Base de Datos 1

Unidad 5

SQL

Historia

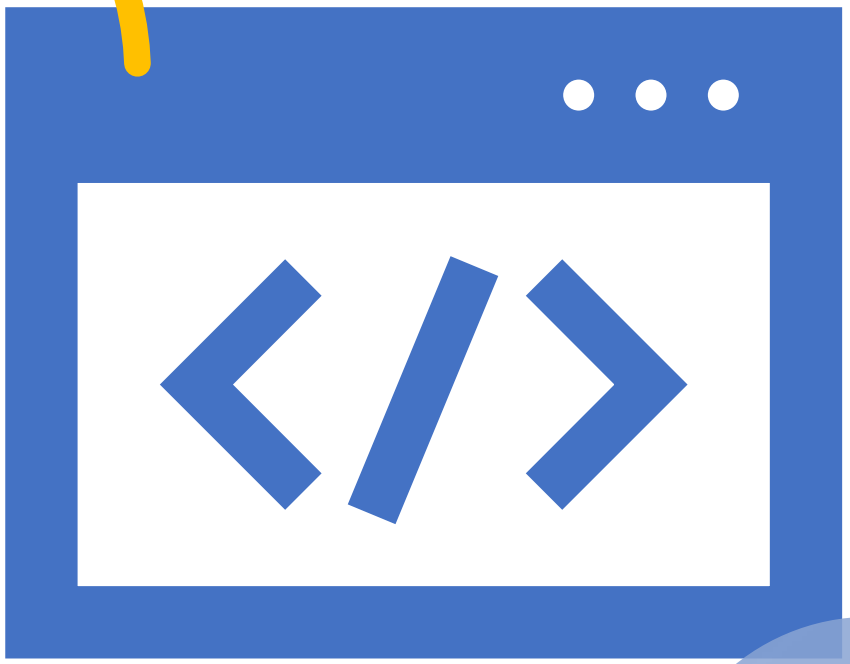
- El lenguaje SQL se introdujo como lenguaje de consulta del Sistema R. Posteriormente, varios sistemas comerciales lo adoptaron como lenguaje para sus bases de datos.
- El nombre SQL está formado por las iniciales de Structured Query Language (Lenguaje de consultas estructuradas).
- Si bien el SQL está definido como un estándar, las bases de datos relacionales tienen sus extensiones propietarias y variaciones.

ANSI/ISO

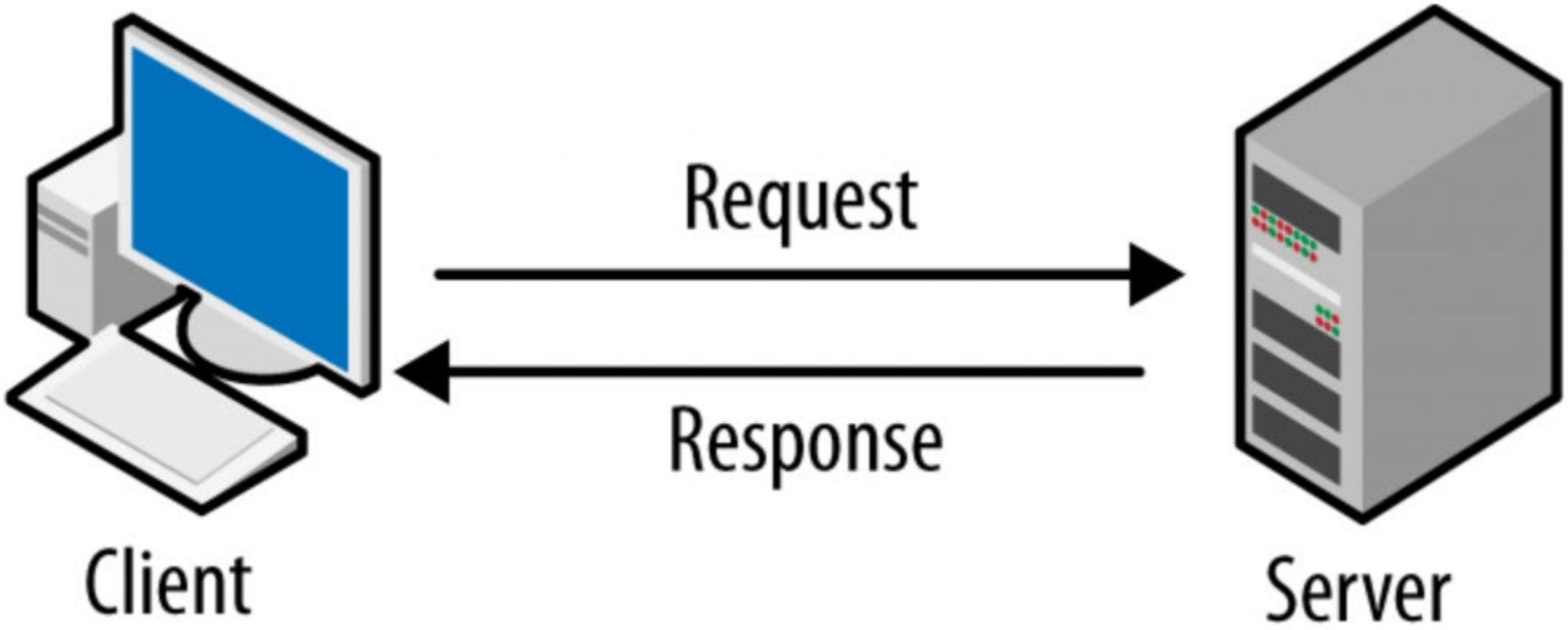
Año	Comentarios
1986	Primera publicación hecha por ANSI. Confirmada por ISO en 1987.
1989	Revisión menor, el agregado más importante fueron las restricciones de integridad. Adoptado como FIPS 127-1.
1992	Revisión mayor (ISO 9075)
1999	Expresiones regulares, consultas recursivas (para relaciones jerárquicas), triggers y algunas características orientadas a objetos.
2003	Introduce algunas características de XML, cambios en las funciones, estandarización del objeto sequence y de las columnas auto numéricas.
2006	ISO/IEC 9075-14:2006 Define las maneras en las cuales el SQL se puede utilizar conjuntamente con XML. Define maneras de importar y guardar datos XML en una base de datos SQL, manipulándolos dentro de la base de datos y publicando el XML y los datos SQL convencionales en forma XML. Además, proporciona facilidades que permiten a las aplicaciones integrar dentro de su código SQL el uso de XQuery, lenguaje de consulta XML publicado por el W3C (World Wide Web Consortium) para acceso concurrente a datos ordinarios SQL y documentos XML.
2008	Permite el uso de la cláusula ORDER BY fuera de las definiciones de los cursores. Incluye los disparadores del tipo INSTEAD OF. Añade la sentencia TRUNCATE
2011	Soporte para bases de datos temporales (cronológicas). Mejoras en las funciones de ventana y cláusula FETCH
2016	Búsqueda de patrones, funciones de tabla polimórficas y compatibilidad con los ficheros JSON.
2019	Arrays Multidimensionales (SQL/MDA)
2023	Se agrega el tipo JSON (SQL/Foundation); Se agrega parte 16, Property Graph Queries (SQL/PGQ)

Categorías de instrucciones

- **DML** (Database Manipulation Language o Lenguaje de manipulación de datos): SELECT, INSERT, UPDATE, DELETE, TRUNCATE
- **DDL** (Database Definition Language o Lenguaje de definición de datos): CREATE, ALTER, DROP
- **DCL** (Database Control Language o Lenguaje de control de datos): GRANT, DENY, REVOKE
- **TCL** (Transaction Control Language o Lenguaje de control de transacciones): COMMIT, ROLLBACK, SAVEPOINT



Cliente / Servidor

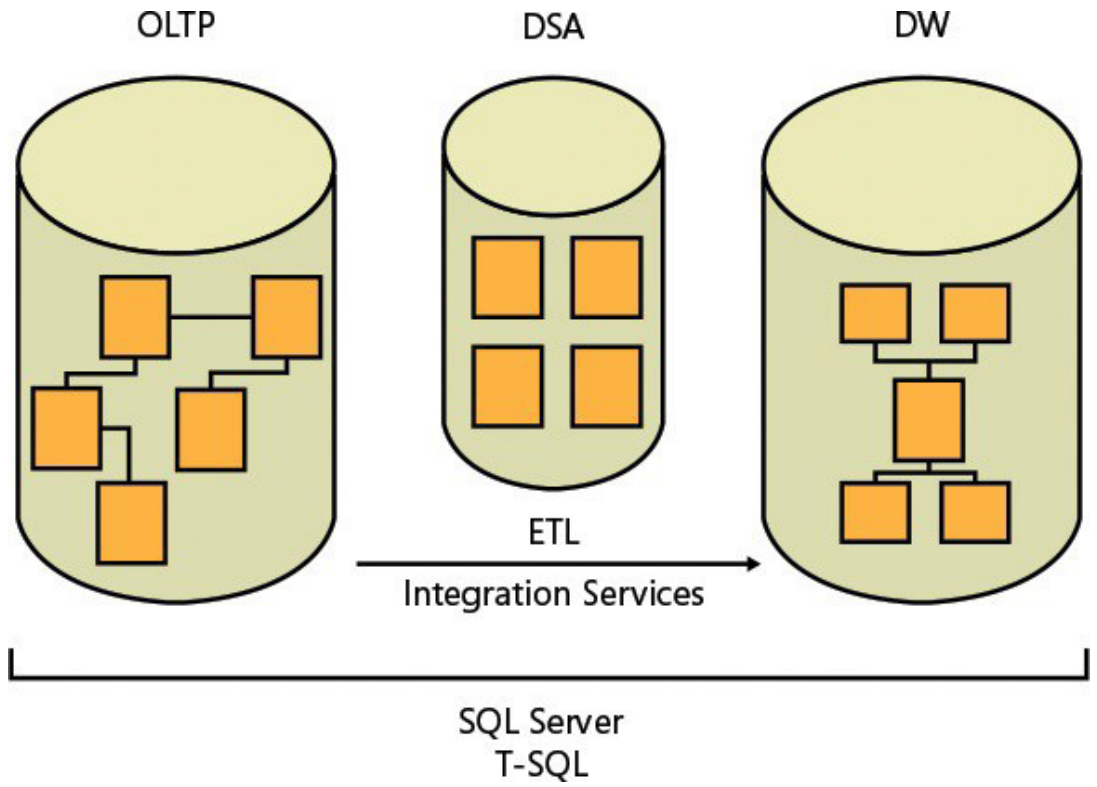


Sistemas de Gestión de Bases de Datos Relacionales

- MS SQL Server
- Oracle Database
- IBM DB2
- SAP ASE (ex Sybase)
- MySQL
- MariaDB
- PostgreSQL
- SQLite

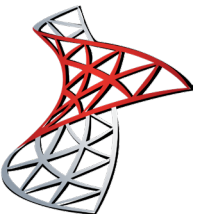
Tipos de Sistemas de Bases de Datos

- **OLTP**: OnLine Transactional Processing
- **DW**: DataWarehouse
- **DSA**: Data-staging área
- **ETL**: Extract, Transform and Load



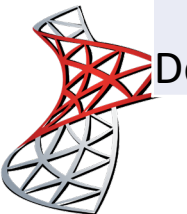
SQL Server – Ediciones Principales

- **Enterprise**: Edición completa, con performance para aplicaciones de misión crítica y con requerimientos de inteligencia de negocios. Provee los más altos niveles de servicio y performance.
- **Standard**: Administración de datos central y capacidad para inteligencia de negocios para aplicaciones que no sean de misión crítica con el mínimo de recursos de IT
- **Developer**: Versión con todas las características orientada a desarrolladores, que permite desarrollar, probar y demostrar aplicaciones basadas en software SQL Server.
- **Web**: Versión especial para proveedores de hosting.
- **Express**: Nivel básico y gratuita. Ideal para el aprendizaje y construcción de aplicaciones de escritorio o pequeños servidores. Permite hasta 10 GB de almacenamiento.
- https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2022



UNPAZ

Características	SQL Server Enterprise	SQL Server Standard	SQL Server Express	SQL Server Developer
Número máximo de núcleos	Máximo del SO	24	4	Máximo del SO
Memoria: Tamaño máximo de pool de buffer por instancia	Máximo del Sistema Operativo	128 GB	1410 MB	Máximo del Sistema Operativo
Memory: Máximo segmento chache Columnstore por instancia	Máximo del Sistema Operativo	32 GB	352 MB	Máximo del Sistema Operativo
Memoria: Máximo memoria optimizada para datos por base de datos	Máximo del Sistema Operativo	32 GB	352 MB	Máximo del Sistema Operativo
Tamaño máximo de base de datos	524 PB	524 PB	10 GB	524 PB
Derecho de uso en producción	Sí	Sí	Sí	No



UNPAZ

MySQL – Ediciones

- **MySQL Community Edition**: Versión gratuita. Licencia GPL. El soporte lo brinda la comunidad.
- Ediciones Comerciales: Brindan acceso a soporte de Oracle 24x7, y otras características comerciales.
  - MySQL **Standard** Edition
  - MySQL **Enterprise** Edition
  - MySQL **Cluster** Carrier Grade Edition

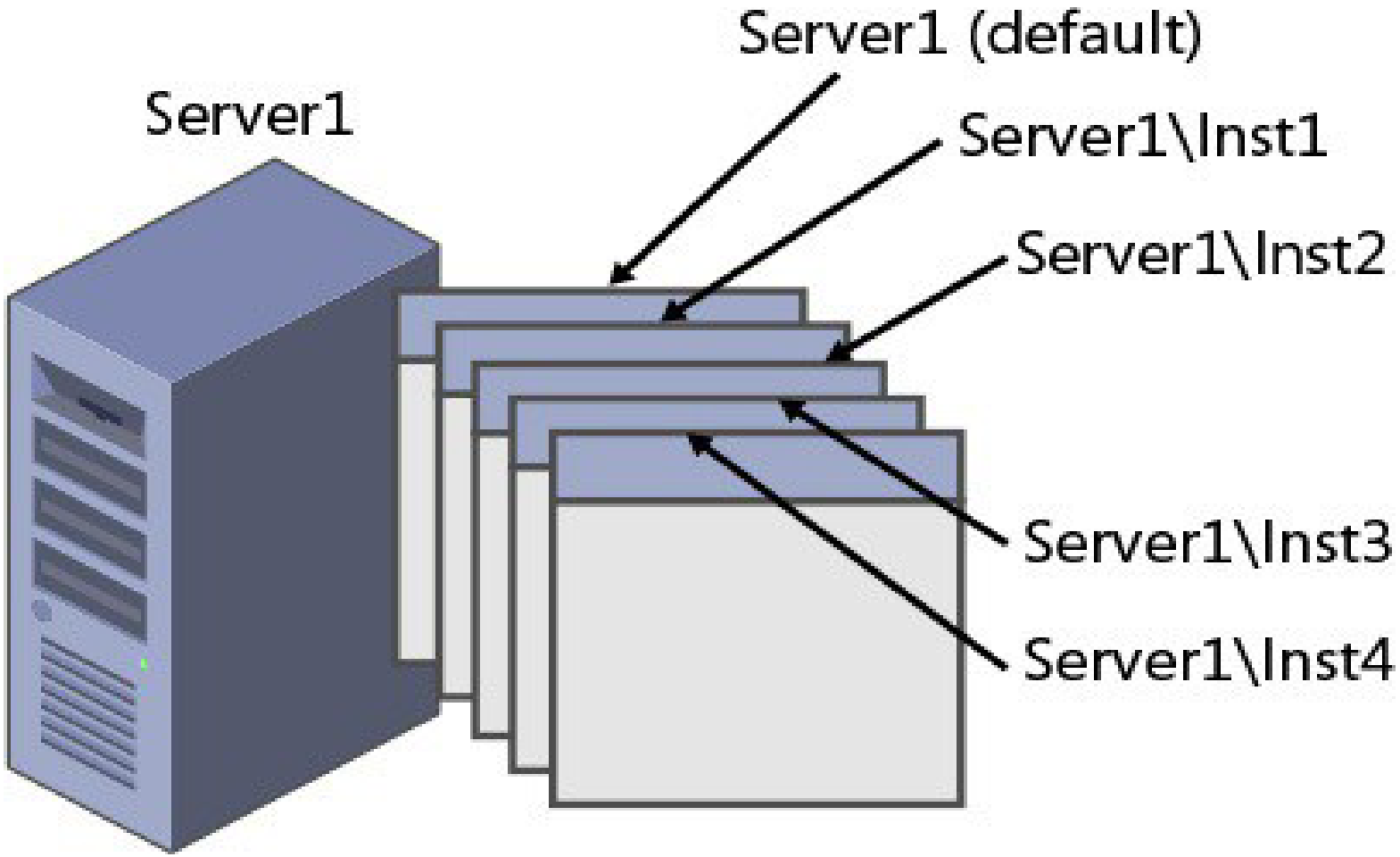
Arquitectura SQL (ABC)

APPLIANCE (APARATO)

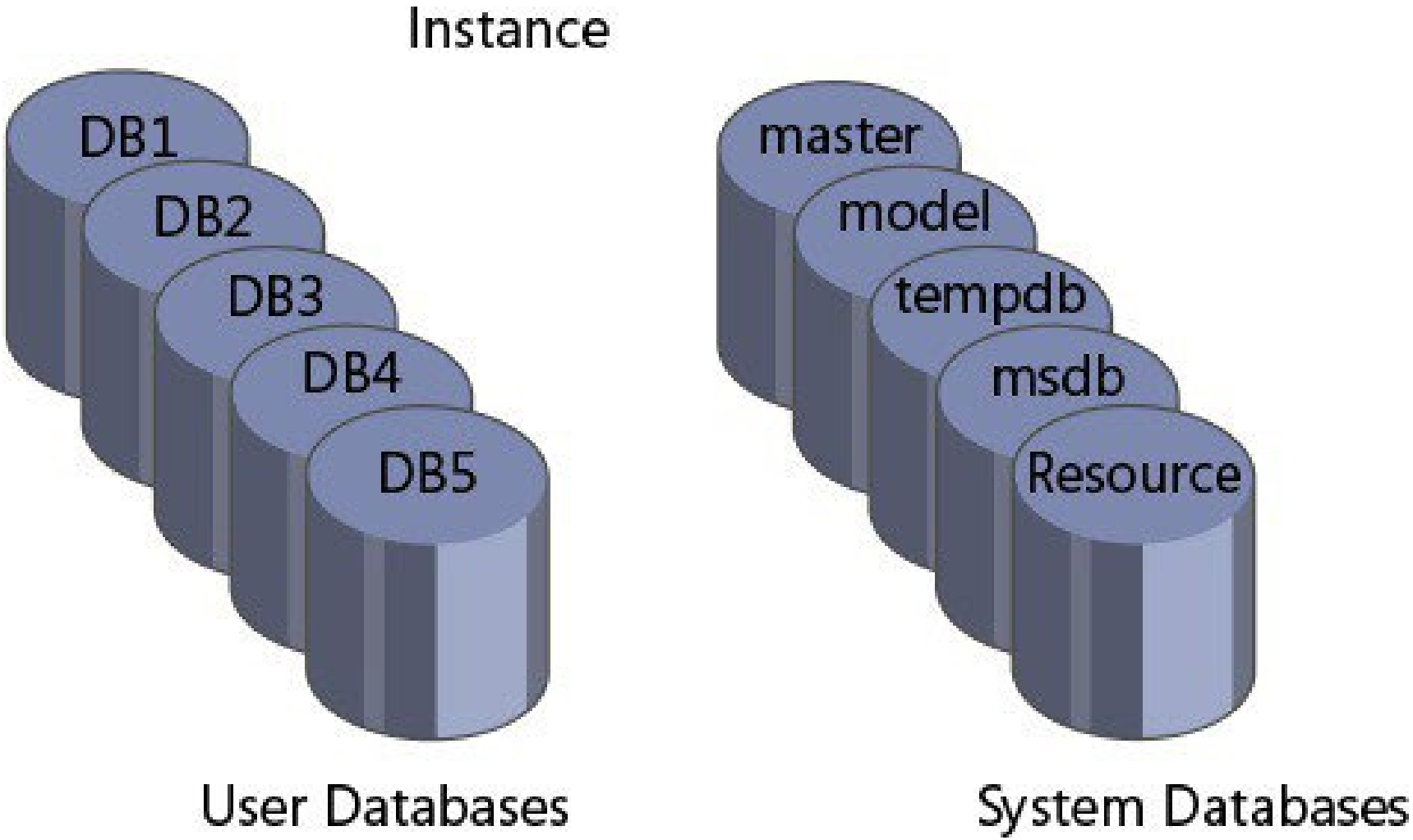
BOX (CAJA)

CLOUD (NUBE)

Instancias


Server1
Server1 (default)
Server1\Inst1
Server1\Inst2
Server1\Inst3
Server1\Inst4

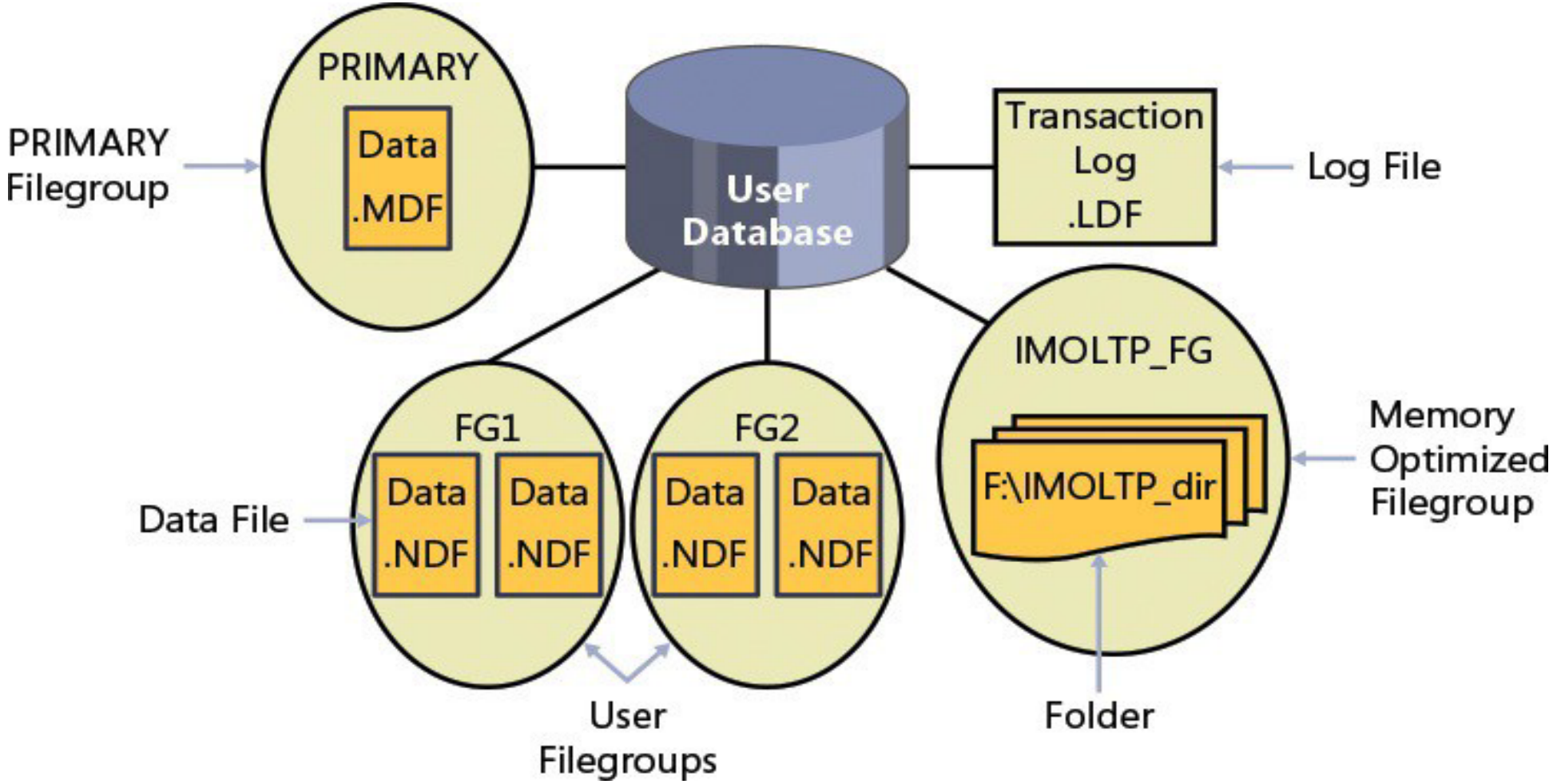
SQL Server - Bases de Datos



MySQL – Bases de Datos

- **mysql**: es la base de datos de sistema que contiene tablas con la información requerida por MySQL server
- **information_schema**: permite el acceso a la metadata de la base de datos
- **performance_schema**: es una característica que permite monitorear la ejecución de MySQL Server a bajo nivel
- **sys**: conjunto de objetos que ayudan al DBA y a los desarrolladores a interpreter los datos colectados por el performance_schema

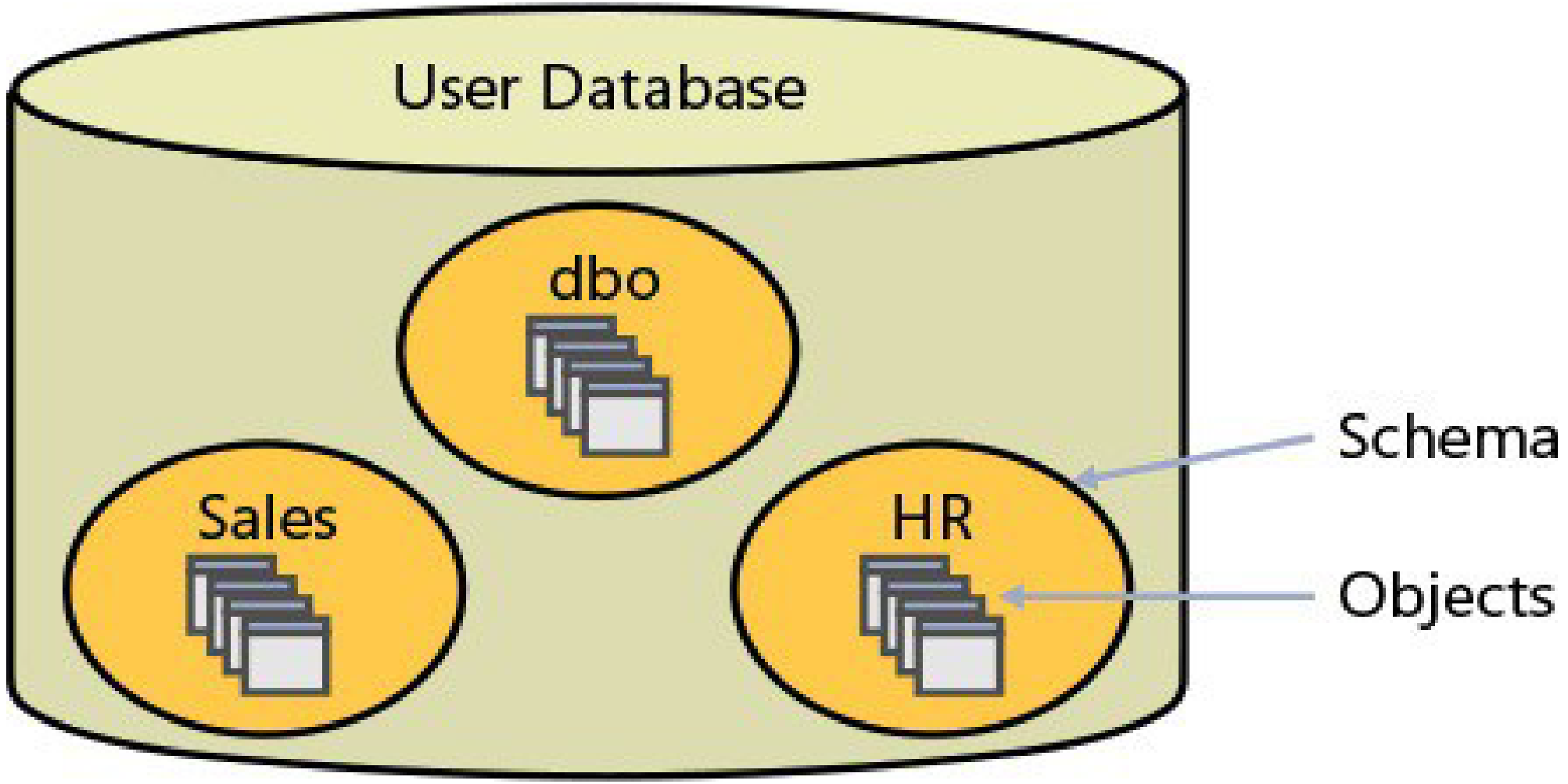
SQL Server – Archivos Físicos de las Bases de Datos de Usuario



MySQL – Archivos físicos

- El path por defecto para los datos es: /var/lib/mysql
  - Cada directorio corresponde a una base (de Sistema o de Usuario)
  - Logs
  - InnoDB tablespaces y logs

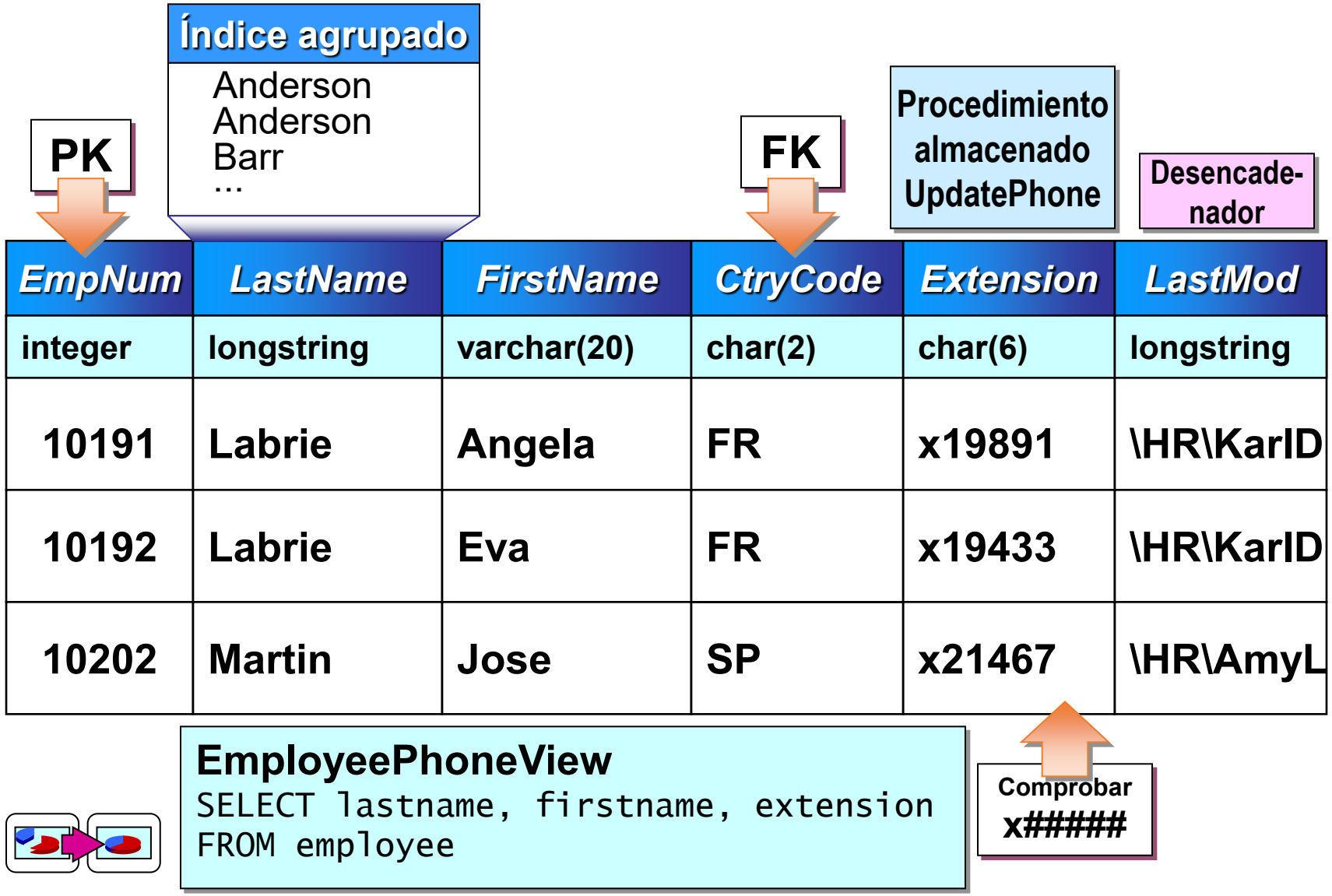
SQL Server – Esquemas y objetos



MySQL - Esquemas

- En MySQL Esquema es un sinónimo de Base de Datos.

Objetos de base de datos



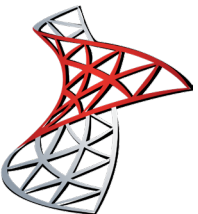
Referencia a los objetos de SQL Server

- Nombres completos

  servidor.baseDeDatos.esquema.objeto

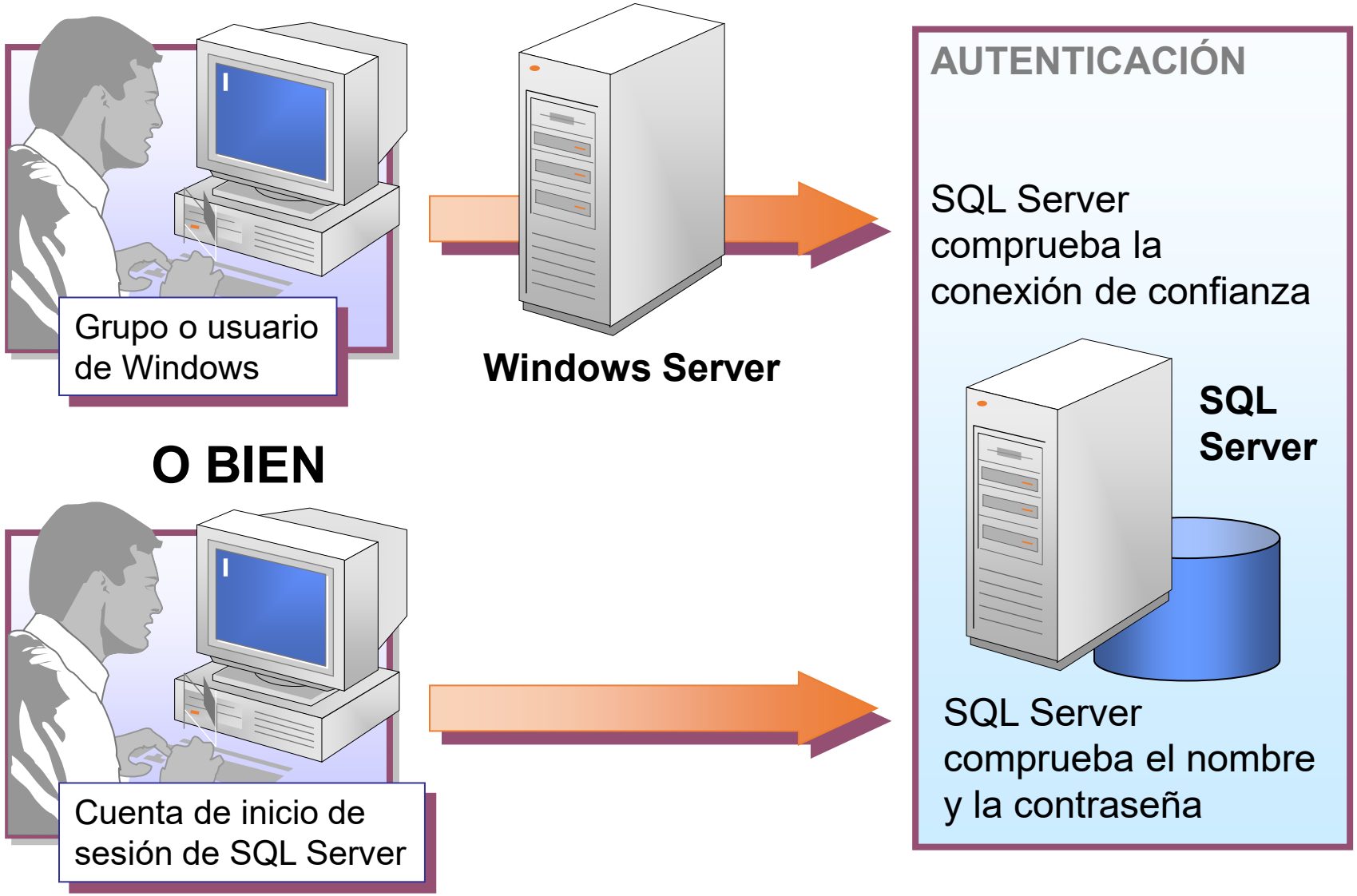
- Nombres parcialmente especificados
  - El servidor predeterminado es la instancia actual del servidor local
  - La base de datos predeterminada es la base de datos actual
  - El esquema predeterminado es dbo

```
CREATE TABLE Northwind.dbo.OrderHistory
.
.
.
```

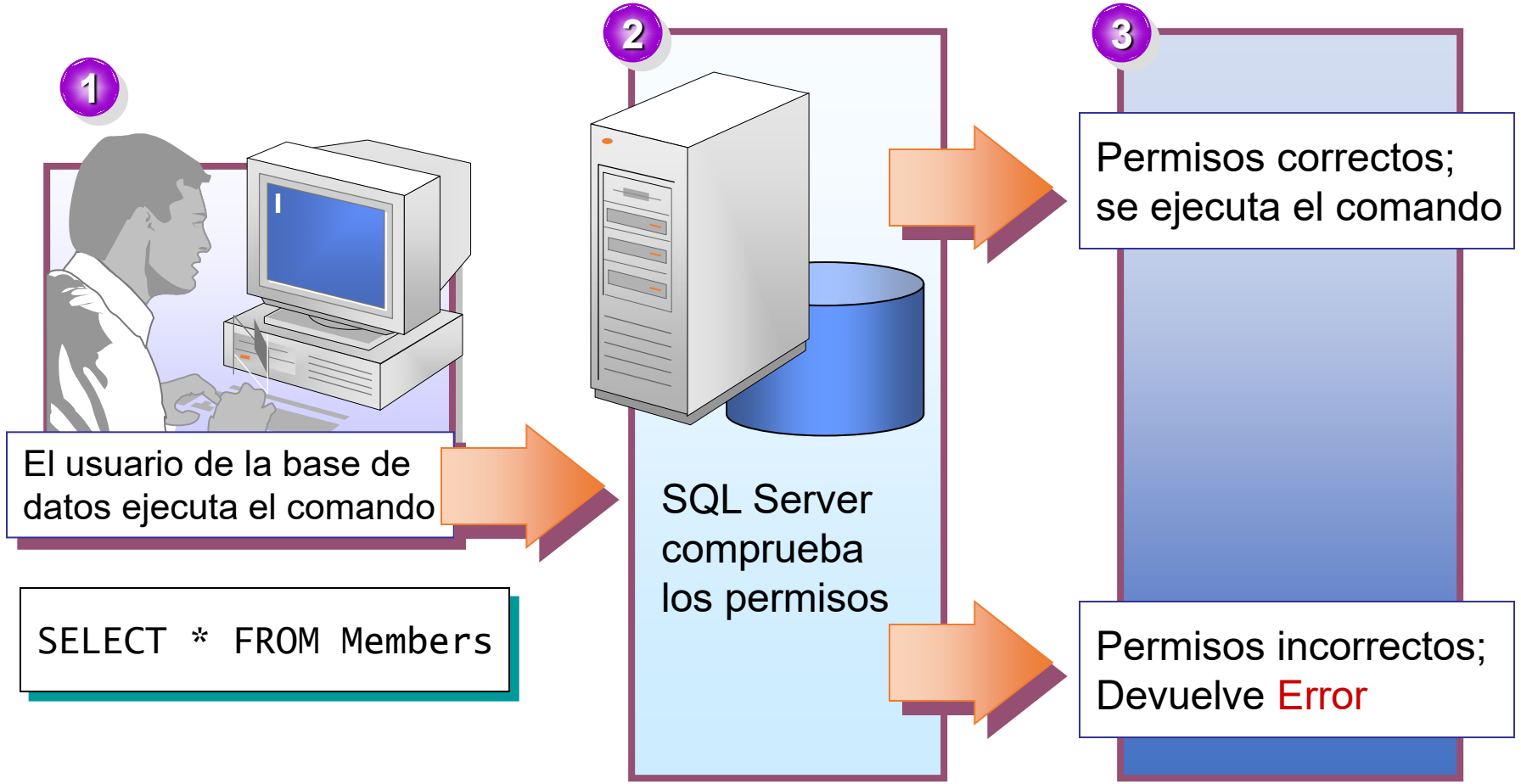


UNPAZ

Autenticación de inicio de sesión

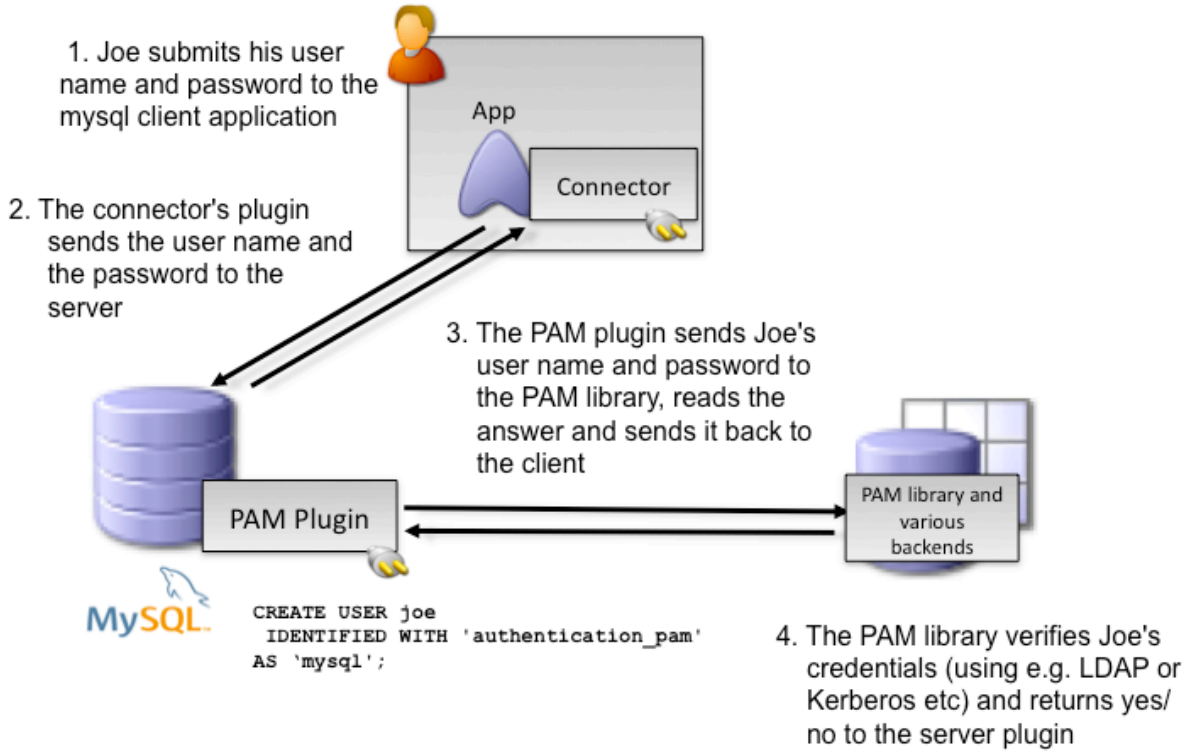


Validación de permisos



MySQL – Autenticación de usuarios

- MySQL (interna)
- Externa
  - External Authentication for LDAP
  - Native Kerberos Authentication
  - External Authentication for Windows
  - External Authentication for PAM



Tipos de Datos

- https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql
- https://dev.mysql.com/doc/refman/8.0/en/data-types.html
- Se pueden agrupar en:
  - Numéricos exactos (Enteros y de punto fijo)
  - Numéricos aproximados
  - Fecha y Hora
  - Cadenas de caracteres
  - Cadenas de caracteres Unicode
  - Cadenas Binarias
  - Otros

Tipos de Datos – Enteros

SQL Server

- **bigint**: -2^63 a 2^63-1
- **bit**: 1, 0 o NULL
- **int**: -2^31 (-2,147,483,648) a 2^31-1 (2,147,483,647)
- **smallint**: -2^15 (-32,768) a 2^15-1 (32,767)
- **tinyint**: 0 a 255

MySQL

- **bigint**: -2^63 a 2^63-1
- **int**: -2,147,483,648 a 2,147,483,647
- **mediumint**: -8,388,608 a 8,388,607
- **smallint**: --32,768 a 32,767
- **tinyint**: 0 a 255

Tipos de Datos – Punto Fijo

SQL Server

- **decimal (p,s)**: p de 1 a 38. s de 0 a p
- **money**: -922,337,203,685,477.5808 a 922,337,203,685,477.5807
- **numeric**: equivalente a decimal
- **smallmoney**: - 214,748.3648 a 214,748.3647

MySQL

- **decimal (p,s)**: p de 1 a 65. s de 0 a p
- **numeric**: equivalente a decimal

Tipos de Datos – Números aproximados

SQL Server

- **float (n)**: n de 1 a 53, siendo n la cantidad de bits usados para la mantisa. - 1.79E+308 a -2.23E-308, 0 y 2.23E-308 a 1.79E+308
- **real**: float(24)

MySQL

- **float (n)**: n de 0 a 23
- **double(n)**: n de 24 a 53

Tipos de Datos – Fecha y Hora

SQL Server

- **date**: YYYY-MM-DD. De 0001-01-01 hasta 9999-12-31
- **datetime2**: YYYY-MM-DD hh:mm:ss[.fracción de segundo]. De 0001-01-01 hasta 9999-12-31 y 00:00:00 hasta 23:59:59.9999999
- **datetime**: De 1 de Enero de 1753 hasta 31 de Diciembre de 9999 y 00:00:00 hasta 23:59:59.997
- **datetimeoffset**: YYYY-MM-DD hh:mm:ss[.nnnnnnn] [{+|-}hh:mm]. El offset va de -14:00 hasta +14:00
- **smalldatetime**: De 1900-01-01 hasta 2079-06-06 y 00:00:00 hasta 23:59:59
- **time**: hh:mm:ss[.nnnnnnn] De 00:00:00.0000000 hasta 23:59:59.9999999

MySQL

- **date**: YYYY-MM-DD. De 1001-01-01 hasta 9999-12-31
- **datetime**: De 1001-01-01 hasta 9999-12-31 y 00:00:00.000000 hasta 23:59:59.999999
- **time**: hh:mm:ss[.nnnnnnn] De -853:59:59.000000 a 838:59:59.000000
- **timestamp**: De 1970-01-01 00:00:01.000000 a 2038-01-19 03:14:07.999999
- **year**: De 1901 a 2155, o 0000

Tipos de Datos – Cadenas de caracteres

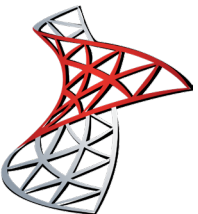
SQL Server

- **char(n)**: n de 1 a 8000. Longitud fija.
- **text**: será removido. No usar
- **varchar(n | max)**: n de 1 a 8000. Longitud variable

MySQL

- **char(n)**: n de 0 a 255. Longitud fija.
- **longtext**: longitude máxima de 4.294.967.295 o 4GB (232 – 1) bytes
- **mediumtext**: longitud máxima de 16.777.215 (224 – 1) bytes
- **text(n)**: longitud máxima de 65535 (216 - 1) bytes
- **tinytext**: longitud máxima 255 (28 - 1) bytes
- **varchar(n | max)**: n de 1 a 65535. Longitud variable

Tipos de Datos – Cadenas de caracteres Unicode



SQL Server

- **nchar(n)**: n de 1 a 4000. Longitud fija. Soporta caracteres Unicode
- **ntext**: será removido. No usar
- **nvarchar(n | max)**: n de 1 a 4000. Longitud variable

MySQL

- No maneja tipos diferenciados, si no que permite utilizar los mismos tipos, diferenciando la capacidad de manejar caracteres UNICODE mediante la utilización de CHARACTER SET, por ejemplo:

```
CREATE TABLE t
(
c1 VARCHAR(20) CHARACTER SET utf8,
c2 TEXT CHARACTER SET latin1 COLLATE
latin1_general_cs
);
```

Tipos de Datos – Cadenas Binarias

SQL Server

- **binary(n)**: n de 1 a 8000. cadena binaria de longitud fija
- **image**: será removido. No usar
- **varbinary(n)**: n de 1 a 8000. cadena binaria de longitud variable

MySQL

- **binary(n)**: similar a char, pero almacena cadena binaria
- **blob(n)**: longitud máxima 65.535 (216 – 1) bytes. Cada blob es almacenado usando un prefijo de 2 bytes que indica la cantidad de bytes en el valor.
- **longblob**: columna blob con longitud máxima de 4.294.967.295 o 4GB (232 – 1) bytes. Usa prefijo de 4 bytes.
- **mediumblob**: columna blob con longitud máxima de 16.777.215 (224 – 1) bytes. Usa prefijo de 3 bytes.
- **tinyblob**: columna blob con longitud máxima de 255 (28 – 1) bytes
- **varbinary(n)**: similar a varchar, pero almacena cadena binaria

Otros Tipos de Datos

SQL Server

- cursor: permite almacenar el resultado de una consulta o tabla y ser recorrido
- hierarchyid: representa posición en una jerarquía
- sql_variant: permite almacenar distintos tipos de datos variant
- table: se utiliza como almacenamiento temporal
- rowversion: número binario único generado automáticamente por la base de datos
- uniqueidentifier: cadena única de 16 bytes
- xml: almacena datos XML
- Spatial Types: Geométricos y Geográficos

MySQL

- JSON
- Spatial Types

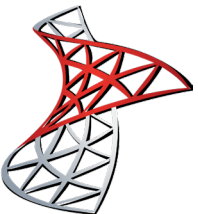
Creación y eliminación de tipos de datos definidos por el usuario

Creación

```
CREATE TYPE ssn
FROM varchar(11) NOT NULL ;
```

Eliminación

```
DROP TYPE ssn ;
```



UNPAZ

Directrices para especificar tipos de datos

- Si la longitud de la columna varía, utilice uno de los tipos de datos variables
- Para tipos de datos numéricos, use los decimales más frecuentes
- Para la moneda utilice el tipo de datos money (en SQL Server, en MySQL DECIMAL(19,4))
- No utilice float y real como claves principales

Creación y eliminación de una tabla

- Creación de una tabla

Nombre de columna	*Tipo de datos*	*NULL o NOT NULL*
CREATE TABLE dbo.Categories (CategoryID CategoryName Description Picture	 int IDENTITY(1,1) nvarchar(15) nvarchar(100) varbinary(max)	 NOT NULL, NOT NULL, NULL, NULL)

- Eliminación de una tabla

```
DROP TABLE tabla
```

Generación de valores de columnas

SQL Server

- Uso de la propiedad Identity
- Uso de la función NEWID y el tipo de datos uniqueidentifier

MySQL

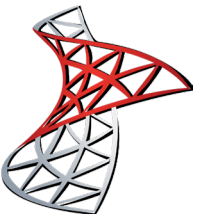
- AUTO_INCREMENT
- UUID()

Uso de la propiedad Identity

- Requisitos para utilizar la propiedad Identity
  - Sólo se permite una columna de identidad por tabla
  - Utilizar con tipos de datos **integer**, **numeric** y **decimal**
- Recuperar información acerca de la propiedad Identity
  - Utilizar IDENT_SEED e IDENT_INCR para información de definición
  - Utilizar **@@identity** para determinar el valor más reciente

IDENTITY [(*seed* , *increment*)]

```
CREATE TABLE Customer
(CustID int IDENTITY(1,1),
 CustName char(30) NOT NULL)
```



UNPAZ

Usando AUTO_INCREMENT

```
CREATE TABLE animals (
id MEDIUMINT NOT NULL
AUTO_INCREMENT,
name CHAR(30) NOT NULL,
PRIMARY KEY (id)
);
INSERT INTO animals (name)
VALUES
('dog'),('cat'),('penguin'),
('lax'),('whale'),('ostrich');

SELECT * FROM animals;
```

```
+----+---------+
| id | name    |
+----+---------+
| 1  | dog     |
| 2  | cat     |
| 3  | penguin |
| 4  | lax     |
| 5  | whale   |
| 6  | ostrich |
+----+---------+
```

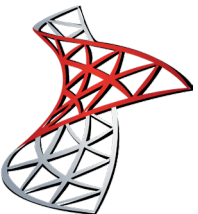
UNPAZ

Uso de la función NEWID y el tipo de datos uniqueidentifier

- Estas características se utilizan juntas
- Asegurar valores únicos globales
- Utilizar con la restricción DEFAULT

```
CREATE TABLE Customer
(CustID uniqueidentifier NOT NULL DEFAULT NEWID(),
 CustName char(30) NOT NULL)
```

```
CustID                               CustName
------------------------------------ ------------------------------
F74E709D-DBDF-4007-AEF6-132989EBA5A0 Wartian Herkku
FA6AE901-A306-44F5-A86E-CECE87505F1C Wellington Importadora
3A77CBF1-CCC5-4FF4-8E3D-40E4AA258F92 Cactus Comidas para Ilevar
D14CB8BE-E92A-47AE-BF39-211374DFCA81 Ernst Handel
0B14C0BD-C80A-4B49-AAE0-7F47B674B843 Maison Dewey
```

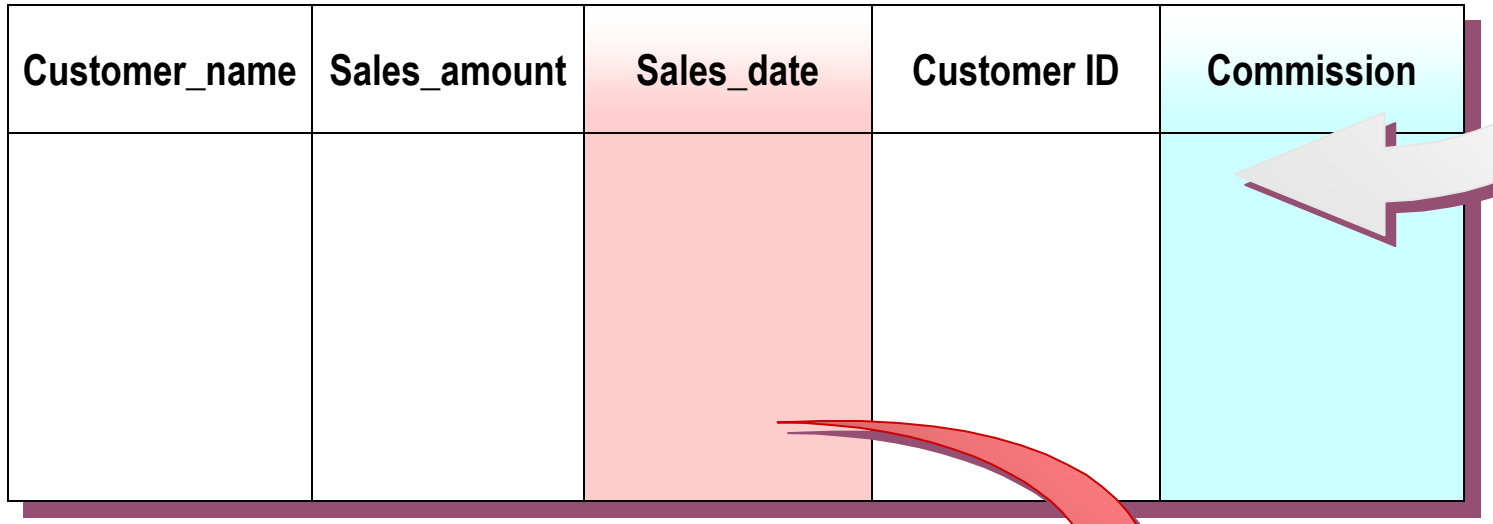


UNPAZ

Agregar y quitar columnas

AGREGAR

```
ALTER TABLE CategoriesNew
    ADD Commission money null
```

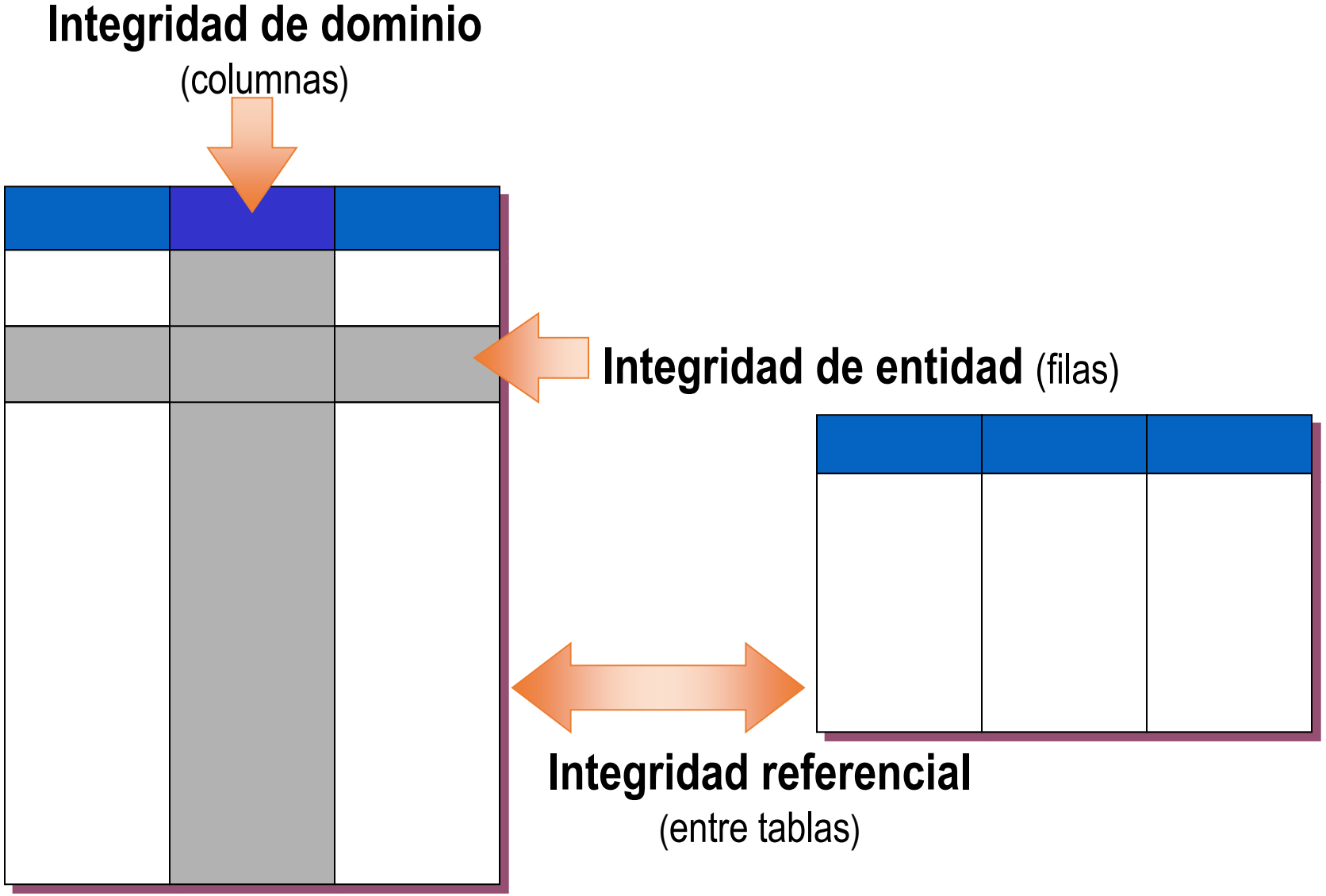


QUITAR

```
ALTER TABLE CategoriesNew
    DROP COLUMN Sales_date
```

UNPAZ

Tipos de integridad de datos



Constraints / Restricciones

- Ayudan a mantener la integridad de los datos. Son reglas que los datos deben cumplir. Se definen en el modelo de datos y el sistema de gestión de base de datos se encarga de que se cumplan.
- Tipos de datos. Si un atributo está definido como fecha, sólo aceptará como valores fechas que sean válidas.
- Si el campo puede tomar valores indefinidos (NULL).
- Clave primaria, que es un identificador unívoco de cada registro (tupla), y por lo tanto no admiten valores indefinidos (NULL).
- Las claves candidatas son otro tipo de restricción que también colaboran con la integridad de datos.
- Las claves foráneas permiten cumplir con la regla de integridad referencial. Una clave foránea es un atributo, o conjunto de atributos, que referencian a una clave primaria (o candidata) de otra tabla (o podría ser de la misma tabla).

Creación de restricciones

Utilizar CREATE TABLE o ALTER TABLE

Puede agregar restricciones a una tabla con datos existentes

Puede aplicar restricciones a una sola columna o a varias columnas

- *Una sola columna, se llama restricción de columna*
- *Varias columnas, se llama restricción de tabla*

UNPAZ

Consideraciones para el uso de restricciones

- Pueden cambiarse sin volver a crear una tabla
- Requieren comprobación de errores en aplicaciones y transacciones
- Comprueban los datos existentes

Tipos de restricciones

- Restricciones DEFAULT
- Restricciones CHECK
- Restricciones PRIMARY KEY
- Restricciones UNIQUE
- Restricciones FOREIGN KEY
  - Integridad referencial

UNIQUE Vs PRIMARY KEY

- La diferencia entre ambos es que la clave primaria no acepta valores null, mientras que una clave única sí las acepta.
- Y la otra diferencia es que una tabla solo puede definir una sola clave primaria, mientras que podemos definir varias claves únicas.

Restricciones DEFAULT

- Sólo se aplica a las instrucciones INSERT
- Sólo una restricción DEFAULT por columna
- No se puede utilizar con la propiedad IDENTITY o el tipo de datos rowversion
- Permite que se especifiquen algunos valores proporcionados por el sistema (en SQL Server)

```
USE Northwind
ALTER TABLE dbo.Customers
ADD
CONSTRAINT DF_contactname DEFAULT 'UNKNOWN'
FOR ContactName
```

UNPAZ

Restricciones CHECK

- Se utilizan con las instrucciones INSERT y UPDATE
- Pueden hacer referencia a otras columnas en la misma tabla
- No pueden contener subconsultas
- Puede haber varias por campo

```
USE Northwind
ALTER TABLE dbo.Employees
ADD
CONSTRAINT CK_birthdate
CHECK (BirthDate > '01-01-1900' AND BirthDate < getdate())
```

UNPAZ

Restricciones PRIMARY KEY

- Sólo una restricción PRIMARY KEY por tabla
- Los valores deben ser exclusivos
- No se permiten valores nulos
- Crea un índice exclusivo en las columnas especificadas

```
USE Northwind
ALTER TABLE dbo.Customers
ADD
CONSTRAINT PK_Customers
  PRIMARY KEY (CustomerID)
```

Restricciones UNIQUE

- Permite **un** valor nulo
- Permite varias restricciones UNIQUE en una tabla
- Definidas con una o más columnas
- Exigida con un índice único

```
USE Northwind
ALTER TABLE dbo.Suppliers
ADD
CONSTRAINT U_CompanyName
  UNIQUE (CompanyName)
```

Restricciones FOREIGN KEY

- Deben hacer referencia a una restricción PRIMARY KEY o UNIQUE
- Proporcionan integridad referencial de una o de varias columnas
- No crean índices automáticamente
- Los usuarios deben tener permisos SELECT o REFERENCES en las tablas a las que se hace referencia
- Usa sólo la cláusula REFERENCES en la tabla de ejemplo

```
USE Northwind
ALTER TABLE dbo.Orders
ADD CONSTRAINT FK_Orders_Customers
  FOREIGN KEY (CustomerID)
  REFERENCES dbo.Customers(CustomerID)
```

UNPAZ

Integridad Referencial

- Restringir (NO ACTION)
- Cascada
- Default
- NULL

FOREIGN KEY

SQL Server

```
FOREIGN KEY ( column [ ,...n
] )
REFERENCES
referenced_table_name [ (
ref_column [ ,...n ] ) ]
[ ON DELETE { NO ACTION |
CASCADE | SET NULL | SET
DEFAULT } ]
[ ON UPDATE { NO ACTION |
CASCADE | SET NULL | SET
DEFAULT } ]
```

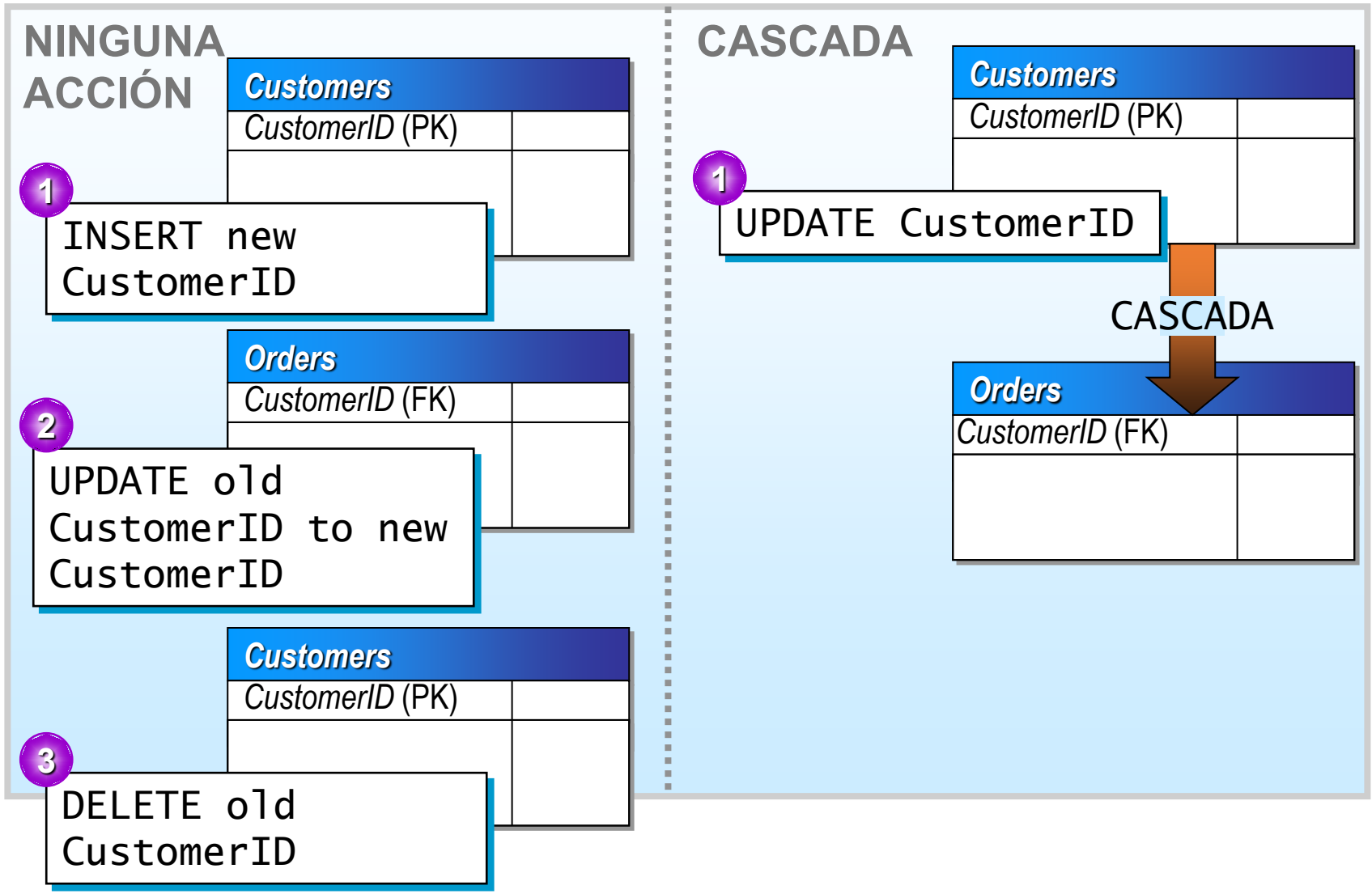
MySQL

```
FOREIGN KEY [index_name]
(col_name, ...)
REFERENCES tbl_name
(col_name,...)
[ON DELETE reference_option]
[ON UPDATE reference_option]

reference_option:
RESTRICT | CASCADE | SET NULL
| NO ACTION | SET DEFAULT
```

UNPAZ

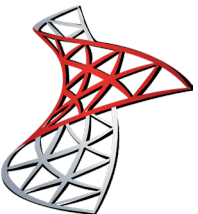
Integridad referencial en cascada



Deshabilitación de la comprobación de las restricciones en los datos existentes

- Se aplica a las restricciones CHECK y FOREIGN KEY
- Utilice la opción WITH NOCHECK cuando agregue una restricción nueva
- Utilizar si los datos existentes no cambian
- Se pueden cambiar los datos existentes antes de agregar restricciones

```
USE Northwind
ALTER TABLE dbo.Employees
WITH NOCHECK
  ADD CONSTRAINT FK_Employees_Employees
  FOREIGN KEY (ReportsTo)
  REFERENCES dbo.Employees(EmployeeID)
```

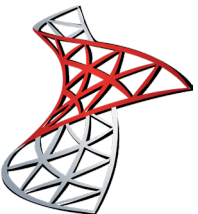


UNPAZ

Deshabilitación de la comprobación de las restricciones al cargar datos nuevos

- Se aplica a las restricciones CHECK y FOREIGN KEY
- Utilizar si:
  - Los datos cumplen las restricciones
  - Carga datos nuevos que no cumplen las restricciones

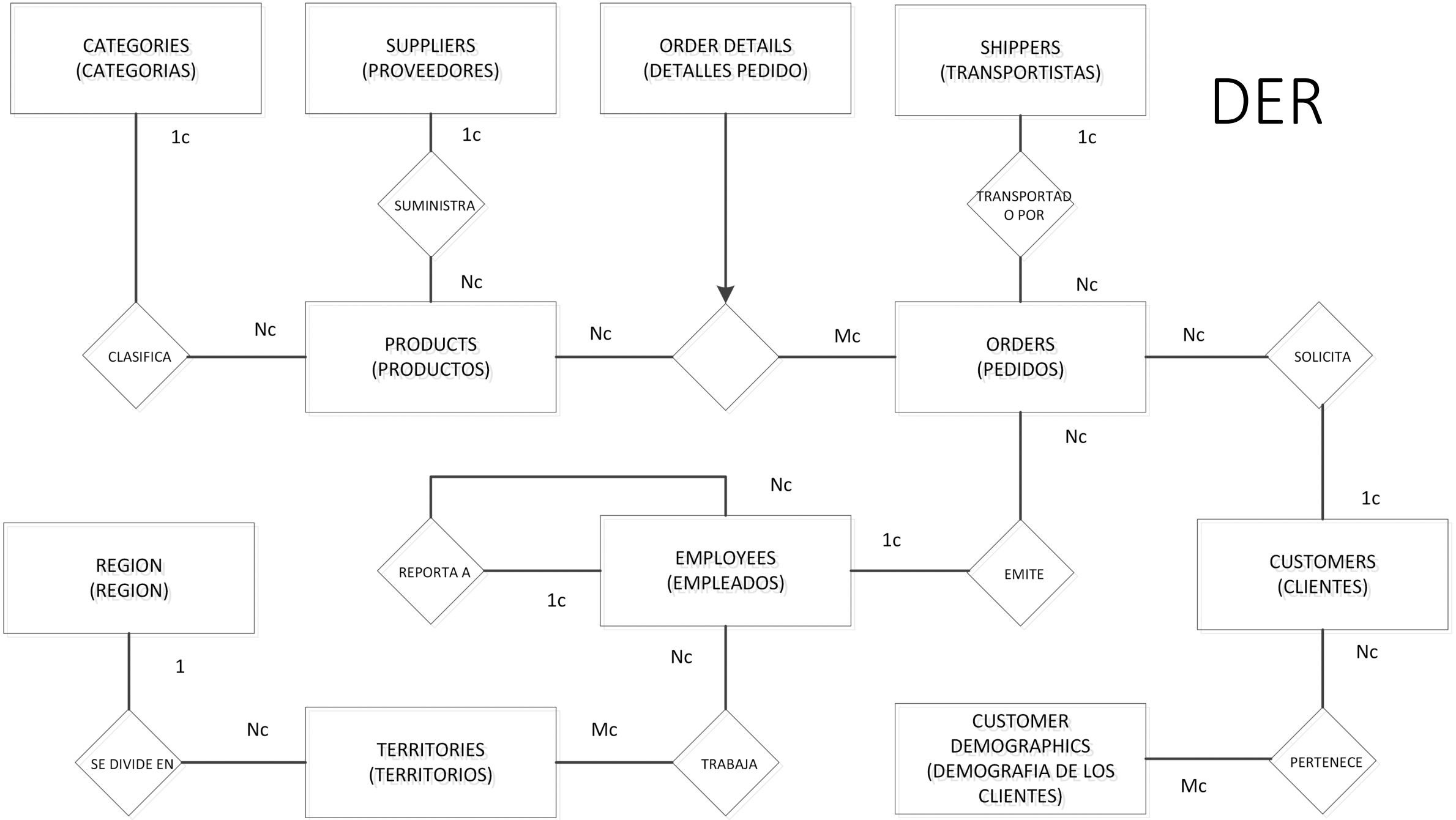
```
USE Northwind
ALTER TABLE dbo.Employees
NOCHECK
  CONSTRAINT FK_Employees_Employees
```



UNPAZ

Northwind

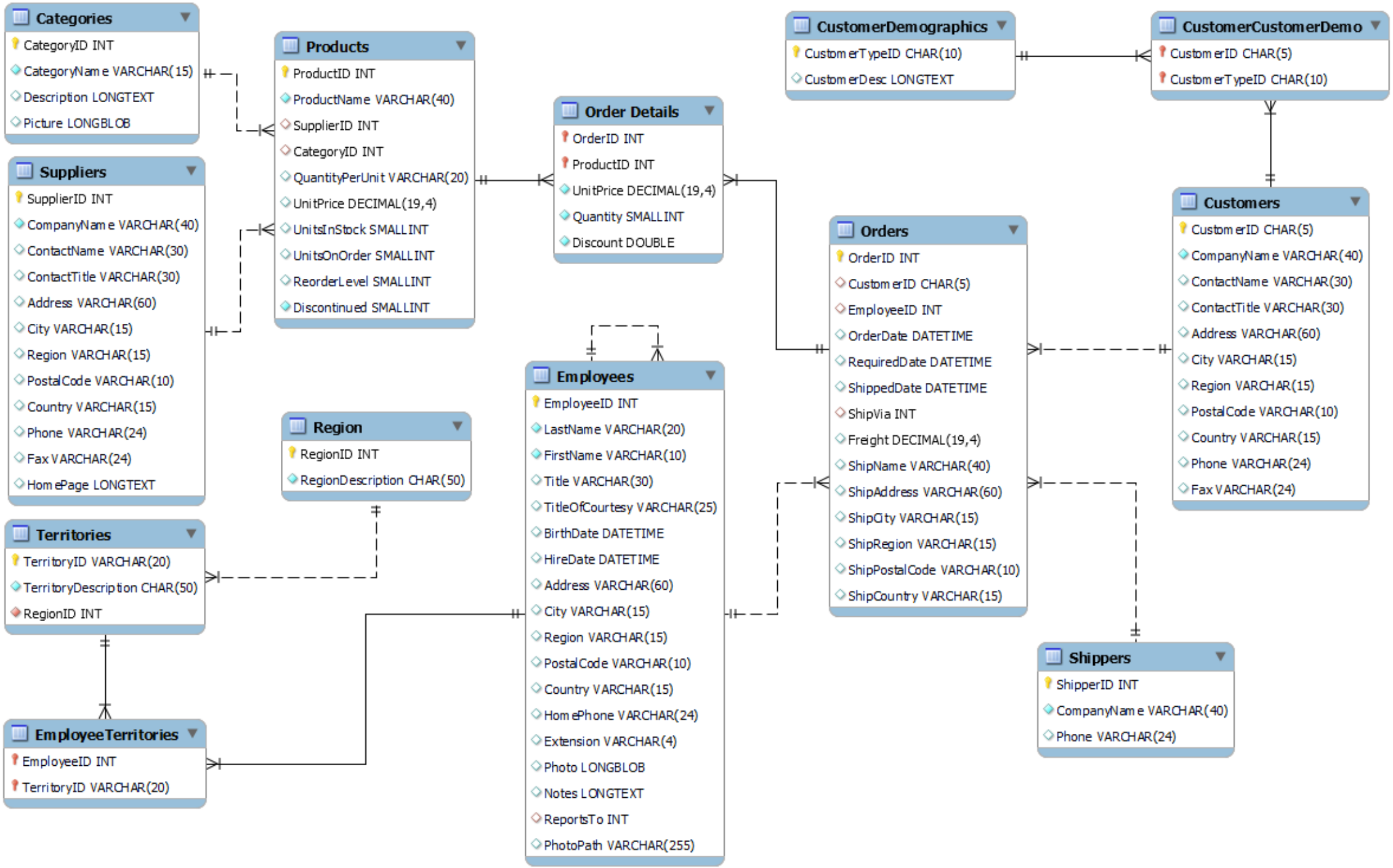
DER


CATEGORIES
(CATEGORIAS)
SUPPLIERS
(PROVEEDORES)
ORDER DETAILS
(DETALLES PEDIDO)
SHIPPERS
(TRANSPORTISTAS)
1c
1c
1c
SUMINISTRA
TRANSPORTADO POR
Nc
Nc
CLASIFICA
Nc
PRODUCTS
(PRODUCTOS)
Nc
Mc
ORDERS
(PEDIDOS)
Nc
SOLICITA
Nc
Nc
1c
REGION
(REGION)
REPORTA A
EMPLOYEES
(EMPLEADOS)
1c
EMITE
CUSTOMERS
(CLIENTES)
1c
Nc
Nc
1
SE DIVIDE EN
Nc
TERRITORIES
(TERRITORIOS)
Mc
TRABAJA
CUSTOMER DEMOGRAPHICS
(DEMOGRAFIA DE LOS CLIENTES)
Mc
PERTENECE

Diccionario de Datos

CATEGORIES	=	@CategoryID + CategoryName + Description + Picture
CUSTOMERDEMOGRAPHICS	=	@CustomerTypeID + CustomerDesc
CUSTOMERS	=	@CustomerID + CompanyName + ContactName + ContactTitle + Address + City + Region + PostalCode + Country + Phone + Fax
EMPLOYEES	=	@EmployeeID + LastName + FirstName + Title + TitleOfCourtesy + BirthDate + HireDate + Address + City + Region + PostalCode + Country + HomePhone + Extension + Photo + Notes + Photopath
ORDER DETAILS	=	UnitPrice + Quantity + Discount
ORDERS	=	@OrderID + OrderDate + RequiredDate + ShippedDate + Freight + ShipName + ShipAddress + ShipCity + ShipRegion + ShipPostalCode + ShipCountry
PRODUCTS	=	@ProductID + ProductName + QuantityPerUnit + UnitPrice + UnitsInStock + UnitsOnOrder + ReorderLevel + Discontinued
REGION	=	@RegionID + RegionDescription
SHIPPERS	=	@ShipperID + CompanyName + Phone
SUPPLIERS	=	@SupplierID + CompanyName + ContactName + ContactTitle + Address + City + Region + PostalCode + Country + Phone + Fax + HomePage
TERRITORIES	=	@TerritoryID + TerritoryDescription

UNPAZ


Categories
CategoryID INT
CategoryName VARCHAR(15)
Description LONGTEXT
Picture LONGBLOB
Products
ProductID INT
ProductName VARCHAR(40)
SupplierID INT
CategoryID INT
QuantityPerUnit VARCHAR(20)
UnitPrice DECIMAL(19,4)
UnitsInStock SMALLINT
UnitsOnOrder SMALLINT
ReorderLevel SMALLINT
Discontinued SMALLINT
Order Details
OrderID INT
ProductID INT
UnitPrice DECIMAL(19,4)
Quantity SMALLINT
Discount DOUBLE
CustomerDemographics
CustomerTypeID CHAR(10)
CustomerDesc LONGTEXT
CustomerCustomerDemo
CustomerID CHAR(5)
CustomerTypeID CHAR(10)
Suppliers
SupplierID INT
CompanyName VARCHAR(40)
ContactName VARCHAR(30)
ContactTitle VARCHAR(30)
Address VARCHAR(60)
City VARCHAR(15)
Region VARCHAR(15)
PostalCode VARCHAR(10)
Country VARCHAR(15)
Phone VARCHAR(24)
Fax VARCHAR(24)
HomePage LONGTEXT
Orders
OrderID INT
CustomerID CHAR(5)
EmployeeID INT
OrderDate DATETIME
RequiredDate DATETIME
ShippedDate DATETIME
ShipVia INT
Freight DECIMAL(19,4)
ShipName VARCHAR(40)
ShipAddress VARCHAR(60)
ShipCity VARCHAR(15)
ShipRegion VARCHAR(15)
ShipPostalCode VARCHAR(10)
ShipCountry VARCHAR(15)
Customers
CustomerID CHAR(5)
CompanyName VARCHAR(40)
ContactName VARCHAR(30)
ContactTitle VARCHAR(30)
Address VARCHAR(60)
City VARCHAR(15)
Region VARCHAR(15)
PostalCode VARCHAR(10)
Country VARCHAR(15)
Phone VARCHAR(24)
Fax VARCHAR(24)
Region
RegionID INT
RegionDescription CHAR(50)
Employees
EmployeeID INT
LastName VARCHAR(20)
FirstName VARCHAR(10)
Title VARCHAR(30)
TitleOfCourtesy VARCHAR(25)
BirthDate DATETIME
HireDate DATETIME
Address VARCHAR(60)
City VARCHAR(15)
Region VARCHAR(15)
PostalCode VARCHAR(10)
Country VARCHAR(15)
HomePhone VARCHAR(24)
Extension VARCHAR(4)
Photo LONGBLOB
Notes LONGTEXT
ReportsTo INT
PhotoPath VARCHAR(255)
Territories
TerritoryID VARCHAR(20)
TerritoryDescription CHAR(50)
RegionID INT
Shippers
ShipperID INT
CompanyName VARCHAR(40)
Phone VARCHAR(24)
EmployeeTerritories
EmployeeID INT
TerritoryID VARCHAR(20)

UNPAZ

DDL – CREATE TABLE

- La sintaxis básica es:

```
CREATE TABLE < tabla > (
atributo1 dominio1 [NOT NULL],
... ,
atributoN dominioN [NOT NULL]);
```

UNPAZ

DDL – DROP TABLE

- La sintaxis básica es:

```
DROP TABLE [IF EXISTS] < tabla >
```

DDL – ALTER TABLE

- La sintaxis básica para agregar un campo es:

```
ALTER TABLE < tabla > ALTER COLUMN (atributo1 dominio1
[NOT NULL]...)
```

```
ALTER TABLE < tabla > DROP COLUMN (atributo1)
```

```
ALTER TABLE < tabla > ADD (atributo1 dominio1 [NOT
NULL]...)
```

```
ALTER TABLE < tabla > ADD/DROP (restricción ...)
```

UNPAZ

DDL – ALTER TABLE - Restricciones

```
[ CONSTRAINT nombre_restriccion ]
{
    [ NULL | NOT NULL ]
    { PRIMARY KEY | UNIQUE }
| [ FOREIGN KEY (columna/s)]
        REFERENCES [ esquema. ] tablareferenciada
            [ ( columnareferenciada/s ) ]
        [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
        [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
    | CHECK ( expresión_logica )
}
```

UNPAZ

DDL – ALTER TABLE - Restricciones

```
CREATE TABLE Person.ContactBackup
    (ContactID int) ;
GO

ALTER TABLE Person.ContactBackup
ADD CONSTRAINT FK_ContactBackup_Contact FOREIGN KEY (ContactID)
    REFERENCES Person.Person (BusinessEntityID) ;
GO

ALTER TABLE Person.ContactBackup
DROP CONSTRAINT FK_ContactBackup_Contact ;
GO

DROP TABLE Person.ContactBackup ;
```

UNPAZ

NULL y la lógica de tres valores

OR	Verdadero	Falso	Desconocido
Verdadero	Verdadero	Verdadero	Verdadero
Falso	Verdadero	Falso	Desconocido
Desconocido	Verdadero	Desconocido	Desconocido

AND	Verdadero	Falso	Desconocido
Verdadero	Verdadero	Falso	Desconocido
Falso	Falso	Falso	Desconocido
Desconocido	Desconocido	Desconocido	Desconocido

NOT	
Verdadero	Falso
Falso	Verdadero
Desconocido	Desconocido

UNPAZ

Comentarios

	--	#	/* */
Ejemplo	-- Comentario	# Comentario	/* Comentario */
ANSI	Sí	No	No
SQL Server	Sí	No	Sí
Oracle	Sí	No	Sí
MySQL	Sí	Sí	Sí

UNPAZ

ESPACIOS EN BLANCO Y EL PUNTO Y COMA

Los espacios en blanco y los saltos de línea son ignorados por los intérpretes SQL, así que se pueden utilizar libremente para darle mayor claridad y legibilidad a las consultas.

El indicador de finalización de una consulta es el punto y coma (;). En algunas herramientas puede omitirse (como en el SQL Managment Studio), pero en otras es fundamental (como en SQLCMD).

UNPAZ

SENSIBILIDAD A MAYÚSCULAS Y MINÚSCULAS

- SQL no es case-sensitive, esto quiere decir que se pueden escribir las consultas indistintamente en mayúsculas o minúsculas. Sin embargo, es una práctica habitual utilizar los comandos y palabras reservadas en MAYÚSCULAS.
- Los nombres definidos por el usuario, como nombres de tablas y nombres de campos, pueden ser case-sensitive o no, dependiendo del sistema operativo utilizado y la configuración del sistema de gestión de base de datos.


Recuperación de Datos

DML

SELECCIONANDO TODAS LAS COLUMNAS Y TODAS LAS FILAS

SINTAXIS

```
SELECT tabla.*
FROM tabla;
   -- O
SELECT *
FROM tabla;
```

- **Ejemplo**

```
/* Obtener todas las columnas de la tabla Region*/
SELECT *
FROM Region;

SELECT *
FROM dbo.Region;

SELECT Region.*
FROM Region;
```

```
RegionID    RegionDescription
----------- --------------------------------------------------
1           Eastern
2           Western
3           Northern
4           Southern
```

UNPAZ

DML

SELECCIONANDO COLUMNAS ESPECÍFICAS

SINTAXIS

```
SELECT tabla.nombre_columna1,
tabla.nombre_columna2
FROM tabla;
   -- O
SELECT
nombre_columna1, nombre_columna2
FROM tabla;
```

- EJEMPLOS

```
/* Selecciona nombre (FirstName) y
apellido (LastName) de la tabla
Empleados (Employees) */

SELECT FirstName, LastName
FROM Employees;
```

```
FirstName  LastName
---------- --------------------
Nancy      Davolio
Andrew     Fuller
Janet      Leverling
Margaret   Peacock
Steven     Buchanan
Michael    Suyama
Robert     King
Laura      Callahan
Anne       Dodsworth
```

UNPAZ

DML

Ordenando registros

Se utiliza la cláusula ORDER BY de la sentencia SELECT.

Ordenando por una columna

Sintaxis

```
SELECT columna1, columna2
FROM tabla
ORDER BY columna1;
```

- Ejemplo

```
/* Selecciona nombre (FirstName) y apellido
(LastName) de la tabla Empleados
(Employees). Ordena por apellido. */

SELECT FirstName, LastName
FROM Employees
ORDER BY LastName;
```

```
FirstName  LastName
---------- --------------------
Steven     Buchanan
Laura      Callahan
Nancy      Davolio
Anne       Dodsworth
Andrew     Fuller
Robert     King
Janet      Leverling
Margaret   Peacock
Michael    Suyama
```

UNPAZ

DML

Ordenar por varias columnas

Sintaxis

```
SELECT columna1, columna2
FROM tabla
ORDER BY columna1, columna2;
```

- Ejemplo

```
/* Selecciona puesto (Title), nombre (FirstName) y apellido
(LastName) de la tabla Empleados (Employees).
Ordena por puesto y luego por apellido. */

SELECT Title, FirstName, LastName
FROM Employees
ORDER BY Title, LastName;
```

```
Title                         FirstName  LastName
----------------------------- ---------- --------------------
Inside Sales Coordinator      Laura      Callahan
Sales Manager                 Steven     Buchanan
Sales Representative          Nancy      Davolio
Sales Representative          Anne       Dodsworth
Sales Representative          Robert     King
Sales Representative          Janet      Leverling
Sales Representative          Margaret   Peacock
Sales Representative          Michael    Suyama
Vice President, Sales         Andrew     Fuller
```

UNPAZ

DML

ORDENAR POR POSICIÓN DE LA COLUMNA

SINTAXIS

SELECT columna1, columna2
FROM tabla
ORDER BY posicion_columna1, posicion_columna2;

- EJEMPLO

```
/* Selecciona puesto (Title), nombre (FirstName) y apellido (LastName) de la tabla Empleados (Employees).
Ordena por puesto (posición 1) y luego por apellido (posición 3). */

SELECT Title, FirstName, LastName
FROM Employees
ORDER BY 1, 3;
```

Title	FirstName	LastName
Inside Sales Coordinator	Laura	Callahan
Sales Manager	Steven	Buchanan
Sales Representative	Nancy	Davolio
Sales Representative	Anne	Dodsworth
Sales Representative	Robert	King
Sales Representative	Janet	Leverling
Sales Representative	Margaret	Peacock
Sales Representative	Michael	Suyama
Vice President, Sales	Andrew	Fuller

UNPAZ

DML

Orden Ascendente y Descendente

Cuando se utiliza ORDER BY los registros son ordenados de manera Ascendente por defecto. Esto puede ser especificado explícitamente con la palabra clave ASC. Para ordenar en orden descendente se utiliza DESC

Sintaxis

```
SELECT columna1, columna2
FROM tabla
ORDER BY posicion_columna1 DESC,
posicion_columna2 ASC;
```

- Ejemplo

```
/* Selecciona puesto (Title),
nombre (FirstName) y apellido
(LastName) de la tabla Empleados
(Employees).
Ordena por puesto ascendente y
luego por apellido descendente.
*/

SELECT Title, FirstName, LastName
FROM Employees
ORDER BY Title ASC, LastName
DESC;
```

```
Title                        FirstName  LastName
---------------------------- ---------- --------------------
Inside Sales Coordinator     Laura      Callahan
Sales Manager                Steven     Buchanan
Sales Representative         Michael    Suyama
Sales Representative         Margaret   Peacock
Sales Representative         Janet      Leverling
Sales Representative         Robert     King
Sales Representative         Anne       Dodsworth
Sales Representative         Nancy      Davolio
Vice President, Sales        Andrew     Fuller
```

UNPAZ

Recuperación de datos

- Recuperar datos de tablas mediante la instrucción SELECT
- Filtrar los datos
- Dar formato al conjunto de resultados
- Como se procesan las consultas
- Como se ordenan

UNPAZ

La cláusula WHERE y los símbolos operadores

- La cláusula WHERE se utiliza para obtener filas específicas de la tabla. Puede contener una o más condiciones que especifican cuáles son las filas que deben ser devueltas.
- Sintaxis

```
SELECT columna1, columna2
FROM tabla
WHERE condiciones;
```

Uso de los operadores de comparación

```
USE northwind
SELECT lastname, city
FROM employees
WHERE country = 'USA'
GO
```

Ejemplo 1

Operador	Descripción
=	Igual a
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
<>	No igual a

lastname	*city*
Davolio	Seattle
Fuller	Tacoma
Leverling	Kirkland
Peacock	Redmond
Callahan	Seattle

OPERADORES

=	Igual a
<>	No Igual a
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

UNPAZ

Verificando por igualdad - Ejemplo

```
/* Selecciona puesto (Title), nombre (FirstName) y apellido
(LastName) de la tabla Empleados (Employees) de todos los
representantes de ventas (Sales Representative). */

SELECT Title, FirstName, LastName
FROM Employees
WHERE Title = 'Sales Representative';
```

```
Title                         FirstName  LastName
----------------------------- ---------- --------------------
Sales Representative          Nancy      Davolio
Sales Representative          Janet      Leverling
Sales Representative          Margaret   Peacock
Sales Representative          Michael    Suyama
Sales Representative          Robert     King
Sales Representative          Anne       Dodsworth
```

UNPAZ

Verificando por no igualdad – Ejemplo

```
/* Selecciona nombre (FirstName) y apellido
(LastName) de la tabla Empleados (Employees)
excluyendo a los representantes de ventas (Sales
Representative). */

SELECT FirstName, LastName

FROM Employees

WHERE Title <> 'Sales Representative';
```

```
FirstName  LastName
---------- --------------------
Andrew     Fuller
Steven     Buchanan
Laura      Callahan
```

UNPAZ

Verificando por mayor o menor que

- Estos operadores pueden utilizarse para comparar número, fechas y cadenas de texto

```
/* Seleccionar nombre (FirstName) y apellido
(LastName) de los empleados (Employees) cuyo apellido
comience con N en adelante */

SELECT FirstName, LastName

FROM Employees

WHERE LastName >= 'N';
```

```
FirstName  LastName

---------- --------------------

Margaret   Peacock

Michael    Suyama
```

```sql
SELECT CustomerID, CompanyName, Region
FROM Customers;

SELECT CustomerID, CompanyName, Region
FROM Customers
WHERE Region = 'WA’;

SELECT CustomerID, CompanyName, Region
FROM Customers
WHERE Region <> 'WA’;
```

- ¿Y los 60 que faltan?

UNPAZ

Verificando por NULL

Se dice que un valor el nulo (NULL) cuando el campo no tiene un valor asignado. No es lo mismo que una cadena vacía. Lo que significa es que el campo no tiene ningún valor asignado.

Para verificar si una columna tiene valor nulo no se utiliza el operador “=”, en su lugar se utiliza la expresión IS NULL.

UNPAZ

Verificando por NULL

```
/* Seleccionar nombre (FirstName) y apellido
(LastName) de todos los empleados (Employees) que no
tengan especificada una región (Region). */

SELECT FirstName, LastName
FROM Employees
WHERE Region IS NULL;
```

```
FirstName  LastName
---------- --------------------
Steven     Buchanan
Michael    Suyama
Robert     King
Anne       Dodsworth
```

UNPAZ

Verificando por NOT NULL

- /* Seleccionar nombre (FirstName) y apellido (LastName) de todos los empleados (Employees) que tengan especificada una región (Region). */

```
SELECT FirstName, LastName
FROM Employees
WHERE Region IS NOT NULL;
```

```
FirstName  LastName
---------- --------------------
Nancy      Davolio
Andrew     Fuller
Janet      Leverling
Margaret   Peacock
Laura      Callahan
```

UNPAZ