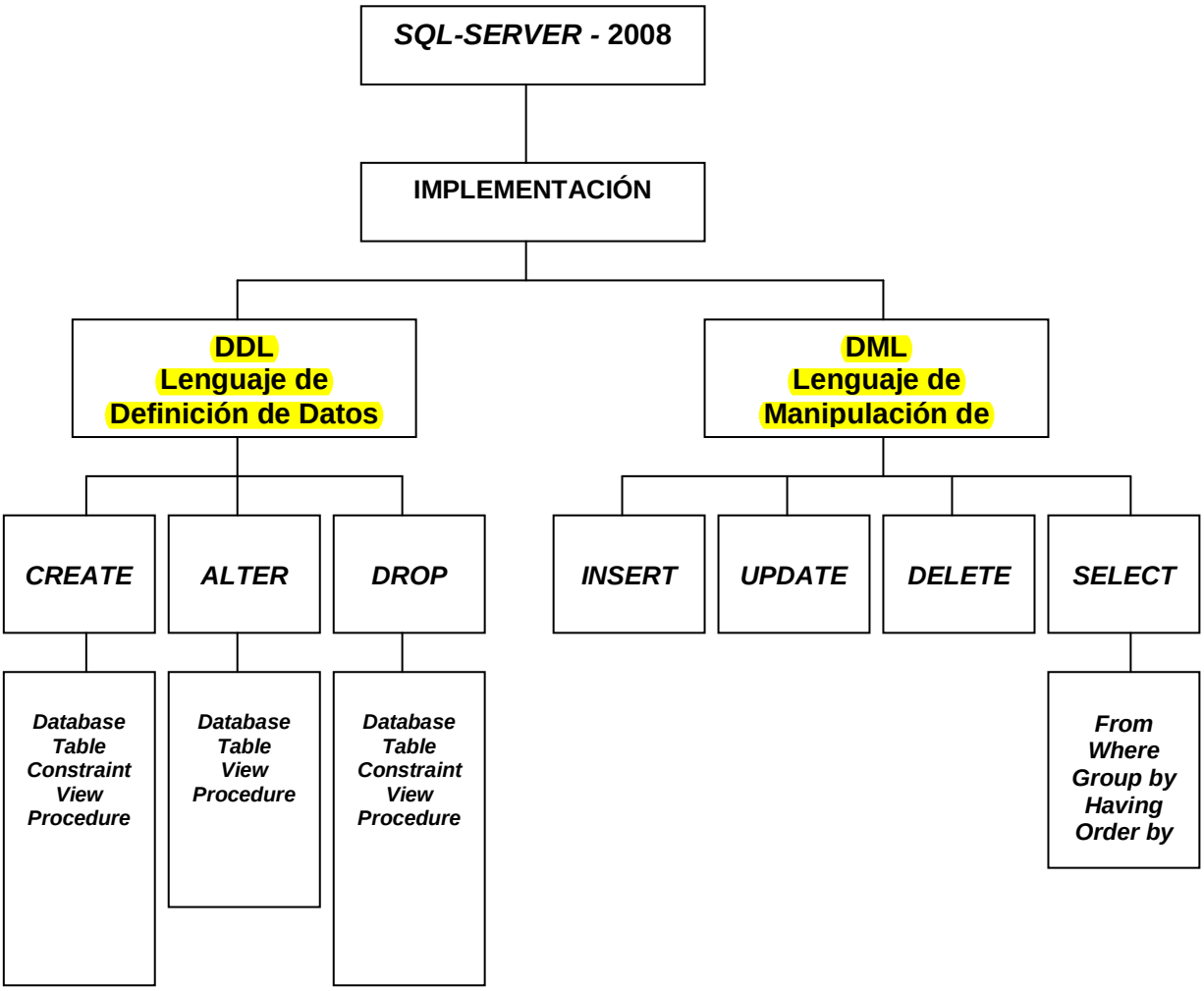
ÍNDICE

Página

Red de contenidos



1.1. TEMA 1: Introducción a las bases de datos

1.1.1. Lenguaje estructurado de consultas (SQL)

El lenguaje de consulta estructurado (*SQL*) es un lenguaje de base de datos normalizado, utilizado por los diferentes motores de bases de datos para realizar determinadas operaciones sobre los datos o sobre la estructura de los mismos.
Pero como sucede con cualquier sistema de normalización, hay excepciones para casi todo. De hecho, cada motor de bases de datos tiene sus peculiaridades y lo hace diferente de otro motor; por lo tanto, el lenguaje *SQL* normalizado (*ANSI*) no nos servirá para resolver todos los problemas, aunque sí se puede asegurar que cualquier sentencia escrita en *ANSI* será interpretable por cualquier motor de datos.

1.1.1.1. Historia del lenguaje estructurado

La historia de *SQL* empieza en 1974 con la definición, por parte de *Donald Chamberlin* y de otras personas que trabajaban en los laboratorios de investigación de *IBM*, de un lenguaje para la especificación de las características de las bases de datos que adoptaban el modelo relacional. Este lenguaje se llamaba *SEQUEL* (*Structured English Query Language*) y se implementó en un prototipo llamado *SEQUEL-XRM* entre 1974 y 1975. Las experimentaciones con ese prototipo condujeron, entre 1976 y 1977, a una revisión del lenguaje (*SEQUEL/2*) que, a partir de ese momento, cambió de nombre por motivos legales y se convirtió en *SQL*.

El prototipo (*System R*), basado en este lenguaje, se adoptó y utilizó internamente en *IBM* y lo adoptaron algunos de sus clientes elegidos. Gracias al éxito de este sistema, que no estaba todavía comercializado, otras compañías empezaron a desarrollar sus productos relacionales basados en *SQL*. A partir de 1981, *IBM* comenzó a entregar sus productos relacionales y, en 1983, empezó a vender *DB2*. En el curso de los años ochenta, numerosas compañías (por ejemplo *Oracle* y *Sybase*, sólo por citar algunas) comercializaron productos basados en *SQL*, que se convierte en el estándar industrial de hecho por lo que respecta a las bases de datos relacionales.

En 1986, el *ANSI* adoptó *SQL* (sustancialmente adoptó el dialecto *SQL* de *IBM*) como estándar para los lenguajes relacionales y en 1987 se transformó en estándar *ISO*. Esta versión del estándar va con el nombre de *SQL/86*. En los años siguientes, éste ha sufrido diversas revisiones que han conducido primero a la versión *SQL/89* y, posteriormente, a la actual *SQL/92*.

El hecho de tener un estándar definido por un lenguaje para bases de datos relacionales abre potencialmente el camino a la intercomunicación entre todos los productos que se basan en él. Desde el punto de vista práctico, por desgracia las cosas fueron de otro modo. Efectivamente, en general cada productor adopta e implementa, en la propia base de datos sólo el corazón del lenguaje *SQL* (el así llamado *Entry level* o al máximo el *Intermediate level*), y lo extiende de manera individual según la propia visión que cada cual tenga del mundo de las bases de datos.

Actualmente, está en marcha un proceso de revisión del lenguaje por parte de los comités *ANSI* e *ISO*, que debería terminar en la definición de lo que en

este momento se conoce como *SQL3*. Las características principales de esta nueva encarnación de *SQL* deberían ser su transformación en un lenguaje *stand-alone* (mientras ahora se usa como lenguaje hospedado en otros lenguajes) y la introducción de nuevos tipos de datos más complejos que permitan, por ejemplo, el tratamiento de datos multimedia.

1.1.1.2. Componentes del *SQL*

El lenguaje *SQL* está compuesto por comandos, cláusulas, operadores y funciones de agregado. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos.

Existen dos (2) tipos de comandos *SQL*:

- Los comandos del Lenguaje de Definición de Datos (DDL) que permiten crear y definir nuevas bases de datos, campos e índices.
- Los comandos del Lenguaje de Manipulación de Datos (DML) que permiten modificar y generar consultas para insertar, modificar o eliminar, así como, ordenar, filtrar y extraer datos de la base de datos.

1.1.1.2.1. Comandos del DDL

Comando	Descripción
CREATE	Utilizado para crear nuevas tablas, campos e índices
DROP	Empleado para eliminar tablas e índices
ALTER	Utilizado para modificar las tablas y agregar campos o cambiar la definición de los campos.

1.1.1.2.2. Comandos del DML

Comando	Descripción
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Utilizado para ingresar registros de datos en la base de datos en una única operación
UPDATE	Utilizado para modificar los valores de los campos y registros especificados
DELETE	Utilizado para eliminar registros de una tabla de una base de datos

1.1.1.2.3. Cláusulas

Comando	Descripción
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

1.1.1.2.4. Operadores Lógicos

Comando	Descripción
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta
NOT	Negación lógica. Devuelve el valor contrario de la expresión

1.1.1.2.5. Operadores de Comparación

Comando	Descripción
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó igual que
>=	Mayor ó igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores
LIKE	Utilizado en la comparación de un modelo
In	Utilizado para especificar registros de una base de datos

1.1.1.3.Funciones para el manejo de fechas y función de conversión

Función	Descripción
GETDATE()	Devuelve la fecha del día
DAY(`fecha`***)***	Devuelve un entero que representa el día (día del mes) de la fecha especificada
MONTH(`fecha`***)***	Devuelve un entero que representa el mes de la fecha especificada
YEAR(`fecha`***)***	Devuelve un entero que representa la parte del año de la fecha especificada
DATEDIFF(*parte,fi,ff****)***	Devuelve el número de límites *datepart* de fecha y hora entre dos fechas especificadas. `DATEDIFF ( parte , fechaInicio , fechaFin)`
DATEPART*(**parte*,`fecha`***)***	Devuelve un entero que representa el parámetro parte especificado (dia, mes o año) del parámetro fecha especificada.
CONVERT	Convierte una expresión de un tipo de datos en otro. `CONVERT ( tipodedato [ ( lengitud ) ] , expresión [ , stilo ] )`

1.1.1.4 Funciones de agregado

Las funciones de agregado se usan dentro de una cláusula *SELECT*, en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

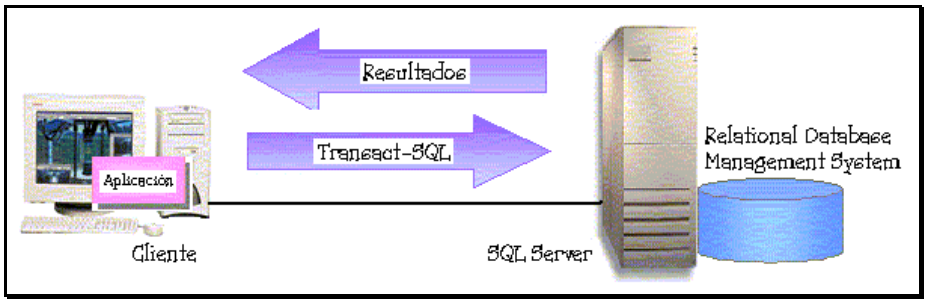
Comando	Descripción
AVG(exp**)**	Se emplea para calcular el promedio de los valores de un campo determinado.
COUNT(*exp****)***	Se emplea para devolver la cantidad de registros de la selección.
SUM(*exp****)***	Se emplea para devolver la suma de todos los valores de un campo determinado.
MAX(*exp****)***	Se emplea para devolver el valor más alto de un campo o expresión especificada.
MIN(*exp****)***	Se emplea para devolver el valor más bajo de un campo o expresión especificada.

1.1.2. IMPORTANCIA DE LA BASE DE DATOS

Las bases de datos son importantes porque permiten almacenar grandes cantidades de información en forma estructurada, consistente e íntegra y dan la posibilidad a un desarrollador de utilizarlas mediante programas (aplicaciones); además, les proporciona a éstos una herramienta bajo la cual puedan reducir considerablemente el tiempo del proceso de búsqueda en profundidad de los datos almacenados.

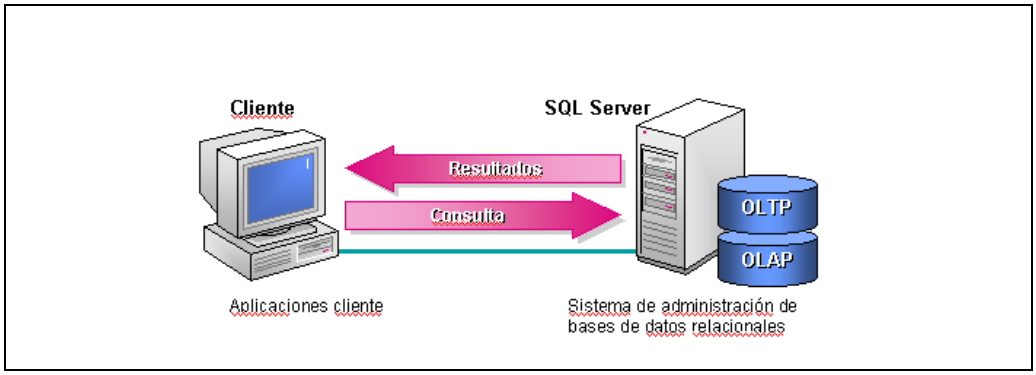
1.1.2.1. Implementación de las base de datos con *SQL SERVER*

SQL Server es un sistema administrador para Bases de Datos relacionales basadas en la arquitectura Cliente / Servidor (*RDBMS*) que usa *Transact SQL* para mandar peticiones entre un cliente y el *SQL Server*.



1.1.2.1.1. Arquitectura Cliente / Servidor

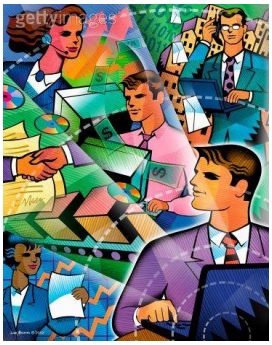
SQL Server usa la arquitectura Cliente / Servidor para separar la carga de trabajo en tareas que se ejecuten en computadoras tipo Servidor y tareas que se ejecuten en computadoras tipo Cliente:

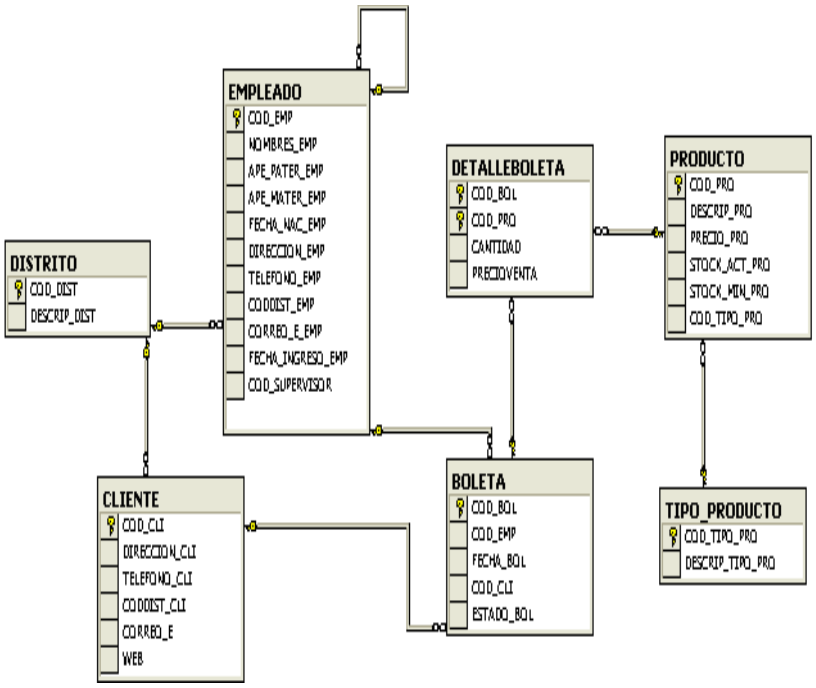


- El Cliente es responsable de la parte lógica y de presentar la información al usuario. Generalmente, el cliente ejecuta en una o más computadoras Cliente, aunque también puede ejecutarse en una computadora que cumple las funciones de Servidor con *SQL Server*.
- *SQL Server* administra bases de datos y distribuye los recursos disponibles del servidor tales como memoria, operaciones de disco, etc. Entre las múltiples peticiones.
- La arquitectura Cliente/Servidor permite desarrollar aplicaciones para realizarlas en una variedad de ambientes.
- *SQL Server* 2008R2 trabaja con dos (2) tipos de bases de datos:

- ***OLTP* (*Online Transaction Processing*)**

Son bases de datos caracterizadas por mantener una gran cantidad de usuarios conectados concurrentemente realizando ingreso y/o modificación de datos. Por ejemplo: entrada de pedidos en línea, inventario, contabilidad o facturación.





1.2.3 TIPOS DE DATOS

Los tipos de datos definen el valor de datos que se permite en cada columna. *SQL Server* proporciona varios tipos de datos diferentes. Ciertos tipos de datos comunes tienen varios tipos de datos de *SQL Server* asociados. Se debe elegir los tipos de datos adecuados que permitan optimizar el rendimiento y conservar el espacio en el disco.

1.2.3.1. Categorías de tipos de datos del sistema

La siguiente tabla asocia los tipos de datos comunes con los tipos de datos del sistema proporcionados por *SQL Server*. La tabla incluye los sinónimos de los tipos de datos por compatibilidad con ANSI[1].

Tipos de datos comunes	Tipos de datos del sistema de SQL Server	Sinómino ANSI	Número de Bytes
Entero	**Int** **bigint** **smallint, tinyint**	Integer - -	4 8 2,1
Numérico Exacto	**decimal[(p[,s])]** **numeric[(p[,s])]**	Dec -	2-17
Numérico Aproximado	**float[(n)]** **real**	double precisión, float[(n)] para n=8-15 float[(n)] para n=1-7	8 4
Moneda	**money,** **smallmoney**	-	8,4
Fecha y hora	**datetime,** **smalldatetime**	-	8 4
Carácter	**char[(n)]** **varchar[(n)]** **text**	character [(n)] char VARYING[(n)] character VARYING[(n)] -	0-8000 0-2 GB
Caracteres Unicote	**nchar[(n)]** **nvarchar[(n)]** **ntext**	-	0-8000 (4000 caracteres) 0-2 GB
Binario	**binary [(n)]** **varbinary [(n)]**	binary VARYING[(n)]	0-8000
Imagen	**Image**	-	0 a 2 GB

Nota: *SQL SERVER* 2008R2 agrega nuevos tipos de datos. Estos nuevos tipos de datos permiten una mejora en el almacenamiento y en el trabajo con tipos de datos fecha y hora, incluyendo múltiples zonas horarias y cálculos mejorados. Estos nuevos tipos de datos son: datetime2, date, time y datetimeoffset.

[1] American Nacional Standards Institute