálculo I | FCT | I |
| 0002 | Ana Rodríguez | 8-299-9999 | Química I | FCT | I |
| 0003 | José Pérez | 3-455-5111 | Base de Datos I | FISC | III |
| 0004 | José Pérez | 3-455-5111 | Base de Datos II | FISC | III |
| 0005 | José Pérez | 3-455-5111 | Sistemas Operativos | FISC | III |
| 0006 | Ana Rodríguez | 8-299-9999 | Química II | FCT | II |

  ➢ La tabla está en 1FN porque tiene una llave primaria (compuesta), todos los valores de cada atributo son atómicos y no hay grupos repetitivos.
  ➢ Como la tabla tiene una llave compuesta, hay que encontrar la 2FN. Para ello se identifican todas las dependencias funcionales de la llave compuesta y son las siguientes:

*cod_curso, cedula → nom_curso*
*cod_curso → facultad, año_estudio*
*cedula → nom_profesor*

- Se observa que hay atributos-no-llave que no dependen de la llave completa, la cual es *cod_curso – cedula*, por lo tanto no está en 2FN. Se separa la relación original y se obtienen las siguientes relaciones o tablas:

CURSO

| **cod_curso** | **cedula** | nom_curso |
|---|---|---|
| 0001 | 8-881-8881 | Cálculo I |
| 0002 | 8-299-9999 | Química I |
| 0003 | 3-455-5111 | Base de Datos I |
| 0004 | 3-455-5111 | Base de Datos II |
| 0005 | 3-455-5111 | Sistemas Operativos |
| 0006 | 8-299-9999 | Química II |

PLAN

| **cod_curso** | facultad | año_estudio |
|---|---|---|
| 0001 | FCT | I |
| 0002 | FCT | I |
| 0003 | FISC | III |
| 0004 | FISC | III |
| 0005 | FISC | III |
| 0006 | FCT | II |

PROFESOR

| **cedula** | nom_profesor |
|---|---|
| 8-881-8881 | John Smith |
| 8-299-9999 | Ana Rodríguez |
| 3-455-5111 | José Pérez |

- **Tercera forma normal (3FN):** Una relación está en 3FN si y sólo si:
  a) Está en la segunda forma normal.
  b) No hay atributos-no-llave que son dependientes de otros atributos-no-llave.

  Considerar lo siguiente:

- ✓ Si la relación no está en 3FN, remover todos los atributos-no-llave que no dependan de la llave primaria y crear otra tabla con esos atributos y el atributo del que dependen.

**Ejemplo:** Observe la siguiente relación:

CURSO

| **cod_curso** | nom_curso | cedula | nom_profesor | facultad |
|---|---|---|---|---|
| 0001 | Cálculo I | 8-881-8881 | John Smith | FCT |
| 0002 | Química I | 8-299-9999 | Ana Rodríguez | FCT |
| 0003 | Base de datos I | 3-455-5111 | José Pérez | FISC |
| 0004 | Base de datos II | 3-455-5111 | José Pérez | FISC |
| 0005 | Sistemas Operativos | 3-455-5111 | José Pérez | FISC |
| 0006 | Química II | 8-299-9999 | Ana Rodríguez | FCT |

- ➢ La tabla está en 2FN porque tiene una llave simple.
- ➢ Hay que encontrar la 3FN. Para ello se identifican las dependencias funcionales de los atributos-no-llave, es decir, todos aquellos atributos que no dependen de la llave.

  *cedula_prof* → *nom_profesor*
- ➢ Se remueve el atributo-no-llave de la tabla original y se crea una nueva tabla con el atributo-no-llave y su determinante. Se obtendrían las siguientes tablas:

CURSO

| **cod_curso** | nom_curso | facultad | cedula_prof |
|---|---|---|---|
| 0001 | Cálculo I | FCT | 8-881-8881 |
| 0002 | Química I | FCT | 8-299-9999 |
| 0003 | Base de datos I | FISC | 3-455-5111 |
| 0004 | Base de datos II | FISC | 3-455-5111 |
| 0005 | Sistemas Operativos | FISC | 3-455-5111 |
| 0006 | Química II | FCT | 8-299-9999 |

PROFESOR

| **cedula_prof** | nom_profesor |
|---|---|
| 8-881-8881 | John Smith |
| 8-299-9999 | Ana Rodríguez |
| 3-455-5111 | José Pérez |
| 3-455-5111 | José Pérez |
| 3-455-5111 | José Pérez |
| 8-299-9999 | Ana Rodríguez |

- **Forma normal de Boyce-Codd (FNBC):** Una relación está en FNBC si cada determinante de una DF puede ser una llave primaria. Toda relación en FNBC también está en 1FN, 2FN y 3FN.

**Descomposición de las relaciones**

El esquema de una base de datos se compone de una relación universal $R = \{A_1, A_2, \dots, A_n\}$ donde $A$ es el conjunto de todos los atributos de la base de datos. A través del uso de las dependencias funcionales, esta relación R se descompone en un conjunto de relaciones denominado $D = \{R_1, R_2, \dots, R_m\}$, lo cual corresponde a la descomposición de $R$ (Elmasri & Navathe, 2011). Al realizar la descomposición de relaciones es importante asegurar que haya una preservación de atributos, de modo que no se pierda ningún atributo. Para ello, cada atributo en $R$ debe aparecer en al menos un esquema de relación $R_i$, lo cual formalmente se define como:

$$\bigcup_{i=1}^{m} R_i = R$$

Para lograr una descomposición sin pérdida de atributos, es conveniente incluir atributos comunes. De esta manera, cuando se hagan operaciones de *join* o *unión*, será posible recuperar la información que ahora está guardada en tablas distintas.

En este sentido, se habla de una propiedad denominada *dependencia de unión sin pérdida*, que de acuerdo con (Connolly & Begg, 2015), asegura que no se generen tuplas espurias cuando las relaciones se unen a través de la operación reunión natural.

**Prácticas de normalización**

A continuación se exponen tres problemas. Normalizar cada una de las relaciones, indicando si se encuentra en primera, segunda o tercera forma normal y si no está en alguna de las formas normales ¿cómo quedaría la relación? Tomar en cuenta que el atributo en color rojo es la llave de la relación.

**Problema #1:**

*membresia (cod_membresia, nombre_miembro, cedula, telefono, dirección, ficha_registro)*

**Solución:**

- Está en 1FN porque tiene llave primaria.
- Está en 2FN porque está en 1FN y porque tiene una llave simple.
- No está en 3FN porque se puede identificar la siguiente DF de un atributo-no-llave:

  *cedula → nombre_miembro*

  Para que la relación esté en 3FN, la relación debe separarse en las siguientes tablas:

  *membresia (cod_membresia, cedula, telefono, dirección, fecha_registro)*
  *miembro (cedula, nombre_miembro)*

**Problema #2:**

*pelicula (cod_pelicula, titulo, año, nom_director, genero, duración, cod_director)*

**Solución:**

- Está en 1FN porque tiene llave primaria.
- Está en 2FN porque está en 1FN y porque tiene una llave simple.
- No está en 3FN porque se identifica la siguiente DF de un atributo-no-llave:

  *cod_director → nom_director*

  Para que la relación esté en 3FN, la relación debe separarse en las siguiente tablas:

  *pelicula (cod_pelicula, titulo, año, cod_director, genero, duracion)*
  *director (cod_director, nom_director)*

**Problema #3:**

*boleto (cod_boleto, nombre_cliente, id_cliente, destino, fecha_vuelo, numero_vuelo)*

**Solución:**

- Está en 1FN porque tiene llave primaria.
- Como tiene una llave compuesta se procede a buscar la 2FN. Se identifican las siguientes DFs de la llave:

  *cod_boleto, numero_vuelo → fecha_vuelo*
  *numero_vuelo → destino*
  *cod_boleto → id_cliente, nombre_cliente*

Se observa que algunos atributos no dependen de la llave completa. Para que la relación esté en 2FN, la relación debe separarse en las siguientes tablas:

*boleto (cod_boleto, numero_vuelo, fecha_vuelo)*
*vuelo (numero_vuelo, destino)*
*pasaje (cod_boleto, id_cliente, nombre_cliente)*

- Se observa que las tres relaciones resultantes están en la 2FN. Sin embargo, en la relación *pasaje* se observa que el atributo *nombre_cliente* depende de un atributo-no-llave, como se comprueba con la siguiente DF:

  *id_cliente → nombre_cliente*

  Para que la relación esté en 3FN, la relación *pasaje* debe separarse en las siguientes tablas:

  *pasaje (cod_boleto, id_cliente)*
  *cliente (id_cliente, nombre_cliente)*

# Capítulo VI: MS SQL – Lenguaje estructurado de consulta

## *Introducción al entorno de trabajo del Gestor de Base de Datos*

El entorno de trabajo del gestor de base de datos MS SQL Server, es similar a las ventanas de herramientas de otras aplicaciones del sistema operativo Windows. Como describen (Microsoft, 2019) y (Tutorials Point, 2016), MS SQL Server es un entorno que integra herramientas que soportan el uso del lenguaje SQL para crear y mantener bases de datos, así como para el análisis de datos y generación de reportes. En la Figura 30 se muestra la ventana de herramientas de MS SQL Server.

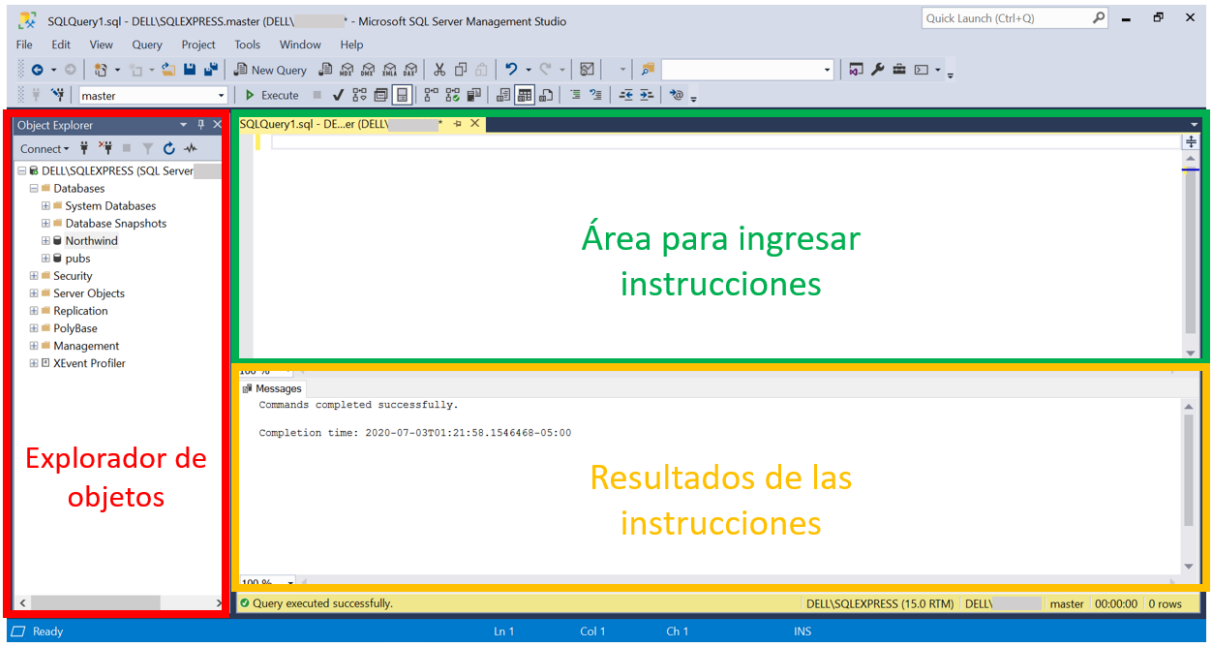



*Figura 30. Ventana de herramientas de MS SQL Server Management Studio.*

La ventana de herramientas de SQL Server se puede dividir en dos grandes áreas:

- **Explorador de objetos:** Permite ver y administrar los objetos que forman parte de las instancias de SQL Server (Microsoft, 2019). En esta área se pueden ver todas las bases de datos instaladas.
- **Área de trabajo:** Se subdivide en un área para ingresar las instrucciones para la consulta y mantenimiento de las bases de datos y en otra área donde se muestran mensajes de error, resultados de las consultas o mensajes que indican si se han completado satisfactoriamente las instrucciones.

Al iniciar el programa SQL Server, es necesario conectarse a un servidor. Se coloca el nombre del servidor y se hace click en el botón “Connect”, que aparecen en la ventana “Conectar a servidor”, mostrada en la Figura 31.

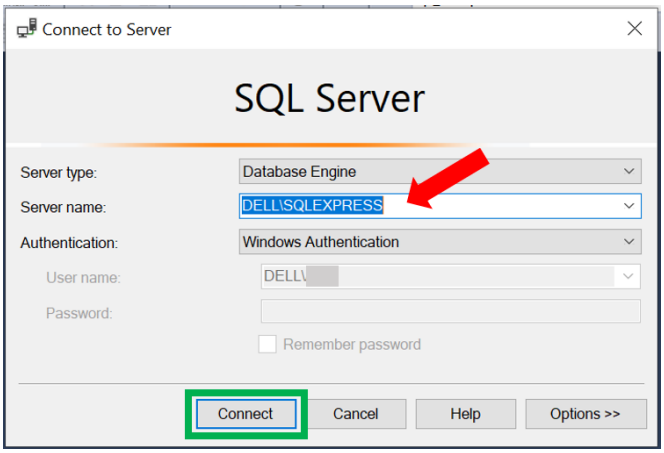



*Figura 31. Ventana "Conectar a servidor".*

Una vez se ha realizado la conexión, en el explorador de objetos se muestra el servidor activo, como se muestra en la Figura 32.

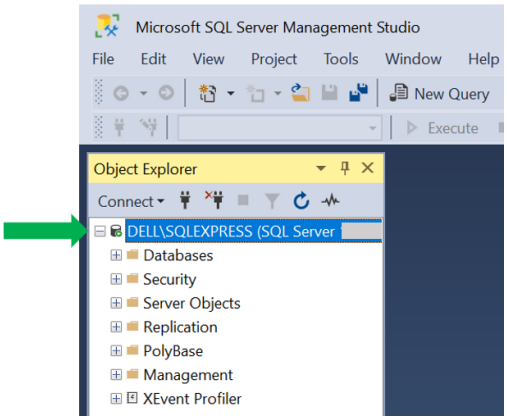



*Figura 32. Servidor activo en el explorador de objetos.*

También se muestra de forma completa la ventana de herramientas con un menú de opciones y una barra de herramientas. Para activar el área de trabajo y empezar a escribir instrucciones, se debe seleccionar la opción “New Query” que se encuentra en la barra de herramientas, como se muestra en la Figura 33.

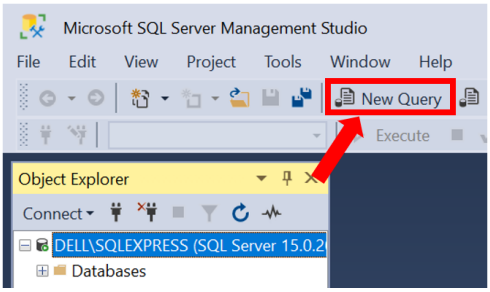



*Figura 33. Opción "New Query" en la barra de herramientas en SQL Server.*

Una vez se ha activado, se pueden ingresar las instrucciones que permiten crear y mantener las bases datos. De igual manera, se activa otra barra de herramientas en donde está la “Execute”, como se muestra en la Figura 34. Esta opción permite ejecutar las instrucciones que se escriban en el área de trabajo.

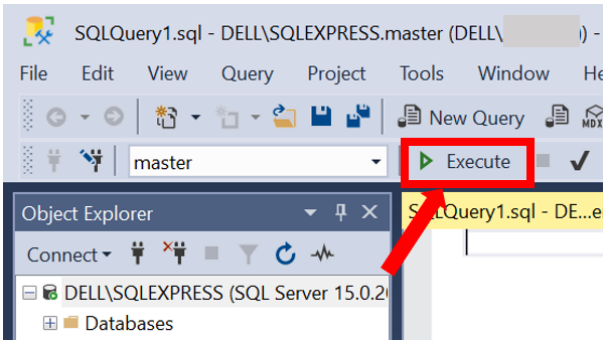



*Figura 34. Opción "Execute" en SQL Server, que permite ejecutar instrucciones.*

En la Figura 34, también se observa que al lado de la opción “Execute”, se muestra la bases de datos que está activa. Antes de empezar a escribir instrucciones, es necesario seleccionar la base de datos que se utilizará. Para ello, se hace click sobre esta opción y se despliega un listado de las bases de datos instaladas, como se observa en la Figura 35.

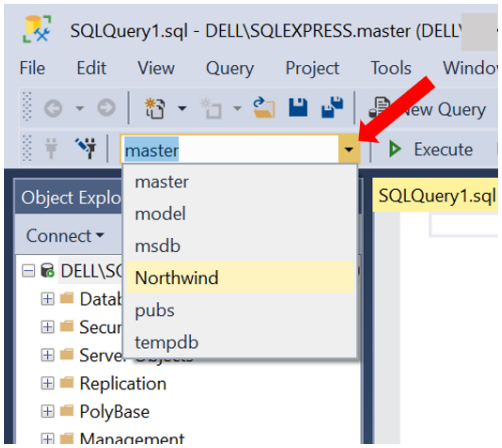



*Figura 35. Listado para activación de una base de datos.*

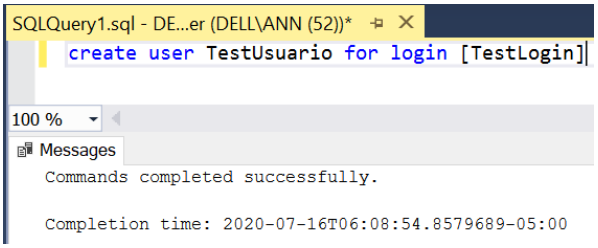
### *Instrucciones de creación de usuarios en el sistema*

Como explica (Tutorials Point, 2016), para poder acceder a una base de datos, un usuario se refiere a una cuenta en la base de datos en MS SQL Server. La sintaxis de la instrucción para crear un usuario es la siguiente:

```
create user nombre_usuario for login nombre_login
```

Para utilizar esta sintaxis es necesario saber el nombre del login, es decir, el login ya debe estar creado. Además, se debe seleccionar la base de datos para la cual se desea crear el login.

**Ejemplo:** Crear un usuario en llamado *TestUsuario* cuyo nombre de login sea *TestLogin*.




## *Instrucciones de definición de datos*

Las instrucciones de definición de datos se utilizan para la creación, eliminación y modificación de los modelos relacionales de las bases de datos (Universidad Privada TELESUP, 2018). Dentro del lenguaje SQL, a estos tipos de instrucciones o comandos se les denomina Data Definition Language (DDL) y permiten definir bases de datos, lo cual incluye crear, alterar y eliminar tablas y establecer las restricciones de estas (Sumathi & Esakkirajan, 2007).

El estándar SQL soporta diferentes tipos de datos, los cuales se utilizan para su definición. Entre los tipos de datos más utilizados, se encuentran los siguientes, descritos por (Silberschatz et al., 2011) y (Camps Paré et al., 2005):

- **char(n):** Almacena una cadena de caracteres de longitud fija, a la cual se le debe especificar la longitud.
- **varchar(n):** Almacena una cadena de caracteres de longitud variable, a la cual se le debe especificar la longitud máxima.
- **integer:** Almacena un número entero.
- **smallint:** Almacena un número entero pequeño.
- **decimal(p,e):** Almacena un número decimal con la cantidad de dígitos indicados en la precisión (p) y con los decimales indicados en la escala (e).
- **real:** Almacena un número decimal que tiene una precisión predefinida.
- **float(p):** Almacena un número decimal, se le debe especificar la precisión (p).
- **date:** Almacena una fecha compuesta por YEAR (año), MONTH (mes), DAY (día).
- **time:** Almacena horas compuestas por HOUR (hora), MINUTE (minuto), SECOND (segundo).
- **timestamp:** Almacena fecha y hora.

Las instrucciones de definición de datos más utilizadas se describen a continuación:

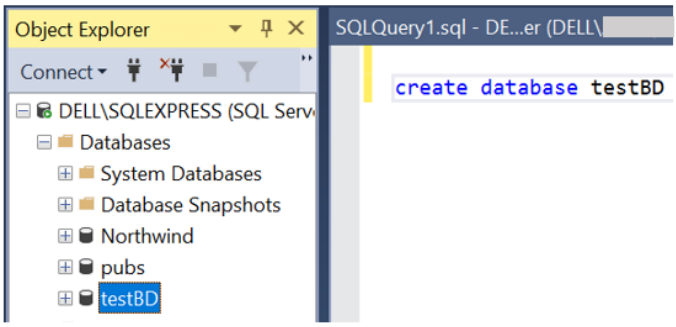
### Create

Esta instrucción sirve para crear las bases de datos y sus tablas.

- **Create database:** Instrucción que sirve para crear una base de datos. La sintaxis es la siguiente:

```
create database nombre_BD
```

**Ejemplo:** Crear una base de datos llamada *testBD*.



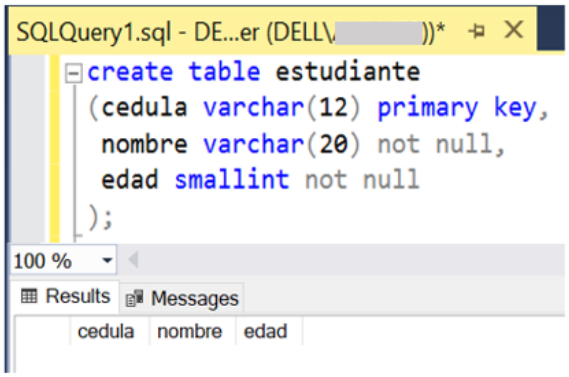

**Create table:** Instrucción que sirve para crear una nueva tabla y sus atributos. Los atributos o columnas de una tabla pueden tener restricciones como las siguientes, descritas por (Codecademy, 2020b) y (Teorey, Lighstone, Nadeau, & Jagadish, 2011):

- **Primary key:** Se utiliza para identificar a una tabla de forma única y puede ser uno o más atributos.
- **Unique:** Indica que la columna tiene un valor diferente para cada fila.
- **Default:** Asigna un valor por defecto cuando no se especifica el valor de un dato en la columna.
- **Not null:** Indica que la columna debe tener un valor, es decir, que no debe ser nulo.
- **Foreign key:** Indica que el atributo es una llave foránea, la cual está referenciada a una llave primaria existente en otra tabla.

La sintaxis básica utilizada para esta instrucción es la siguiente:

```
create table nombre_tabla
(columna1 tipo_dato,
 columna2 tipo_dato,
 columnaN tipo_dato
);
```

**Ejemplo:** Utilizar la base de datos “testDB” y crear una tabla llamada *estudiante*, la cual debe tener los siguientes atributos y sus restricciones: *cedula* es de tipo varchar de longitud 12 y es la llave primaria, *nombre* es de tipo varchar de longitud 20 y *edad* es de tipo smallint y no puede ser nulo:



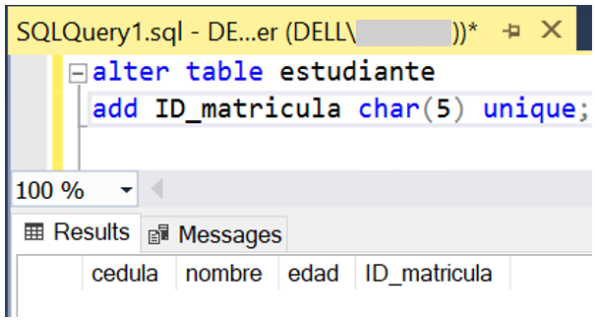

**Alter**

Esta instrucción es utilizada para agregar columnas nuevas y eliminar o modificar las columnas existentes en una tabla. Esta instrucción se utiliza con los siguientes cláusulas, descritos por (Sumathi & Esakkirajan, 2007):

- **Add:** Permite agregar una columna nueva a la tabla. Para ello se utiliza la siguiente sintaxis:

```
alter table nombre_tabla
add nombre_columna tipo_dato;
```

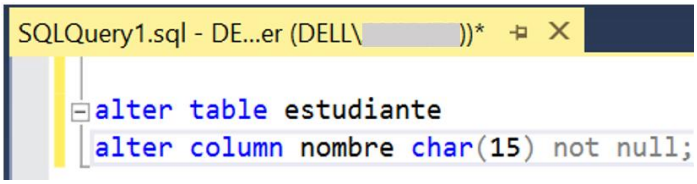
  **Ejemplo:** Agregar una nueva columna a la tabla *estudiante*, denominada *ID_matricula*. Es de tipo char de longitud 5 y su valor debe ser único para cada fila.




- **Alter column:** Permite modificar el tipo de dato de una columna. La sintaxis es la siguiente:

```
alter table nombre_tabla
alter column nombre_columna tipo_dato;
```

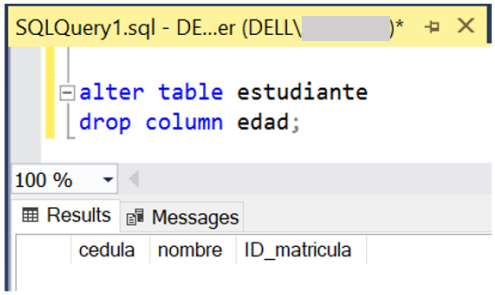
  **Ejemplo:** Modificar el tipo de dato de la columna *nombre* al tipo char de longitud 15.




- **Drop column:** Permite eliminar una columna de una tabla. Se utiliza la siguiente sintaxis:

```
alter table nombre_tabla
drop column nombre_columna;
```

**Ejemplo:** Eliminar la columna *edad* de la tabla *estudiante*.



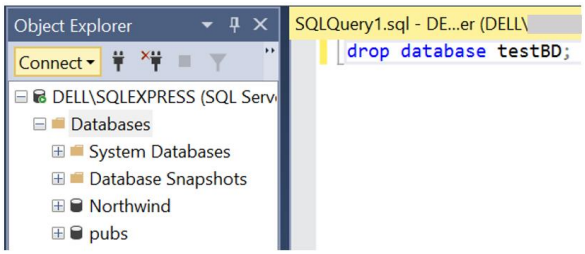

**Otras**

Otras instrucciones permiten eliminar una base de datos existente o la tabla de una base de datos. Estas instrucciones son:

- **Drop database:** Permite eliminar una base de datos existente. Se utiliza la siguiente sintaxis:

```
drop database nombre_BD;
```

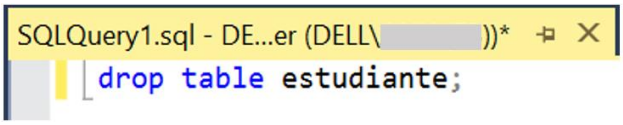
**Ejemplo:** Eliminar la base de datos *testBD*.




- **Drop table:** Permite eliminar un tabla existente en una base de datos. La sintaxis que se debe utilizar es la siguiente:

```
drop table nombre_tabla;
```

**Ejemplo:** Eliminar la tabla *estudiante* de la base de datos *testBD*.




## *Instrucciones de manipulación de datos*

Las instrucciones de manipulación de datos, también llamados Data Manipulation Language (DML), son comandos que permiten mantener bases de datos y hacer consultas en estas, lo cual incluye actualizar, modificar y consultar datos (Sumathi & Esakkirajan, 2007).

### Estructura básica

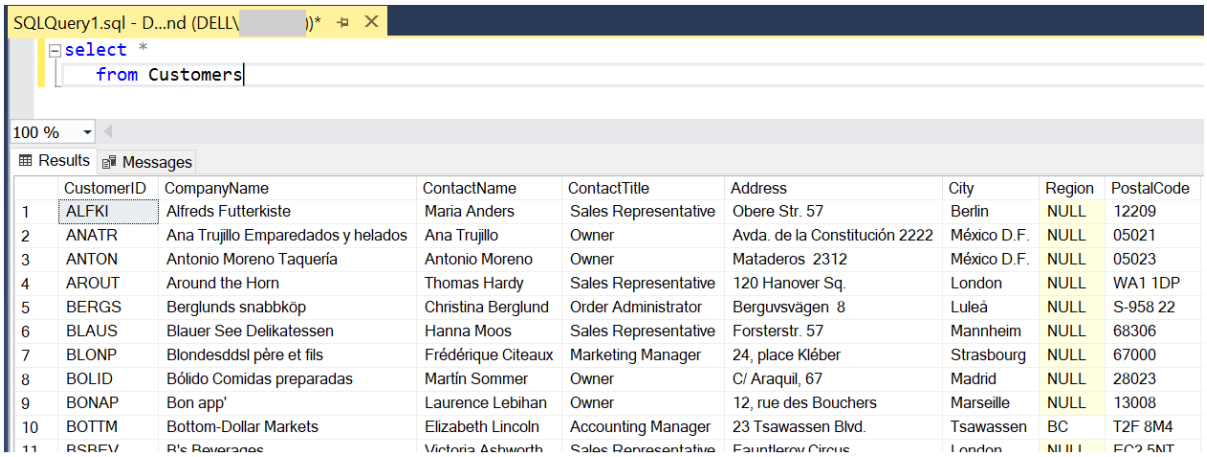
La estructura básica de las consultas en SQL se compone de las cláusulas *select*, *from* y *where*. *Select* es la cláusula base para todas las consultas (Teorey et al., 2011). A continuación se describen estas cláusulas, de acuerdo con (Universidad Privada TELESUP, 2018):

- **Cláusula SELECT:** Muestra las columnas de una tabla en el orden que se haya especificado.
- **Cláusula WHERE:** Especifica la tabla o tablas de las que se recupera la información.
- **Cláusula FROM:** Especifica las restricciones o criterios de selección de las filas.

Para mostrar todas las columnas o atributos de una tabla se utiliza la siguiente sintaxis:

```
select *
   from nombre_tabla
```

**Ejemplo:** Utilizando la base de datos *Northwind*, hacer una consulta que muestre todos los atributos de la tabla *Customers*.

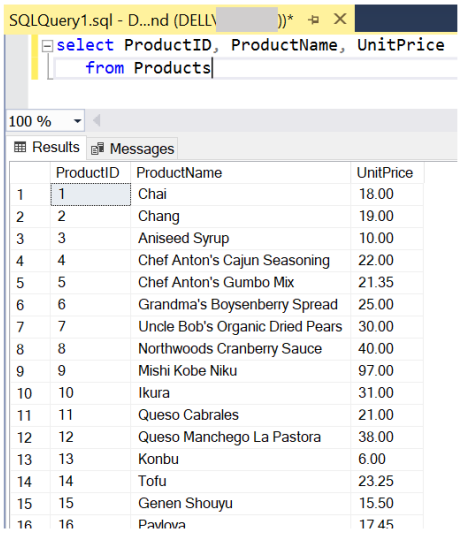



| | CustomerID | CompanyName | ContactName | ContactTitle | Address | City | Region | PostalCode |
|---|---|---|---|---|---|---|---|---|
| 1 | ALFKI | Alfreds Futterkiste | Maria Anders | Sales Representative | Obere Str. 57 | Berlin | NULL | 12209 |
| 2 | ANATR | Ana Trujillo Emparedados y helados | Ana Trujillo | Owner | Avda. de la Constitución 2222 | México D.F. | NULL | 05021 |
| 3 | ANTON | Antonio Moreno Taquería | Antonio Moreno | Owner | Mataderos 2312 | México D.F. | NULL | 05023 |
| 4 | AROUT | Around the Horn | Thomas Hardy | Sales Representative | 120 Hanover Sq. | London | NULL | WA1 1DP |
| 5 | BERGS | Berglunds snabbköp | Christina Berglund | Order Administrator | Berguvsvägen 8 | Luleå | NULL | S-958 22 |
| 6 | BLAUS | Blauer See Delikatessen | Hanna Moos | Sales Representative | Forsterstr. 57 | Mannheim | NULL | 68306 |
| 7 | BLONP | Blondesddsl père et fils | Frédérique Citeaux | Marketing Manager | 24, place Kléber | Strasbourg | NULL | 67000 |
| 8 | BOLID | Bólido Comidas preparadas | Martín Sommer | Owner | C/ Araquil, 67 | Madrid | NULL | 28023 |
| 9 | BONAP | Bon app' | Laurence Lebihan | Owner | 12, rue des Bouchers | Marseille | NULL | 13008 |
| 10 | BOTTM | Bottom-Dollar Markets | Elizabeth Lincoln | Accounting Manager | 23 Tsawassen Blvd. | Tsawassen | BC | T2F 8M4 |
| 11 | BSBEV | B's Beverages | Victoria Ashworth | Sales Representative | Fauntleroy Circus | London | NULL | EC2 5NT |

Si se desean mostrar columnas específicas de una tabla, se utiliza la siguiente sintaxis:

```
select nombre_columna1, nombre_columna2, nombre_columnaN
   from nombre_tabla
```

**Ejemplo:** Utilizando la base de datos *Northwind*, hacer una consulta que muestre el código de producto, el nombre del producto y el precio unitario, de la tabla *Products*.

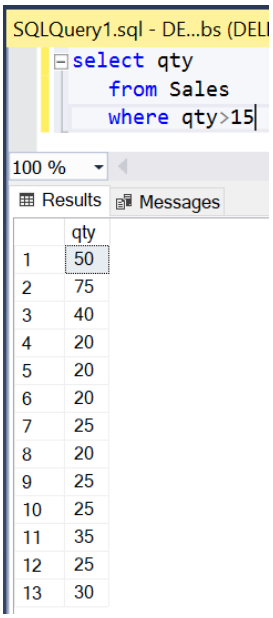



| | ProductID | ProductName | UnitPrice |
|---|---|---|---|
| 1 | 1 | Chai | 18.00 |
| 2 | 2 | Chang | 19.00 |
| 3 | 3 | Aniseed Syrup | 10.00 |
| 4 | 4 | Chef Anton's Cajun Seasoning | 22.00 |
| 5 | 5 | Chef Anton's Gumbo Mix | 21.35 |
| 6 | 6 | Grandma's Boysenberry Spread | 25.00 |
| 7 | 7 | Uncle Bob's Organic Dried Pears | 30.00 |
| 8 | 8 | Northwoods Cranberry Sauce | 40.00 |
| 9 | 9 | Mishi Kobe Niku | 97.00 |
| 10 | 10 | Ikura | 31.00 |
| 11 | 11 | Queso Cabrales | 21.00 |
| 12 | 12 | Queso Manchego La Pastora | 38.00 |
| 13 | 13 | Konbu | 6.00 |
| 14 | 14 | Tofu | 23.25 |
| 15 | 15 | Genen Shouyu | 15.50 |
| 16 | 16 | Pavlova | 17.45 |

Si se desean mostrar todas las columnas de una tabla, pero sólo las que cumplen con una restricción especificada, se utiliza la siguiente sintaxis:

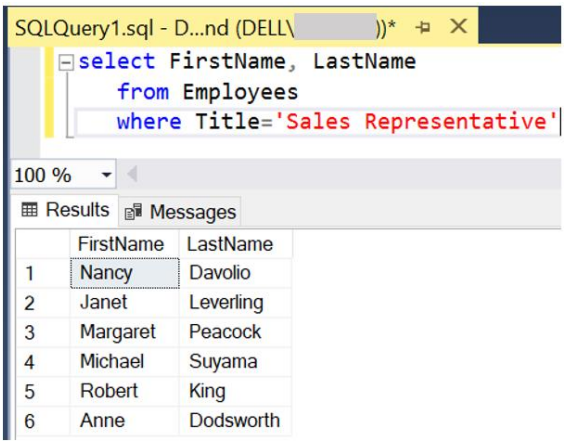
```
select nombre_columna1, nombre_columna2, nombre_columnaN
   from nombre_tabla
   where condición
```

**Ejemplo #1:** Utilizando la base de datos *pubs*, hacer una consulta que muestre los registros con *qty* mayor que 15, que están en la tabla *sales*.




| | qty |
|---|---|
| 1 | 50 |
| 2 | 75 |
| 3 | 40 |
| 4 | 20 |
| 5 | 20 |
| 6 | 20 |
| 7 | 25 |
| 8 | 20 |
| 9 | 25 |
| 10 | 25 |
| 11 | 35 |
| 12 | 25 |
| 13 | 30 |

**Ejemplo #2:** Utilizando la base de datos *Northwind*, hacer una consulta que muestre el nombre y el apellido de todos los empleados cuyo *Title* es “Sales Representative”.




| | FirstName | LastName |
|---|---|---|
| 1 | Nancy | Davolio |
| 2 | Janet | Leverling |
| 3 | Margaret | Peacock |
| 4 | Michael | Suyama |
| 5 | Robert | King |
| 6 | Anne | Dodsworth |

Como se ha visto en los ejemplos anteriores, dentro de la cláusula *where* se pueden utilizar los operadores de comparación, los cuales se describen a continuación:

| Operador | Descripción |
|---|---|
| = | Igual a |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |
| <> | Distinto a |

- **Operación renombramiento:** Esta operación permite renombrar cualquier atributo que se muestre como resultado de una consulta, para lo cual se coloca la cláusula *as* seguido del nombre deseado (Elmasri & Navathe, 2016). El renombramiento sólo es temporal y demora el tiempo que dure la consulta. En SQL, existen diferentes maneras de realizar esta operación. A continuación se describe un ejemplo, el cual se utilizará para demostrar el uso de los distintos formatos que se pueden usar para la operación renombramiento.

  **Ejemplo:** Utilizando la base de datos *pubs*, realizar una consulta que muestre los registros de la columna *au_fname*, de la tabla *authors*. Renombrar la columna como "Nombre del autor".

  ✓ **Formato #1:** Utilizando la palabra *as*, seguido del nombre deseado entre comillas simples. En este caso, el nombre puede tener más de dos palabras y se puede dejar espacio entre ellas. La sintaxis sería:

  ```
  nombre_columna as 'nombre_deseado'
  ```


| | Nombre del autor |
|---|---|
| 1 | Abraham |
| 2 | Reginald |
| 3 | Cheryl |
| 4 | Michel |
| 5 | Innes |
| 6 | Ann |
| 7 | Marjorie |
| 8 | Morningstar |
| 9 | Burt |
| 10 | Sheryl |
| 11 | Livia |
| 12 | Charlene |
| 13 | Stearns |
| 14 | Heather |
| 15 | Michael |
| 16 | Sylvia |

  ✓ **Formato #2:** Utilizando la palabra *as*, seguido del nombre deseado sin comillas. En este caso, si el nombre tiene más de dos palabras, no se puede dejar espacio entre ellas. La sintaxis sería:

  ```
  nombre_columna as nombre_deseado
  ```


- ✓ **Formato #3:** Utilizando el signo *igual (=)*, seguido del nombre de la columna que se quiere renombrar. En este caso, primero se coloca el nombre deseado. Al igual que el formato anterior, si el nombre tiene más de dos palabras, no se puede dejar espacio entre ellas. La sintaxis sería:

```
nombre_deseado = nombre_columna
```

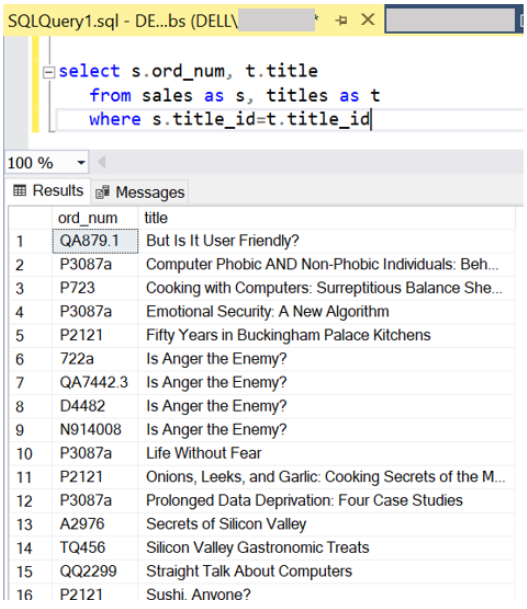

- ✓ **Formato #4:** Utilizando corchetes (paréntesis cuadrados) y colocar dentro de estos el nombre deseado. La sintaxis sería:

```
nombre_columna [nombre_deseado]
```


| | Nombre del autor |
|---|---|
| 1 | Abraham |
| 2 | Reginald |
| 3 | Cheryl |
| 4 | Michel |
| 5 | Innes |
| 6 | Ann |
| 7 | Marjorie |
| 8 | Morningstar |
| 9 | Burt |
| 10 | Sheryl |
| 11 | Livia |
| 12 | Charlene |
| 13 | Stearns |
| 14 | Heather |
| 15 | Michael |
| 16 | Sylvia |

La operación renombramiento también puede utilizarse para renombrar tablas, de modo que sea más conveniente utilizar un nombre más corto dentro de una consulta.

**Ejemplo:** Utilizando la base de datos *pubs*, mostrar el número de orden y el título de todos los libros que se hayan vendido.

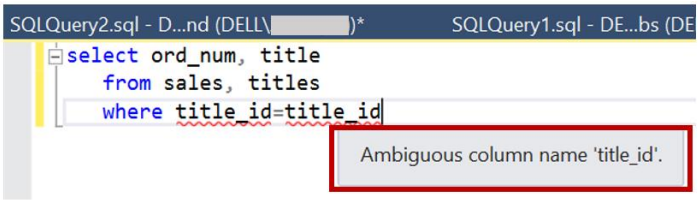



| | ord_num | title |
|---|---|---|
| 1 | QA879.1 | But Is It User Friendly? |
| 2 | P3087a | Computer Phobic AND Non-Phobic Individuals: Beh... |
| 3 | P723 | Cooking with Computers: Surreptitious Balance She... |
| 4 | P3087a | Emotional Security: A New Algorithm |
| 5 | P2121 | Fifty Years in Buckingham Palace Kitchens |
| 6 | 722a | Is Anger the Enemy? |
| 7 | QA7442.3 | Is Anger the Enemy? |
| 8 | D4482 | Is Anger the Enemy? |
| 9 | N914008 | Is Anger the Enemy? |
| 10 | P3087a | Life Without Fear |
| 11 | P2121 | Onions, Leeks, and Garlic: Cooking Secrets of the M... |
| 12 | P3087a | Prolonged Data Deprivation: Four Case Studies |
| 13 | A2976 | Secrets of Silicon Valley |
| 14 | TQ456 | Silicon Valley Gastronomic Treats |
| 15 | QQ2299 | Straight Talk About Computers |
| 16 | P2121 | Sushi, Anyone? |

Note que en la Figura 45, el nuevo nombre de las tablas corresponde a las letras al inicio de los nombres de las columnas. Esto permite referenciar cada columna con su correspondiente tabla, haciendo uso del nuevo nombre de esta última. Esto también es útil para comparar columnas que tienen el mismo nombre en diferentes

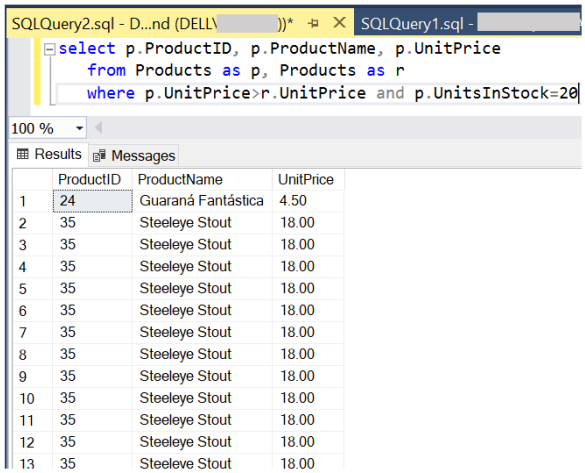
tablas, como sucede con *title_id*. Al utilizar el nuevo nombre de las columnas, se indica que la consulta debe hacerse a través del *title_id*, pues ambas tablas tienen esta columna.

Si no se hace de esta manera, el gestor de base de datos no sabe a cuál columna se hacer referencia, dado que en este caso, ambas tablas tienen una columna llamada *title_id*. La Figura 46 muestra el mismo ejemplo, pero sin utilizar la operación de renombramiento para las tablas.




- **Variable tupla:** La variable tupla se utiliza para renombrar una tabla, cuando se quiere hacer una comparación de tuplas o registros dentro de una misma tabla (Silberschatz et al., 2011).

  **Ejemplo:** Utilizando la base de datos *Northwind*, mostrar el ID, el nombre y el precio unitario de todos los productos con unidades en stock iguales a 20, cuyo precio unitario sea mayor al del producto con el menor precio unitario.



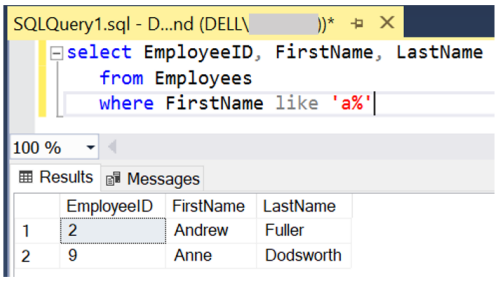

| | ProductID | ProductName | UnitPrice |
|---|---|---|---|
| 1 | 24 | Guaraná Fantástica | 4.50 |
| 2 | 35 | Steeleye Stout | 18.00 |
| 3 | 35 | Steeleye Stout | 18.00 |
| 4 | 35 | Steeleye Stout | 18.00 |
| 5 | 35 | Steeleye Stout | 18.00 |
| 6 | 35 | Steeleye Stout | 18.00 |
| 7 | 35 | Steeleye Stout | 18.00 |
| 8 | 35 | Steeleye Stout | 18.00 |
| 9 | 35 | Steeleye Stout | 18.00 |
| 10 | 35 | Steeleye Stout | 18.00 |
| 11 | 35 | Steeleye Stout | 18.00 |
| 12 | 35 | Steeleye Stout | 18.00 |
| 13 | 35 | Steeleye Stout | 18.00 |

- **Operaciones sobre cadenas:** En SQL, las cadenas se colocan entre comillas simples. Además, SQL no hace distinción entre mayúsculas y minúsculas. La operación sobre cadena que más comúnmente se utiliza es la de encontrar coincidencias dentro de una base de datos. Para ello, se utiliza el operador *like*, el cual permite encontrar patrones especificados en la cláusula *where* (Codecademy,

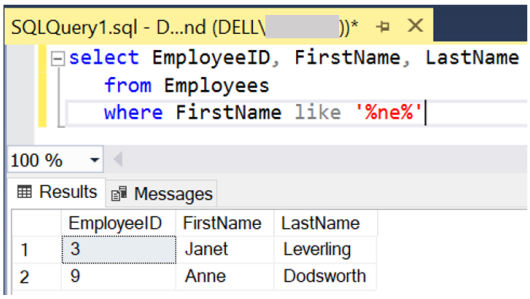
2020d). Las operaciones sobre cadenas pueden realizarse utilizando alguno de los caracteres especiales que se describen a continuación:

- **Porcentaje (%):** Encuentra coincidencias con cualquier subcadena.

  **Ejemplo #1:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el ID, el nombre y el apellido de todos los empleados, cuyo nombre empieza con la letra “a”.

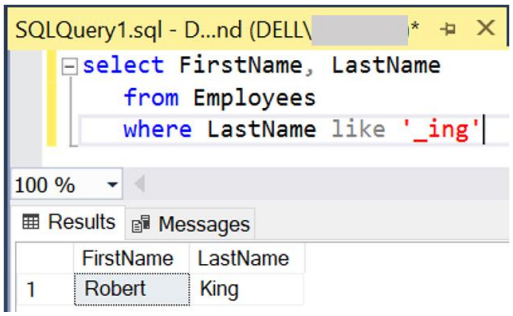



  **Ejemplo #2:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el ID, el nombre y el apellido de todos los empleados, cuyo nombre contiene la subcadena “ne”.

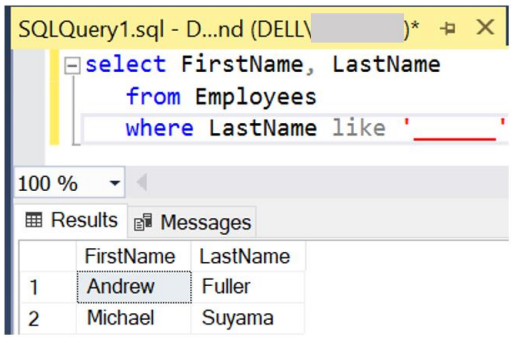



- **Guión bajo (_):** Encuentra coincidencias con cualquier carácter.

  **Ejemplo #1:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre y el apellido de todos los empleados, cuyo apellido contiene un solo carácter antes de la subcadena “ing”.



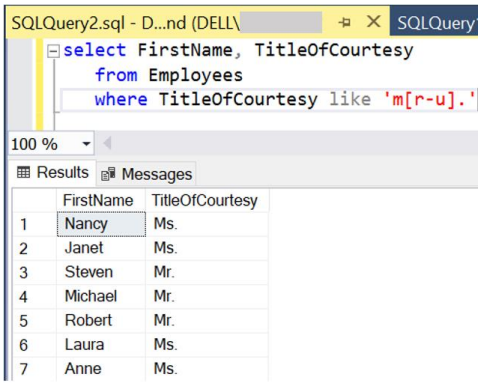

**Ejemplo #2:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre y el apellido de todos los empleados, cuyo apellido contiene exactamente seis caracteres.




Observe que en la Figura 51, se coloca seis veces el carácter de guión bajo.

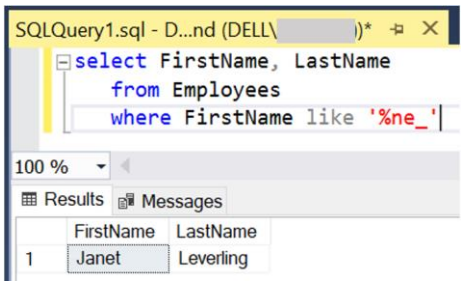
- **Guión(-):** Muestra coincidencias con el rango de caracteres especificado, incluyendo esos caracteres.

**Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre de todos los empleados, cuyo nombre empieza por la letra “m” y luego contiene cualquier letra entre “r” y “u”.



Estos caracteres especiales pueden utilizarse juntos, como se muestra en los siguientes ejemplos:

**Ejemplo #1:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre y el apellido de todos los empleados cuyo nombre contiene cualquier carácter o caracteres antes de la subcadena “ne” y tiene un solo carácter después de esta subcadena.

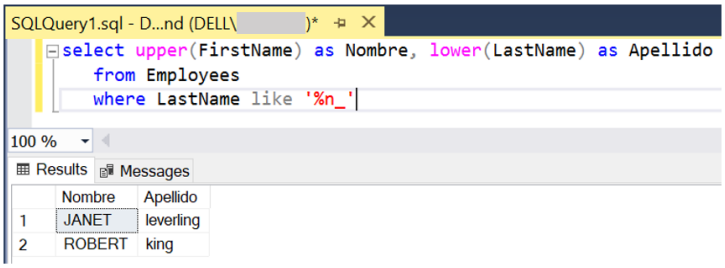



**Ejemplo #2:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre de todos los empleados, cuyo nombre empieza por la letra “m” y luego contiene cualquier letra entre “a” y “h”. No importa qué caracteres estén después de estas letras.


También existen funciones que permiten trabajar con cadenas, algunas de estas son las siguientes:

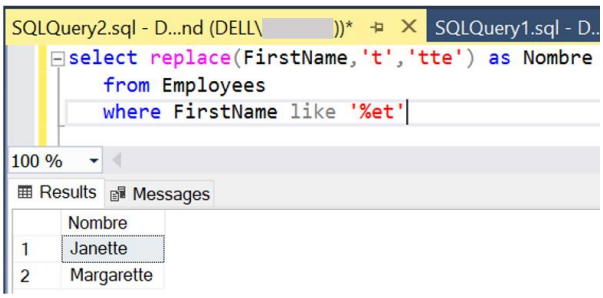
- **Upper(cadena):** Convierte a una cadena en mayúscula.
- **Lower(cadena):** Convierte a una cadena en minúscula.

**Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre (renombrar la columna como “Nombre”) y el apellido (renombrar la columna como “Apellido”) de todos los empleados cuyo nombre contiene cualquier carácter o caracteres antes de la subcadena “n” y tiene un solo carácter después de esta subcadena. Mostrar los nombres en mayúsculas y los apellidos en minúsculas.



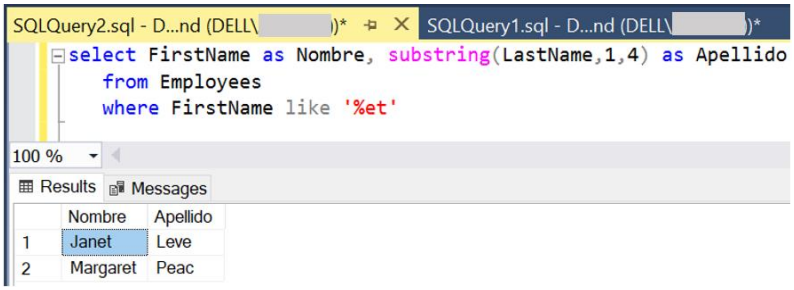

- **Replace(cadena,cadena_anterior,cadena_nueva):** Reemplaza todas las ocurrencias de una subcadena con otra subcadena.

  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre (renombrarlo como “Nombre”) de todos los empleados cuyo nombre contiene cualquier carácter o caracteres antes de la subcadena “et”. Para cada nombre, reemplazar la subcadena “t” por “tte”.



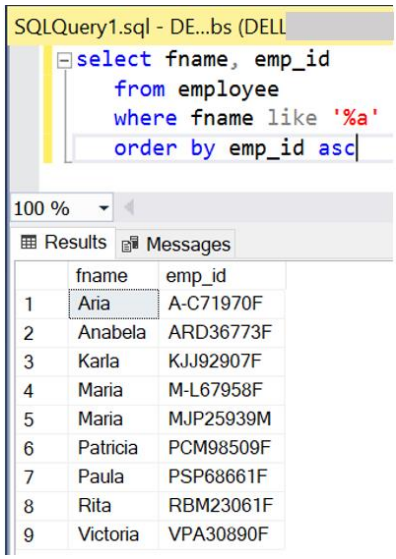

- **Substring(cadena,pos_inicio,cant_caracteres):** Extrae algunos caracteres de una cadena. Se debe indicar la posición de inicio para la extracción de caracteres y la cantidad de caracteres que se deben extraer, considerando que en SQL, la posición inicial de toda cadena es 1.

  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre (renombrarlo como “Nombre”) y el apellido (renombrar la columna como “Apellido”) de todos los empleados cuyo nombre contiene cualquier carácter o caracteres antes de la subcadena “et”. Para cada apellido, mostrar solamente los cuatro primeros caracteres.

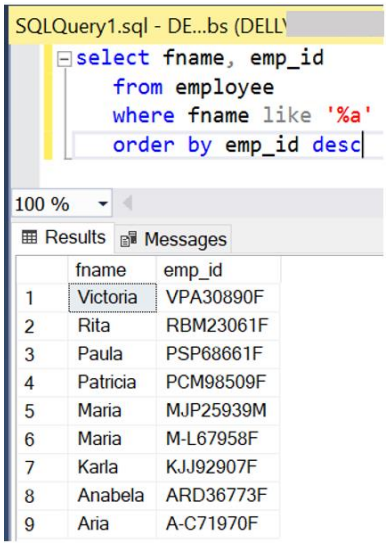



- **Orden en la presentación de tuplas:** Las tuplas que resultan de las consultas que se hagan en SQL, se pueden ordenar utilizando la cláusula *order by*. Por defecto, esta cláusula ordena las tuplas en orden ascendente. Sin embargo, se puede especificar el orden, utilizando *desc* para un orden descendente o *asc* para un orden ascendente.

  **Ejemplo:** Utilizando la base de datos *pubs*, realizar una consulta que muestre el nombre y el ID de todos los empleados, cuyo nombre termine con la letra “a”, en orden ascendente según el ID. Luego, realizar la misma consulta, pero mostrando los resultados en orden descendente según el ID.



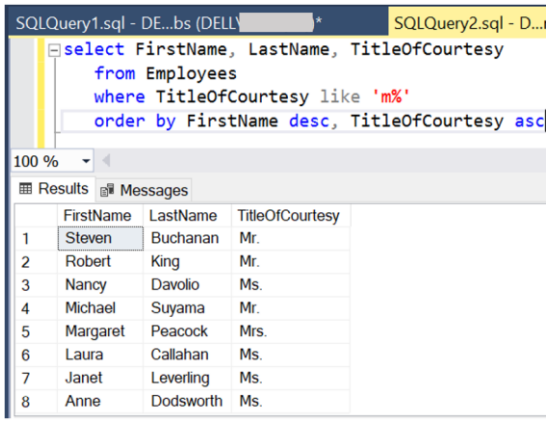

| | fname | emp_id |
|---|---|---|
| 1 | Aria | A-C71970F |
| 2 | Anabela | ARD36773F |
| 3 | Karla | KJJ92907F |
| 4 | Maria | M-L67958F |
| 5 | Maria | MJP25939M |
| 6 | Patricia | PCM98509F |
| 7 | Paula | PSP68661F |
| 8 | Rita | RBM23061F |
| 9 | Victoria | VPA30890F |




| | fname | emp_id |
|---|---|---|
| 1 | Victoria | VPA30890F |
| 2 | Rita | RBM23061F |
| 3 | Paula | PSP68661F |
| 4 | Patricia | PCM98509F |
| 5 | Maria | MJP25939M |
| 6 | Maria | M-L67958F |
| 7 | Karla | KJJ92907F |
| 8 | Anabela | ARD36773F |
| 9 | Aria | A-C71970F |

También se pueden usar las palabras *desc* y *asc* de manera combinada.

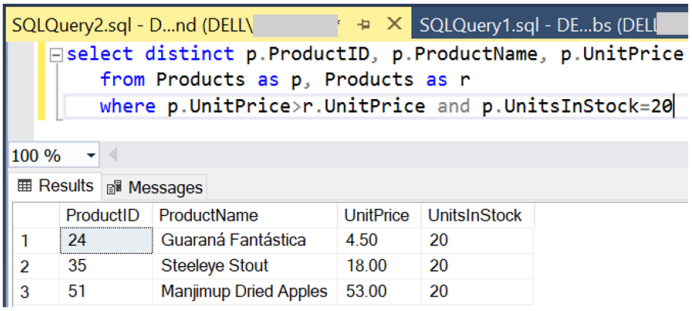
**Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre, el apellido y el título de cortesía de todos los empleados, cuyo título de cortesía inicie con la letra “m”. Mostrar el nombre en orden descendente y el título de cortesía en orden ascendente.




| | FirstName | LastName | TitleOfCourtesy |
|---|---|---|---|
| 1 | Steven | Buchanan | Mr. |
| 2 | Robert | King | Mr. |
| 3 | Nancy | Davolio | Ms. |
| 4 | Michael | Suyama | Mr. |
| 5 | Margaret | Peacock | Mrs. |
| 6 | Laura | Callahan | Ms. |
| 7 | Janet | Leverling | Ms. |
| 8 | Anne | Dodsworth | Ms. |

- **Duplicados:** Muchas veces, al realizar consultas en una base de datos, se pueden obtener tuplas duplicadas. Para evitar esto se utiliza la cláusula *distinct*, que permite seleccionar tuplas que cumplan con lo solicitado en la consulta, sin que se muestren duplicados.

  **Ejemplo:** Utilizando la base de datos *Northwind*, mostrar el ID, el nombre y el precio unitario de todos los productos con unidades en stock iguales a 20, cuyo precio unitario sea mayor al del producto con el menor precio unitario.




| | ProductID | ProductName | UnitPrice | UnitsInStock |
|---|---|---|---|---|
| 1 | 24 | Guaraná Fantástica | 4.50 | 20 |
| 2 | 35 | Steeleye Stout | 18.00 | 20 |
| 3 | 51 | Manjimup Dried Apples | 53.00 | 20 |

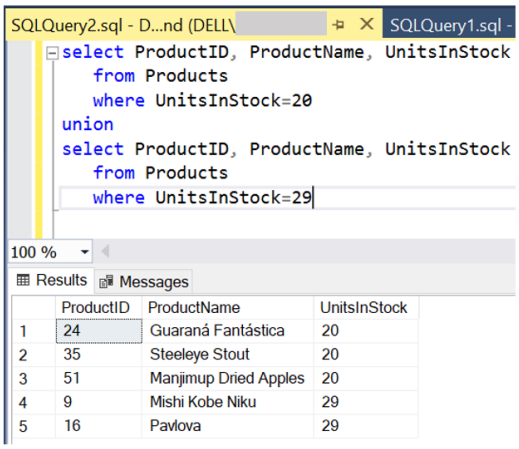
  Este mismo ejemplo, descrito en la página 80 en el punto “Variable tupla”, muestra resultados duplicados ya que no se utilizó la cláusula *distinct*.

**Operaciones sobre conjuntos**

En SQL existen operaciones que permiten operar sobre múltiples tablas y corresponden a las operaciones matemáticas de unión e intersección (Silberschatz et al., 2011).

- **Operación unión:** Permite combinar los resultados mostrados a partir de múltiples *select* y controla los duplicados (Codecademy, 2020c). Los resultados pueden combinarse a partir de una o más tablas. Como explica (W3Schools, 2020d), las cláusulas *select* deben tener la misma cantidad de columnas y estas deben ser del mismo tipo de dato y deben estar en el mismo orden.

**Ejemplo #1:** Utilizando la base de datos *Northwind*, realizar una consulta que permita mostrar el ID, el nombre del producto y las unidades en stock, de todos los productos cuya unidad en stock sea igual a 20 o igual a 29.




| | ProductID | ProductName | UnitsInStock |
|---|---|---|---|
| 1 | 24 | Guaraná Fantástica | 20 |
| 2 | 35 | Steeleye Stout | 20 |
| 3 | 51 | Manjimup Dried Apples | 20 |
| 4 | 9 | Mishi Kobe Niku | 29 |
| 5 | 16 | Pavlova | 29 |

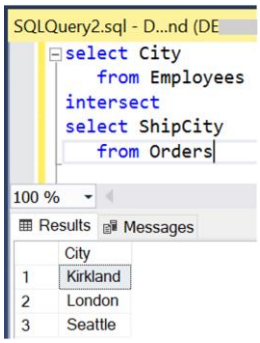
**Ejemplo #2:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre todas las ciudades en las tablas Employees y Orders.


| | City |
|---|---|
| 1 | Aachen |
| 2 | Albuquerque |
| 3 | Anchorage |
| 4 | Århus |
| 5 | Barcelona |
| 6 | Barquisimeto |
| 7 | Bergamo |
| 8 | Berlin |
| 9 | Bern |
| 10 | Boise |
| 11 | Bräcke |
| 12 | Brandenburg |
| 13 | Bruxelles |
| 14 | Buenos Aires |
| 15 | Butte |
| 16 | Campinas |
| 17 | Caracas |
| 18 | Charleroi |
| 19 | Colchester |
| 20 | Cork |

- **Operación intersección:** Permite combinar los resultados a partir de múltiples consultas y muestra los resultados no duplicados (SQLServerTutorial, 2020). Se muestran los resultados que tienen en común las tablas consultadas. Al igual que la operación anterior, las cláusulas *select* deben tener la misma cantidad de

columnas y estas deben ser del mismo tipo de dato y deben estar en el mismo orden.

**Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre todas las ciudades que tienen en común las tablas *Employees* y *Orders*.

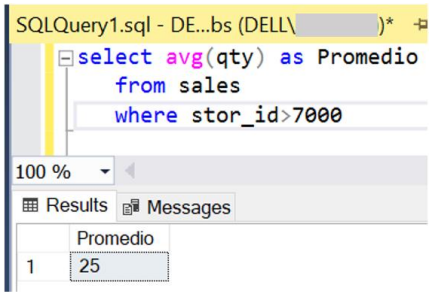



**Funciones de agregación**

Las funciones de agregación son funciones que devuelven un sólo valor a partir del cálculo de un conjunto de valores de entrada (Silberschatz et al., 2011). Estas funciones de agregación toman como argumento el nombre de una columna y pueden utilizarse con otras cláusulas, como *distinct*, para controlar los duplicados en los resultados, o *where* para especificar criterios. A continuación se describen estas funciones de agregación, de acuerdo con lo expuesto por (Codecademy, 2020a):

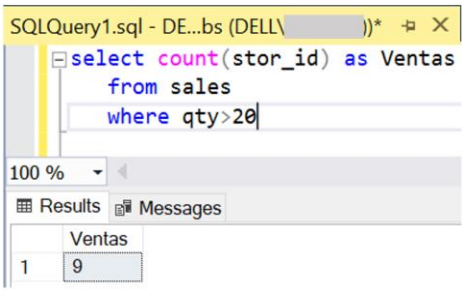
- **Avg():** Devuelve el valor promedio de la columna. El tipo de dato de la columna dentro del argumento debe ser numérico.

  **Ejemplo:** Utilizando la base de datos *pubs*, realizar una consulta que muestre el promedio de la cantidad de productos vendidos, cuyo *stor_id* es mayor que 7000. Renombra la columna como *Promedio*.

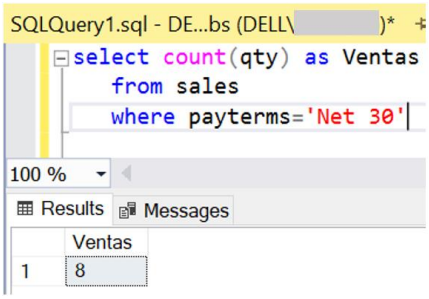



- **Count():** Devuelve la cantidad de tuplas que coinciden con el criterio especificado.

**Ejemplo #1:** Utilizando la base de datos *pubs*, realizar una consulta a través del *stor_id*, *que* muestre la cantidad total de ventas cuyo *qty* es mayor que 20. Renombrar la columna como *Ventas*.

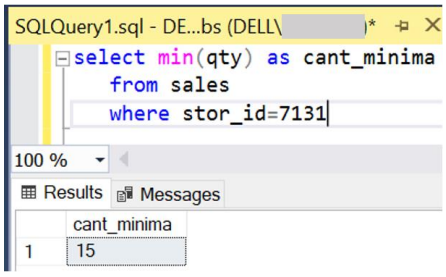



**Ejemplo #2:** Utilizando la base de datos *pubs*, realizar una consulta a través del *qty*, *que* muestre la cantidad de registros cuyo *payterms* es “Net 30”. Renombrar la columna como *Ventas*.



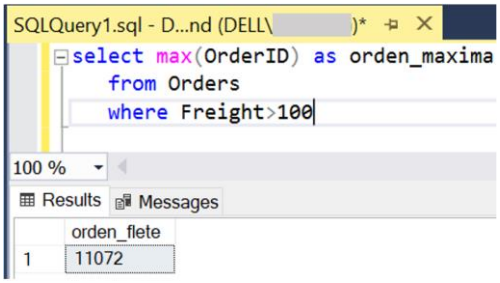

- **Min():** Devuelve el valor mínimo en la columna.

  **Ejemplo:** Utilizando la base de datos *pubs*, realizar una consulta que muestre el *qty* mínimo de entre todas las ventas cuyo *stor_id* es 7131. Renombrar la columna como *cant_minima*.




- **Max():** Devuelve el valor máximo en la columna.

**Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el número de orden, cuyo *Freight* es el mínimo de entre todos los *Freight* mayores que 100. Renombrar la columna como *orden_maxima*.




- **Sum():** Devuelve el valor de la suma de todos los valores en la columna. El tipo de dato de la columna dentro del argumento debe ser numérico.

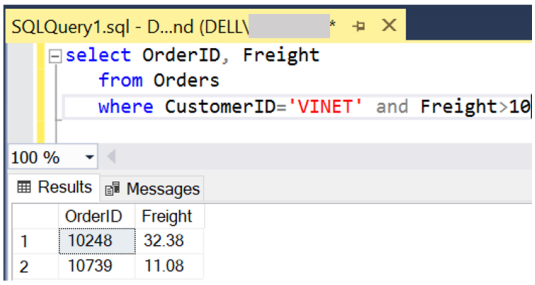
  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre la suma total del flete (*Freight*), para todos los *CustomerID* igual a "VINET". Renombrar la columna como *suma_flete*.


**Valores nulos**

Muchas veces, dentro de las tablas de una base de datos pueden existir campos que no tengan ningún valor almacenado, es decir, contienen valores nulos, los cuales pueden generar problemas al realizar diversas operaciones (Silberschatz et al., 2011). Para ello, en SQL existen diversas formas de trabajar con estos tipos de valores, como es el caso de los operadores lógicos, que suelen utilizarse en la cláusula *where*. Los operadores lógicos permiten hacer comparaciones de valores no nulos con valores nulos, dando como resultado un valor desconocido, tal como describe (Silberschatz et al., 2011). Estos operadores lógicos son los que permiten realizar operaciones booleanas y se describen a continuación, basado en lo explicado en (W3Schools, 2020a):

- **And:** Muestra resultados si todas las condiciones separadas por el operador son ciertas.

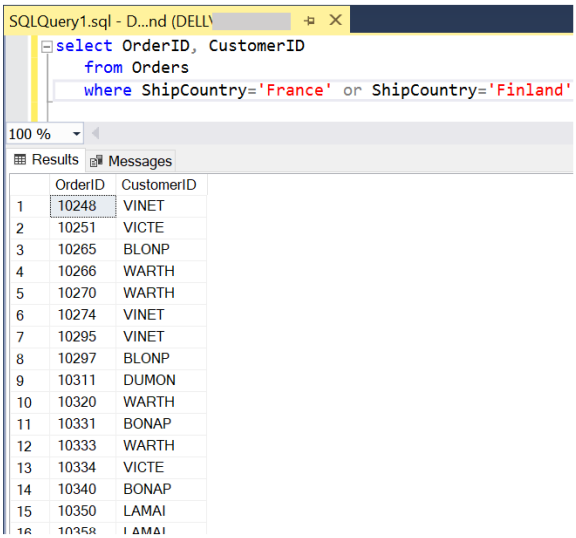
**Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el ID de la orden y el flete (*Freight*) de todas las órdenes cuyo *CustomerID* sea “VINET” y con un *Freight* mayor que 10.




| | OrderID | Freight |
|---|---|---|
| 1 | 10248 | 32.38 |
| 2 | 10739 | 11.08 |

- **Or:** Muestra resultados si alguna de las condiciones separadas por el operador es cierta.

  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el ID de la orden y el ID del cliente, que tengan como país de envío a “France” o a “Finland”.




| | OrderID | CustomerID |
|---|---|---|
| 1 | 10248 | VINET |
| 2 | 10251 | VICTE |
| 3 | 10265 | BLONP |
| 4 | 10266 | WARTH |
| 5 | 10270 | WARTH |
| 6 | 10274 | VINET |
| 7 | 10295 | VINET |
| 8 | 10297 | BLONP |
| 9 | 10311 | DUMON |
| 10 | 10320 | WARTH |
| 11 | 10331 | BONAP |
| 12 | 10333 | WARTH |
| 13 | 10334 | VICTE |
| 14 | 10340 | BONAP |
| 15 | 10350 | LAMAI |
| 16 | 10358 | LAMAI |

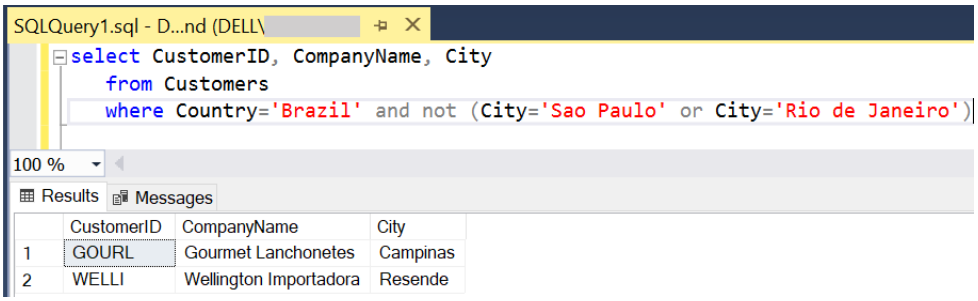
- **Not:** Muestra resultados si la condición no es cierta.

  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el ID de la orden y el ID del cliente, que no tengan como país de envío a “Germany”.


| | OrderID | CustomerID |
|---|---|---|
| 1 | 10248 | VINET |
| 2 | 10250 | HANAR |
| 3 | 10251 | VICTE |
| 4 | 10252 | SUPRD |
| 5 | 10253 | HANAR |
| 6 | 10254 | CHOPS |
| 7 | 10255 | RICSU |
| 8 | 10256 | WELLI |
| 9 | 10257 | HILAA |
| 10 | 10258 | ERNSH |
| 11 | 10259 | CENTC |
| 12 | 10261 | QUEDE |
| 13 | 10262 | RATTC |
| 14 | 10263 | ERNSH |
| 15 | 10264 | FOLKO |
| 16 | 10265 | BLONP |

Estos operadores lógicos también pueden combinarse en una sola consulta, como se muestra a continuación.

**Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el ID del cliente, el nombre de la compañía y la ciudad, de todos los clientes cuyo país sea “Brazil”, pero que no sean de la ciudad de “Sao Paulo” o de “Rio de Janeiro”.

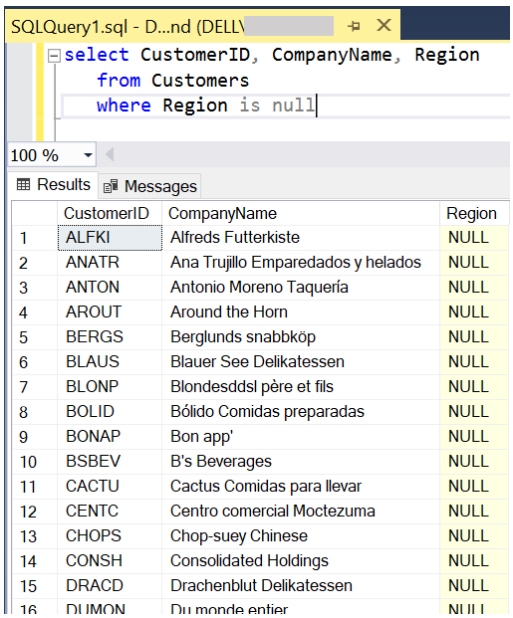



| | CustomerID | CompanyName | City |
|---|---|---|---|
| 1 | GOURL | Gourmet Lanchonetes | Campinas |
| 2 | WELLI | Wellington Importadora | Resende |

Existen otros operadores que también permiten trabajar con valores nulos, como lo son el operador *is null* y el operador *is not null*. A continuación se describen estos operadores, de acuerdo con lo explicado por (Elmasri & Navathe, 2016) y (W3Schools, 2020c):

- **Is null:** Permite comprobar si hay valores nulos.

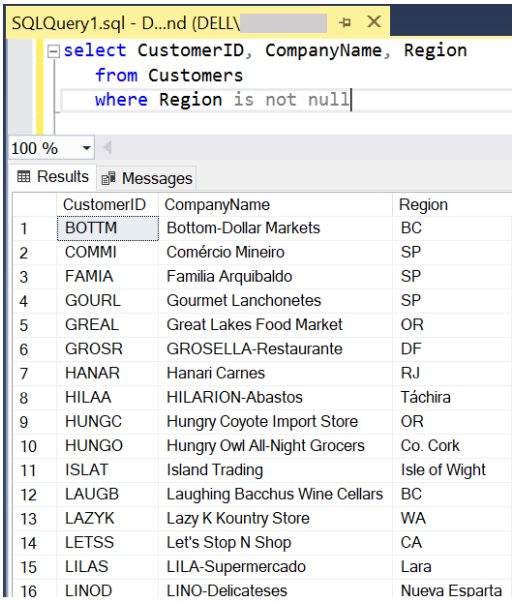
  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el ID del cliente, el nombre de la compañía y la región, de todos los clientes cuya región esté vacía.




| | CustomerID | CompanyName | Region |
|---|---|---|---|
| 1 | ALFKI | Alfreds Futterkiste | NULL |
| 2 | ANATR | Ana Trujillo Emparedados y helados | NULL |
| 3 | ANTON | Antonio Moreno Taquería | NULL |
| 4 | AROUT | Around the Horn | NULL |
| 5 | BERGS | Berglunds snabbköp | NULL |
| 6 | BLAUS | Blauer See Delikatessen | NULL |
| 7 | BLONP | Blondesddsl père et fils | NULL |
| 8 | BOLID | Bólido Comidas preparadas | NULL |
| 9 | BONAP | Bon app' | NULL |
| 10 | BSBEV | B's Beverages | NULL |
| 11 | CACTU | Cactus Comidas para llevar | NULL |
| 12 | CENTC | Centro comercial Moctezuma | NULL |
| 13 | CHOPS | Chop-suey Chinese | NULL |
| 14 | CONSH | Consolidated Holdings | NULL |
| 15 | DRACD | Drachenblut Delikatessen | NULL |
| 16 | DUMON | Du monde entier | NULL |

- **Is not null:** Permite comprobar si hay valores no nulos.

  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el ID del cliente, el nombre de la compañía y la región, de todos los clientes cuya región no esté vacía.



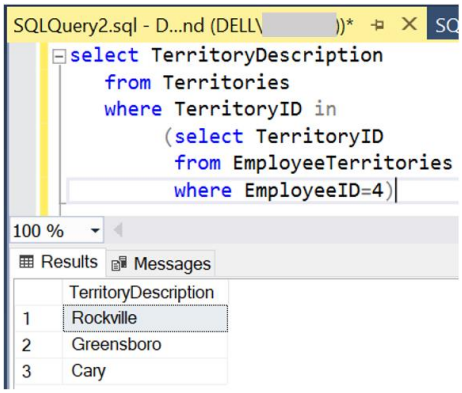

| | CustomerID | CompanyName | Region |
|---|---|---|---|
| 1 | BOTTM | Bottom-Dollar Markets | BC |
| 2 | COMMI | Comércio Mineiro | SP |
| 3 | FAMIA | Familia Arquibaldo | SP |
| 4 | GOURL | Gourmet Lanchonetes | SP |
| 5 | GREAL | Great Lakes Food Market | OR |
| 6 | GROSR | GROSELLA-Restaurante | DF |
| 7 | HANAR | Hanari Carnes | RJ |
| 8 | HILAA | HILARION-Abastos | Táchira |
| 9 | HUNGC | Hungry Coyote Import Store | OR |
| 10 | HUNGO | Hungry Owl All-Night Grocers | Co. Cork |
| 11 | ISLAT | Island Trading | Isle of Wight |
| 12 | LAUGB | Laughing Bacchus Wine Cellars | BC |
| 13 | LAZYK | Lazy K Kountry Store | WA |
| 14 | LETSS | Let's Stop N Shop | CA |
| 15 | LILAS | LILA-Supermercado | Lara |
| 16 | LINOD | LINO-Delicateses | Nueva Esparta |

### Subconsultas anidadas

Una subconsulta anidada es una consulta que está dentro de otra consulta y permiten probar la pertenencia a conjuntos, comparación de conjuntos y comprobación de relaciones vacías o de tuplas duplicadas (Silberschatz et al., 2011). Estos se definen a continuación, de acuerdo con lo descrito por (Silberschatz et al., 2011):

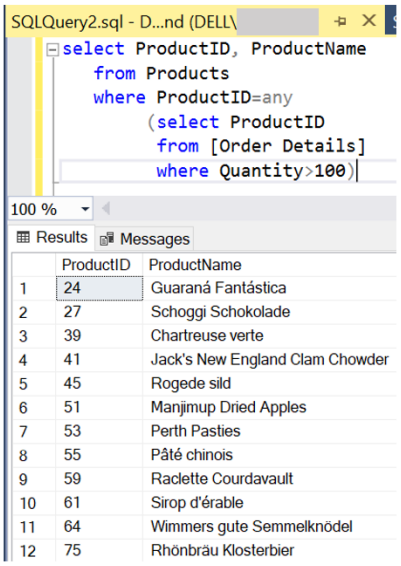
- **Pertenencia a conjuntos:** Permite probar la pertenencia de tuplas en una relación, para lo cual se puede utilizar el operador *in*.

  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre de los territorios (“TerritoryDescription”) del empleado con ID igual a 4.



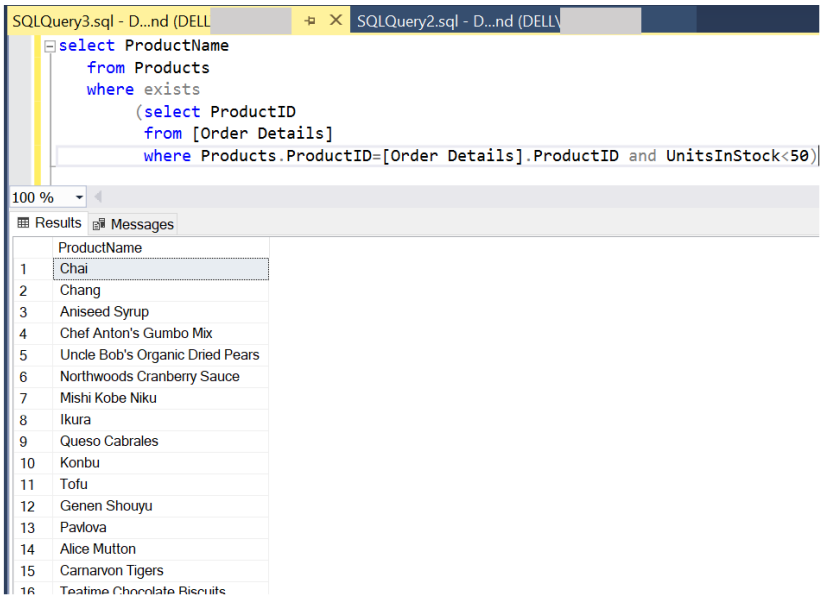

- **Comparación de conjuntos:** Para comparar conjuntos se utilizan los operadores de comparación junto con alguno de los siguientes operadores: *all* o *any*. *All* muestra los registros si todos cumplen las condiciones dentro de la subconsulta, mientras que *any* muestra los registros si alguno de estos cumple con alguna de las condiciones dentro de la subconsulta (W3Schools, 2020b).

  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre y el ID de los productos que tienen una orden con cantidades mayores a 100.



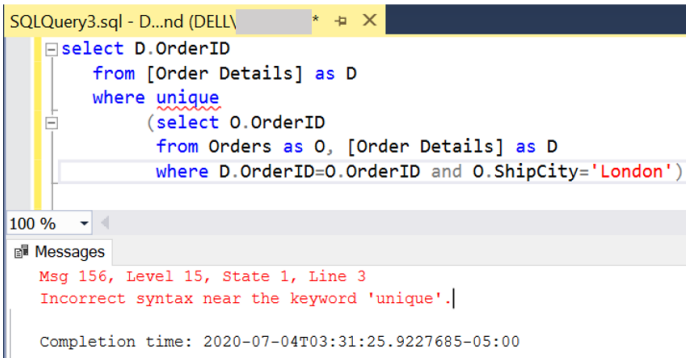
- **Comprobación de relaciones vacías:** SQL permite verificar si una subconsulta tiene tuplas en su resultado. Para ello se utiliza el operador *exists*, que muestra los registros que resultan de la subconsulta, si estos no están vacíos.

  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el nombre de los productos que tengan unidades en stock menores que 50 y que estén registrados en los detalles de las órdenes (“Order Details”).




- **Comprobación de tuplas duplicadas:** Para verificar si hay tuplas duplicadas en el resultado de una subconsulta, se utiliza el operador *unique*.

  **Ejemplo:** Utilizando la base de datos *Northwind*, realizar una consulta que muestre el ID de las órdenes que se enviaron una sola vez a Londres (“London”).



### *Vistas*

Una vista es una tabla virtual que se deriva a partir de los datos de otras tablas que han sido previamente creadas con la instrucción *create table* (Teorey et al., 2011). De acuerdo con (Hernandez, 2003) y (Teorey et al., 2011), algunas de las razones por las que las vistas son útiles son:

- Permiten establecer consultas complejas de antemano, de modo que un usuario principiante de SQL pueda invocarlas a través de consultas más sencillas.
- Proveen mayor seguridad en el uso de una base de datos, ya que el administrador de la base de datos puede controlar los datos que cada usuario ve, a través de la asignación de diferentes vistas.
- Proveen independencia de los datos, ya que se pueden modificar, actualizar y eliminar datos de las tablas base, pero no es necesario cambiar la consulta de la vista. Si es necesario cambiar la definición de la vista, esto debe hacerlo el administrador de la base de datos.
- Permiten trabajar simultáneamente con datos que provengan de múltiples tablas, siempre que estas estén relacionadas.
- Reflejan información actualizada, es decir, cualquier cambio que se haga en las tablas base se refleja en la vista.

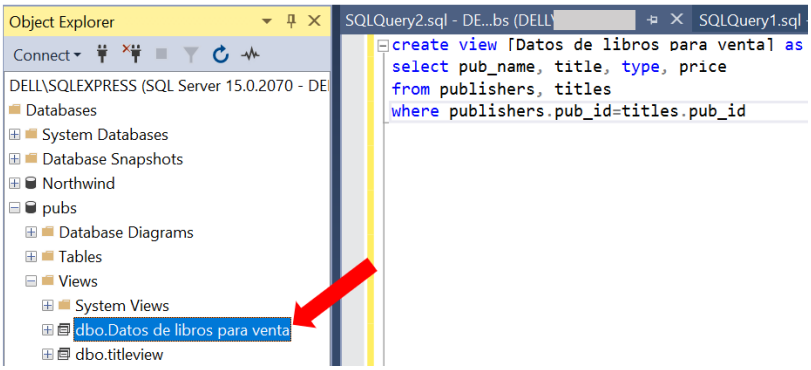
Una columna de una vista que deriva de la columna de una tabla base, hereda las restricciones que se apliquen a esa columna de la tabla base (Sumathi & Esakkirajan, 2007).

Para la creación de una vista se utiliza la sentencia *create view*, como se indica con la sintaxis siguiente:

```
create view nombre_vista as
select nombre_columna1, nombre_columna2, nombre_columnaN
from nombre_tabla
where condición
```

**Ejemplo #1:** Utilizando la base de datos *Northwind*, crear una vista a partir de la tabla *Customers*, denominada *clientes_suramerica* y que contenga el nombre de la compañía, el nombre de contacto, el título del contacto, la ciudad y el país. Los países de América del Sur contenidos en la tabla *Customers* son los siguientes: “Argentina”, “Brazil”, “Mexico” y “Venezuela”.


**Ejemplo #2:** Utilizando la base de datos *pubs*, crear una vista a partir de las tablas *publishers* y *titles*, denominada *Datos de libros para venta* y que contenga el nombre de la editora, el título del libro, el tipo y el precio.




Una vez se han creado las vistas, se pueden hacer consultas, utilizando la misma sintaxis para las consultas que se hacen en las tablas de las bases de datos. A continuación se muestran ejemplos, utilizando las vistas creadas en los ejemplos anteriores.

**Ejemplo #1:** Realizar una consulta que muestre todas las columnas de la vista *clientes_suramerica*, creada en la base de datos *Northwind*. Luego hacer una consulta que muestre el nombre de la compañía y el nombre de contacto de todos los clientes que son de “Buenos Aires” o “Sao Paulo”.

SQLQuery1.sql - D...nd (DELL\ SQLQuery2.sql - DE...bs (DELL\

```
select * from clientes_suramerica
```

100 %

Results Messages

| | CompanyName | ContactName | ContactTitle | City | Country |
|---|---|---|---|---|---|
| 1 | Ana Trujillo Emparedados y helados | Ana Trujillo | Owner | México D.F. | Mexico |
| 2 | Antonio Moreno Taquería | Antonio Moreno | Owner | México D.F. | Mexico |
| 3 | Cactus Comidas para llevar | Patricio Simpson | Sales Agent | Buenos Aires | Argentina |
| 4 | Centro comercial Moctezuma | Francisco Chang | Marketing Manager | México D.F. | Mexico |
| 5 | Comércio Mineiro | Pedro Afonso | Sales Associate | Sao Paulo | Brazil |
| 6 | Familia Arquibaldo | Aria Cruz | Marketing Assistant | Sao Paulo | Brazil |
| 7 | Gourmet Lanchonetes | André Fonseca | Sales Associate | Campinas | Brazil |
| 8 | GROSELLA-Restaurante | Manuel Pereira | Owner | Caracas | Venezuela |
| 9 | Hanari Carnes | Mario Pontes | Accounting Manager | Rio de Janeiro | Brazil |
| 10 | HILARION-Abastos | Carlos Hernández | Sales Representative | San Cristóbal | Venezuela |
| 11 | LILA-Supermercado | Carlos González | Accounting Manager | Barquisimeto | Venezuela |
| 12 | LINO-Delicateses | Felipe Izquierdo | Owner | I. de Margarita | Venezuela |
| 13 | Océano Atlántico Ltda. | Yvonne Moncada | Sales Agent | Buenos Aires | Argentina |
| 14 | Pericles Comidas clásicas | Guillermo Ferná... | Sales Representative | México D.F. | Mexico |
| 15 | Que Delícia | Bernardo Batista | Accounting Manager | Rio de Janeiro | Brazil |
| 16 | Queen Cozinha | Lúcia Carvalho | Marketing Assistant | Sao Paulo | Brazil |
| 17 | Rancho grande | Sergio Gutiérrez | Sales Representative | Buenos Aires | Argentina |
| 18 | Ricardo Adocicados | Janete Limeira | Assistant Sales Ag... | Rio de Janeiro | Brazil |
| 19 | Tortuga Restaurante | Miguel Angel Pa... | Owner | México D.F. | Mexico |
| 20 | Tradição Hipermercados | Anabela Domin... | Sales Representative | Sao Paulo | Brazil |
| 21 | Wellington Importadora | Paula Parente | Sales Manager | Resende | Brazil |

SQLQuery1.sql - D...nd (DELL\ ))* SQLQuery2.sql - DE...b

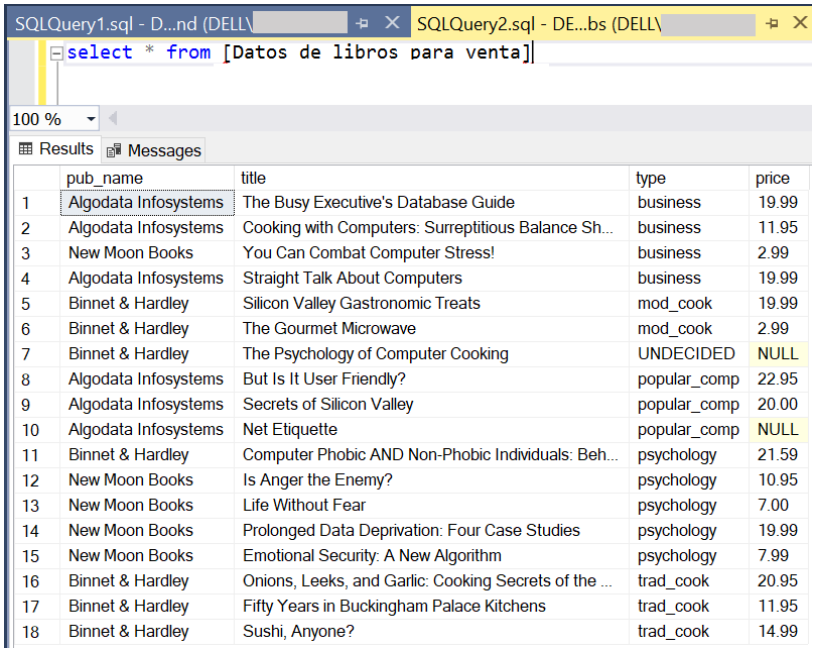
```
select CompanyName, ContactName
    from clientes_suramerica
    where City='Buenos Aires' or City='Sao Paulo'
```

100 %

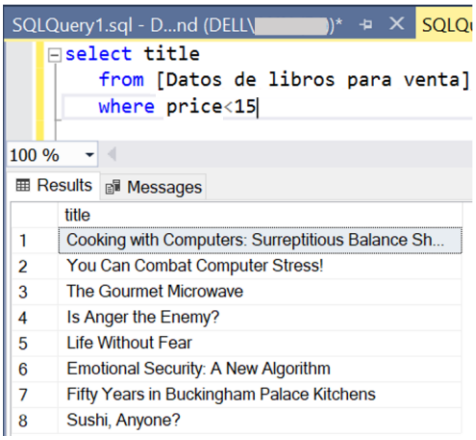
Results Messages

| | CompanyName | ContactName |
|---|---|---|
| 1 | Cactus Comidas para llevar | Patricio Simpson |
| 2 | Comércio Mineiro | Pedro Afonso |
| 3 | Familia Arquibaldo | Aria Cruz |
| 4 | Océano Atlántico Ltda. | Yvonne Moncada |
| 5 | Queen Cozinha | Lúcia Carvalho |
| 6 | Rancho grande | Sergio Gutiérrez |
| 7 | Tradição Hipermercados | Anabela Domingues |

**Ejemplo #2:** Realizar una consulta que muestre todas las columnas de la vista *Datos de libros para venta*, creada en la base de datos *pubs*. Luego hacer una consulta que muestre el título de todos los libros cuyo precio sea menor que 15.




| | pub_name | title | type | price |
|---|---|---|---|---|
| 1 | Algodata Infosystems | The Busy Executive's Database Guide | business | 19.99 |
| 2 | Algodata Infosystems | Cooking with Computers: Surreptitious Balance Sh... | business | 11.95 |
| 3 | New Moon Books | You Can Combat Computer Stress! | business | 2.99 |
| 4 | Algodata Infosystems | Straight Talk About Computers | business | 19.99 |
| 5 | Binnet & Hardley | Silicon Valley Gastronomic Treats | mod_cook | 19.99 |
| 6 | Binnet & Hardley | The Gourmet Microwave | mod_cook | 2.99 |
| 7 | Binnet & Hardley | The Psychology of Computer Cooking | UNDECIDED | NULL |
| 8 | Algodata Infosystems | But Is It User Friendly? | popular_comp | 22.95 |
| 9 | Algodata Infosystems | Secrets of Silicon Valley | popular_comp | 20.00 |
| 10 | Algodata Infosystems | Net Etiquette | popular_comp | NULL |
| 11 | Binnet & Hardley | Computer Phobic AND Non-Phobic Individuals: Beh... | psychology | 21.59 |
| 12 | New Moon Books | Is Anger the Enemy? | psychology | 10.95 |
| 13 | New Moon Books | Life Without Fear | psychology | 7.00 |
| 14 | New Moon Books | Prolonged Data Deprivation: Four Case Studies | psychology | 19.99 |
| 15 | New Moon Books | Emotional Security: A New Algorithm | psychology | 7.99 |
| 16 | Binnet & Hardley | Onions, Leeks, and Garlic: Cooking Secrets of the ... | trad_cook | 20.95 |
| 17 | Binnet & Hardley | Fifty Years in Buckingham Palace Kitchens | trad_cook | 11.95 |
| 18 | Binnet & Hardley | Sushi, Anyone? | trad_cook | 14.99 |

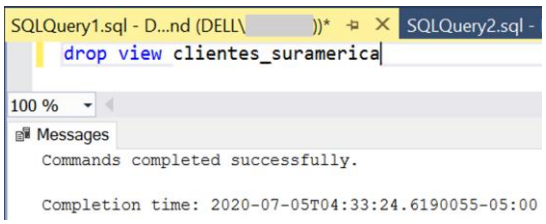


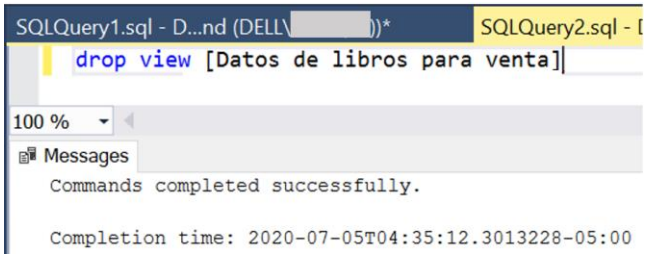

| | title |
|---|---|
| 1 | Cooking with Computers: Surreptitious Balance Sh... |
| 2 | You Can Combat Computer Stress! |
| 3 | The Gourmet Microwave |
| 4 | Is Anger the Enemy? |
| 5 | Life Without Fear |
| 6 | Emotional Security: A New Algorithm |
| 7 | Fifty Years in Buckingham Palace Kitchens |
| 8 | Sushi, Anyone? |

Finalmente, para borrar una vista, se debe utilizar la siguiente sintaxis:

```
drop view nombre_vista
```

**Ejemplo:** Borrar la vista *clientes_suramerica*, creada en la base de datos *Northwind* y luego borrar la vista *Datos de libros para venta*, creada en la base de datos *pubs*.






## *Comandos de modificación de la base de datos*

Estos comandos o sentencias permiten agregar, eliminar o cambiar información con SQL en la base de datos (Silberschatz et al., 2011).

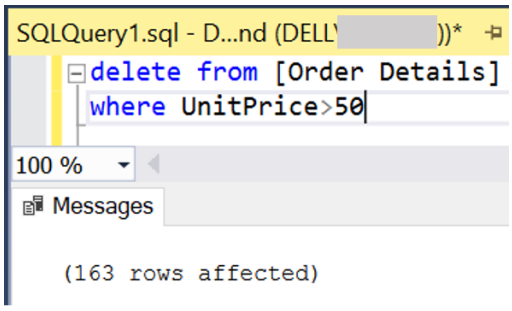
### Borrado (DELETE)

Este comando permite eliminar tuplas completas, mas no es posible eliminar el valor de un atributo; su sintaxis es similar a la de una consulta (Silberschatz et al., 2011):

```
delete from nombre_tabla
where condicion
```

Como explica (Silberschatz et al., 2011), la condición dentro de la cláusula *where* puede ser tan compleja como una cláusula *select*, o bien puede no utilizarse, para lo cual se eliminarían todas las tuplas de la tabla. En el primer caso se eliminarían todas la tuplas que cumplan con la condición indicada por la cláusula *where*, mientras que en el último caso, la tabla existiría dentro de la base de datos, pero estaría vacía.

- **Ejemplo:** Utilizando la base de datos *Northwind*, eliminar todos los productos de la tabla *Order Details*, cuyo precio unitario sea mayor que 50.




### Inserción (INSERT)

Este comando permite insertar tuplas en una tabla, se utiliza la cláusula *values* y los valores a insertar deben ser del tipo de dato apropiado para cada columna (Mannino, 2007). Para insertar valores en una base de datos, se utiliza la siguiente sintaxis:

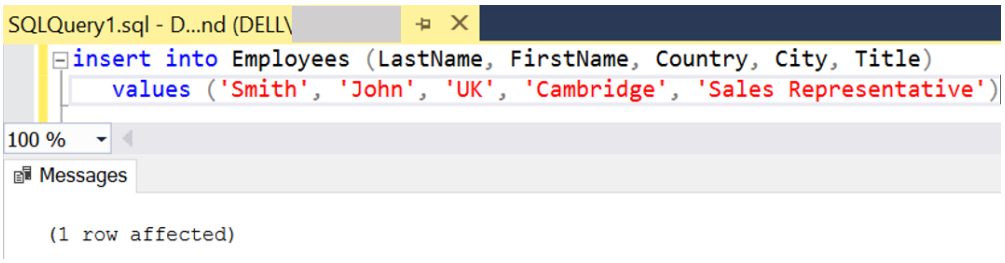
```
insert into nombre_tabla (nombre_columna1, nombre_columna2,…)
   values (valor1, valor2, valor3)
```

Es necesario que los datos estén en el mismo orden de la columna a la que corresponden, pero no importa si no están en el mismo orden en el que están en la tabla.

Esta sintaxis se puede utilizar para insertar datos en todas las columnas o sólo en algunas.

- **Ejemplo:** Utilizando la base de datos *Northwind*, insertar los siguientes datos en la tabla *Employees*:

| Columna | Valor |
|---|---|
| Apellido (LastName) | Smith |
| Nombre (FirstName) | John |
| Título (Title) | Sales Representative |
| Ciudad (City) | Cambridge |
| País (Country) | UK |

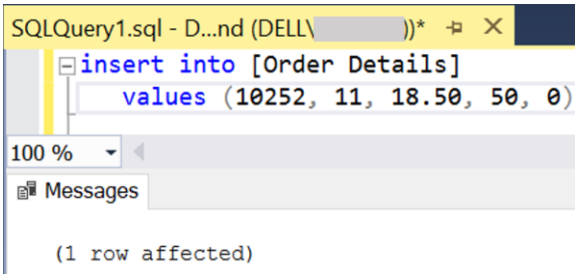



Si se van a insertar datos en todas las columnas de la tabla, se puede omitir colocar el nombre de las columnas y sólo se colocan los valores a insertar. Los valores deben estar en el mismo orden en el que están las columnas en la tabla y deben ser del tipo de dato apropiado para cada columna. La sintaxis sería la siguiente:

```
insert into nombre_tabla
   values valor_columna1, valor_columna2,…valor_columnaN
```

- **Ejemplo:** Utilizando la base de datos *Northwind*, insertar los siguientes datos en la tabla *Customers*:

| Columna | Valor |
|---|---|
| OrderID | 10252 |
| ProductID | 11 |
| Precio Unitario (UnitPrice) | 18.50 |
| Quantity | 50 |
| Discount | 0 |



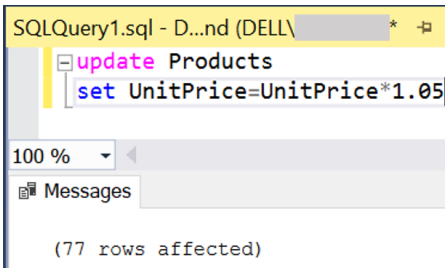

**Actualizaciones (UPDATE)**

Este comando permite la modificación de una o más tuplas en una o más columnas, sin embargo, es necesario tomar en cuenta que si se desea modificar la llave primaria, es probable que las reglas para actualización de las tuplas referenciadas no lo permitan (Mannino, 2007). La sintaxis para modificar una tabla es la siguiente:

```
update nombre_tabla
set nombre_columna1=valor1, nombre_columna2=valor2,…
where condicion
```

Si no se utiliza la cláusula *where*, se actualizan todos los registros de la tabla, ya que la cláusula permite especificar los registros que se deben actualizar.

- **Ejemplo #1:** Utilizando la base de datos *Northwind*, asignarles a todos los productos un aumento del 5 por ciento en el precio unitario (UnitPrice), en la tabla *Products*.




- **Ejemplo #2:** Utilizando la base de datos *Northwind*, modificar el *CustomerID* a “VINET” y el *EmployeeID* a “9”, para todas las órdenes cuyo país de envío (ShipCountry) sea “Germany”, en la tabla *Orders*.

**Actualización de vistas**

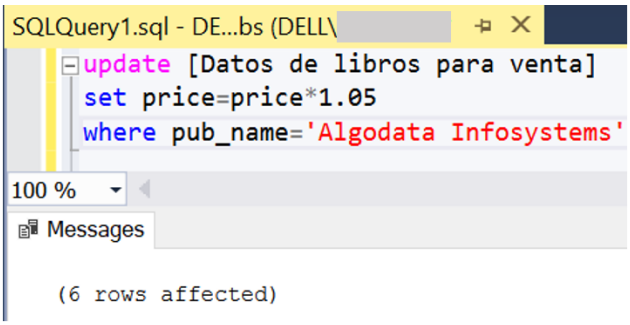
En el caso de las vistas, existen algunas condiciones para que a estas se les pueda aplicar el comando de modificación. Las condiciones son las siguientes, de acuerdo con (Silberschatz et al., 2011):

a) En la cláusula *select*, no debe tener funciones de agregación, expresiones o la cláusula *distinct*.
b) La cláusula *from* sólo tiene una tabla.
c) No hay subconsultas.
d) Los atributos no listados en la cláusula *select* no deben tener la restricción *not null* y no deben ser parte de una llave primaria.

La sintaxis tanto para insertar como para modificar datos en una vista, es similar a la utilizada para las tablas. Para modificar datos en una vista se utiliza la siguiente sintaxis:

```
update nombre_vista
set nombre_columna1=valor1, nombre_columna2=valor2,…
where condicion
```

**Ejemplo:** Utilizando la vista *Datos de libros para venta*, en la base de datos *pubs*, asignarle un aumento del 5 por ciento al precio de todos los libros cuyo *pub_name* sea “Algodata Infosystems”.



# Bibliografía


Bender, C. M., Deco, C., González Sanabria, J. S., & Ponce Gallegos, J. C. (2014). *Tópicos avanzados de bases de datos* (1ra Ed.). Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn).

Camps Paré, R., Casillas Santillán, L. A., Costal Costa, D., Gibert Ginestá, M., Martín Escofet, C., & Pérez Mora, O. (2005). *Bases de Datos*. Formación de Posgrado, Universitat Oberta de Catalunya.

Cardona, H., Masso, J. E., Mera, M. F., Roa, S. M., Ruano, E. F., Torres, M. D., & Vidal, M. I. (2014). *Diseño e Implementación de Bases de Datos desde una Perspectiva Práctica*. Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn).

Churcher, C. (2007). *Beginning Database Design: From Novice to Professional*. Apress.

Codecademy. (2020a). Cheatsheet/Learn SQL: Aggregate Functions. Retrieved from https://www.codecademy.com/learn/learn-sql/modules/learn-sql-aggregate-functions/cheatsheet

Codecademy. (2020b). Cheatsheets/Learn SQL: Manipulation. Retrieved from https://www.codecademy.com/learn/learn-sql/modules/learn-sql-manipulation/cheatsheet

Codecademy. (2020c). Cheatsheets/Learn SQL: Multiple Tables. Retrieved from https://www.codecademy.com/learn/learn-sql/modules/learn-sql-multiple-tables/cheatsheet

Codecademy. (2020d). Cheatsheets/Learn SQL: Queries. Retrieved from https://www.codecademy.com/learn/learn-sql/modules/learn-sql-queries/cheatsheet

Connolly, T., & Begg, C. (2015). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6a Ed.). Pearson Education Limited.

Coronel, C., Morris, S., & Rob, P. (2011). *Database Systems: Design, Implementation and Management* (9a Ed.). Cengage Learning.

Elmasri, R., & Navathe, S. B. (2011). *Fundamentals of Database Systems* (6a Ed.). Addison-Wesley.

Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7a Ed.). Pearson.

Garcia-Molina, H., Ullman, J. D., & Widom, J. (2009). *Database Systems: The Complete Book* (2a Ed.). Pearson Prentice Hall.

Gupta, P., Mata-Toledo, R. A., & Monger, M. D. (2011). Database Development Life Cycle. *Journal of Information Systems and Operations Management*.

Hernandez, M. J. (2003). *Database Design for Mere Mortals: A Hands-On Guide to Relational Database Design* (2nd ed.). Addison-Wesley.

Kroenke, D. M., & Auer, D. J. (2012). *Database Processing: Fundamentals, Design, and Implementation* (12a Ed.). Pearson.

Lumbreras, R. (n.d.). *Material de Apoyo para el curso Introducción al Diseño de Base de Datos*. Dirección de Cómputo para la Docencia, UNAM.

Mannino, M. V. (2007). *Administración de Bases de Datos: Diseño y desarrollo de aplicaciones* (3era Ed.). McGraw-Hill Interamericana.

Marqués, M. (2011). *Bases de datos*. Universitat Jaume I.

Microsoft. (2019). ¿Qué es SQL Server Management Studio (SSMS)? Retrieved from https://docs.microsoft.com/es-es/sql/ssms/sql-server-management-studio-ssms?view=sql-server-ver15

Millán, M. E. (2012). *Fundamentos de Bases de Datos: Notas de referencia*. Programa Editorial Universidad del Valle. https://doi.org/10.25100/peu.47

Mittal, P. (2012). *Analysis and Design of Information Systems*. Excel Books Private Limited for Lovely Professional University.

Ricardo, C. M. (2009). *Base de datos*. McGraw-Hill Educación.

Rosas Hernandez, O. A. (2018). *Bases de Datos*. Compilando Conocimiento.

Sharma, N., Perniu, L., Chong, R. F., Iyer, A., Nandan, C., Mitea, A.-C., … Danubianu, M. (2010). *Database Fundamentals* (1a Ed.). IBM Corporation.

Sharmila, P., & Umarani, R. (2011). A walkthrough of Requirement Elicitation Techniques. *International Journal of Engineering Research and Applications (IJERA)*, *1*(4), 1583–1586.

Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). *Database System Concepts* (6a Ed.). McGraw-Hill.

SQLServerTutorial. (2020). SQL Server INTERSECT. Retrieved from https://www.sqlservertutorial.net/sql-server-basics/sql-server-intersect/

Sumathi, S., & Esakkirajan, S. (2007). *Fundamentals of Relational Database Management Systems*. Springer.

Teorey, T., Lighstone, S., Nadeau, T., & Jagadish, H. V. (2011). *Database Modeling and Design: Logical Design* (5th ed.). Morgan Kaufmann.

Tutorials Point. (2016). *MS SQL Server*.

Universidad Privada TELESUP. (2018). *Modelamiento de Base de Datos*. Universidad Privada TELESUP.

W3Schools. (2020a). SQL AND, OR and NOT Operators. Retrieved from https://www.w3schools.com/sql/sql_and_or.asp

W3Schools. (2020b). SQL ANY and ALL Operators. Retrieved from https://www.w3schools.com/sql/sql_any_all.asp

W3Schools. (2020c). SQL NULL Values. Retrieved from https://www.w3schools.com/sql/sql_null_values.asp

W3Schools. (2020d). SQL UNION Operator. Retrieved from https://www.w3schools.com/sql/sql_union.asp

# Anexos 1: Pruebas Rápidas

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #1

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte: Cierto y Falso

1. _____ La base de datos son una parte esencial de todo sistema de información y le han permitido a organizaciones grandes y pequeñas automatizar sus procedimientos.

2. _____ El enfoque tradicional es el sistema basado en archivos y su estudio es el fundamento para el desarrollo del enfoque en base de datos.

3. _____ La integración de datos permite el control de la redundancia, esto significa que solo se almacenan varias copias de un mismo dato de ser necesario.

4. _____ La base de datos cuenta con propiedades que permiten la protección de los datos, ya se de accesos no autorizados por parte de los usuarios o de programas que puedan hacer mal uso de los datos.

5. _____ El costo del diseño, creación y mantenimiento de una base de datos puede resultar más bajo, que si se asignaran porciones de un presupuesto a todos los departamentos de una organización.

II Parte: Desarrollo

1. Explique cuales son algunas limitaciones que presenta el sistema manual

BUENA SUERTE

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #2

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte: Desarrollo

1. Explique 2 desventajas del uso de un sistema de base datos.
2. Explique 2 ventajas del uso de un sistema de base de datos.
3. Explique brevemente la evolución de la base de datos y las nuevas tendencias
4. Describa que se refiere con DBMS
5. Resuma los tres niveles de arquitectura ANSI/SPARC

BUENA SUERTE

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #3

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte: Llene los espacios

1. Son elementos que asisten al administrado de la base de datos en el mantenimiento del sistema: _____.

2. Son dos ejemplos de utilitarios: _____ y _____.

3. Son funcionalidades adicionales que pueden utilizar los usuarios, el diseñador de la base de datos y el propio DBMS: _____

4. La aplicación de la base de datos se ejecuta en la maquina cliente desde la cual se solicita alguna funcionalidad del sistema de la base de datos en la maquina servidor: _____

5. La máquina cliente es simplemente una interfaz y no se pueden realizar solicitudes a la base de datos de forma directa: _____.

II Parte: Desarrollo

1. Explique todo lo que sabe sobre Esquemas, instancias y estado de una base de datos

BUENA SUERTE

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #4

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte: Cierto y Falso

1. _____ La independencia lógica permite que se realicen modificaciones en el esquema conceptual sin que esto afecte a los esquemas externos.

2. _____ La independencia física permite que se realicen modificaciones en el esquema interno sin esto afecte al esquema conceptual y por consiguiente, a los esquemas externos.

3. _____ El modelo de datos físico representa la forma en que se almacenan los datos, describiendo información tal como la estructura de los registros y las rutas de acceso de estos.

4. _____ El modelo de datos jerárquico representa los datos como un grupo de registros organizados en un grafo.

5. _____ El software y el hardware son dos ejemplos de componentes de puente.

II Parte: Desarrollo

1. Enumere las etapas en el desarrollo de un sistema de base datos

BUENA SUERTE

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #5

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte: Desarrollo

1. Mencione 3 puntos sobre la importancia de la modelización conceptual.
2. Explique dos componentes básicos de un modelo entidad – relación
3. Que entiende por restricciones de cardinalidad, explique
4. Mencione los objetivos de la definición del modelo de datos relacionales.
5. Defina tres de los conceptos básicos del modelo relacional.

BUENA SUERTE

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #6

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte: Selección Multiple

1. Es uno o más atributos que identifican a una tupla en una relación.
   a. Definición de claves b. valores nulos c. dominios

2. Permite definir las características de los atributos de una relación
   a. Definición de claves b. dominios c. otros

3. Cuanto un atributo en una tupla significa que es un valor reconocido
   a.Dominios b. Definición de Claves c. Valores nulos

4. Utiliza una sola relación R para tomar aquellas tuplas que satisfagan la condición especifica.
   a. Selección (σ) b. Proyección (Π) c. Unión (∪)

5. Utiliza dos relaciones R y S para definir otra relación que contiene todas las tuplas de ambas relaciones, sin incluir duplicados
   a. Selección (σ) b. Proyección (Π) c. Unión (∪)

II Parte: Desarrollo

1. Explique lo que es Algebra relacional extendida y describa las operaciones que forman parte de ella.

BUENA SUERTE

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #7

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte: Cierto y Falso

1. _____ La anomalía de inserción ocurre cuando al insertar un registro, se requiere de la adición de datos no relacionados al dicho registro.

2. _____ La anomalía de eliminación ocurre cuando al eliminar un registro, inadvertidamente se eliminan otros datos no relacionados.

3. _____ La anomalía de actualización ocurre cuando al modificar un registro, es necesario modificar otros registros.

4. _____ La normalización puede considerarse como una serie de pasos que se aplican para asegurar que se cumple una determinada forma normal.

5. _____ Las dependencias funcionales también pueden utilizarse para la determinación de claves, o bien para determinar posibles claves.

II Parte: Desarrollo

1. Explique qué significa normalización.

BUENA SUERTE

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #8

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte: Llene los espacios

1. El _____ permite ver administrar los objetos que forman parte de las instancias de SQL Server.

2. Se subdivide en un área para ingresar las instrucciones para la consulta y mantenimiento de las bases de datos: _____.

3. Escriba 3 tipos de datos mas utilizados : _____, _____ y _____.

4. Instrucción que es utilizada para agregar columnas nuevas y eliminar o modificar las columnas existentes: _____.

5. Son tres clausulas de la estructura básica: _____, _____ y _____.

II Parte:

1. Explique brevemente para que se utiliza la variable tupla.

BUENA SUERTE

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #9

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte: Llene los espacios

1. Son dos funciones que nos permiten trabajar con cadenas:
   _____ y _____.

2. Remplaza todas las ocurrencias de una subcadena con otra subcadenad:
   _____

3. Extrae algunos caracteres de una cadena:
   _____

4. Permite combinar los resultados mostrados a partir de múltiples select y controla los duplicados: _____.

5. Permite combinar los resultados a partir de múltiples consultas y muestra los resultados no duplicados: _____

II Parte: Desarrollo

1. Describa las funciones de agregación.

BUENA SUERTE

Universidad Tecnológica de Panamá
Facultad de Ingeniería en Sistemas Computacionales
Herramientas aplicadas a la Inteligencia Artificial
Parcial #10

Nombre: _____ Cédula: _____ Grupo: _____
Profesor: Dr. Carlos A. Rovetto Puntos Obtenidos: _____/

I Parte. Desarrollo

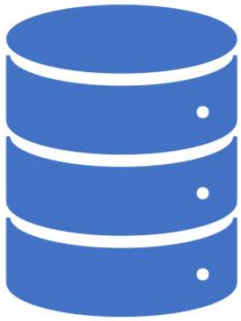
1. Explique en que consiste los valores nulos
2. Explique el operador lógico AND y presente 1 ejemplo del mismo
3. Explique el operador lógico OR y presente 1 ejemplo del mismo
4. Mencione las utilidades de las vistas
5. Describa dos de los comandos de modificación de la base de datos.

BUENA SUERTE

# Anexos 2: Presentaciones

# SISTEMA DE BASE DE DATOS I

Autor: Dr. Carlos A. Rovetto



1

## Capítulo I

2

## Capítulo I: Conceptos sobre base de datos

Las bases de datos son una parte esencial de todo sistema de información y le han permitido a organizaciones grandes y pequeñas automatizar sus procedimientos.

En la actualidad, casi todos los sistemas que utilizamos almacenan datos que pueden ser consultados o recuperados por un usuario y la manera en que se almacenan estos datos es a través de los sistemas de bases de datos.

3

## *Enfoque tradicional vs enfoque de base de datos para el tratamiento de datos e información*

El enfoque tradicional es el sistema basado en archivos y su estudio es el fundamento para el desarrollo del enfoque de base de datos, define al sistema basado en archivos como un conjunto de programas de aplicación, en el que cada programa define y gestiona sus propios datos y le brinda servicios con esos datos al usuario final. Este tipo de enfoque permitió digitalizar al obsoleto sistema en el que se almacenaban datos de forma manual, el cual contaba con muchas limitaciones para el procesamiento de la información.

4

A continuación se definen estas limitaciones, basado en las descripciones de:

**Dependencia de los datos:** Dentro del código en el que se programa la aplicación están definidas la estructura física y la manera en la que se almacenan los archivos con los datos.

**Duplicación de datos:** Debido a la descentralización de los datos en el enfoque tradicional, es necesario que los datos se repiten en diferentes archivos.

**Consultas fijas y proliferación de programas de aplicación:** Para obtener información sobre los datos almacenados se requiere de consultas, las cuales forman parte del código de la aplicación y por lo tanto, es dependiente de su desarrollador.

**Incompatibilidad de formatos de archivos:** Los datos se asocian al lenguaje de programación utilizado para escribir el código de la aplicación porque la estructura de los archivos está ligada a la aplicación.

**Separación y aislamiento de datos:** Esto dificulta el acceso a los datos, pues estos están en archivos separados.

5

## Ventajas de un ambiente de bases de datos

- **Control de redundancia:** La integración de los datos permite el control de la redundancia, esto significa que sólo se almacenan varias copias de un mismo dato si es necesario.
- **Compartición de datos:** La información contenida en la base de datos está distribuida, lo que quiere decir que esta le pertenece a toda la organización.
- **Consistencia de los datos:** Este es el resultado del control de la redundancia, ya que existe coherencia entre los datos.
- **Mejor integridad:** Se logra a través de la definición de restricciones o *constraints*, las cuales no son más que reglas que la base de datos debe cumplir y estas pueden ser aplicadas a las propiedades dentro de un registro
- **Mayor seguridad:** La base de datos cuenta con propiedades que permiten la protección de los datos, ya sea de accesos no autorizados por parte de los usuarios o de programas que puedan hacer mal uso de los datos
- **Equilibrio de requisitos en conflicto:** Los usuarios o los departamentos dentro de una organización, pueden tener necesidades distintas en cuanto a la información contenida en la base de datos
- **Economía en escala:** El costo del diseño, creación y mantenimiento de una base de datos puede resultar más bajo, que si se asignarán porciones de un presupuesto a todos los departamentos de una organización
- **Control sobre la concurrencia:** Los sistemas de bases de datos permiten que varios usuarios ingresen datos de forma simultánea
- **Mejor accesibilidad a los datos:** Los sistemas de bases de datos les permiten a los usuarios la interacción con los datos almacenados, a través de un lenguaje de consulta que le brinda al usuario la información necesaria para tomar decisiones
- **Asignación de estándares:** Corresponde a las reglas de la organización con relación a la representación de los datos almacenados en la base de datos.
- **apidez en el desarrollo de nuevas aplicaciones:** La base de datos de una organización es un patrón para el desarrollo de nuevas aplicaciones, ya que estas almacenan datos que generalmente se requieren para realizar este proceso.
- **Mayor capacidad de respaldo y recuperación:** Los sistemas de bases de datos cuentan con un mecanismo que permite respaldar regularmente la información almacenada.

6

## Desventajas de un ambiente de bases de datos

**Alto costo de software:** Los sistemas de bases de datos son paquetes de software completos y complejos, por lo tanto, son costosos.

**Alto costo de hardware:** Es necesario invertir dinero en hardware que cuente con las capacidades de almacenamiento y procesamiento necesarias para ejecutar el sistema de base de datos.

**Alto costo de programación:** De igual forma, es necesario invertir dinero en la capacitación o contratación de programadores que puedan sacar el mejor provecho al sistema de base de datos. Esto debido a la complejidad de este tipo de software.

**Aumento de vulnerabilidad en la seguridad:** El resto de las aplicaciones que dependan del sistema de base de datos, pueden verse afectadas si alguna de estas falla y dar lugar a datos erróneos dentro de la base de datos.

**Aumento en la dificultad de recuperación:** Cuando la base de datos falla, muchos procesos denominados *transacciones*, pueden estar ocurriendo en ese momento

7

## Evolución de las bases de datos y nuevas tendencias

El uso de computadoras dentro de grandes y pequeñas organizaciones permitió automatizar el procesamiento de la información y por ello, es importante conocer cómo se originó lo que hoy se conoce como base de datos

| Rango de tiempo | Características |
|---|---|
| 1950 a inicios de 1960 | • Se utilizaban cintas magnéticas y tarjetas perforadas para el almacenamiento de datos.<br>Para procesar los datos estos se leían a partir de una o más cintas y se escribían en una cinta nueva. En el caso de las tarjetas perforadas, la salida de los datos era a través de la impresión.<br>Las cintas magnéticas y las tarjetas perforadas sólo se podían leer de forma secuencial y los volúmenes de datos eran mucho mayores que los de la memoria principal. |
| Finales de 1960 a 1970 | • El procesamiento de datos dio un gran giro gracias al incremento en el uso de los discos duros, ya que estos permitían acceder de forma directa a los datos.<br>Disminuyó el tiempo de acceso a los datos, debido a que estos ya no debían ser leídos de forma secuencial. Era posible crear redes y bases de datos jerárquicas que permitían que programadores manipularan los datos, a través de estructuras de datos como listas y árboles.<br>Edgar F. Codd definió en un artículo las bases del modelo relacional y las técnicas para la consulta de datos en este modelo, dando así origen a las bases de datos relacionales. |
| Década de 1980 | • IBM desarrolló un sistema de base de datos denominado System R, como un proyecto para la aplicación de técnicas para la construcción de una base de datos relacional eficiente. Esto surgió tras la publicación del artículo de Codd, el cual aunque era de gran interés académico, no pudo ser aplicado de forma práctica al inicio debido a varias limitaciones por el uso de las bases de datos de red y jerárquicas.<br>El proyecto System R dio origen al uso de SQL como lenguaje de consulta y a partir de este acontecimiento, las bases de datos relacionales empezaron a ser más utilizadas.<br>Las bases de datos relacionales eventualmente empezaron a reemplazar a las bases de datos de red y jerárquicas, debido a su fácil uso.<br>Empezaron a desarrollarse investigaciones relacionadas a las bases de datos paralelas, distribuidas y orientadas a objetos. |

8

## Evolución de las bases de datos y nuevas tendencias

| Rango de tiempo | Características |
|---|---|
| Inicios de 1990 | • El uso del lenguaje SQL, el cual está diseñado para realizar consultas, comenzó a convertirse en una importante aplicación dentro del área de las bases de datos.<br>Surgió el amplio uso de herramientas para el análisis de grandes cantidades de datos.<br>Los proveedores de sistemas de bases de datos comenzaron a comercializar productos basados en bases de datos paralelas y con soporte relacional de objetos. |
| Década de 1990 | • La implementación y el uso de las bases de datos aumentó extensivamente gracias al crecimiento en el uso del Internet.<br>Los sistemas de bases de datos ahora debían ser capaces de soportar altas cantidades de transacciones y estar disponibles para funcionar 24/7. |
| Década del 2000 y el presente | • En la primera mitad de la década del 2000 surgió el uso de XML y su correspondiente lenguaje de consulta XQuery como una nueva tecnología en las bases de datos. Sin embargo, las bases de datos relacionales tienen un uso más extendido.<br>Aumentó el uso de sistemas de bases de datos de código abierto, principalmente PostgreSQL y MySQL.<br>Durante la segunda mitad de la década, debido al incremento en el análisis de grandes volúmenes de datos, surgió el uso de sistemas de bases de datos especializados para estos procesos.<br>Los sistemas de bases de datos han empezado a agregar funcionalidades que permitan manipular los requerimientos de almacenamiento de grandes sitios web.<br>Ha surgido la importancia de las técnicas para los procesos relacionados a la minería de datos. |

9

## Conceptos Básicos

Las bases de datos son una parte esencial de toda organización que utilice computadoras para el almacenamiento y procesamiento de información.

En la actualidad, esta tecnología se aplica en una gran variedad de áreas como negocios, medicina, ingeniería, educación, investigación científica, entre otras; incluyendo a individuos que utilicen las bases de datos para uso personal. Considerando estos hechos es importante conocer diferentes conceptos relacionados a las bases de datos con el fin de comprender su funcionamiento.

10

## Bases de datos

Una base de datos es un conjunto compartido de datos que están relacionados de forma lógica y cuentan con descripciones que le permiten a una organización tomar decisiones relacionadas con esa información.

Por otro lado, los datos se obtienen a partir de diversas fuentes y luego se registran en la base de datos que se diseña y se crea con un fin específico. Por ello, estos datos no son escogidos al azar, mantienen coherencia entre ellos. Al analizar la información necesaria para crear la base de datos de una organización, se pueden identificar tres elementos:

- **Entidad:** Es un objeto concreto que se representa en la base de datos. Estos pueden ser una persona, un lugar, una cosa, un evento o un concepto.
- **Atributo:** Es una propiedad que describe al objeto que está registrado en la base de datos.
- **Relación:** Es la correspondencia o vínculo entre entidades.

11

## DBMS

El Sistema de Gestión de Bases de Datos (SGBD), conocido en inglés como Database Management System (DBMS), es un software que cuenta con herramientas que permiten definir, crear, mantener y manipular a una base de datos.

El DBMS es el programa que funciona como interfaz entre el usuario y la base de datos, a través de accesos controlados y por consiguiente, dependiendo de los permisos que los usuarios tengan, estos obtienen diferentes vistas para la manipulación de los datos.

12

## Sin embargo, el DBMS cuenta con otros elementos y características importantes que se describirán a continuación:

- **Arquitectura general:** La arquitectura del DBMS se compone de tres niveles que se denominan *arquitectura ANSI/SPARC*, porque fue inicialmente propuesta por estas dos organizaciones: el American National Standards Institute (ANSI) y el Standards Planning And Requirements Committe (SPARC). Esta arquitectura es una herramienta que permite la visualización de los niveles en el sistema de base de datos y cada nivel no es más que una descripción de los datos. A continuación se describen los tres niveles que componen a la arquitectura ANSI/SPARC:
  - **Nivel interno:** Corresponde a la representación y el almacenamiento físico de la base de datos, tanto a nivel de software (sistema de gestión de la base de datos) como a nivel de hardware (equipo informático para el almacenamiento de la base de datos).
  - **Nivel conceptual:** Corresponde a una descripción de la base de datos completa, exceptuando detalles sobre el almacenamiento físico de los datos. Se describen elementos como las entidades, tipos de datos, relaciones y restricciones.
  - **Nivel externo:** Corresponde a la representación de las vistas de los distintos usuarios de la base de datos

13

## Sin embargo, el DBMS cuenta con otros elementos y características importantes que se describirán a continuación:

- **Lenguaje de definición y manipulación:** Ambos lenguajes le proveen al usuario la facilidad de realizar tareas con los datos almacenados en la base de datos. El lenguaje de definición de datos o *Data Definition Language (DDL)*, permite definir el tipo, la estructura y las restricciones de los datos, mientras que el lenguaje de manipulación de datos o *Data Manipulation Language (DML)*, permite la inserción, eliminación, actualización y recuperación de datos (Connolly & Begg, 2015). El DML más utilizado es *Structured Query Language (SQL)*.
- **Funciones del DBMS:** Tras un vistazo general sobre las diversas capacidades del DBMS, conviene profundizar en sus funcionalidades

14

| Tipos de interfaces | Características |
|---|---|
| Interfaces basadas en menús para clientes o navegación web | • A través de listas de opciones, denominadas menús, el usuario es guiado para que haga una solicitud.<br>• El menú funciona como una guía paso a paso para que el usuario escoja los comandos que se muestran. No es necesario que el usuario memorice los comandos o la sintaxis del lenguaje de consulta.<br>• En las interfaces de clientes suelen utilizarse menús desplegables, mientras que en las de navegación los usuarios pueden explorar entre el contenido de la base de datos. |
| Aplicaciones para dispositivos móviles | • El usuario acceda a sus datos a través de una interfaz incorporada en el código de la aplicación.<br>• Por lo general, el usuario debe iniciar sesión en una cuenta creada en la aplicación para que esta le proporcione un menú con opciones para acceder a sus datos. |
| Interfaces basadas en formularios | • Se basa en el uso de formularios que el usuario llena. Las entradas del formulario son los datos que serán insertados o bien, el usuario llena algunas de las entradas y el DBMS se encarga de recuperar datos para llenar los entradas restantes.<br>• Muchos DBMS cuentan con lenguajes específicos para que los programadores definan estos formularios. |

- **Interfaces, utilitarios, herramientas de aplicación y recursos de comunicación:** A continuación se describen estos elementos.
  - **Interfaces:** Permiten la interacción entre el usuario y las diversas funcionalidades del DBMS. A continuación se describen los tipos de interfaces más utilizados:

15

| Tipos de interfaces | Características |
|---|---|
| Interfaces basadas en búsqueda de palabras clave | • El usuario inserta palabras en lenguaje natural y el DBMS las asocia con documentos en sitios o páginas web desde un motor de búsqueda.<br>• El DBMS utiliza índices predeterminados de palabras y a través de funciones de clasificación, recupera y le presenta los resultados al usuario en orden decreciente. |
| Interfaces para usuarios paramétricos | • Son interfaces específicas para usuarios inexpertos, es decir, aquellos que no tienen conocimiento sobre base de datos y sólo la utilizan a través de la aplicación.<br>• El diseño e implementación de este tipo de interfaces incluye una interfaz específica para cada tipo de usuario paramétrico que se conoce utilizará la aplicación. |
| Interfaces para el Administrador de la Base de Datos (DBA) | • Son de uso exclusivo para aquellos que tienen la tarea de administrar la base de datos, ya que estas interfaces incluyen comandos únicos.<br>• Los comandos le permiten al administrador realizar diferentes tareas como crear cuentas, determinar los permisos de estas, definir parámetros del sistema, entre otros. |

- **Interfaces, utilitarios, herramientas de aplicación y recursos de comunicación:** A continuación se describen estos elementos.
  - **Interfaces:** Permiten la interacción entre el usuario y las diversas funcionalidades del DBMS. A continuación se describen los tipos de interfaces más utilizados:

16

- **Utilitarios:** Son elementos que asisten al administrado de la base de datos en el mantenimiento del sistema. A continuación se detallan los utilitarios más utilizados:
  - **Inserción:** Carga de archivos de datos a la base de datos, conversión del archivo al formato especificado por el utilitario y transferencia de datos de un DBMS a otro.
  - **Monitoreo de rendimiento:** Monitoreo del uso de la base de datos, generación de estadísticas que orientan al administrado en la toma de decisiones relacionadas a la base de datos.
  - **Reorganización del almacenamiento de la base de datos:** Reordenamiento de archivos dentro de la base de datos y creación de nuevos accesos para mejorar el rendimiento de la base de datos.
  - **Respaldo:** Creación de copia de seguridad de la base de datos, la cual luego se almacena en un medio de almacenamiento masivo. Suele utilizarse el respaldo de tipo incremental porque ahorra espacio de almacenamiento.

17

- **Herramientas de aplicación y recursos de comunicación:** Son funcionalidades adicionales que pueden utilizar los usuarios, el diseñador de la base de datos y el propio DBMS. A continuación se especifican estas funcionalidades:
  - **Sistema de diccionario de datos:** Almacenamiento de un catálogo con información sobre esquemas, restricciones, usuarios y estándares.
  - **Ambiente para el desarrollo de aplicaciones:** Entorno para desarrollar aplicaciones para la base de datos, tales como el diseño de la base datos, desarrollo de la interfaz gráfica, consultas y actualización de los datos.
  - **Recursos de comunicación:** Acceso remoto a la base de datos desde los puestos de trabajo y computadoras personales.

18

**Arquitectura cliente/servidor:** Como explica (Elmasri & Navathe, 2016), la arquitectura cliente/servidor se utiliza en entornos en los que hay un gran número de elementos de software y hardware que están conectados a través de una red.

- **Arquitectura de dos niveles:** La aplicación de la base de datos se ejecuta en la máquina cliente desde la cual se solicita alguna funcionalidad del sistema de base de datos en la máquina servidor.
- **Arquitectura de tres niveles:** La máquina cliente es simplemente una interfaz y no se pueden realizar solicitudes a la base de datos de forma directa.

19

## Esquemas, instancias y estado de una base de datos

Es importante diferenciar los conceptos *esquema* e *instancia*, pues el esquema corresponde al diseño lógico de la base de datos, mientras que la instancia es una imagen del estado de los datos. A continuación, con base en lo explicado por (Elmasri & Navathe, 2016), se describirán con mayor detalle estos conceptos:

- **Esquemas:** Describe a la base de datos y se define durante el diseño de la base de datos. Y pueden ser Esquema a nivel interno, Esquema a nivel conceptual, Esquema a nivel externo.
- **Instancias:** El estado de los datos en un instante determinado se denomina *estado de la base de datos* y se pueden distinguir tres tipos de estados, de acuerdo con (Elmasri & Navathe, 2016):Estado vacío, Estado inicial, Estado actual.

20

## Independencia lógica y física de datos

Considerando la arquitectura de tres niveles que se describió anteriormente, uno de los principales motivos de su aplicación en un DBMS es la de proveer la independencia de los datos. Se pueden distinguir dos tipos de independencia de datos que se definen a continuación, de acuerdo con las descripciones de (Elmasri & Navathe, 2016):

- **Independencia lógica:** Permite que se realicen modificaciones en el esquema conceptual sin que esto afecte a los esquemas externos.
- **Independencia física:** Permite que se realicen modificaciones en el esquema interno sin esto afecte al esquema conceptual y por consiguiente, a los esquemas externos.

21

## Modelo de datos

Anteriormente se mencionó que las bases de datos son una representación de la vida real y para lograr esto, es necesario describir la estructura de la base de datos, lo cual corresponde a los datos, sus relaciones y sus restricciones.

**Definición:** es un conjunto de conceptos que permiten la descripción de la estructura de una base de datos y brinda los recursos requeridos para lograr esta abstracción.

**Abstracción:** Este término se refiere a que un modelo de datos detalla las características esenciales sobre los datos de modo que se mejore la comprensión.

22

## Clasificación

A continuación se describe la clasificación de modelos de datos propuesta por (Connolly & Begg, 2015):

- **Modelo de datos basado en objetos:** Describe a la base de datos a través de los conceptos de entidad, atributo y relación.
  - **Entidad-relación (ER):** este tipo de modelo se compone de entidades que pertenecen a un conjunto y que están asociadas a través de relaciones.
- **Modelo de datos basado en registros:** Describe a la base de datos a través de un conjunto de registros de formato fijo y con una cantidad fija de campos.
  - **Modelo de datos relacional:** Representa los datos y sus relaciones a través de tablas, donde cada una de ellas tiene columnas que permiten la relación entre las tablas.
  - **Modelo de datos en red:** Representa los datos como un grupo de registros organizados en un grafo.
  - **Modelo de datos jerárquico:** Al igual que el tipo de modelo anterior, representa los datos como un grupo de registros y a las relaciones como conjuntos.
  - **Modelo de datos físico:** Representa la forma en que se almacenan los datos, describiendo información tal como la estructura de los registros y las rutas de acceso de estos.

23

*Usuarios en un ambiente de base de datos*

| Rol | | Funciones |
|---|---|---|
| Administradores de datos y de la base de datos | Administrador de datos (DA) | • Planificar la base de datos.<br>• Diseñar el modelo conceptual y lógico de la base de datos.<br>• Desarrollar y mantener estándares, políticas y procedimientos.<br>• Consultar con sus superiores y orientarlos, de modo que el desarrollo de la base de datos vaya de acuerdo con los objetivos de la organización. |
| | Administrador de la base de datos (DBA) | • Desarrollar el diseño y la implementación de la base de datos.<br>• Controlar la seguridad e integridad de la base de datos.<br>• Asegurar el correcto funcionamiento de las aplicaciones para los usuarios.<br>• Autorizar los permisos de acceso a la base de datos (Elmasri & Navathe, 2016).<br>• Tener un conocimiento detallado del DBMS que utilizará la organización, ya que su función es más técnica que la del DA. |

24

## *Usuarios en un ambiente de base de datos*

| Rol | | Funciones |
|---|---|---|
| Diseñadores de la base de datos | Diseñador lógico | • Responder al qué debe contener la base de datos.<br>• Identificar los datos, en términos de entidades y atributos, las relaciones entre esos datos y sus correspondientes restricciones.<br>• Comprender detalladamente sobre los datos y las restricciones que deben aplicarse, de acuerdo con las reglas de la organización.<br>• Desarrollar el modelo de datos considerando a los futuros usuarios de la base de datos. |
| | Diseñador físico | • Responder al cómo representar la base de datos.<br>• Desarrollar el diseño físico de la base de datos, considerando el diseño lógico.<br>• Diseñar las tablas y las restricciones de integridad correspondientes.<br>• Seleccionar las estructuras de almacenamiento y los métodos de acceso a los datos.<br>• Diseñar las medidas de seguridad que requieran los datos. |
| Desarrolladores de aplicaciones | | • Desarrollar e implementar aplicaciones con funcionalidades que cumplan los requerimientos de los usuarios finales.<br>• Fundamentar el desarrollo de las aplicaciones en las especificaciones dadas por los analistas de sistemas. |
| Usuarios finales | Usuarios paramétricos | • Por lo general, no tiene conocimiento sobre el DBMS (usuarios inexpertos).<br>• Acceder a los datos a través de aplicaciones que facilitan en la mayor medida de lo posible las tareas a realizar.<br>• Realizar solicitudes a la base de datos a través del uso de comandos simples o escogiendo opciones presentadas en un menú. |
| | Usuarios avanzados | • Conocer la estructura de la base de datos.<br>• Conocer las funcionalidades con las que cuenta el DBMS.<br>• Utilizar un lenguaje de consulta específico para realizar las tareas requeridas. |

25

## *Componentes de un ambiente de base de datos*

| | Componente | Descripción |
|---|---|---|
| Máquina | Hardware | • Partes físicas del sistemas informático que se requieren para ejecutar el DBMS y las aplicaciones.<br>• Puede variar desde una computadora personal hasta una red de computadoras.<br>• Depende de las necesidades de la organización y del DBMS que se utilizará. |
| | Software | • Comprende el DBMS, las aplicaciones, el sistema operativo y el programa de red, si el DBMS se utiliza a través de una red.<br>• Las aplicaciones suelen escribirse en un lenguaje de programación de tercera generación (Java, Visual Basic, C, etc.) o en un lenguaje de cuarta generación (SQL) dentro de un lenguaje de tercera generación.<br>• Algunos DBMS incluyen su propio lenguaje de cuarta generación, lo que facilita el desarrollo de aplicaciones. |
| Puente | Datos | • Es el componente más importante dentro del ambiente de base de datos.<br>• Funciona con un puente que crea el vínculo máquina-humano.<br>• Se compone de los datos contenidos en la base de datos, incluyendo los metadatos (datos sobre los datos). |
| Humano | Procedimientos | • Son las instrucciones y pautas que encaminan el diseño y el uso de la base de datos.<br>• Algunas de estas instrucciones pueden ser el inicio de sesión en el DBMS y la creación copias de respaldo. |
| | Personas | • Corresponde a todas aquellas personas que forman parte del entorno de la base de datos, quienes desempeñan diferentes roles en este. |

26

# Capítulo II

27

# Capítulo II: Fases en el desarrollo y construcción de una base de datos

***Ciclo de vida del desarrollo de sistemas de bases de datos***

Al ser las bases de datos una parte fundamental del sistema de información de una organización, su ciclo de vida debe enfocarse en los requerimientos más imprescindibles; por ello, el ciclo de vida de un sistema de base de datos está vinculado al ciclo de vida del sistema de información de la organización

28

## Etapas en el desarrollo de un sistema de base de datos

**Planificación de la base de datos:** Administración de las actividades que permiten que las etapas del ciclo de vida del desarrollo de la base de datos se realicen de manera eficiente y eficaz.

**Definición del sistema:** Establecimiento del alcance y los límites del sistema de base de datos y de las vistas de los usuarios.

**Recolección y análisis de requisitos:** Recolección y análisis de la información sobre la organización, que debe ser incorporada en el sistema de base de datos y que permitirá identificar los requerimientos para el desarrollo de dicho sistema.

**Diseño de la base de datos:** Creación de un diseño que incorpore la misión y objetivos del sistema, así como los estándares especificados como requisitos para el sistema de base de datos.

**Selección del DBMS:** Corresponde a la selección de un DBMS que cumpla con los requisitos del sistema de base de datos.

29

## Etapas en el desarrollo de un sistema de base de datos

- **Diseño de la aplicación:** Corresponde al diseño de la interfaz de usuario y de los programas de aplicación que permitirán el uso de la base de datos.
- **Prototipado:** Es una etapa opcional que consiste en construir un modelo de trabajo del sistema en desarrollo
- **Implementación:** Implica utilizar las sentencias DDL del DBMS seleccionado o una GUI (Interfaz Gráfica de Usuario), para definir el diseño físico de la base de datos y de los diseños de las aplicaciones
- **Conversión y carga de datos:** Consiste en transferir los datos al nuevo sistema de base de datos y esta etapa es imprescindible cuando el nuevo sistema reemplaza a un sistema antiguo.
- **Prueba:** Consiste en ejecutar el nuevo sistema de base de datos con el propósito de encontrar errores y validar que se cumplan los requisitos de los usuarios.
- **Mantenimiento:** Consiste en monitorear y darle mantenimiento al sistema de base de datos, una vez ha sido instalado y se ha iniciado su ejecución dentro de la organización.

30

## *Técnicas de determinación de hechos*

- **Introducción**

El desarrollo de un sistema de base de datos requiere de información sobre la organización, por lo que es necesario determinar hechos que permitirán darle estructura al sistema.

- **Definición**

La determinación de hechos se define como un procedimiento formal en el que se aplican técnicas para recolectar acontecimientos sobre las terminologías utilizadas dentro de la organización, problemas que presenta el sistema actual, oportunidades que generaría el nuevo sistema, restricciones sobre los datos y los usuarios y los requisitos para el nuevo sistema de base de datos

31

## *Técnicas de determinación de hechos*

- **Descripción de las técnicas**

Existen diferentes técnicas de determinación de hechos y el desarrollador de la base de datos puede elegir utilizar más de una de ellas.

- **Revisión de documentación:** Consiste en analizar todos aquellos documentos de la organización que podrían aportar información valiosa, tales como manuales, descripción de puestos de trabajo, reportes, formularios y cualquier tipo de documento relacionado al sistema actual.
- **Observación:** Se analizan a las personas mientras realizan sus labores dentro de la organización, con el fin obtener una perspectiva sobre el funcionamiento del sistema actual.
- **Investigación:** Consiste en investigar problemas similares que permitan obtener información sobre posibles soluciones para el sistema de la organización.
- **Cuestionarios:** Consiste en documentos con preguntas ordenadas y formales para recolectar datos y es útil en casos en los que no es posible realizar entrevistas

32

# Capítulo III

33

## Capítulo III: Modelaje conceptual de base de datos

- ***Importancia de la modelización conceptual***

Dentro de la etapa de diseño de la base de datos, la modelización conceptual de la base de datos es un proceso indispensable para la identificación de los datos que se requieren en el sistema en desarrollo. Como explica (Connolly & Begg, 2015), este proceso es importante porque:

- Los desarrolladores y diseñadores de la base de datos obtienen información acerca de la naturaleza de los datos y cómo estos son utilizados por la organización.
- Se asegura el cumplimiento de los requisitos de los usuarios de la base de datos.
- Permite la comunicación de información dado que se evitan ambigüedades sobre los datos.

34

## *Componentes básicos de un modelo entidad-relación*

**Entidad:** Una entidad corresponde a un objeto o cosa en el mundo real con existencia propia, ya sea física o abstracta.

**Tipos de entidades**

Las entidades pueden clasificarse en dos categorías y estas se describen a continuación:

- **Entidades fuertes:** Son entidades cuya existencia no depende de la existencia de otra entidad. Este tipo de entidades cuentan con un atributo que es una llave primaria que permite la identificación de cada una de las ocurrencias de la entidad. Su representación en el modelo ER es la mostrada en la Figura 12.
- **Entidades débiles:** Son entidades cuya existencia depende de la existencia de otra entidad.

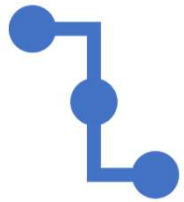
35

## Componentes básicos de un modelo entidad-relación

- **Relación:** Una relación es el vínculo que existe entidades y estas relaciones corresponden a la realidad.

**Tipos de relación**

- Los tipos de relación entre entidades corresponden al conjunto de asociaciones que existen entre los distintos tipos de entidades que se representan



36

## Componentes básicos de un modelo entidad-relación

- **Grado:** El grado de una relación se refiere a la cantidad de entidades asociadas a una determinada relación. Se pueden identificar los siguientes grados de relación, según lo explicado por (Coronel et al., 2011):
  - **Unitaria o recursiva:** Es aquella relación en la que hay asociación con una sola entidad.
  - **Binaria:** Es aquella relación en la que se asocian dos entidades. Es el grado de relación más común
  - **Ternaria:** Es aquella relación en la que hay asociación entre tres entidades.
  - **Múltiple:** Es cuando existe más de una relación entre dos entidades.

37

## Atributos

Los atributos son las propiedades que describen a una entidad o relación y toman valores que describen cada ocurrencia de una determinada entidad.

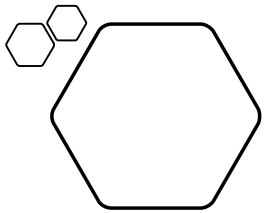
Los atributos pueden clasificarse de la siguiente manera indicada en el cuadro a continuación, según las explicaciones de (Coronel et al., 2011):

- **Simples o compuestos:** Los atributos simples son aquellos que no se subdividen, mientras que los atributos compuestos son aquellos que puede subdividirse en otros atributos
- **Univaluados o multivaluados:** Un atributo univaluado es aquel que sólo puede tomar un solo valor, mientras que un atributo multivaluado es aquel que puede tomar múltiples valores.
- **Derivados:** Se refiere a un atributo cuyo valor es el resultado del cálculo de otros atributos, es decir, su valor depende de otros atributos

38

## Atributos de las relaciones

Como indica (Connolly & Begg, 2015), las relaciones entre entidades también pueden tener atributos. En este caso, se representa con el mismo símbolo utilizado para representar el atributo de una entidad, pero se utiliza una línea punteada asociada a la relación



39

## Restricciones estructurales

Las restricciones estructurales son aquellas que toman las entidades que forman parte de una relación y estas corresponden a las políticas o términos de la organización o del usuario.

- **Relaciones uno a uno (1:1):** La ocurrencia de una entidad se asocia con una sola ocurrencia de otra entidad, es decir, una sola entidad A se relaciona con una sola entidad B y viceversa.
- **Relaciones uno a muchos (1:M):** La ocurrencia de una entidad se asocia con más de una ocurrencia de otra entidad, es decir, una sola entidad A se relaciona con muchas entidades B y una entidad B se relaciona con una sola entidad A.
- **Relaciones muchos a muchos (M:M):** Más de una ocurrencia de una entidad se asocia con más de una ocurrencia de otra entidad, es decir, una entidad A se relaciona con muchas entidades B y viceversa.

40

## Restricciones de cardinalidad

La cardinalidad se define como el número máximo posible de ocurrencias que puede tener una entidad que pertenece a una determinada relación (Connolly & Begg, 2015). La cardinalidad puede ser de cuatro tipos, de acuerdo con lo descrito por (Cardona et al., 2014): cero a uno (0,1), uno a uno (1,1), cero a muchos (0,M) y uno a muchos (1,M).

41

Para comprender estos conceptos es conveniente definir dos tipos especiales de entidades denominados superclases y subclases. Una *superclase* es una entidad tiene uno o más subgrupos de ocurrencias, mientras que una *subclase* es una entidad que se considera un subgrupo de ocurrencias de otra entidad. Considerando lo anterior, se habla de dos procesos de abstracción, los cuales se describen a continuación de acuerdo con lo definido por (Connolly & Begg, 2015):

- **Especialización:** Consiste en identificar las características que no son comunes entre los miembros o subgrupos de una entidad, con el fin de maximizar las diferencias.
- **Generalización:** Consiste en identificar las características comunes entre entidades con el fin de minimizar las diferencias ellas.


42

## Conversión de un esquema ER a tablas

Un modelo ER puede convertirse a un conjunto de tablas que representen el problema en estudio. De hecho, como explica (Sumathi & Esakkirajan, 2007), la implementación de una base de datos se hace a través del modelo relacional que se compone de tablas mapeadas a través de las llaves de las entidades, de allí que a este proceso se le denomine *mapeo*. Para ello, se debe considerar lo siguiente, basado en lo descrito por:

- Cada entidad corresponde a una tabla.
- Los atributos de una entidad corresponden a las columnas de la tabla, cada columna lleva el nombre de un atributo.
- Las filas (denominadas tuplas) de una tabla son las ocurrencias de cada entidad y estos toman valores para cada atributo.
- El grado de la relación corresponde a la cantidad de columnas (atributos) de la tabla, mientras que la cardinalidad corresponde a la cantidad de tuplas de la tabla. (Sharma et al., 2010)

43

## Capítulo IV

44

- ***Definición del modelo de datos relacionales***

El modelo de datos relacionales fue introducido por Edgar F. Codd en el año 1970. Este modelo se compone de un conjunto de estructuras básicas denominadas *tablas*, que forman relaciones entre los datos almacenados en ellas. El modelo de datos relacional cuenta con diferentes objetivos, los cuales se especifican en:

- **Independencia lógica de los datos:** El usuario interpreta a las relaciones como la estructura lógica que conforma a la base de datos.
- **Independencia física de los datos:** El usuario desconoce la estructura de datos utilizada para almacenar los datos en la base de datos
- **Simplicidad y uniformidad:** Los datos se representan de una sola forma a través de valores explícitos que permiten formar las relaciones de la base de datos.

## Capítulo IV: Fundamentos del modelo relacional

45

## Definición de conceptos básicos del modelo relacional

Para estudiar el modelo relacional es primordial definir conceptos asociados a este. (Marqués, 2011) define estos conceptos de la siguiente manera:

- **Relación:** Es una tabla compuesta por filas y columnas. Las filas corresponden a registros individuales, mientras que las columnas son los campos de esos registros.
- **Atributo:** Es el nombre de una columna en una relación. Son los elementos de una relación.
- **Dominio:** Es el conjunto de valores permitidos para un atributo determinado. Cada atributo se define sobre un dominio, pero sobre un dominio se pueden definir varios atributos.
- **Tupla:** Es una fila en una relación y un conjunto de tuplas conforman el elemento de una relación.
- **Grado de una relación:** Cantidad de atributos que contiene la relación. El grado de una relación por lo general es constante.
- **Cardinalidad de una relación:** Cantidad de tuplas que contiene la relación. La cardinalidad de una relación puede variar frecuentemente, debido a la actualización de tuplas.

46

## Restricciones de integridad del modelo

Para que una base de datos represente a la realidad a través de los datos que almacena, es primordial aplicar restricciones de integridad. La integridad es precisamente aquella característica de los datos que permite que estos representen adecuadamente al mundo real.

- **Definiciones de claves:** Una *clave* o *llave* es uno o más atributos que identifican a una tupla en una relación.
  - **Llave candidata:** Es un atributo o combinación de atributos que pueden servir como llave primaria en una relación
  - **Llave primaria (PK):** Es uno o más atributos que se escogen a partir de las llaves candidatas e identifican de forma única a cada tupla de la relación.
  - **Llave foránea (FK):** Es uno o más atributos que hacen referencia a una tupla en otra relación.

47

## Restricciones de integridad del modelo

- **Valores nulos:** Cuando un atributo en una tupla tiene un valor *nulo*, significa que el valor es desconocido.
- **Dominios:** El dominio permite definir las características de los atributos de una relación
- **Otros:** Existen otras reglas de integridad que son importantes y por lo tanto, es necesario describirlas
  - **Regla de integridad de entidad:** La llave primaria de una relación no puede contener valores nulos, dado que si una llave primaria tuviera un valor nulo entonces algunas tuplas no podrían identificarse al no contener un valor único.
  - **Regla de integridad referencial:** La llave foránea debe contener un valor que coincida con el valor que contiene la llave primaria con la cual se asocia o un valor nulo

48

# Operaciones en el modelo relacional: álgebra relacional

- **Selección (σ):** Utiliza una sola relación R para tomar aquellas tuplas que satisfagan la condición especificada (predicado).
- **Proyección (Π):** Utiliza una sola relación R para tomar algunos atributos y definir otra relación. La nueva relación contiene los valores de los atributos especificados y no muestra tuplas duplicadas.
- **Unión (U):** Utiliza dos relaciones R y S para definir otra relación que contiene todas las tuplas de ambas relaciones, sin incluir duplicados
- **Diferencia de conjuntos (−):** Utiliza dos relaciones R y S para producir una nueva relación que muestra las tuplas que están en una relación pero no en la otra.
- **Producto cartesiano (×):** Utiliza dos relaciones R y S para generar una nueva relación que combina las tuplas de ambas relaciones. Además, la nueva relación contiene todos los atributos de ambas relaciones.
- **Renombramiento (ρ):** Permite cambiar el nombre de una relación o de los atributos de una relación resultante.
- **Intersección (∩):** Utiliza dos relaciones R y S para generar una nueva relación que contiene las tuplas que pertenecen a ambas relaciones.
- **Reunión natural (⋈):** Utiliza dos relaciones R y S para generar una nueva relación que contiene las tuplas que coinciden con los atributos en común de ambas relaciones.
- **División (/):** Utiliza dos relaciones R y S para generar una nueva relación que contiene todas las tuplas de R que coinciden con las tuplas de S.

49

# Álgebra relacional extendida

El álgebra relacional extendida permite que se realicen operaciones aritméticas en combinación con las operaciones básicas del álgebra relacional. A continuación se describen las operaciones que forman parte del álgebra relacional extendida:

- **Proyección generalizada:** Funciona igual que la operación proyección, con la diferencia de que se pueden utilizar funciones aritméticas.
- **Funciones de agregación:** Son funciones que toman como entrada un conjunto de valores y da como resultado un solo valor. Las funciones de agregación más utilizadas son:
  - **sum:** Devuelve la suma de los valores asociados a un atributo.
  - **avg:** Devuelve el promedio de los valores asociados a un atributo.
  - **count:** Devuelve la cantidad de valores que hay.
  - **min:** Devuelve el valor más pequeño en el atributo.
  - **max:** Devuelve el valor más grande en el atributo.
  - **Reunión externa:** Es parecido a la operación unión, pero además el resultado es una relación que incluye las tuplas de una relación R que no coinciden con los atributos de una relación S.

50

# Capítulo V

51

# Capítulo V: Normalización

***Justificación:*** La normalización es una técnica de diseño de base de datos, que permite identificar las relaciones entre los atributos que son óptimas para la representación de los datos de la organización.

La redundancia de datos significa que se puede encontrar un mismo dato en más de una ubicación dentro de las tablas de la base de datos. Esto a su vez, conlleva problemas que se describirán a continuación, basado en lo explicado por (Mannino, 2007):

- **Anomalía de inserción:** Ocurre cuando al insertar un registro, se requiere de la adición de datos no relacionados al dicho registro.
- **Anomalía de eliminación:** Ocurre cuando al eliminar un registro, inadvertidamente se eliminan otros datos no relacionados.
- **Anomalía de actualización:** Ocurre cuando al modificar un registro, es necesario modificar otros registros.

52

## *Concepto de dependencias funcionales*

Una dependencia funcional (DF) es un tipo de restricción de integridad que describe la relación entre los atributos de una tabla. La representación de una dependencia funcional entre dos atributos es la siguiente, como indica (Elmasri & Navathe, 2011): $A \rightarrow B$, donde $A$ (lado izquierdo o LHS) y $B$ (lado derecho o RHS) son conjuntos de atributos que son subconjuntos de una relación $R$. $A \rightarrow B$ significa que $B$ es funcionalmente dependiente de $A$, $A$ determina a $B$ si existe al menos un valor de $B$ para cada valor de $A$. El lado izquierdo de la dependencia funcional se denomina **determinante**.

53

## Formas normales y axiomas

Las formas normales permiten la eliminación de las redundancias cuando se hace el proceso de normalización. Una forma normal (NF) es definida por (Mannino, 2007) como *"una regla sobre las dependencias permisibles"*.

| Axioma | Definición |
|---|---|
| Reflexividad | Si $B$ es un subconjunto de $A$, entonces $A \rightarrow B$. |
| Aumentatividad | Si $A \rightarrow B$ y $C$ es otro atributo, entonces $AC \rightarrow BC$. |
| Transitividad | Si $A \rightarrow B$ y $B \rightarrow C$, entonces $A \rightarrow C$. |
| Auto-determinación | $A \rightarrow A$ |
| Descomposición | Si $A \rightarrow BC$, entonces $A \rightarrow B$ y $A \rightarrow C$. |
| Unión | Si $A \rightarrow B$ y $A \rightarrow C$, entonces $A \rightarrow BC$. |
| Composición | Si $A \rightarrow B$ y $C \rightarrow D$, entonces $AC \rightarrow BD$. |

54

## Cálculo de la clausura de atributos

Para el cálculo del cierre de atributos se tomará una relación $R$ con un conjunto de atributos, se calculará el cierre $(A)$. (Sharma et al., 2010) explica el proceso de la siguiente manera:

- Primero establecer el cierre como $(A) = A$.
- Para cada DF, si $A \rightarrow B$, entonces agregar $B$ al cierre $(A)$. De esta manera queda el cierre $(A) \cup B$.
- Para cada subconjunto de $A$, hacer $C$ un subconjunto de $A$ para obtener una DF trivial $A \rightarrow C$. Si $C \rightarrow D$ y $D$ no es subconjunto de $A$, entonces agregar $D$ al cierre $(A)$.
- Repetir el paso 3 hasta que no haya más conjuntos de atributos que agregar al cierre $(A)$.

55

## Cierre de un conjunto de dependencias funcionales

El cierre de un conjunto de DFs se refiere a todas las dependencias funcionales que están implícitas para un conjunto dado de una DF determinada (Sharma et al., 2010), es decir, todas las dependencias funcionales que pueden derivar de esa DF. Para un conjunto de dependencias funcionales denominado $F$, su cierre de conjunto de dependencias funcionales se denota como $F^+$.

56

## Recubrimiento Minimal

El recubrimiento minimal, denotado como $F_c$ se refiere a un conjunto de dependencias funcionales de modo que el cierre de dependencias funcionales y el conjunto $F$ de DFs dado para una relación $R$ son equivalentes (Sharma et al., 2010). Esto quiere decir que $F^+ = F_c^+$.

- Se debe considerar lo siguiente:
- Cada dependencia funcional en el recubrimiento minimal tiene un solo atributo en el lado derecho.
- Reducir los atributos en el lado izquierdo de cada dependencia funcional, genera cambios en el cierre del recubrimiento minimal.
- Las dependencias funcionales no son redundantes, por lo que la eliminación de alguna de las dependencias en el recubrimiento minimal hará que el cierre de este cambie.

57

## Determinación de las Claves

Las dependencias funcionales también pueden utilizarse para la determinación de claves, o bien para determinar posibles claves. De acuerdo con (Mannino, 2007), se debe considerar lo siguiente para determinar las claves:

- Un determinante es una llave candidata si al establecer la dependencia funcional $A \rightarrow B$, $A$ y $B$ se colocan juntos en una tabla sin otras columnas o atributos.
- Un determinante es una llave candidata cuando determina otras columnas o atributos de una tabla.
- Un determinante es una llave primaria si no existen otras llaves candidatas y si dicho determinante no permite valores nulos.

58

## *Normalización*

El proceso de normalización puede considerarse como una serie de pasos que se aplican para asegurar que se cumple una determinada forma normal. Cada paso corresponde a una forma normal y como explica (Connolly & Begg, 2015), las restricciones de las relaciones se vuelven más estrictas y por consiguiente, se disminuye la posibilidad de que la futura base de datos presente anomalías de actualización.

- **Formas normales basadas en DFs**

En esta sección se describirán las principales formas normales que generalmente se aplican a las relaciones de un modelo relacional. A continuación se describirán las diferentes formas normales, basado en las explicaciones de (Churcher, 2007):

- **Primera forma normal (1FN):** Este es el nivel de normalización más básico. Una relación está en 1FN si y sólo si:
  - Todos los atributos son valores atómicos, es decir, cada columna de la tabla sólo puede tener un valor para cada registro.
  - La relación no tiene grupos repetitivos.
  - La relación tiene una llave primaria.
  - Ninguna clave candidata acepta valores nulos.

59

## Formas normales basadas en DFs

- **Segunda forma normal (2FN):** Una relación está en 2FN si y sólo si:
  - Está en la primera forma normal.
  - Cada atributo-no-llave es completamente dependiente de la llave primaria completa, no sólo de una parte de la llave.
- **Tercera forma normal (3FN):** Una relación está en 3FN si y sólo si:
  - Está en la segunda forma normal.
  - No hay atributos-no-llave que son dependientes de otros atributos-no-llave.
- **Forma normal de Boyce-Codd (FNBC):** Una relación está en FNBC si cada determinante de una DF puede ser una llave primaria. Toda relación en FNBC también está en 1FN, 2FN y 3FN.

60

## Descomposición de las relaciones

El esquema de una base de datos se compone de una relación universal $R = \{A_1, A_2, \dots, A_n\}$ donde $A$ es el conjunto de todos los atributos de la base de datos. A través del uso de las dependencias funcionales, esta relación R se descompone en un conjunto de relaciones denominado $D = \{R_1, R_2, \dots, R_m\}$, lo cual corresponde a la descomposición de $R$ (Elmasri & Navathe, 2011). Al realizar la descomposición de relaciones es importante asegurar que haya una preservación de atributos, de modo que no se pierda ningún atributo. Para ello, cada atributo en $R$ debe aparecer en al menos un esquema de relación $R_i$, lo cual formalmente se define como:

$$\bigcup_{i=1}^{m} R_i = R$$

61


62

## Capítulo VI: MS SQL – Lenguaje estructurado de consulta

- ***Introducción al entorno de trabajo del Gestor de Base de Datos***

El entorno de trabajo del gestor de base de datos MS SQL Server, es similar a las ventanas de herramientas de otras aplicaciones del sistema operativo Windows.

La ventana de herramientas de SQL Server se puede dividir en dos grandes áreas:

- **Explorador de objetos:** Permite ver y administrar los objetos que forman parte de las instancias de SQL Server (Microsoft, 2019). En esta área se pueden ver todas las bases de datos instaladas.
- **Área de trabajo:** Se subdivide en un área para ingresar las instrucciones para la consulta y mantenimiento de las bases de datos y en otra área donde se muestran mensajes de error, resultados de las consultas o mensajes que indican si se han completado satisfactoriamente las instrucciones.

63

## *Instrucciones de creación de usuarios en el sistema*

Como explica (Tutorials Point, 2016), para poder acceder a una base de datos, un usuario se refiere a una cuenta en la base de datos en MS SQL Server. La sintaxis de la instrucción para crear un usuario es la siguiente:

```
create user nombre_usuario for login nombre_login
```

Para utilizar esta sintaxis es necesario saber el nombre del login, es decir, el login ya debe estar creado. Además, se debe seleccionar la base de datos para la cual se desea crear el login.

64

## *Instrucciones de definición de datos*

Las instrucciones de definición de datos se utilizan para la creación, eliminación y modificación de los modelos relacionales de las bases de datos

El estándar SQL soporta diferentes tipos de datos, los cuales se utilizan para su definición. Entre los tipos de datos más utilizados, se encuentran los siguientes, descritos por:

- **char(n):** Almacena una cadena de caracteres de longitud fija, a la cual se le debe especificar la longitud.
- **varchar(n):** Almacena una cadena de caracteres de longitud variable, a la cual se le debe especificar la longitud máxima.
- **integer:** Almacena un número entero.
- **smallint:** Almacena un número entero pequeño.
- **decimal(p,e):** Almacena un número decimal con la cantidad de dígitos indicados en la precisión (p) y con los decimales indicados en la escala (e).
- **real:** Almacena un número decimal que tiene una precisión predefinida.
- **float(p):** Almacena un número decimal, se le debe especificar la precisión (p).
- **date:** Almacena una fecha compuesta por YEAR (año), MONTH (mes), DAY (dia).
- **time:** Almacena horas compuestas por HOUR (hora), MINUTE (minuto), SECOND (segundo).
- **timestamp:** Almacena fecha y hora.

65

## Las instrucciones de definición de datos más utilizadas se describen a continuación:

- **Create:** Esta instrucción sirve para crear las bases de datos y sus tablas.
  - **Create database:** Instrucción que sirve para crear una base de datos. La sintaxis es la siguiente: create database nombre_BD
  - **Create table:** Instrucción que sirve para crear una nueva tabla y sus atributos. Los atributos o columnas de una tabla pueden tener restricciones como las siguientes:
    - **Primary key:** Se utiliza para identificar a una tabla de forma única y puede ser uno o más atributos.
    - **Unique:** Indica que la columna tiene un valor diferente para cada fila.
    - **Default:** Asigna un valor por defecto cuando no se especifica el valor de un dato en la columna.
    - **Not null:** Indica que la columna debe tener un valor, es decir, que no debe ser nulo.
    - **Foreign key:** Indica que el atributo es una llave foránea, la cual está referenciada a una llave primaria existente en otra tabla.

66

**Alter**

Esta instrucción es utilizada para agregar columnas nuevas y eliminar o modificar las columnas existentes en una tabla.

- **Add:** Permite agregar una columna nueva a la tabla. Para ello se utiliza la siguiente sintaxis:
- **Alter column:** Permite modificar el tipo de dato de una columna.
- **Drop column:** Permite eliminar una columna de una tabla.

**Otras**

Otras instrucciones permiten eliminar una base de datos existente o la tabla de una base de datos. Estas instrucciones son:

- **Drop database:** Permite eliminar una base de datos existente
- **Drop table:** Permite eliminar un tabla existente en una base de datos.

67

## *Instrucciones de manipulación de datos*

Las instrucciones de manipulación de datos, también llamados Data Manipulation Language (DML), son comandos que permiten mantener bases de datos y hacer consultas en estas, lo cual incluye actualizar, modificar y consultar datos.

- **Estructura básica**

La estructura básica de las consultas en SQL se compone de las cláusulas *select*, *from* y *where*. *Select* es la cláusula base para todas las consultas (Teorey et al., 2011). A continuación se describen estas cláusulas, de acuerdo con (Universidad Privada TELESUP, 2018):

- **Cláusula SELECT:** Muestra las columnas de una tabla en el orden que se haya especificado.
- **Cláusula WHERE:** Especifica la tabla o tablas de las que se recupera la información.
- **Cláusula FROM:** Especifica las restricciones o criterios de selección de las filas.

68

## *Instrucciones de manipulación de datos*

- **Operación renombramiento:** Esta operación permite renombrar cualquier atributo que se muestre como resultado de una consulta, para lo cual se coloca la cláusula *as* seguido del nombre deseado.
- **Variable tupla:** La variable tupla se utiliza para renombrar una tabla, cuando se quiere hacer una comparación de tuplas o registros dentro de una misma tabla.
- **Operaciones sobre cadenas:** En SQL, las cadenas se colocan entre comillas simples. Además, SQL no hace distinción entre mayúsculas y minúsculas.
- **Operaciones sobre cadenas:** En SQL, las cadenas se colocan entre comillas simples. Además, SQL no hace distinción entre mayúsculas y minúsculas.
  - **Porcentaje (%):** Encuentra coincidencias con cualquier subcadena.
  - **Guión bajo (_):** Encuentra coincidencias con cualquier carácter
  - **Guión(-):** Muestra coincidencias con el rango de caracteres especificado, incluyendo esos caracteres.

  También existen funciones que permiten trabajar con cadenas, algunas de estas son las siguientes:
  - **Upper(cadena):** Convierte a una cadena en mayúscula.
  - **Lower(cadena):** Convierte a una cadena en minúscula.
  - **Replace(cadena,cadena_anterior,cadena_nueva):** Reemplaza todas las ocurrencias de una subcadena con otra subcadena.
  - **Substring(cadena,pos_inicio,cant_caracteres):** Extrae algunos caracteres de una cadena.

69

## *Instrucciones de manipulación de datos*

- **Orden en la presentación de tuplas:** Las tuplas que resultan de las consultas que se hagan en SQL, se pueden ordenar utilizando la cláusula *order by*. Por defecto, esta cláusula ordena las tuplas en orden ascendente. Sin embargo, se puede especificar el orden, utilizando *desc* para un orden descendente o *asc* para un orden ascendente.
- **Duplicados:** Muchas veces, al realizar consultas en una base de datos, se pueden obtener tuplas duplicadas.

70

## Operaciones sobre conjuntos

En SQL existen operaciones que permiten operar sobre múltiples tablas y corresponden a las operaciones matemáticas de unión e intersección

- **Operación unión:** Permite combinar los resultados mostrados a partir de múltiples *select* y controla los duplicados
- **Operación intersección:** Permite combinar los resultados a partir de múltiples consultas y muestra los resultados no duplicados

71

## Funciones de agregación

Las funciones de agregación son funciones que devuelven un sólo valor a partir del cálculo de un conjunto de valores de entrada

- **Avg():** Devuelve el valor promedio de la columna. El tipo de dato de la columna dentro del argumento debe ser numérico.
- **Count():** Devuelve la cantidad de tuplas que coinciden con el criterio especificado
- **Min():** Devuelve el valor mínimo en la columna.
- **Max():** Devuelve el valor máximo en la columna.
- **Sum():** Devuelve el valor de la suma de todos los valores en la columna. El tipo de dato de la columna dentro del argumento debe ser numérico.

72

# Valores Nulos

Muchas veces, dentro de las tablas de una base de datos pueden existir campos que no tengan ningún valor almacenado, es decir, contienen valores nulos, los cuales pueden generar problemas al realizar diversas operaciones

- **And:** Muestra resultados si todas las condiciones separadas por el operador son ciertas.
- **Or:** Muestra resultados si alguna de las condiciones separadas por el operador es cierta.
- **Not:** Muestra resultados si la condición no es cierta.

Existen otros operadores que también permiten trabajar con valores nulos, como lo son el operador *is null* y el operador *is not null*.

- **Is null:** Permite comprobar si hay valores nulos
- **Is not null:** Permite comprobar si hay valores no nulos.

73

# Subconsultas anidadas

Una subconsulta anidada es una consulta que está dentro de otra consulta y permiten probar la pertenencia a conjuntos, comparación de conjuntos y comprobación de relaciones vacías o de tuplas duplicadas

- **Pertenencia a conjuntos:** Permite probar la pertenencia de tuplas en una relación, para lo cual se puede utilizar el operador *in*.
- **Comparación de conjuntos:** Para comparar conjuntos se utilizan los operadores de comparación junto con alguno de los siguientes operadores: *all* o *any*.
- **Comprobación de relaciones vacías:** SQL permite verificar si una subconsulta tiene tuplas en su resultado.
- **Comprobación de tuplas duplicadas:** Para verificar si hay tuplas duplicadas en el resultado de una subconsulta, se utiliza el operador *unique*.

74

# Vistas

Una vista es una tabla virtual que se deriva a partir de los datos de otras tablas que han sido previamente creadas con la instrucción *create table*.

75

# *Comandos de modificación de la base de datos*

Estos comandos o sentencias permiten agregar, eliminar o cambiar información con SQL en la base de datos

- **Borrado (DELETE)**

Este comando permite eliminar tuplas completas, mas no es posible eliminar el valor de un atributo

**Inserción (INSERT)**

- Este comando permite insertar tuplas en una tabla, se utiliza la cláusula *values* y los valores a insertar deben ser del tipo de dato apropiado para cada columna (Mannino, 2007).

**Actualizaciones (UPDATE)**

Este comando permite la modificación de una o más tuplas en una o más columnas, sin embargo, es necesario tomar en cuenta que si se desea modificar la llave primaria, es probable que las reglas para actualización de las tuplas referenciadas no lo permitan

**Actualización de vistas**

- En el caso de las vistas, existen algunas condiciones para que a estas se les pueda aplicar el comando de modificación.

76