

Herramientas Computacionales

Teoría de bases de datos y SQL

Enero de 2023

Manuel Rueda Álvaro (mrueda@afi.es)



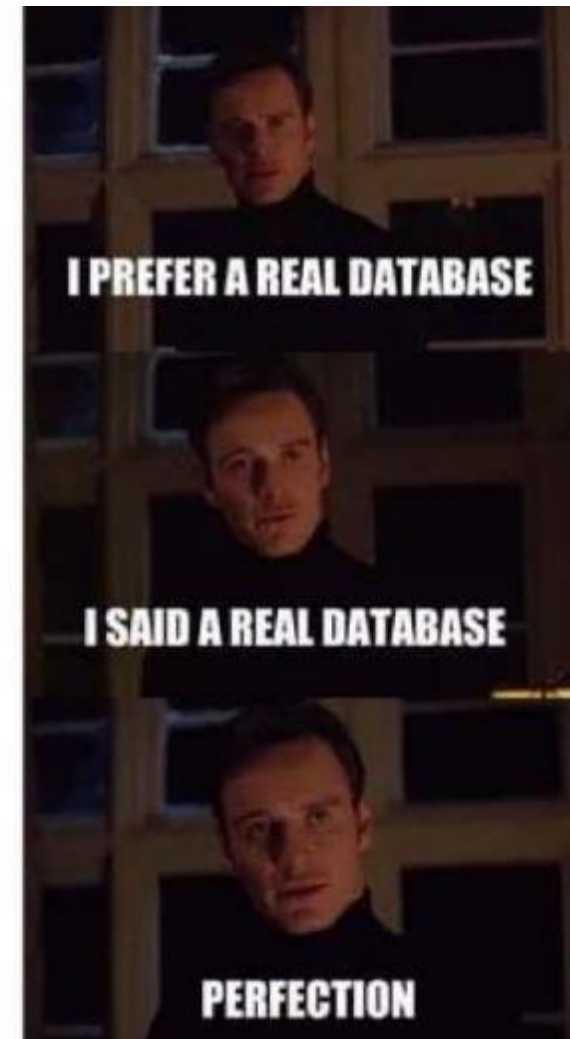
Índice

1. Introducción a Bases de Datos
2. SQL Basics
3. Creación de Bases de Datos
4. Instalación de Software
5. Introducción al lenguaje SQL (DDL)
6. Carga de datos desde fichero
7. Introducción al lenguaje SQL (MDL)
8. Otras acciones
9. Materiales

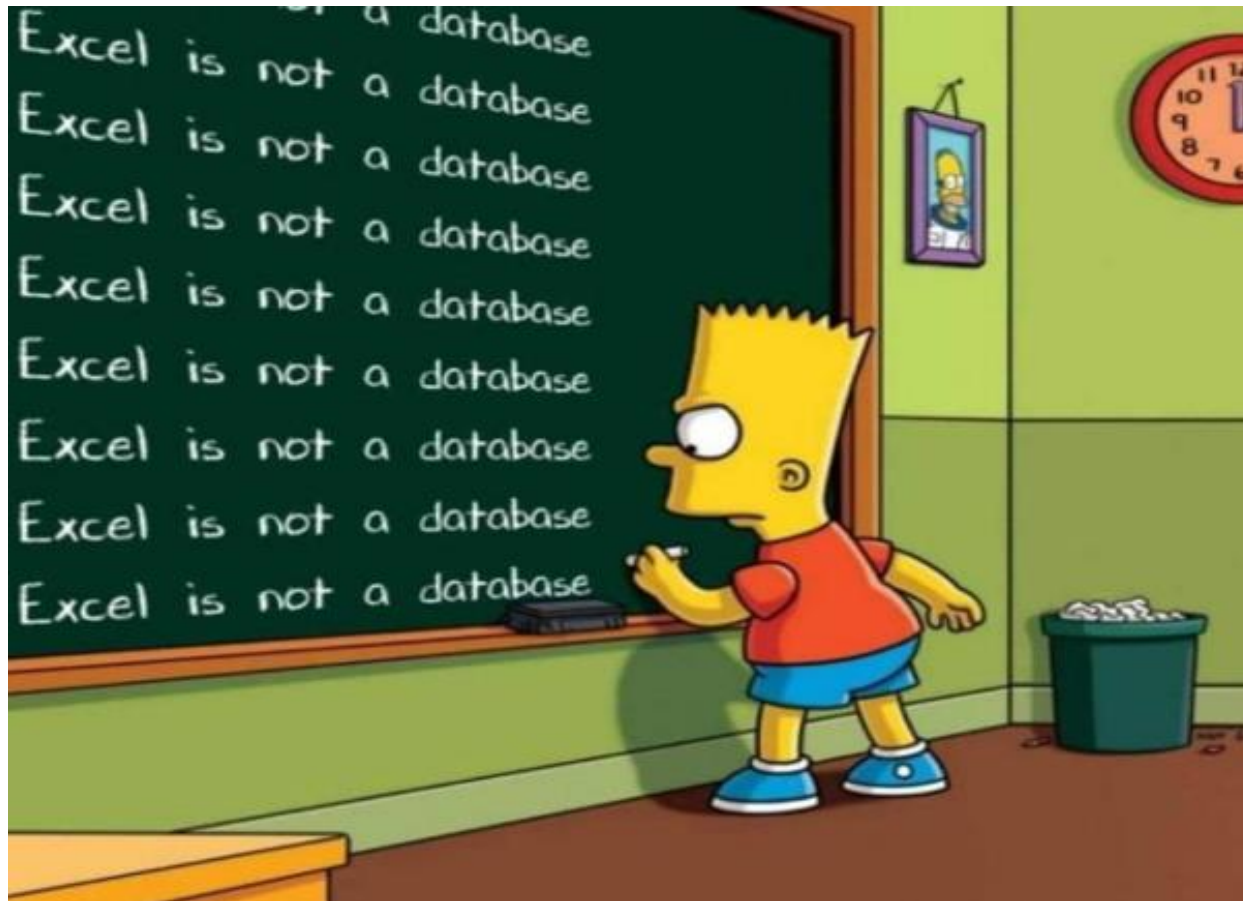


1. Introducción a Bases de Datos

¿Por qué aprender de BBDD si tenemos Excel ?



Aunque lo parezca, no es una Base de Datos



¿Por qué Excel no es una BBDD ?

NEWS

[Home](#) | [Coronavirus](#) | [Climate](#) | [Video](#) | [World](#) | [UK](#) | [Business](#) | [Tech](#) | [Science](#) | [Stories](#) | [Entertainment & Arts](#)

[Tech](#)

Excel: Why using Microsoft's tool caused Covid-19 results to be lost

The badly thought-out use of Microsoft's Excel software was the reason nearly 16,000 coronavirus cases went unreported in England.

And it appears that Public Health England (PHE) was to blame, rather than a third-party contractor.

The issue was caused by the way the agency brought together logs produced by commercial firms paid to analyse swab tests of the public, to discover who has the virus.

They filed their results in the form of text-based lists - known as CSV files - without issue.

PHE had set up an automatic process to pull this data together into Excel templates so that it could then be uploaded to a central system and made available to the NHS Test and Trace team, as well as other government computer dashboards.

Bases de datos y Gestores de Bases de datos

Base de datos: es un soporte que permite almacenar información:

- SQL Server
- Oracle
- Etc

Gestor de base de datos: programa que permite introducir, almacenar, ordenar y manipular información:

- Microsoft Access
- Microsoft SQL Server Management
- Toad for Oracle
- Etc

¿Qué es una base de datos ?

- Es, básicamente, un almacén de información.
- Normalmente, los datos de este almacén tienen algún tipo de relación o vínculo entre ellos.
- Existen múltiples tipos de bases de datos:
 - Relacionales.
 - Multidimensionales.
 - Clave-valor.
 - Documentales.
 - De grafos.
 - ...

Tipos de bases de datos

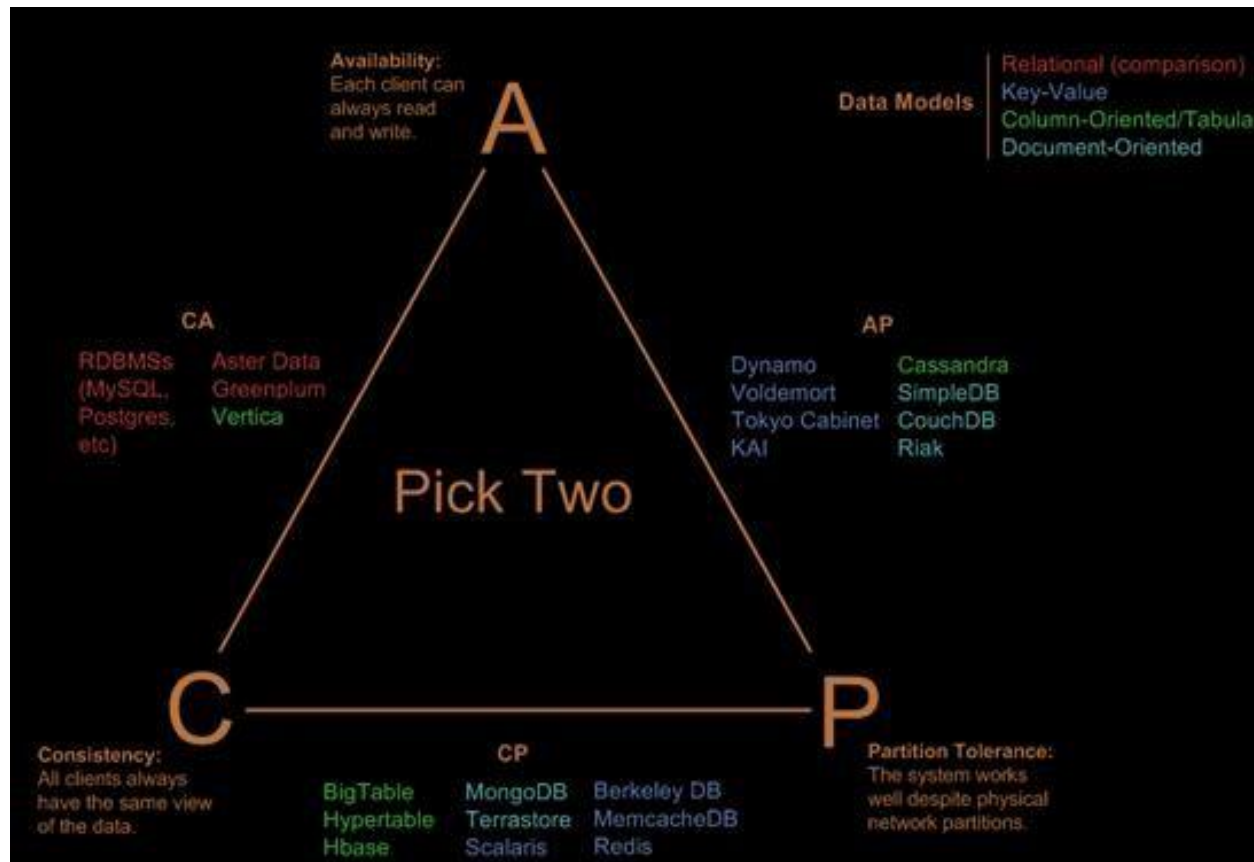
- Cada uno de los diferentes tipos establece unas “normas” en términos de:
 - ¿Qué estructura tiene el contenido?
 - Tabular, objetos, grafos...
 - ¿Cómo se representa el contenido?
 - Tablas, documentos, nodos...
 - ¿Cómo se representan las relaciones?
 - Claves (primarias/foráneas), por referencia o elementos embebidos, aristas (dirigidas/no dirigidas)...
- Cada uno de los diferentes tipos ofrece una serie de capacidades en términos de:
 - Consistencia de datos.
 - Disponibilidad (*availability*) de los datos.
 - Particionado de los datos.

SQL...¿Por qué?



Teorema CAP

- Teorema CAP (Consistency, Availability, Partition tolerance)



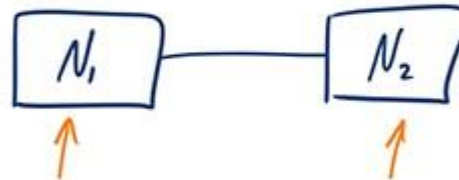
Teorema CAP

- Teorema CAP (Consistency, Availability, Partition tolerance)

Consistency



Availability

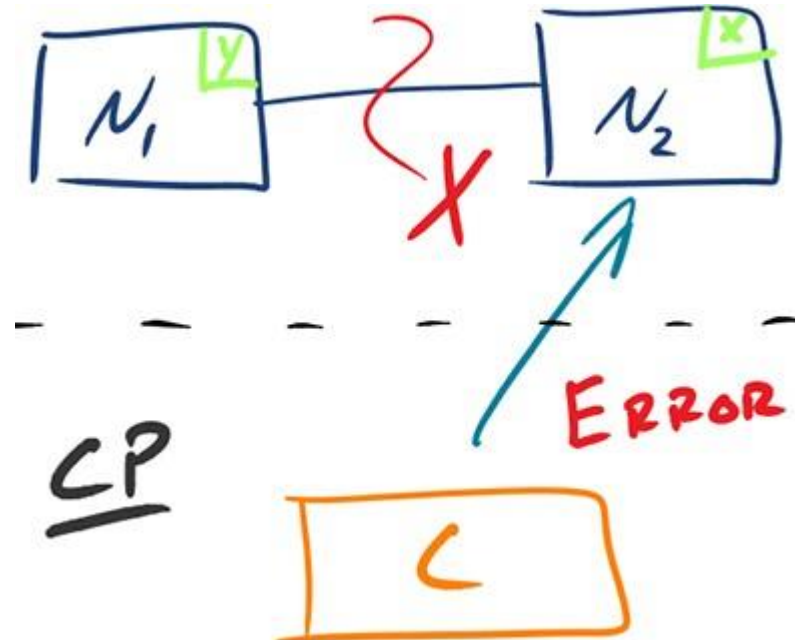


Partition Tolerance



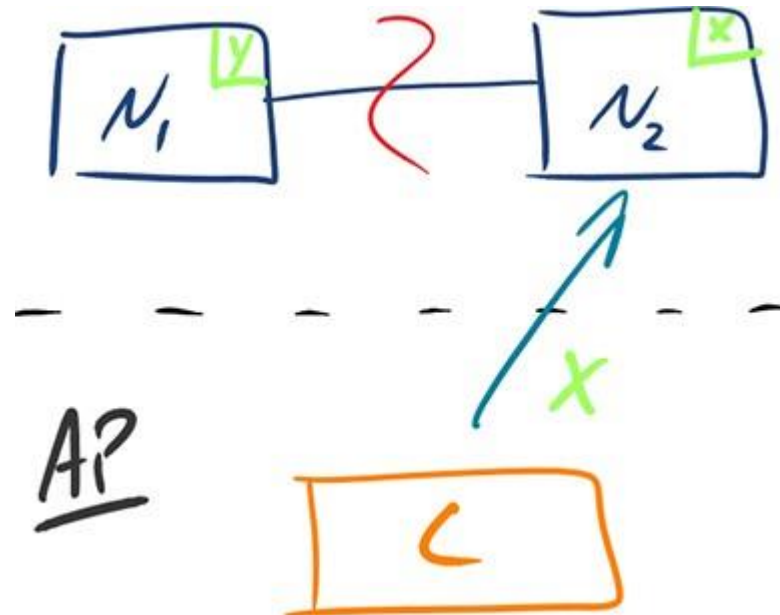
Teorema CAP

- Teorema CAP (Consistency, Availability, Partition tolerance)



Teorema CAP

- Teorema CAP (Consistency, Availability, Partition tolerance)



Modelo de una base de datos

El modelo de una base de datos determina la estructura de la misma. Es decir, la manera en que los datos serán almacenados. Algunos ejemplos:

- Modelo relacional (basado en tablas)
- Modelo dimensional (evolución del modelo relacional)
- Modelo orientado a objetos
- Modelo de grafos
- ...

Para poder acceder a la información de una base de datos emplearemos un lenguaje de consultas (**SQL** es el más extendido)

Tipos de Bases de Datos

Ficheros planos:

- La información se almacena en una sola tabla/fichero.
- Ej: Microsoft Excel, ficheros de texto plano.

Bases de datos relacionales:

- La información se clasifica según diferentes criterios y se va estructurando en diferentes tablas relacionadas.
- Cada una de las tablas creadas, almacena el resultado de la clasificación realizada de manera única.

Bases de datos NO relacionales:

- La información se guarda en colecciones y estas pueden tener información heterogénea.
- Suelen guardar la información en formato JSON.
- Ej: MongoDB, DocumentDB.

Ficheros/Tablas planas

Hospital						
Fecha	Nombre	Dirección	Tfno	Diagnóstico	Tratamiento	Médico
06/12/2095	Cabera Ortíz, Pedro	C/Mayor 12 4D	101232	Apendicitis	Cirugía	Dra. Sanz
05/05/2095	García García, Ana	Avda. Arroyos, 54	256699	Gripe	Frenadol	Dr. Pérez
12/01/2096	Santos Gremio, Luis	C/Berruguete, 74	369856	Sarampión	Augmentine	Dr. Pérez
12/01/2096	Cabera Ortíz, Pedro	C/Mayor 12 4D	101232	Sinusitis	Sinus	Dr. Alonso
23/05/2095	García García, Ana	Avda. Arroyos, 54	256699	Sarampión	Clamoxil	Dra. Sanz
06/12/2095	Cabera Ortíz, Pedro	C/Mayor 12 4D	101232	Sinusitis	Sinus	Dr. Pérez
01/01/2096	Santos Gremio, Luis	C/Berruguete, 74	369856	Amigdalitis	Clamoxil	Dr. Alonso
25/02/2095	Cabera Ortíz, Pedro	C/Mayor 12 4D	101232	Amigdalitis	Clamoxil	Dra. Sanz

Ficheros/Tablas planas

Problemas:

- El proceso de obtener la información es **lento** y requiere de la generación de programas.
 - Dependencia de los datos y criterios del programador.
- **Redundancia** en la información.
 - Ejemplo: el médico “Dra. Sanz” se ha introducido 3 veces, con el error operacional que conlleva.
- No existe control sobre la integridad de la información, y no es fácilmente manipulable.
 - Ejemplo: ¿cuántas visitas ha realizado el paciente “Cabrera Ortíz, Pedro”?

Bases de datos relacionales

La información se puede clasificar en:

- **Pacientes**

Pacientes			
Nombre	Dirección	Tfno.	Id Paciente
Cabera Ortíz, Pedro	C/Mayor 12 4D	101232	1
García García, Ana	Avda. Arroyos, 54	256699	2
Santos Gremio, Luis	C/Berruguete, 74	369856	3

- **Médicos**

Médicos			
Nombre	Especialidad	Tfno	Id Médico
Dra. Sanz	1
Dr. Pérez	2
Dr. Alonso	3

- **Visitas**

Visitas				
Id Paciente	Id Médico	Fecha	Diagnóstico	Tratamiento
1	1	06/12/2095	Apendicitis	Cirugía
2	2	05/05/2095	Gripe	Frenadol
3	2	12/01/2096	Sarampión	Augmentine
1	3	12/01/2096	Sinusitis	Sinus
2	1	23/05/2095	Sarampión	Clamoxil
1	2	06/12/2095	Sinusitis	Sinus
3	3	01/01/2096	Amigdalitis	Clamoxil
1	1	25/02/2095	Amigdalitis	Clamoxil

Bases de datos relacionales

Se establecen las relaciones:

Pacientes			
Nombre	Dirección	Tfno.	Id Paciente
Cabera Ortiz, Pedro	C/Mayor 12 4D	101232	1
García García, Ana	Avda. Arroyos, 54	256699	2
Santos Gremio, Luis	C/Berruguete, 74	369856	3

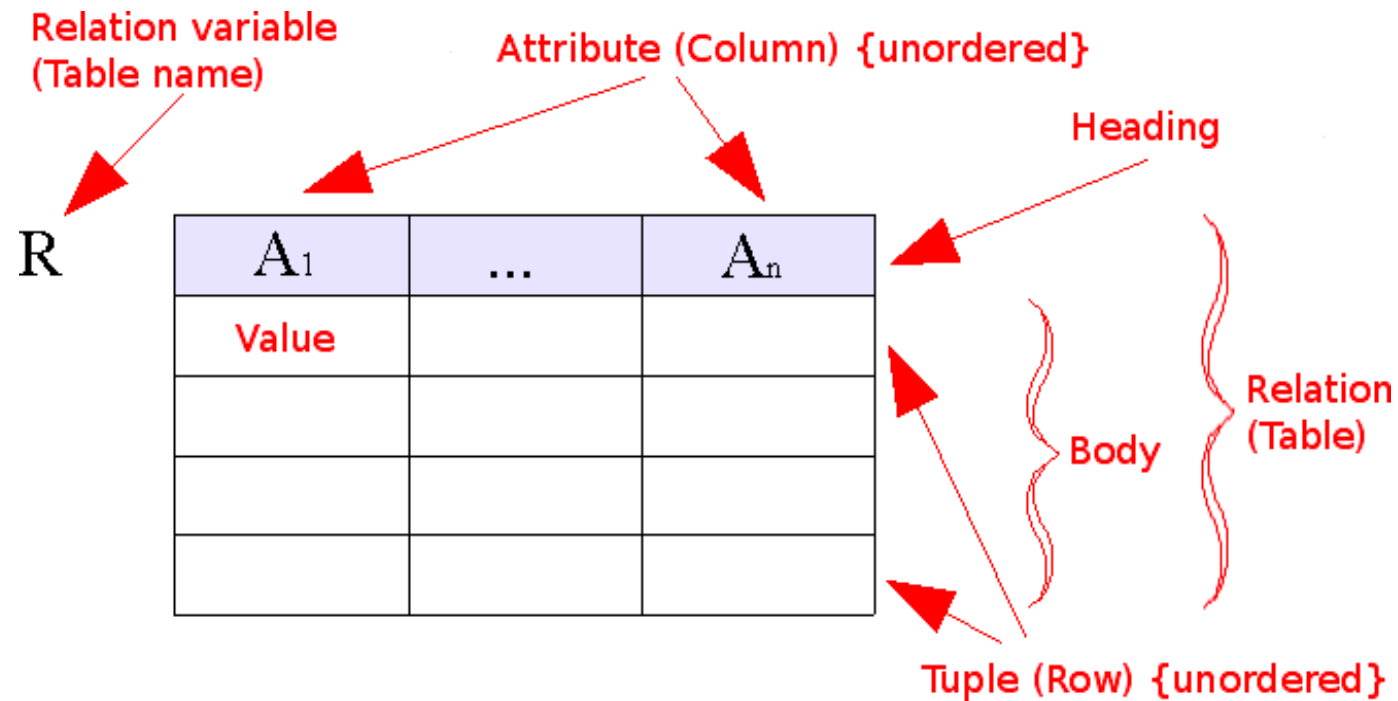
Médicos			
Nombre	Especialidad	Tfno	Id Médico
Dra. Sanz	1
Dr. Pérez	2
Dr. Alonso	3

Visitas				
Id Paciente	Id Médico	Fecha	Diagnóstico	Tratamiento
1	1	06/12/2095	Apendicitis	Cirugía
2	2	05/05/2095	Gripe	Frenadol
3	2	12/01/2096	Sarampión	Augmentine
1	3	12/01/2096	Sinusitis	Sinus
2	1	23/05/2095	Sarampión	Clamoxil
1	2	06/12/2095	Sinusitis	Sinus
3	3	01/01/2096	Amigdalitis	Clamoxil
1	1	25/02/2095	Amigdalitis	Clamoxil

¿Qué es una base de datos relacional ?

- Es un tipo de base de datos que cumple con el **modelo relacional** basado en la lógica de predicados o lógica de primer orden, que establece que los datos están:
 - Estructurados y regidos mediante un **esquema**.
 - Agrupados en **relaciones o tablas**.
 - Definidos mediante un conjunto de **campos o columnas**.
 - Representados como **registros o filas**.
 - Conectados entre sí mediante **claves primarias y/o claves foráneas**.
- Siguen las normas/reglas de los **sistemas transaccionales**.
- Se lleva a cabo su uso (acceso, definición y consulta) mediante el **lenguaje SQL**.
- Todas las tareas se realizan desde un **gestor de base de datos relacionales o RDBMS** (Relational Database Management System).

Modelo Relacional



Sistemas transaccionales

- Un sistema transaccional, o sistema orientado a transacciones, es aquel que trabaja con operaciones indivisibles, llamadas transacciones.
- Una transacción es una operación (o conjunto de operaciones) que se ejecutan garantizando el cumplimiento de las propiedades ACID:
 - **Atomicidad:** cada transacción se ejecuta completamente y de manera correcta o no tiene impacto en el sistema. Si una parte de la transacción falla en un momento de su ejecución, entonces dicha transacción no produce cambios.
 - **Consistencia:** cualquier transacción lleva el sistema de un estado válido a otro estado válido.
 - **Aislamiento (Isolation):** varias ejecuciones concurrentes de transacciones sobre el sistema se procesan una tras otra, evitando bloqueos.
 - **Durabilidad:** una vez finalizada la operación, esta persiste en el sistema aunque se produzcan errores o fallos.

Elementos de una base de datos relacional

Relaciones

Dominios

**Procedimientos
almacenados**

Restricciones

Claves

- Clave primaria
- Clave foránea
- Clave índice

Estructuras

- Esquema
- Datos

Elementos de una base de datos relacional

Relaciones

Las relaciones permiten almacenar y consultar los datos. Existen dos tipos:

- **Relación base o tabla:** Almacena datos.
- **Relación derivada o consulta o vista:** No almacena datos. Se calculan al aplicar operaciones relacionales. Expresan información de varias tablas como si fuera una única tabla.

Elementos de una base de datos relacional

Restricciones

Son limitaciones sobre los campos de las tablas de base de datos. No son fundamentales pero permiten mejorar la organización de los datos.

Dominios

Tipos de datos que serán almacenados.

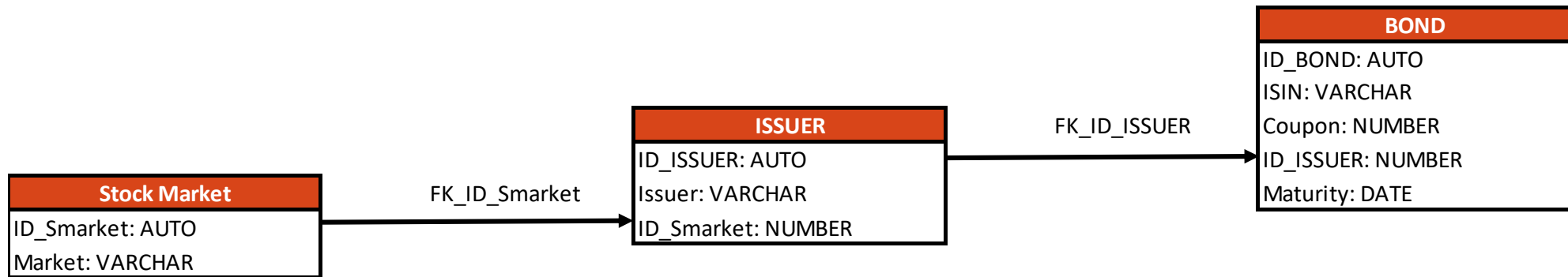
Elementos de una base de datos relacional

Claves

- **Clave primaria:** clave única que permite identificar un registro de la tabla.
- **Clave foránea:** referencia a una clave de otra tabla. No tienen porque ser claves únicas en la tabla donde están, pero sí donde están referenciadas. **Integridad de la información.**
- **Clave índice:** permiten un acceso más rápido a los datos. Se pueden crear como combinación de los campos de una tabla.

Elementos de una base de datos relacional

Claves



Elementos de una base de datos relacional

Procedimientos almacenados

Código ejecutable que se guarda en la base de datos. Cada motor de base de datos tiene su propio lenguaje de programación (por ejemplo: SQL Server tiene Transact-SQL, Oracle tiene PL/SQL...).

Estructuras

- **Esquema:** define la estructura de base de datos (tablas, columna, tipos de datos, claves primarias, relaciones, índices...)
- **Datos:** registros de información. Filas de una tabla (tuplas). El número de filas de una tabla es denominado cardinalidad. El número de columnas se denomina aridad o grado.

Bajando a la base de datos

- **Registro:** es el concepto básico en el almacenamiento de datos. Agrupa la información asociada a un elemento de un conjunto.
- **Campos:** cada una de las partes en las que se desglosa la información de los registros.
- **Tabla:** es el conjunto de registros homogéneos con la misma estructura.



Tipos de campos

- **Campos fundamentales:** aquellos que definen, de forma única, el registro.
- **Campos secundarios:** aquellos que complementan al campo fundamental y proporcionan información descriptiva del registro.

NOTA

En el ejemplo anterior, tendríamos:

- **Campos fundamentales:** Nombre.
- **Campos secundarios:** Dirección y Teléfono.

Bases de datos relacionales

Conceptos Básicos de BBDD

- **Tabla:** Una tabla de datos es el objeto que se define y utiliza para almacenar los datos. Contiene información sobre un tema o asunto particular.
- **Relación:** Las relaciones son los vínculos entre dos campos de distintas tablas que permiten asegurar la integridad de la información.
- **Consultas/Querys:** Una consulta es el objeto que proporciona una visión personal de los datos de las tablas ya creadas.

NOTA

Existen varios tipos de consultas, como veremos más adelante, para seleccionar, actualizar, borrar datos, ..., pero en principio se utilizan para extraer de las tablas los datos que cumplen ciertas condiciones.

Bases de datos relacionales

Claves Primarias

- Una buena gestión y clasificación de la información en bases de datos relacionales requiere de la definición de claves primarias en cada una de las tablas presentes.
- **Clave Primaria:** Es un campo que:
 - Nunca podrá ser nulo.
 - Identifica de manera única a cada registro.
- La clave primaria puede ser:
 - Un campo único, numérico o alfanumérico.
 - Un conjunto de campos.

Bases de datos relacionales

Claves Primarias

En nuestro ejemplo, las claves primarias son:

1. Tabla Pacientes:

- Clave primaria definida con un campo único.
- La clave primaria es el campo **Id Paciente**.

2. Tabla Médicos:

- Clave primaria definida con un campo único.
- La clave primaria es el campo **Id Medico**.

3. Tabla Médicos: Es un campo que:

- Clave primaria definida con campos múltiples.
- La clave primaria es la combinación de los campos **Id Paciente**, **Id Medico** y **Fecha**.

Bases de datos relacionales

Claves Foráneas

- Las claves extranjeras generan vínculos “fuertes” entre distintas tablas.
- Generalmente, asocia la clave primaria de una tabla (A) a un campo de otra tabla (B), de tal forma que restringe los valores posibles que podemos introducir en B.
- Esto nos permite generar cierta integridad en nuestra base de datos.
- Así pues, si A no tiene definido un valor concreto, no podrá ser asignado en B.
- Adicionalmente, tampoco podremos eliminar valores de A que se encuentren asignados en B, por lo que esto nos dará una idea de la prelación a la hora de eliminar elementos en la base de datos.

Bases de datos relacionales

Claves Foráneas

- Supongamos que tenemos una tabla con la información de todas las divisas que trabaja el banco: tabla "Divisas", y en la tabla "Operaciones" aparezca la divisa de una determinada operación.
- Tendremos una relación clave foránea entre la tabla "Divisas", campo "Id Divisa", frente a la tabla "Operaciones", campo "Id Divisa".

ID_DIVISA	DESC	ID_OPERACION	ID_DIVISA	IMPORTE
EUR	Euro	1	EUR	786
GBP	Libra	2	EUR	785
JPY	Yen	3	DKK	1453
DKK	Corona Danesa	4	EUR	-1875
		5	JPY	586542

- En nuestra tabla "Operaciones", no podemos insertar una nueva línea en USD, por ejemplo, ya que no está registrada en la tabla "Divisas" y tendríamos un error de integridad

Bases de datos relacionales

Diseño del modelo de datos

- El modelo de datos es el conjunto de Tablas y relaciones mediante el cual se va a clasificar y almacenar la información.
- Un buen modelo de datos determina la efectividad de la base de datos en cuanto a integridad, introducción y manipulación de la información.
- El diseño del modelo de datos es pues el paso más importante y requiere hacerlo prácticamente con lápiz y papel.
- Es conveniente antes de crear la base de datos pensar en el modelo de datos en estos términos:
 - Clasificación de la información y diseño de tablas que garanticen la integridad.
 - Definición de relaciones.
 - Determinación de claves primarias.

EJERCICIO

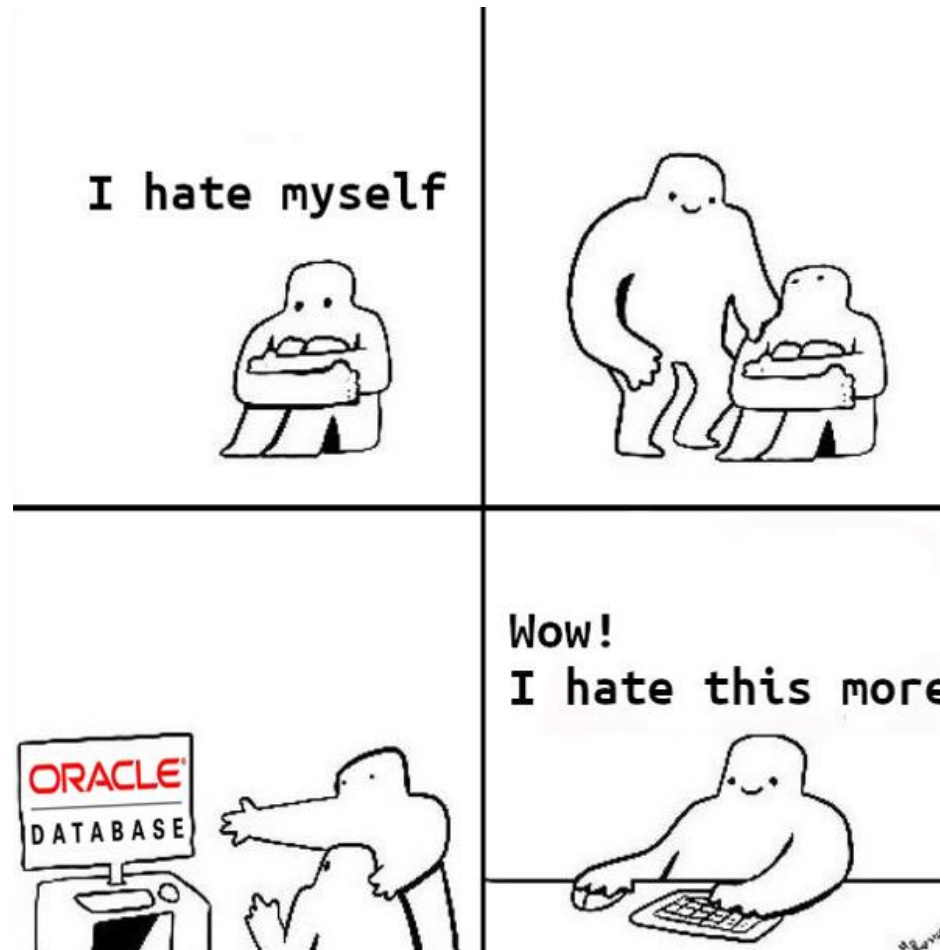
Diseño de un modelo de datos:

- Necesitamos diseñar un modelo de datos para la cartera de préstamos con garantía hipotecaria que tenemos en nuestro banco.
- En general, habrá una serie de clientes que serán los que tienen contratados productos en nuestros bancos (pueden tener más de un contrato asociado). Para el análisis de riesgos, todos estos clientes estarán clasificados con un rating (tipo S&P, Moodys) que nos dará una idea de su probabilidad de impago.
- Todos los contratos con garantía hipotecaria están vinculados a algún tipo de índice y pagan algún tipo de interés. Datos básicos del contrato serán, entre otros, su fecha de inicio, fecha fin, valor de la deuda...
- Además, los contratos están colateralizados con garantías inmobiliarias (puede ocurrir que con más de una garantía o también que una misma garantía aparezca en varios contratos).
- Así pues, las garantías inmobiliarias serán activos de distinta tipología (garaje vivienda, trastero, local comercial,...) que contarán con distintos valores de tasación a lo largo del tiempo.
- Los contratos pueden estar en distintas divisas, y debemos almacenar los tipos de cambio vs euro para poder revaluar la cartera.



2. SQL Basics

Comenzamos nuestro viaje...



SQL (Structured Query Language)

- Es un estándar ANSI (American National Standards Institute) desde 1986. Desde su primera versión ha sufrido varias revisiones a lo largo de los años.
- No es case-sensitive, se puede escribir en mayúscula o minúsculas. Por claridad, es importante escribir las sentencias manteniendo un estilo.
- Propone la división de todas las tareas posibles a realizar en una base de datos en dos sub lenguajes:
 - **Lenguaje de definición de datos (DDL, Data Definition Language):**
definición y modificación de objetos de base de datos (tablas, índices, esquemas, etc.).
 - **Lenguaje de manipulación de datos (DML, Data Manipulation Language):**
consulta y manipulación de los datos.

RDBMS

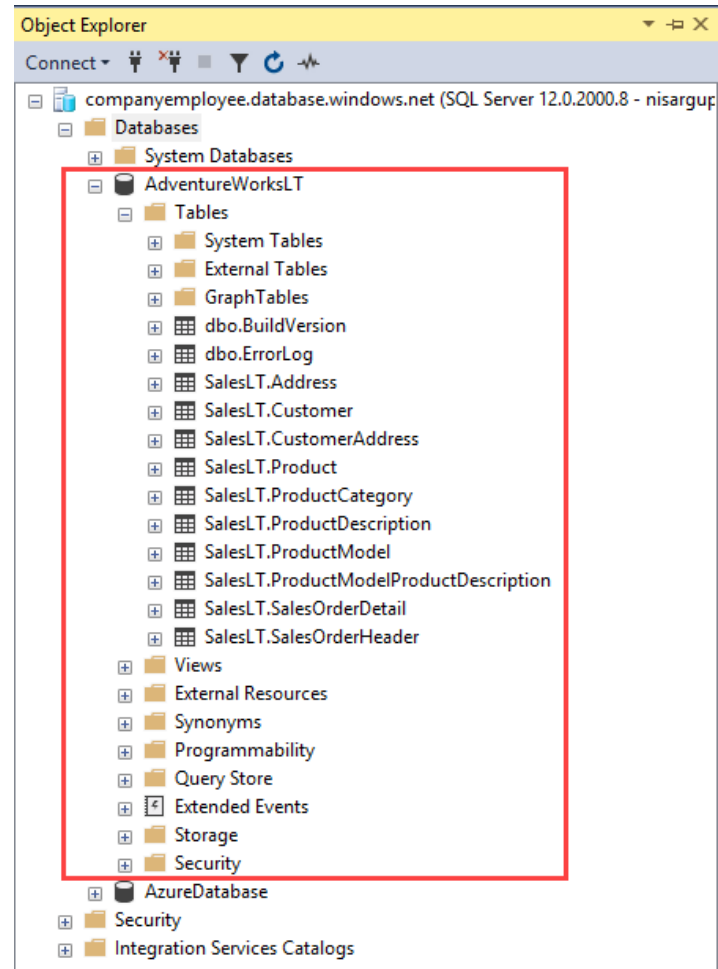
- Permiten llevar a cabo todas las tareas necesarias sobre una base de datos:
 - Crear nuevas bases de datos.
 - Crear nuevas tablas en una base de datos.
 - Establecer permisos en las tablas, procedimientos y puntos de vista.
 - Insertar registros en una base de datos.
 - Actualizar registros en una base de datos.
 - Eliminar registros de una base de datos.
 - Ejecutar consultas en una base de datos.
 - Crear procedimientos almacenados en una base de datos.
 - Crear vistas en una base de datos.
- Aunque SQL es un estándar, cada proveedor posee algunas especificidades o modificaciones propias que hacen que no sean 100% compatibles.



Elementos SQL

- **Base de datos (DATABASE):** se puede definir como un conjunto de objetos (tablas, índices, restricciones, usuarios, datos...).
- **Tabla (TABLE):** estructura fija de campos.
- **Campo (COLUMN, FIELD):** son las columnas de una tabla.
- **Registro (REGISTER, ROW):** son las filas de una tabla (también se pueden denominar tuplas).
- **Clave primaria (PK, PRIMARY KEY):** identificador unívoco de una fila. Todas las PKs tienen asociado un índice siempre.
- **Clave foránea (FK, FOREIGN KEY):** establece una relación entre dos tablas. La relación se establece a una PK de otra tabla.

Elementos SQL



Tipos de datos

- **Numérico:** para almacenar números enteros o decimales. [Numeric Types Overview \(MySQL\) - Numeric Types \(SQL Server\)](#)
 - **Exactos:**
 - **INT:** para números enteros.
 - **BIT o BOOL:** para elementos del tipo booleano o *flags*.
 - **DECIMAL(precision, scale) o NUMERIC (precision, scale):** donde precision es el número de dígitos totales y scale el número de dígitos de la parte decimal.
 - **Aproximados:**
 - **FLOAT:** números con precisión simple. Tiene una precisión aproximada de 7 decimales.
 - **DOUBLE:** números con precisión doble. Tiene una precisión aproximada de 15 decimales.
 - **REAL**

Tipos de datos

- **Fecha y hora:** para almacenar campos de tipo fecha o de tipo horario. [Date and Time Type Overview \(MySQL\)](#) - [Date and Time Types \(SQL Server\)](#)
 - **DATE:** almacena únicamente fechas ('yyyy-mm-dd').
 - **DATETIME:** combina fecha y hora ('yyyy-mm-dd hh:mm:ss').
 - **TIMESTAMP:** número de segundos transcurrido desde '1970-01-01 00:00:00' UTC.
- **Cadenas:** para almacenar cadenas de texto. [String Type Overview \(MySQL\)](#) - [String Type \(SQL Server\)](#)
 - **CHAR(n):** cadenas de longitud fija (n). Se puede establecer el CHARACTER SET a almacenar en la definición del campo.
 - **VARCHAR(n):** cadenas de longitud variable (n indica la longitud máxima). Se puede establecer el CHARACTER SET al almacenar en la definición del campo.
 - **TEXT, BLOB:** para campos de texto/binario más grandes. Por ejemplo: cuerpos de noticias, ficheros, etc...
- **NULL:** representa un valor desconocido en cualquiera de los tipos de datos anteriores.

Tipos de datos

Para una aplicación de música necesitaremos una tabla que contenga la información referida a los álbumes:

id	Título	Autor	Fecha	Precio	Nº canciones
1	'It's Only Rock 'n' Roll'	'Rolling Stones'	'1974-10-18'	12.50	10
2	'Let it be'	'The Beatles'	'1970-05-08'	15.99	12
3	'Thriller'	'Michael Jackson'	'1982-11-30'	17.99	9

INT	TEXT	VARCHAR(50)	DATE	FLOAT	INT
-----	------	-------------	------	-------	-----

Tipos de datos

Diseño de tablas: Tipo de datos en sqlite

- **Null:** valor “vacío”.
- **Integer:** para valores enteros.
- **Real:** para valores con coma flotante.
- **Text:** para guardar cadenas de texto.
- **Blob:** para guardar información binaria.

Operadores básicos

Tipo	Operador	Acción
Aritméticos	+	Operación de suma.
	-	Operación de resta.
	*	Operación de multiplicación.
	/	Operación de división.
	%	Operación de módulo (resto de la división).
Lógicos	AND	TRUE si ambas expresiones booleanas son TRUE.
	OR	TRUE si cualquiera de las dos expresiones booleanas es TRUE.
	NOT	Invierte el valor de cualquier otro operador booleano.

Operadores básicos

Tipo	Operador	Acción
Comparación	=	Igualdad.
	<>	Desigualdad.
	>	Mayor que...
	>=	Mayor o igual que...
	<	Menor que...
	<=	Menor o igual que...
	IS NULL	Selecciona únicamente los valores nulos (NULL).
	IS NOT NULL	Selecciona todos aquellos valores no nulos (NULL).
	BETWEEN	TRUE si el operando está dentro de un intervalo.
	IN	TRUE si el operando es igual a uno de la lista de expresiones.
	LIKE	TRUE si el operando coincide con un patrón.
	EXISTS	TRUE si una subconsulta contiene cualquiera de las filas.



3. Creación de Bases de Datos

Creación de base de datos

Las relaciones

- Al hacer uso de las relaciones, **se evita la duplicidad de datos, ahorrando memoria y espacio en disco, aumentando la velocidad de ejecución y facilitando al usuario el tratamiento de los datos.**
- Para poder relacionar tablas entre sí, se deberá especificar **un campo común** que contenga el mismo valor en las dos tablas, y dicho campo será la clave principal en cada una de ellas.
- Las tablas se relacionan dos a dos, donde una de ellas será la **tabla principal** de la que parte la relación y la otra será **la tabla secundaria** destino de la relación.

Modelo de Entidad-Relación

Herramienta para el **modelado de datos**, permite representar las entidades relevantes de un sistema de información, sus relaciones y propiedades (atributos).

Elementos de un modelo entidad-relación:

- **Entidad:** representa una cosa u objeto del mundo real. Pueden ser objetos con existencia física (por ejemplo: persona...) o con existencia conceptual (por ejemplo: cuenta bancaria...).
- **Atributo:** son las características que describen o definen las entidades/relaciones.
- **Relación:** establece dependencias entre entidades. La **cardinalidad** de una relación indica el número de entidades con las que puede estar relacionada otra entidad.

Modelo de Entidad-Relación

La relación puede ser:

- **Uno a uno (1:1):** un registro de una entidad A se relaciona con un registro de una entidad B.
- **Uno a varios (1:n):** un registro de una entidad A se relaciona con cero o varios registros de una entidad B.
- **Varios a uno (n:1):** un registro de una entidad B se puede relacionar con cero o varios registros de una entidad A.
- **Varios a varios (n:m):** una entidad A se relaciona con varios registros de la entidad B y viceversa.

Modelo de Entidad-Relación

Tipos de relación

EJEMPLO

- **Relación Uno a Uno:** Tenemos dos tablas, una con los datos de diferentes poblaciones y otra con una lista de alcaldes. Una población solo puede tener un alcalde, y un alcalde lo será únicamente de una población.
- **Relación Uno a Varios:** Un fondo de inversión indexado sólo puede tener un índice de referencia asociado, pero un mismo índice puede estar referenciado a varios fondos.
- **Relación Varios a Varios:** Un cliente puede ser partícipe de varios fondos, y un fondo puede ser comprado por varios partícipes. **Las relaciones varios a varios se suelen representar definiendo una tabla intermedia.** En el ejemplo anterior sería definir una tabla que contenga las posiciones de los clientes de un banco.

Creación de base de datos

Integridad referencial

- La **integridad referencial** es un sistema de **reglas** que utilizan los motores de bases de datos para **asegurarse que las relaciones entre registros de tablas relacionadas son válidas** y que no se borren o cambien datos relacionados de forma accidental.
- Al exigir integridad referencial en una relación, le estamos diciendo al sistema de base de datos que **no nos deje introducir datos en la tabla secundaria si previamente no se han introducido en la tabla principal**.

EJEMPLO

En el ejemplo anterior de uno a varios, no nos dejará introducir un índice de referencia a un fondo, si previamente éste no está dado de alta en la tabla índices.

Creación de base de datos

Integridad referencial

La integridad referencial dispone de dos acciones:

- **Actualizar registros en cascada:** Cuando se cambie el valor del campo de la tabla principal, automáticamente cambiarán los valores de sus registros relacionados en la tabla secundaria.

ILUSTRACIÓN

Si cambiamos el nombre de un índice, S&P por S&P_Total_Return, en la tabla índices, automáticamente todos los fondos asociados al S&P cambiarán al S&P_Total_Return.

- **Eliminar registros en cascada:** Cuando se elimine un registro de la tabla principal se borrarán también los registros relacionados en la tabla secundaria.

ILUSTRACIÓN

Si eliminamos el índice S&P de la tabla índices, automáticamente todos los fondos asociados al S&P se borrarán de la tabla Fondo/índice.

Diagrama Entidad-Relación

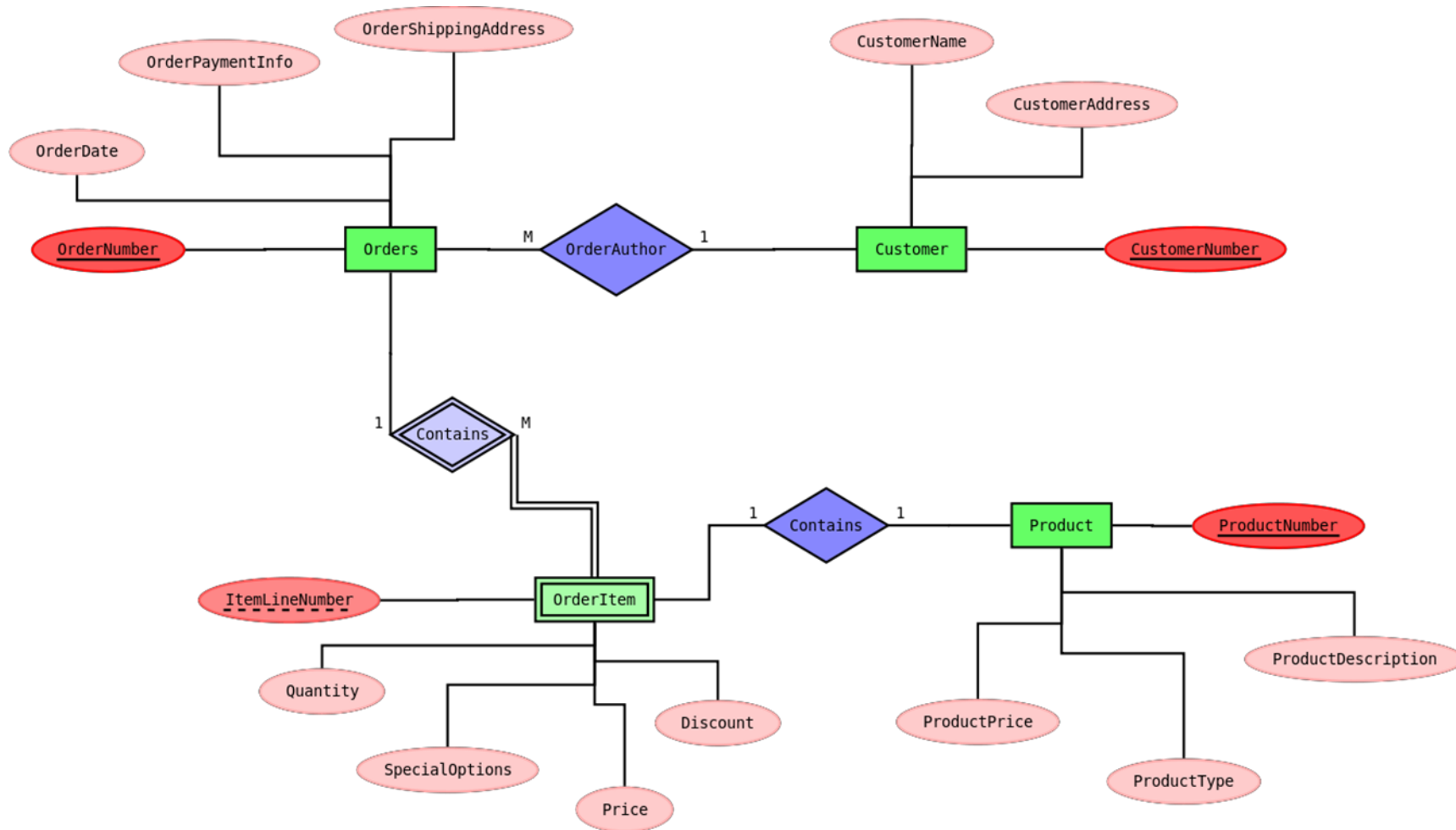
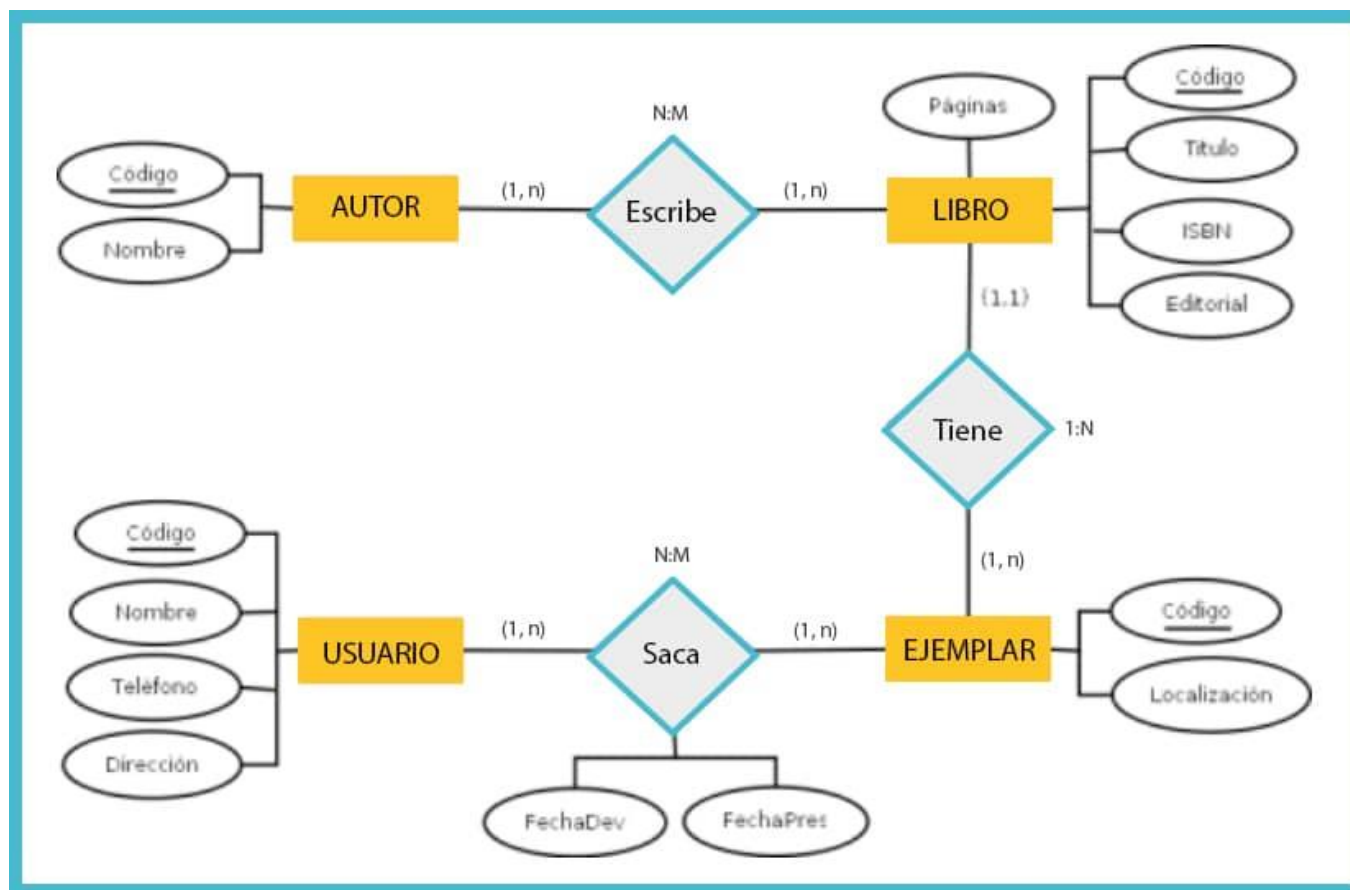


Diagrama Entidad-Relación



Del diagrama entidad relación a las tablas

- **Entidades:** las entidades se convierten en tablas. Los atributos de la entidad en columnas de la tabla.
- **Relaciones:** en general las relaciones se transforman también en tablas. En el caso de que la relación contenga atributos siempre genera tablas.
 - **Uno a uno (1:1):** cada entidad de la relación se convierte en una tabla. La relación aparece como clave primaria en una de las tablas y como clave foránea en la otra tabla.
 - **Uno a varios (1:n):** cada entidad de la relación se convierte en una tabla. Se pasa la clave primaria de la entidad con cardinalidad 1 a la tabla de la otra entidad como clave foránea.
 - **Varios a varios (n:m):** tanto las entidades como la relación se convierten en tablas. Las claves primarias de las entidades aparecen como clave primaria en la relación.

Ejemplo práctico

- Supongamos que vamos a desarrollar una aplicación para ver videos (aka Youtube). Necesitaremos:
 - Vídeos
 - Youtuber
 - Canal
 - Usuarios
 - Listas de reproducción

Crear un modelo entidad relación para la base de datos de la aplicación.

Convertir modelo en tablas de una base de datos.

Ejemplo práctico

Vídeo

- Título
- Duración
- Número de reproducciones
- Id

Youtuber

- Nombre
- Biografía
- Id

Canal

- Título
- Año de lanzamiento
- Id

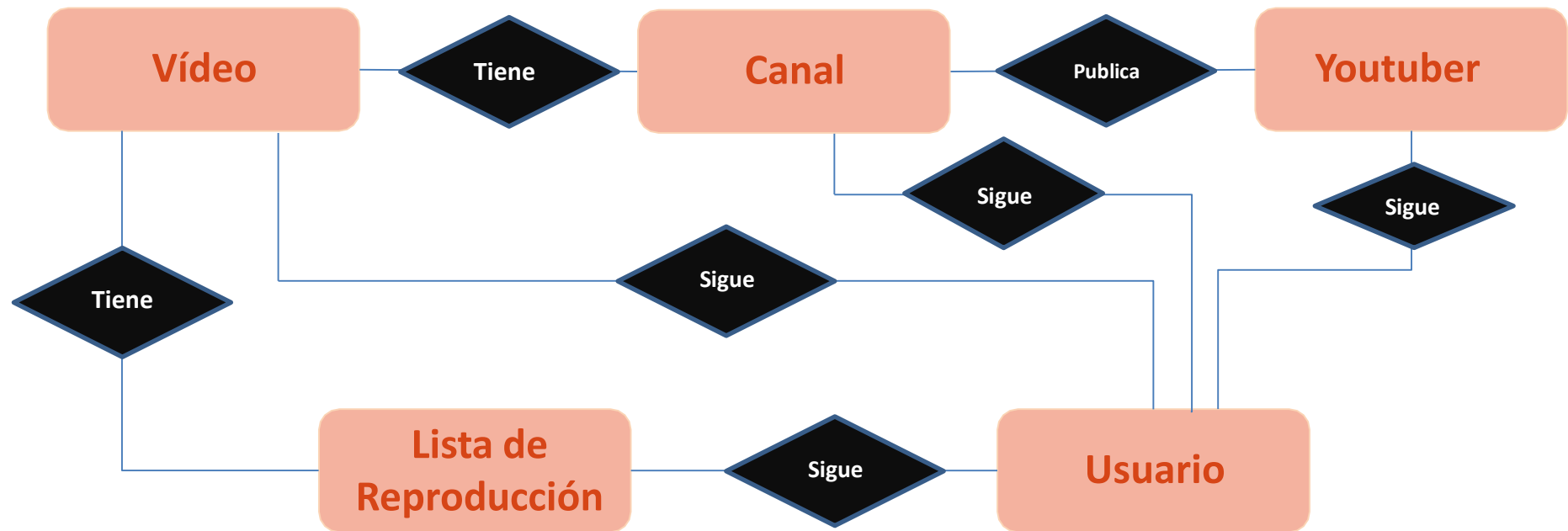
Lista de Reproducción

- Título
- Pública / privada
- Id

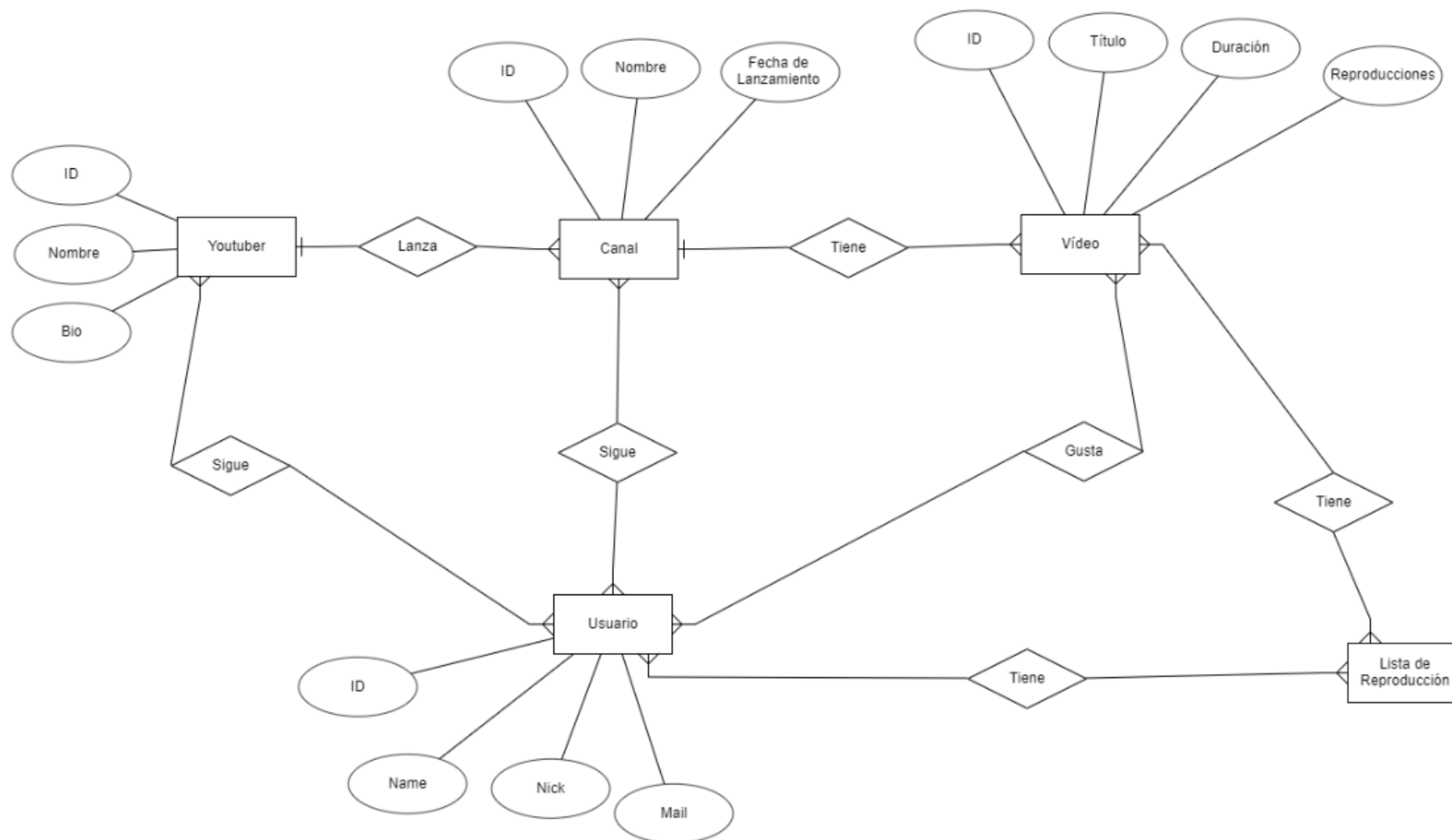
Usuario

- Nombre
- Nombre de usuario
- Correo electrónico
- Id

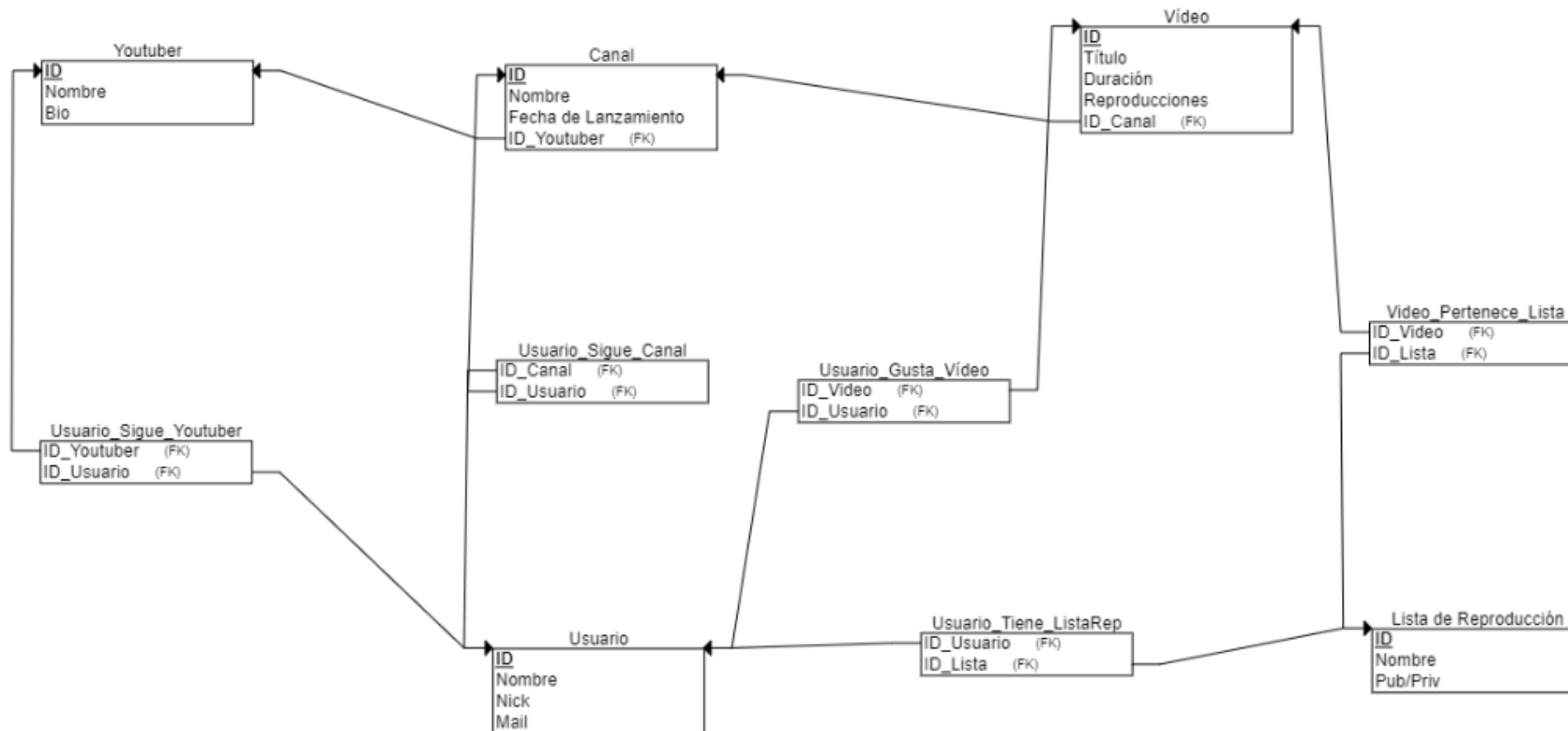
Ejemplo práctico



Ejemplo práctico



Ejemplo práctico



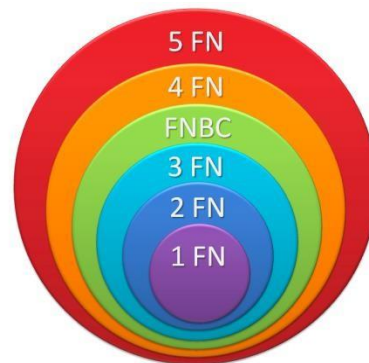
EJERCICIO

Ahora vosotros (Opcional)

- Supongamos que vamos a desarrollar una aplicación de apoyo a BME, donde poder gestionar la información respecto a sus clientes, posiciones, accionariado de cada empresa... Necesitaremos:
 - Usuario
 - Balance (Por usuario)
 - Transacciones
 - Leader Board (De cada cuenta, mejores transacciones)
 - Empresa (Número acciones total)
 - Accionistas (por empresa)
- Crea un modelo entidad relación para la base de datos de la aplicación.
- Conviértelo en tablas de una base de datos.

Normalización de bases de datos

- A través de una serie de reglas o normas aplicadas en el momento del diseño de una base de datos relacional podremos:
 - Evitar redundancia de datos.
 - Disminuir problemas derivados de la actualización de datos en tablas.
 - Proteger la integridad de datos.
- Estas reglas se denominan Formas Normales. Las tres primeras formas normales son suficientes para cumplir las necesidades de la mayoría de base de datos.
- En la práctica es complicado hacer cumplir todas las reglas de las Formas Normales.



Primera forma normal: 1FN (I)

- Una tabla está en primera forma (1FN) normal sí y solo sí es isomorfa a alguna relación, es decir, si se cumplen las siguientes condiciones:
 - **No existe orden en las filas** (de arriba a abajo). El orden no es significativo en la vista.
 - **No existe orden en las columnas** (de izquierda a derecha).
 - **No hay filas duplicadas**. Tiene una clave primaria.
 - Cada intersección de fila y columna **contiene exactamente un valor** del dominio aplicable. Los campos son atómicos. No puede contener columnas con valores nulos o más de un valor.
- Si una tabla no cumple la 1FN, no cumple tampoco el modelo relacional.

Primera forma normal: 1FN (II)

Stock	ISIN	Sector	Emisiones
VOLKSWAGEN INTL FIN NV	VOW GR Equity	Automobiles & Parts	XS1910948675
TELECOM ITALIA SPA	TIT IM Equity	Telecommunications	XS0486101024
VALFORTEC SL	8976822Z SM Equity	Utilities	ES0305542005
UNICREDIT SPA	UCG IM Equity	Banks	IT0005199267

- La tabla anterior guarda información de stocks/emisiones.
- ¿Qué pasa si queremos guardar más de una emisión por stock?

Primera forma normal: 1FN (III)

Stock	ISIN	Sector	Emisiones
VOLKSWAGEN INTL FIN NV	VOW GR Equity	Automobiles & Parts	XS1910948675 XS2152061904
TELECOM ITALIA SPA	TIT IM Equity	Telecommunications	XS0486101024
VALFORTEC SL	8976822Z SM Equity	Utilities	ES0305542005
UNICREDIT SPA	UCG IM Equity	Banks	IT0005199267

- No cumple la 1FN porque hay más de un valor de dominio en la columna Emisiones.

Primera forma normal: 1FN (IV)

Stock	ISIN	Sector	Emisiones (I)	Emisiones (II)	Emisiones (III)
VOLKSWAGEN INTL FIN NV	VOW GR Equity	Automobiles & Parts	XS1910948675	NULL	NULL
TELECOM ITALIA SPA	TIT IM Equity	Telecommunications	XS0486101024	XS2152061904	NULL
VALFORTEC SL	8976822Z SM Equity	Utilities	ES0305542005	NULL	NULL
UNICREDIT SPA	UCG IM Equity	Banks	IT0005199267	NULL	NULL

- No cumple la 1FN porque las columnas Emisiones 2 y Teléfono 3 contienen valores nulos.

Primera forma normal: 1FN (V)

Stock	ISIN	Sector	Emisiones
VOLKSWAGEN INTL FIN NV	VOW GR Equity	Automobiles & Parts	XS1910948675
TELECOM ITALIA SPA	TIT IM Equity	Telecommunications	XS0486101024, XS2152061904
VALFORTEC SL	8976822Z SM Equity	Utilities	ES0305542005
UNICREDIT SPA	UCG IM Equity	Banks	IT0005199267

- No cumple la 1FN porque la columna Emisiones puede ahora almacenar o una emisión o una lista de las mismos. No existe un tipo/dominio definido para la columna Teléfono (la columna no es atómica).

Primera forma normal: 1FN (VI)

Stock	ISIN	Sector
VOLKSWAGEN INTL FIN NV	VOW GR Equity	Automobiles & Parts
TELECOM ITALIA SPA	TIT IM Equity	Telecommunications
VALFORTEC SL	8976822Z SM Equity	Utilities
UNICREDIT SPA	UCG IM Equity	Banks

Stock	Emisiones
VOLKSWAGEN INTL FIN NV	XS1910948675
VOLKSWAGEN INTL FIN NV	XS2152061904
TELECOM ITALIA SPA	XS0486101024
VALFORTEC SL	ES0305542005
UNICREDIT SPA	IT0005199267

- **Solución:** emplear dos tablas diferentes: una para guardar los nombres de los stocks, la otra para guardar las emisiones.

Segunda forma normal: 2FN (I)

- Una tabla está en segunda forma normal (2FN) si está en 1FN y además, sí y solo sí, dada una clave primaria, cualquier atributo que no constituya la clave primaria, depende íntegramente de la clave primaria al completo.
- Cuando una tabla no cumple la 2FN, puede sufrir problemas a la hora de mantener la información actualizada de manera correcta.
- Existen casos para los que aún cumpliendo la 2FN, es posible que se den problemas de actualización. Para estos casos existe la 3FN.

Segunda forma normal: 2FN (II)

Gestora	Fondo	Dirección Gestora
Afi SGIC	AFI Global FI	Marqués de Villamejor 5, 28006 Madrid
AFI Inversiones Globales	Fondo Bolsa Social	Marqués de Villamejor 5, 28006 Madrid
Santander AM	Santander AM Latin American Equity Opportunities	Serrano 69, 28006 Madrid
Santander AM	Santander AM Euro Equity	Serrano 69, 28006 Madrid

- La tabla anterior guarda información de gestoras, sus fondos y la dirección de la gestora.
- La clave primaria de la tabla está conformada por las columnas “Gestora” y “Fondo”.
- Este diseño de tabla no cumple la 2FN, ya que la columna “Dirección Gestora” no depende de la clave primaria en su totalidad, únicamente de “Gestora”.

Segunda forma normal: 2FN (III)

Gestora	Fondo
Afi SGIC	AFI Global FI
AFI Inversiones Globales	Fondo Bolsa Social
Santander AM	Santander AM Latin American Equity Opportunities
Santander AM	Santander AM Euro Equity

Gestora	Dirección Gestora
Afi SGIC	Marqués de Villamejor 5, 28006 Madrid
Santander AM	Serrano 69, 28006 Madrid

- **Solución:** emplear dos tablas diferentes: una para guardar los fondos, la otra para guardar las direcciones.

Tercera forma normal: 3FN (I)

- Una tabla está en tercera forma normal (3FN) si está en 2FN y además, sí y solo sí, ningún atributo no primario (atributo no perteneciente a la clave primaria) de la tabla es dependiente transitivamente (a través de otro) de una clave primaria. Es decir, un atributo no primario depende de otro atributo no primario.
- La 3FN, soluciona los problemas de actualización no contemplados por la 2FN.

Tercera forma normal: 3FN (II)

Bond ISIN	Maturity Date	Emisor	Sector
XS0288429532	22/02/2027	GE CAPITAL EURO FUNDING	Industrial Goods & Services
ES0284940006	07/07/2027	PHARMA MAR SA	Healthcare
ES0276252014	19/11/2030	MELIA HOTELS INTL SA	Travel & Leisure
XS1861206636	01/11/2028	GOLDMAN SACHS GROUP INC	Financial Services

- La tabla anterior guarda información de bonos, así como de sus emisores.
- La clave primaria de la tabla está conformada por la columna “Bond ISIN”.
- Este diseño de tabla no cumple la 3FN, ya que la columna “Sector” depende transitivamente de la clave primaria, a través de la columna “Emisor”.

Tercera forma normal: 3FN (III)

Bond ISIN	Maturity Date	Emisor
XS0288429532	22/02/2027	GE CAPITAL EURO FUNDING
ES0284940006	07/07/2027	PHARMA MAR SA
ES0276252014	19/11/2030	MELIA HOTELS INTL SA
XS1861206636	01/11/2028	GOLDMAN SACHS GROUP INC

Emisor	Sector
GE CAPITAL EURO FUNDING	Industrial Goods & Services
PHARMA MAR SA	Healthcare
MELIA HOTELS INTL SA	Travel & Leisure
GOLDMAN SACHS GROUP INC	Financial Services

- **Solución:** emplear dos tablas diferentes: una para guardar los bonos -> emisores, la otra para guardar las emisores -> sectores.



Kahoot Time



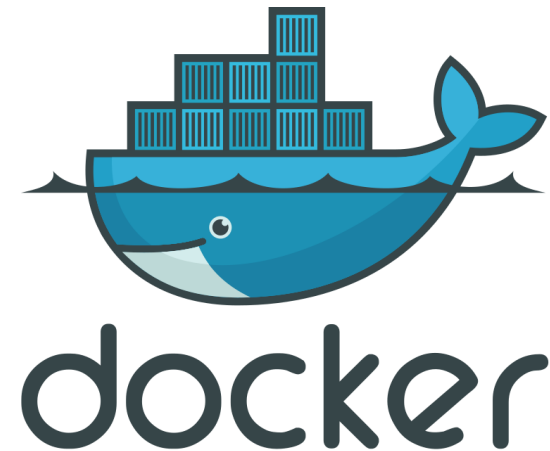
4. Instalación de Software

SQL Express

- Microsoft SQL Server es un sistema de gestión de base de datos relacional, desarrollado por la empresa Microsoft.
- Dentro de las opciones disponibles, SQL Server Express es una edición gratuita de SQL Server ideal para el desarrollo y la producción de aplicaciones de escritorio, aplicaciones web y pequeñas aplicaciones de servidor.
- Para poder descargarlo: <https://www.microsoft.com/es-es/sql-server/sql-server-downloads>

SQL Express en Mac

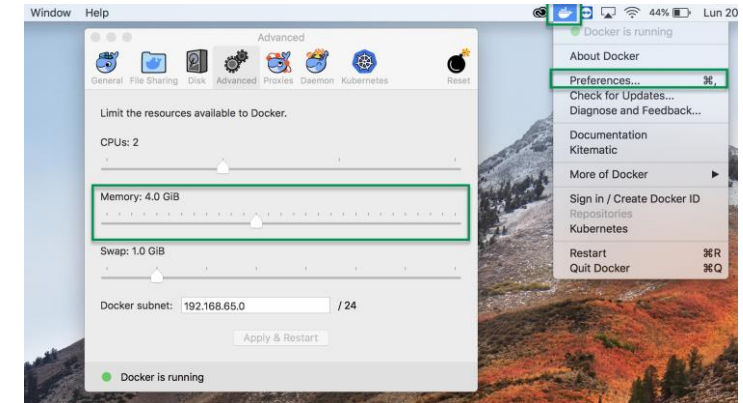
- La instalación de SQL Server en Mac se realiza mediante **Docker**, o a través de una **máquina virtual** con **Windows como OS**.
- **Docker** es un programa que permite crear **contenedores** de una aplicación en el cual incluir todo lo necesario para que la aplicación funcione correctamente. Así se puede tener la aplicación en diferentes servidores que tengan **Docker** instalado sin configurar todo manualmente y resolver el problema conocido de: *en mi máquina funciona*.
- Empezamos instalando **docker**:
<https://www.docker.com/get-started>



SQL Express en Mac

- Hay que configurar **Docker** para que funcione con 4 Gb de memoria. Este es un requisito de SQL Server.

Para esto dar clic el icono de Docker -> Preferences -> Advanced -> Memory -> 4 GB



- Obtener la imagen de **SQL Server** desde la línea de comandos:

```
sudo docker pull microsoft/mssql-server-linux:2017-latest
```

- Correr la imagen de SQL Server; se necesita aceptar los términos de la licencia, definir el password para el usuario y definir el puerto en el cual va a correr SQL Server:

```
sudo docker run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=M!P!ssw0rd!1521' -p 1433:1433 -d  
microsoft/mssql-server-linux:2017-latest
```


SQL Express en Mac

- Se puede ejecutar el siguiente comando para comprobar que SQL server esta corriendo correctamente:

docker ps -a

```
MacBook-Pro-de- :~ $ docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
fa4ca8e7f0b4	microsoft/mssql-server-linux:2017-latest	"/opt/mssql/bin/sqls..."	5 seconds ago	Up 3 seconds	0.0.0.0:1433->1433/t

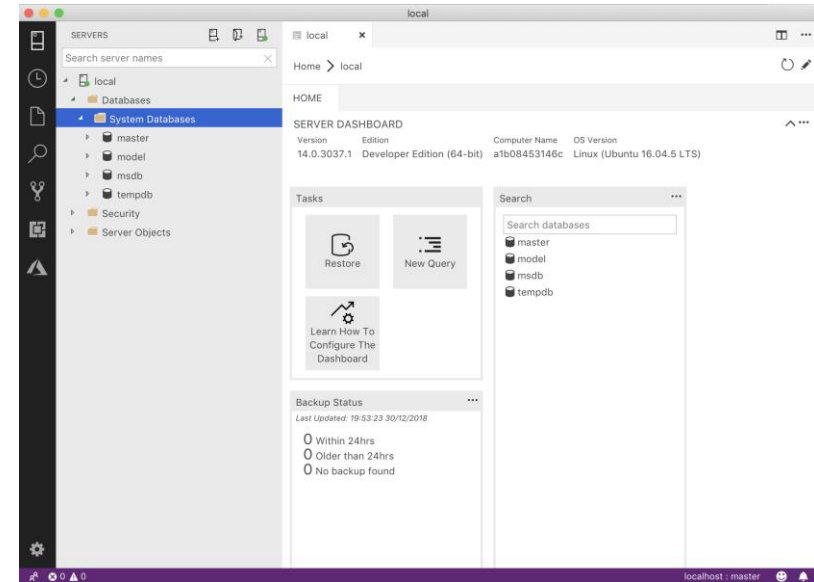
SQL Server Management Studio (SMSS)

- SQL Server Management Studio es una aplicación de software lanzada por primera vez con Microsoft SQL Server 2005 que se utiliza para configurar, administrar y administrar todos los componentes dentro de Microsoft SQL Server.
- Esto es, vista la teoría, es el RDBMS que vamos a utilizar.
- Para poder descargarlo: <https://docs.microsoft.com/es-es/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>

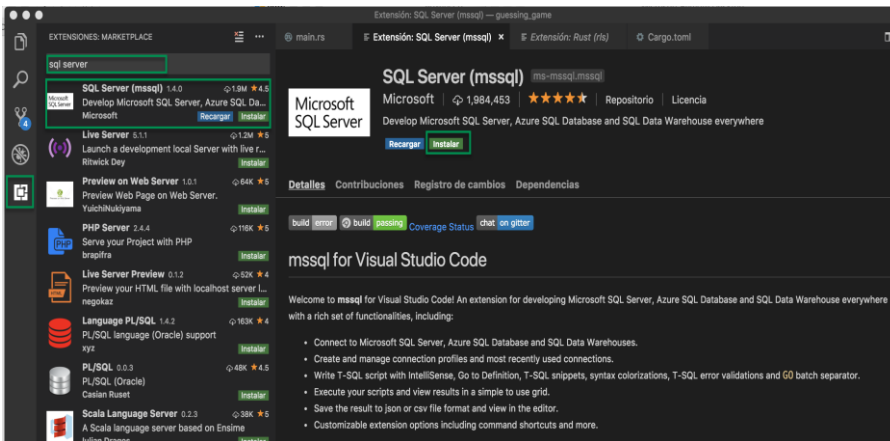
RDBMS en Mac

- SQL Management Studio no se encuentra disponible para Mac; existen 2 alternativas:
- **Azure Data Studio** antes llamado SQL Operations Studio:

<https://docs.microsoft.com/en-us/sql/azure-data-studio/download?view=sql-server-2017>



- Utilizar la extensión de SQL Server(mssql) en **Visual Studio Code**:



EJERCICIO

Vamos a explorar por primera vez nuestro entorno

- Abrimos SMSS y entramos en nuestro servidor local.
- Exploramos los scripts generadores de tablas ejemplo para ver la sintaxis SQL por primera vez.
- Ejecutamos estos scripts y exploramos tablas y datos.

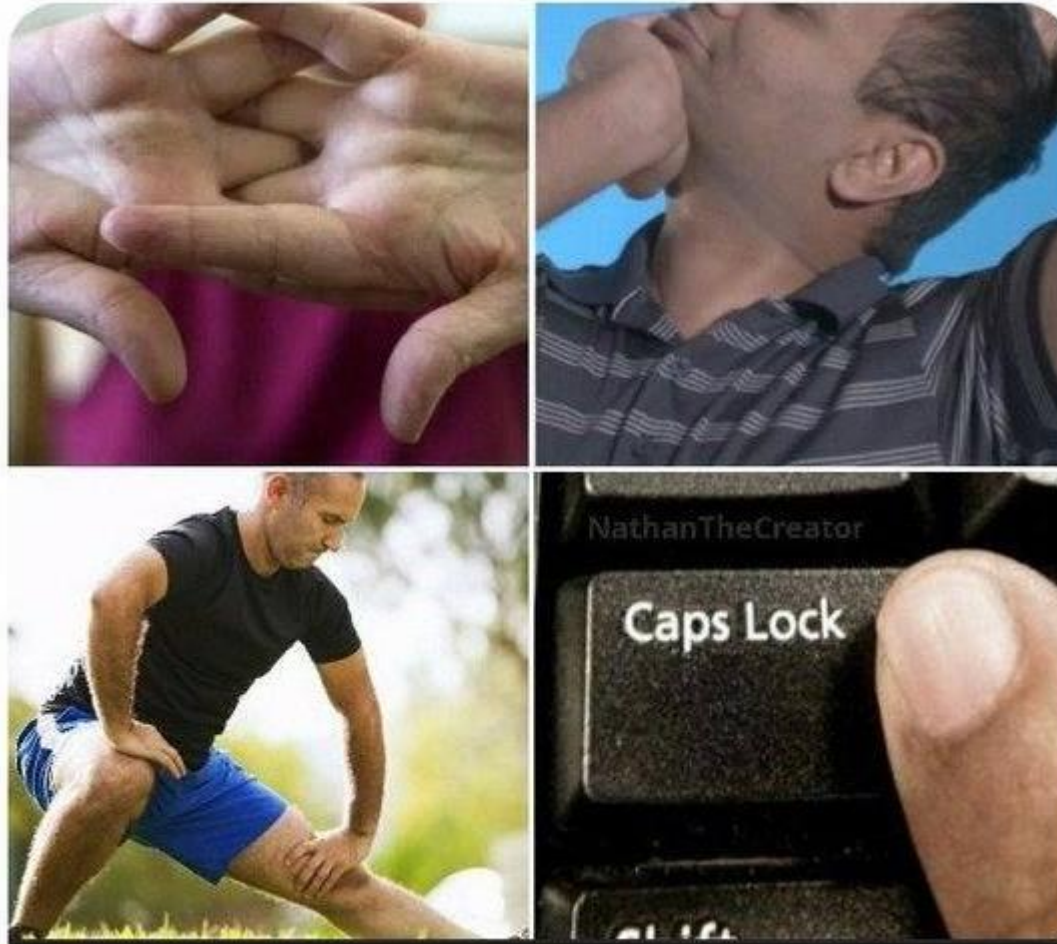


5. Introducción al lenguaje SQL (DDL)

SQL: DDL Vs DML

- Es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar varios tipos de operaciones sobre ésta.
- Las sentencias SQL se dividen en dos categorías: Lenguaje de definición de datos (data definition language (DDL)) y Lenguaje de manipulación de datos (data manipulation language (DML)).
- Las sentencias DDL se utilizan para crear y modificar la estructura de las tablas así como otros objetos de la base de datos.
- Qué veremos:
 - CREATE
 - ALTER
 - DROP
 - TRUNCATE

SQL... case sensitive ?



Consultas de definición básicas

CREATE

- Para crear una base de datos vacía:
- Opción 1:

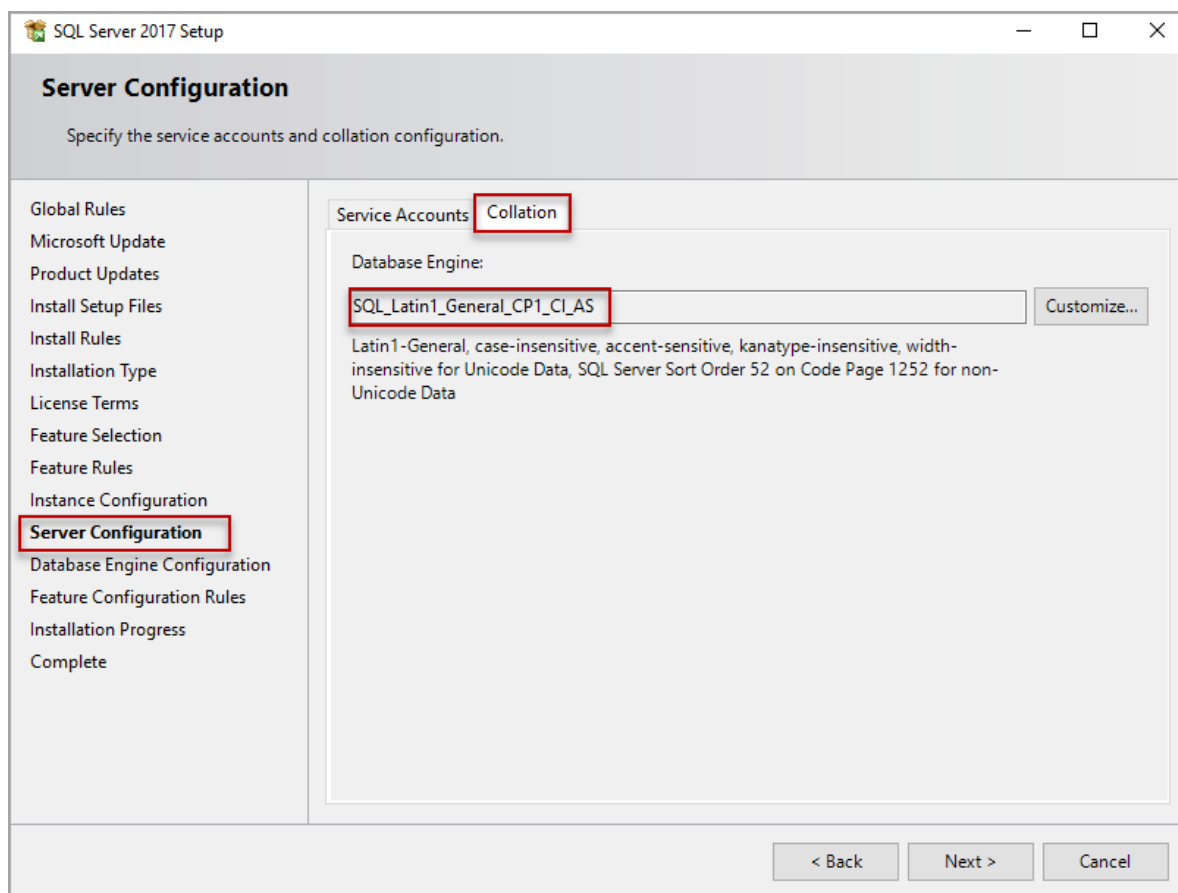
CREATE DATABASE <database_name>;

- Opción 2:

**CREATE DATABASE <database_name>
COLLATION <character_set>;**

¿Qué es el collation ?

- El collation es un conjunto de reglas que le indican al motor de la base de datos cómo comparar y ordenar los datos de caracteres en SQL Server.



Consultas de definición básicas

DROP

- Para crear eliminar una base de datos:

DROP DATABASE <database_name>;

- <database_name>: nombre de la base de datos a crear o eliminar.

Consultas de definición básicas

USE

- Para usar una base de datos en concreto:

USE <database_name>;

- Desde el momento en que ejecutemos esta sentencia todas las operaciones (queries, updates, creates, etc.) se ejecutarán en la base de datos seleccionada.

Consultas de definición básicas

CREATE TABLE

- Para crear nuevas tablas en base de datos:

```
CREATE TABLE <table_name> (  
    <column_name1> data_type1,  
    <column_name2> data_type2,  
    <column_name3> data_type3,  
    ...  
);
```

- **<table_name>**: nueva tabla a crear.
- **<column_name1> data_type1...:** columnas de la tabla junto con el tipo de datos que tiene cada una de ellas.

Restricciones - Recordatorio

- Las restricciones (CONSTRAINTS) sirven para establecer reglas y comprobaciones sobre los datos. En el caso de que no se cumpla la restricción, la acción realizada sobre la tabla (inserción o actualización) se aborta y el motor de base de datos lanzará un error. Existen los siguientes tipos:
 - **Clave primaria - PRIMARY KEY**
 - **Clave foránea - FOREIGN KEY**
 - **Valor por defecto - DEFAULT**
 - **Obligatoriedad - NOT NULL**
 - **Valor único - UNIQUE**
 - **Comprobación - CHECK**
- **¡Atención!** Las restricciones garantizan la integridad de la información de la base de datos, pero a su vez consumen recursos en el momento de inserción/actualización/borrado. Sacrifican rendimiento por integridad.

Consultas de definición básicas

Clave Primaria

Es posible añadir una clave primaria a una tabla en el momento de creación de la misma. La clave primaria debe contener valores únicos y no nulos. Al crear una clave primaria se creará un índice asociado.

```
CREATE TABLE <table_name>
(
    <column_name> data_type,
    ...,
    [CONSTRAINT <pk_name>] PRIMARY KEY (<column_name>, ...)
);
```

- **<pk_name>**: nombre de la clave primaria a crear. El nombre es opcional, si no se pone uno, el sistema de base de datos asignará uno por defecto.
- **<column_name>...**: columna o columnas que formarán parte de la clave primaria de la tabla.

Consultas de definición básicas

Clave Foránea

Es posible añadir una clave foránea a una tabla en el momento de creación de la misma. La clave foránea hace referencia a una campo que es clave primaria de otra tabla.

```
CREATE TABLE <table_name> (  
    <column_name> data_type,  
    ...,  
    [CONSTRAINT <fk_name>] FOREIGN KEY (<column_name>)  
    REFERENCES <table_name_ref>(<pk_table_name_ref>)  
);
```

- **<fk_name>**: nombre de la clave foránea a crear. El nombre es opcional, si no se pone uno, el sistema de base de datos asignará uno por defecto.
- **<column_name>...**: columna que tiene la restricción.
- **<table_name_ref>...**: nombre de la tabla a la que hace referencia la clave foránea.
- **<pk_table_name_ref>...**: campo de la tabla a la que hace referencia la clave foránea.

Consultas de definición básicas

Valor por defecto

Permite establecer un valor por defecto para una columna en el caso de dejarla vacía en el momento de la inserción.

```
CREATE TABLE <table_name> (  
    <column_name> data_type DEFAULT <default_value>,  
    ...,  
);
```

- o **<default_value>...:** valor por defecto de la columna. Tiene que estar acorde con el tipo de dato que almacena la columna.

Consultas de definición básicas

Obligatoriedad

Establece que un campo/columna no puede contener nunca valores nulos.

```
CREATE TABLE <table_name> (  
    <column_name> data_type NOT NULL,  
    ...,  
);
```

Consultas de definición básicas

Constraints

Es posible añadir condiciones de manera que limiten los valores que pueden tomar los registros en una columna. Por ejemplo: `column_value1 > 10 AND column_value2 = 1000`

```
CREATE TABLE <table_name>
(
    <column_name> data_type,
    ...,
    [CONSTRAINT <chk_name>] CHECK (<chk_condition>)
);
```

- **<chk_name>**: nombre de la comprobación/restricción que se creará. El nombre es opcional, si no se pone uno, el sistema de base de datos asignará uno por defecto.
- **<chk_condition>...:** expresión booleana para comprobar el contenido del registro.

EJERCICIO

Vamos a crear nuestra primera BBDD y nuestras primeras tablas

- Crear una nueva base de datos llamada **cambioDivisaDB**.
- Crear una tabla que almacene la siguiente información:
 - Identificador del país almacenado como un número entero.
 - Nombre del país (como máximo un país tendrá 50 caracteres en este campo).
- ¿Qué campo será la clave primaria de la tabla?
- Crear una tabla que almacene la siguiente información:
 - Identificador de la divisa almacenado como un número entero.
 - Nombre de la divisa (como máximo un país tendrá 20 caracteres en este campo).
 - Abreviatura de la divisa (siempre tendrá longitud = 3).
- ¿Cuál es la clave primaria?

EJERCICIO

Vamos a crear nuestra primera BBDD y nuestras primeras tablas

- Crear una tabla que almacene la siguiente información:
 - Identificador del país almacenado como un número entero.
- ¿Cuál es/son las claves primarias? ¿Hay claves foráneas? ¿Cuáles?
- Crear una tabla que almacene la siguiente información:
 - Divisa origen (identificador).
 - Divisa destino (identificador).
 - Tipo de cambio (cantidad).
- ¿Cuál es/son las claves primarias? ¿Hay claves foráneas? ¿Cuáles?

Consultas de definición básicas

Añadir columnas

Mediante esta sentencia se pueden **añadir nuevas columnas** a una tabla de base de datos. Las columnas nuevas se añadirán al final de la tabla.

```
ALTER TABLE <table_name>  
ADD <column_name> <data_type>;
```

Es posible establecer restricciones como las que hemos visto antes en el momento de creación de las nuevas columnas.

Consultas de definición básicas

Eliminar columnas

Mediante esta sentencia se pueden **eliminar columnas existentes** de una tabla de base de datos.

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

¡Atención! El proceso de eliminación de columnas debe realizarse con cuidado, no se pierda información.

Consultas de definición básicas

Modificación de columnas: Tipo de dato

Mediante esta sentencia se puede **modificar el tipo de dato** que contiene una columna de una tabla de base de datos.

```
ALTER TABLE <table_name>  
ALTER COLUMN <column_name> <data_type>;
```

¡Atención! Al cambiar una columna de tipo de dato los registros ya guardados en la tabla deben ser compatibles con el tipo de dato. En caso contrario se producirá un error o una pérdida de información (por ejemplo al cambiar la precisión de un campo numérico).

Consultas de definición básicas

Modificación de columnas: Renombrar

Mediante esta sentencia se puede **modificar el nombre de una columna** de una tabla de base de datos.

```
EXEC sp_rename '<table_name>.<old_column_name>', '<new  
column_name> ', 'COLUMN';
```


Consultas de definición básicas

Modificación de tablas: Clave Primaria

También es posible **añadir una clave primaria** a una tabla después de su creación (en el caso de que no la tuviera).

```
ALTER TABLE <table_name>  
ADD [CONSTRAINT <pk_name >] PRIMARY KEY (<column_name>, ...);
```

Y borrarla:

```
ALTER TABLE <table_name>  
DROP CONSTRAINT <pk_name >;
```

Consultas de definición básicas

Modificación de tablas: Clave Foránea

También es posible **añadir claves foráneas** a una tabla después de su creación.

```
ALTER TABLE <table_name>  
ADD [CONSTRAINT <fk_name >] FOREIGN KEY (<column_name>, ...)  
REFERENCES <table_name_ref>(<pk_table_name_ref>);
```

Y borrarla:

```
ALTER TABLE <table_name>  
DROP CONSTRAINT <fk_name >;
```

Consultas de definición básicas

Modificación de tablas: Clave Foránea

También es posible **añadir restricciones** a una tabla después de su creación.

```
ALTER TABLE <table_name>  
ADD [CONSTRAINT <chk_name >] CHECK (<chk_condition>);
```

Y borrarla:

```
ALTER TABLE <table_name>  
DROP CONSTRAINT <chk_name >;
```

Consultas de definición básicas

Borrado de tablas

Es posible **eliminar una tabla y todo su contenido** de la base de datos.

DROP TABLE <table_name>

¡Atención! Al borrar una tbla perderemos todo su contenido.

EJERCICIO

A partir de ahora, vamos a utilizar trabajar con las tablas que nos generará el script **Employees.sql**

- Ejecutamos el script **Employees.sql** para generar las tablas con la información que utilizaremos a partir de ahora.
- Exploramos las tablas y la estructura de la información que estas contienen. Generamos el diagrama para entender las relaciones.

EJERCICIO

Vamos a aplicar lo aprendido y alterar la base de datos

- Se necesita almacenar el **presupuesto máximo** y el **presupuesto consumido** de cada **departamento**, para ello se deberá modificar la tabla incluyendo dos nuevas columnas “**presupuesto_max**” y “**presupuesto_consumido**” que permita guardar estos datos.
- Se decide que **no** va a ser necesario guardar **el presupuesto máximo** de cada departamento. **Eliminar la columna.**
- Modificar la columna “**presupuesto_consumido**” para que solo acepte **enteros** y Renombrarla como “**presupuesto**”.



6. Carga de datos desde fichero

Carga de datos desde fichero

Es muy común realizar cargas de datos en tablas desde ficheros. Tanto en SQL Server como en otras bases de datos comerciales existe una sentencia que realiza dicha tarea.

Es posible configurar el formato del fichero donde se encuentran los datos, de manera que el gestor de base de datos detecte el comienzo y final de cada columna correctamente.

```
BULK INSERT <table_name>  
FROM <file_path>  
WITH (FIELDTERMINATOR '<field_separator>',  
ROWTERMINATOR '<line_separator>');
```

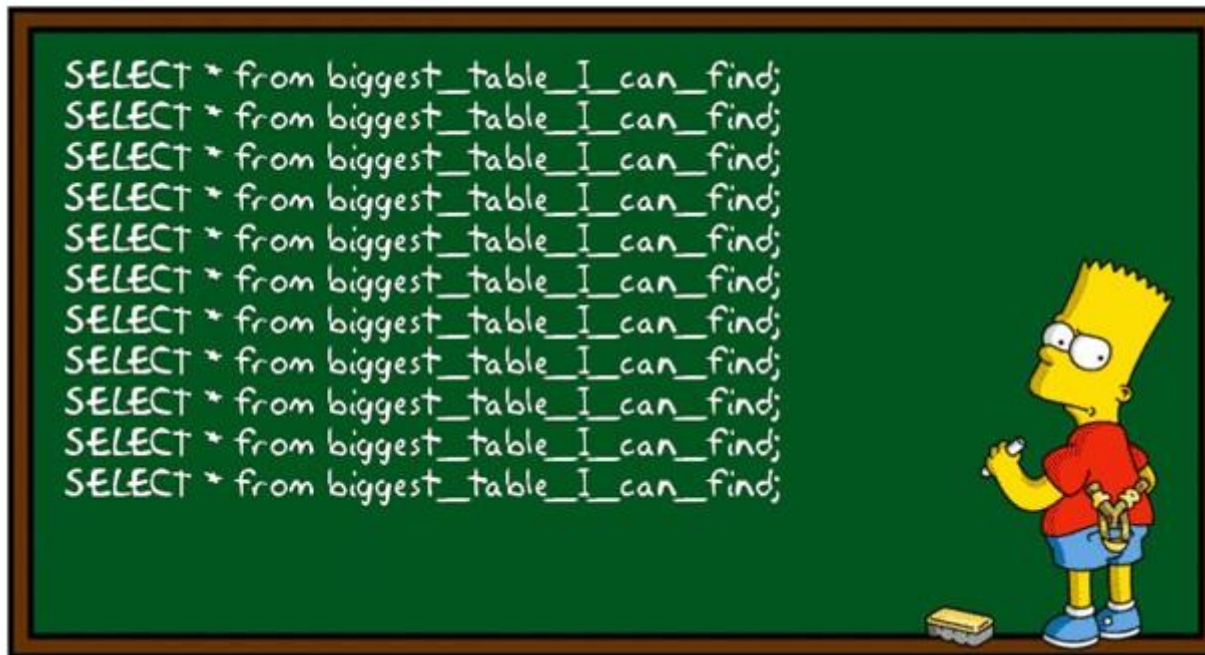
- **<file_path>**: ruta al fichero que contiene los datos a cargar.
- **<table_name>**: nombre de la tabla donde se realizará la carga.
- **<field_separator>**: carácter separador de columnas/campos. Normalmente la coma o el tabulador.
- **<line_separator>**: carácter separador de líneas. Normalmente el salto de línea.

<https://docs.microsoft.com/en-us/sql/t-sql/statements/bulk-insert-transact-sql?view=sql-server-ver15>



7. Introducción al lenguaje SQL (DML)

SQL...Cuidado



Lenguaje SQL (DML)

- Nos permitirá realizar consultas para recuperar la información de la base de datos con el filtro deseado y realizar cambios sobre la misma.
- ¿Qué veremos ?
 - Consultas de selección básica.
 - Inserción de registros.
 - Borrado de registros.
 - Actualización de registros.
 - Consultas de selección de múltiples tablas.

Cuidado con los errores



Consultas de selección básica

SELECT ... FROM ...

Para seleccionar información filtrada (o no) de la base de datos.

ILUSTRACIÓN

SELECT

InstrumentID, MarketId

FROM

Tinstruments

Resultado: Los instrumentos de nuestra BBDD junto con el mercado al que pertenece cada uno de ellos.

	InstrumentID	MarketId
1	BBVA:SM	MDR Stock
2	TEF:SM	MDR Stock
3	SAN:SM	MDR Stock
4	USDEUR:CUR	NY Stock
5	TSCO:LN	LN Stock
6	TSLA:US	NY Stock
7	IBE5Y:CORP	MDR Stock

En el caso de que se quieran mostrar todos los campos de una tabla, se usa el carácter *.

Consultas de selección básica

SELECT ... AS ... FROM ...

- Vamos a utilizar la palabra clave **AS <nueva denominación>** para renombrar campos o tablas.
- La ventaja de utilizar alias para las tablas es más relevante cuando se realizan cruces entre varias de ellas. Cuando renombramos una tabla accedemos a sus campos mediante **<nueva denominación tabla>.<nombre campo>**

EJEMPLO

SELECT

Instrumentos.InstrumentId **AS** Instrumento,
Instrumentos.MarketId **AS** Mercado

FROM

Tinstruments **AS** Instrumentos

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- Obtener un listado de todos los empleados.
- Seleccionar únicamente el nombre, apellido y fecha_contrato.
- Renombrar los campos de la siguiente manera:
 - nombre -> first_name
 - apellido -> last_name
 - fecha_contrato -> hire_date
- Seleccionar todos los campos de la tabla departamento.
- Seleccionar todos los campos de las tablas departamento y responsable_departamento;
¿Ambigüedad?

Consultas de selección básica

SELECT DISTINCT ... FROM ...

¿Y si queremos los mercados a los que pertenecen nuestros instrumentos?

Si hacemos: “**SELECT** MarketId **FROM** Tstruments”, obtendremos los mercados, pero repetidos. Para evitarlo, usamos la palabra clave “**DISTINCT**”:

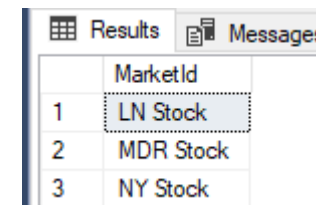
ILUSTRACIÓN

SELECT DISTINCT

MarketId

FROM

Tstruments



	MarketId
1	LN Stock
2	MDR Stock
3	NY Stock

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- Hallar las fechas de contrato de los empleados sin repeticiones.

Consultas de selección básica

SELECT TOP ... FROM ...

¿Y si queremos seleccionar únicamente unos cuantos instrumentos?

Utilizamos la palabra clave: “**TOP <número de registros>**”:

ILUSTRACIÓN

SELECT TOP 2

*

FROM

Tinstruments

Results		Messages			
	RefId	InstrumentId	MarketId	Description	TipoInstrumento
1	1	BBVA:SM	MDR Stock	BBVA Bank	Equity
2	2	TEF:SM	MDR Stock	Telefonica	Equity

Observación: En SQLite, la sintaxis es: **SELECT * FROM Tabla LIMIT 2**

Consultas de selección básica

SELECT < Función > ... FROM ...

- Este lenguaje también nos permite, directamente en la consulta, realizar ciertas operaciones básicas sobre los datos.
- Las funciones más utilizadas y las palabras clave que las ejecutan son:
 - Media: **AVG**
 - Contar: **COUNT**
 - Máximo: **MAX**
 - Mínimo: **MIN**
 - Suma: **SUM**
 - Desviación Típica: **StDev**
 - Varianza: **VAR**
- La sintaxis es **< palabra clave función > (< nombre columna >)**. Por ejemplo, **SELECT MAX(Price) FROM TPrices** nos devuelve el mayor precio guardado en la base de datos.

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- Hallar el salario medio de los empleados.
- Redondear el resultado anterior con 2 decimales.
- Hallar el número total de departamentos.

Consultas de selección básica

SELECT ... FROM ... WHERE ...

- Utilizamos la palabra clave **WHERE** cuando queramos añadir filtros y condiciones a nuestras consultas.

EJEMPLO

```
SELECT *  
FROM Tinstruments  
WHERE MarketId = 'LN STOCK'
```

Nos devolverá los instrumentos de nuestra base que pertenecen al mercado de Londres

Results		Messages			
	RefId	InstrumentId	MarketId	Description	TipoInstrumento
1	5	TSCO:LN	LN Stock	Tesco	Equity

Consultas de selección básica

SELECT ... FROM ... WHERE ... AND/OR

- Dentro de la condición de la consulta podemos añadir resultados lógicos mediante las palabras claves **AND** y **OR**.

EJEMPLO

```
SELECT *  
FROM TPrices  
WHERE (Fecha = convert(datetime, '22/02/2018', 103)  
OR Fecha = convert(datetime, '22/02/2018', 103) )  
AND Valor > 7
```

Results		Messages	
	RefId	Fecha	Valor
1	2	2018-02-22	7.62900918529954
2	5	2018-02-22	200.698924513211
3	6	2018-02-22	330.745805828884
4	7	2018-02-22	99.2530306533681

Nos devolverá los precios cuyas fechas correspondan al miércoles 21 de febrero o al jueves 22 y que superen las 7 unidades.

Consultas de selección básica

SELECT ... FROM ... WHERE ... IN

- Utilizamos la palabra clave **IN** cuando queremos filtrar por una lista de valores posibles.
- La sintaxis es **<nombre campo> IN (<valor1, valor2, ...>)**.

EJEMPLO

```
SELECT *  
FROM TInstruments  
WHERE MarketId IN ('LN STOCK', 'NY STOCK')
```

Nos devolverá los instrumentos de nuestra base que pertenecen al mercado de Londres o al de Nueva York.

Consultas de selección básica

SELECT ... FROM ... WHERE ... IN

- Normalmente **IN** se utiliza tomando como lista de valores posibles el resultado de una consulta.

EJEMPLO

```
SELECT *  
FROM TPrices  
WHERE RefId IN (SELECT RefId FROM TInstruments WHERE MarketId = 'MDR Stock')  
AND Fecha = Convert(Datetime, '22/02/2018',103)
```

Nos devolverá los precios que existan para el 13 de Junio de los instrumentos del mercado de Madrid.

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- ¿En qué año fue contratada Julia Rodríguez?
- ¿Qué empleados son responsables de departamento?
- ¿De qué departamentos es responsable Pilar Luján?
- ¿Qué día dejó de ser responsable Jesús Zaragoza del departamento con `id_departamento = 'd001'`?
- Hallar el nombre de empleado que cobra el máximo salario.
- Hallar la fecha del contrato del empleado con `id = 8`.

Consultas de selección básica

SELECT ... FROM ... WHERE ... BETWEEN

- Normalmente **IN** se utiliza tomando como lista de valores posibles el resultado de una consulta.

EJEMPLO

```
SELECT *  
FROM TPrices  
WHERE RefId = (SELECT RefId FROM TInstruments WHERE InstrumentId = 'TEF:SM')  
AND Fecha BETWEEN CONVERT(datetime, '15/02/2018', 103)  
AND CONVERT(datetime, '28/02/2018', 103)
```

Nos devolverá los precios de Telefónica entre el 15 de febrero y el 28 de Junio (ambos incluidos).

Consultas de selección básica

`SELECT ... FROM ... WHERE ... LIKE {patrón}`

- Utilizamos la palabra clave **LIKE** cuando queremos realizar una búsqueda (en campos de texto) basada en un patrón determinado.
- Para especificar el patrón se utilizan los caracteres especiales:
 - **%**: para especificar cualquier cadena de caracteres.
 - **_**: para especificar cualquier carácter.
- Ejemplos de patrones:
 - **'A_Z'**: toda cadena que empiece por A, esté seguida de otro carácter, y termine con Z.
 - **'STOCK%'**: toda cadena que empiece por STOCK y esté seguida de cualquier cadena (también puede ser la cadena vacía).
 - **'%EF%'**: toda cadena que contenga en su interior la cadena EF.

Consultas de selección básica

SELECT ... FROM ... WHERE ... LIKE {patrón}

ILUSTRACIÓN

- Si, por ejemplo, queremos recuperar de la base de datos los bonos de Diciembre: imaginemos que el campo **InstrumentId** de los bonos se ha construido como **'Bond' + 3 primeras letras del mes en inglés + Vcto.**
- ¿Cómo lo hacemos?

```
SELECT *  
FROM Tinstruments  
WHERE InstrumentId LIKE 'BondDec%'
```

	InstrumentId	InstrumentType	Mxid	MarketId
1	BondDec12	BOND	BUI012141DK	STOCKLON

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- Hallar el nombre completo de los empleado que fueron contratados a partir del 7 de julio de 2009 y cuyo apellido contenga la letra A.
- Hallar los ids, nombres y salarios de 5 empleados que tuvieron actualización de este último entre 2004 y ayer.
- Encontrar aquellos puestos que comienzan por la letra S.
- Encontrar los empleados que tienen la cadena 'la' en su nombre.

Consultas de selección básica

SELECT ... FROM ... [WHERE] ... ORDER BY

- Cuando realizamos una consulta, los registros son devueltos sin ningún order preestablecido.
- Para poder ordenar el resultado usamos la palabra clave **ORDER BY**.
- Su sintaxis corresponde a: **ORDER BY <nombre campo> [ASC, DESC]**. Donde **ASC** corresponde a orden ascendente y **DESC** a orden descendente.
- Especificar el tipo de orden es opcional, si no se especifica será ascendente.
- Se pueden elegir varios campos para ordenar, separados por comas: **ORDER BY <campo1> [ASC, DESC], <campo2> [ASC, DESC]...**
- Por ejemplo: **SELECT * FROM Tprices ORDER BY Fecha**, nos devuelve precios ordenados por fecha de manera ascendente.

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- Hallar las fechas de contrato de los empleados sin repeticiones y ordenadas de forma ascendente.
- Hallar toda la información de las tablas empleado y salario de aquellos empleados que tuvieron actualización de salario de este último entre 2003 y ayer ordenando los resultados según la fecha de contrato de mayor a menor y el nombre alfabéticamente ascendente.
- Obtener un listado de empleados con su nombre, apellidos y cargo. Ordena el listado según el cargo de manera ascendente.
- Acabar de ordenar el listado anterior, por nombre y apellidos de manera ascendente.
- Obtener un listado de los empleados según su salario actual, ordenar de manera descendente.

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- Hacer un ranking con los 3 empleados que más ganan.
- ¿Quiénes están entre el puesto 5 y 10?
- Extraer los 3 últimos empleados contratados.

Consultas de selección básica

SELECT ... FROM ... [WHERE] ... GROUP BY

- Cuando utilizamos la palabra clave **GROUP BY** lo que se consigue es combinar los registros con valores idénticos (en la lista de campos especificados) en un único registro.
- Es necesario incluir en el **SELECT** una función de agregación (COUNT, SUM, AVG), que será la que se ejecute sobre los registros combinados.
- La sintaxis es: **SELECT <campo_agrupación>, <función_agregación> (<campo2>) FROM ... GROUP BY <campo_agrupación>**.
- Se pueden añadir varios campos de agrupación dentro del **GROUP BY**, separados por comas.

Consultas de selección básica

SELECT ... FROM ... [WHERE] ... GROUP BY

ILUSTRACIÓN

Queremos saber cuántos precios se han descargado de cada instrumento. Tendremos que agrupar por instrumento y luego contar los precios.

```
SELECT RefId, COUNT(*) AS NumReg  
FROM TPrices  
GROUP BY RefId
```

Que da como resultado:

	RefId	NumReg
1	1	21
2	2	21
3	3	21
4	4	21
5	5	21
6	6	21
7	7	21

Consultas de selección básica

SELECT ... FROM ... [WHERE] ... GROUP BY ... HAVING

- Si queremos además añadir alguna condición sobre el resultado de la función de agregación, utilizamos la palabra clave **HAVING**.
- La sintaxis es:

SELECT <campo_agrupación>, <función_agregación> (<campo2>)
FROM ...

GROUP BY <campo_agrupación>

HAVING <condición_lógica_sobre_función_de_agregación>

ILUSTRACIÓN

Los instrumentos cuyo precio medio guardado en la base de datos supera las 11 unidades:

SELECT RefId, **AVG(Valor)** **AS** PrecioMedio

FROM TPrices

GROUP BY RefId

HAVING **AVG(Valor)** > 11

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- ¿Cuántos empleados hay de cada sexo?
- ¿Cuántos empleados tiene cada departamento?
- ¿Cuál es el salario medio de cada departamento que tenga, mínimo, dos empleados ?
- ¿Qué nombres de empleado (columna first_name) se repiten más de una vez? ¿Cuántas veces?

Consultas de selección básica

SELECT de SELECT

- Nos permite la posibilidad de hacer una selección del resultado de una consulta.
- En general, cualquier consulta da como resultado una tabla, que puede ser a su vez, consultable.

- La sintaxis es:

```
SELECT campo1_1, campo2_1  
FROM (  
SELECT campo 1 as campo1_1, campo 2 as campo2_1,...  
)
```

- Este tipo de técnica puede ser útil cuando tenemos que realizar consultas de alta complejidad o bien si queremos condensar ciertos núcleos de información.

Consultas de selección básica

SINTAXIS COMPLETA SELECT

SELECT [ALL | DISTINCT | TOP]

<nombre_campo> [, <nombre_campo2>, ...]

FROM

<nombre_tabla> [AS <nombre_vista>]

[**WHERE** <condición> [AND | OR | LIKE | IN | BETWEEN <condición>]]

[**GROUP BY** <nombre_campo> [, <nombre_campo2>, ...]]

[**ORDER BY** <nombre_campo> [ASC | DESC] [, <nombre_campo2> [ASC | DESC], ...]]

Consultas de selección básica

Funciones SQL

- Igual que las funciones de agregación, realizan una determinada operación sobre los valores de una de las columnas.

Función	Operación
UPPER(column)	Convierte el valor de la columna a mayúsculas.
LOWER(column)	Convierte el valor de la columna a minúsculas.
LEN (column)	Devuelve la longitud de la columna (número de caracteres).
ROUND(column, decimals)	Redondea el valor de una columna a los decimales especificados.
GETDATE()	Devuelve el día y la hora del sistema.
ISNULL(column, value)	Convierte el valor de la columna por el parámetro <i>value</i> en el caso de que sea NULL.

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- Crear una consulta que devuelva el nombre de un empleado en mayúscula y el apellido en minúsculas.
- Obtener un listado de todos los empleados cuyo apellido tenga más de 5 caracteres.

Inserción de registros

INSERT INTO

- Cuando queremos insertar un registro en una tabla, utilizamos la sintaxis:

INSERT INTO <nombre_tabla> [(<campo1>, <campo2>, ...)]

VALUES (<valor_campo1>, <valor_campo2>, ...)

ILUSTRACIÓN

INSERT INTO Tinstruments ([RefId], [InstrumentId], [MarketId], [Description], [TipoInstrumento])

VALUES (1, 'REP:SM','MDR Stock','Repsol','Equity')

Insertaría el instrumento **REP:SM**, junto con sus características, en la tabla Tinstruments.

En este caso, hemos rellenado todos los campos. Si se dejara sin especificar alguno, se rellenaría al valor por defecto (normalmente nulo).

Inserción de registros

INSERT INTO

OBSERVACIÓN

- Cuando vamos a insertar un registro, es necesario conocer las relaciones entre las tablas.
- No podemos insertar un valor sin antes haberlo rellenado en la tabla “madre”.
- Tampoco podremos insertar un registro en una tabla con las mismas **PK** que uno ya existente en dicha tabla.
- En caso de que vayamos a indicar los valores para todos los campos existentes en la tabla, no es necesario listarlos antes de la palabra **VALUES**.

INSERT INTO <nombre_tabla>

VALUES (<valor_campo1>, <valor_campo2>, ..., <valor_campoN>)

Inserción de registros

INSERT INTO ... SELECT

- La sentencia **INSERT** permite también insertar varios registros si se combina con un **SELECT**. La lista de selección de la subconsulta debe coincidir con la lista de campos de la instrucción **INSERT** y no quebrantar ninguna relación de la tabla destino.
- Utilizamos la sintaxis:

```
INSERT INTO <nombre_tabla> [(<campo1>, <campo2>, ...)]  
SELECT <campo1>, <campo2>, ... FROM <nombre_tabla_origen>
```

ILUSTRACIÓN

Si queremos rellenar una nueva tabla, TPricesMargin, para que contenga los precios originales más un margen de beneficio del 1%:

```
INSERT INTO TPricesMargin  
(SELECT RefId, Fecha, Valor*1.01 FROM TPrices)
```

EJERCICIO

Vamos a aplicar lo aprendido e insertar algunos datos

- Insertar el departamento de Contabilidad con id = 'd005' en la tabla correspondiente.
- Se sabe que el día 1 de junio de este año se va a proceder al aumento del sueldo en 2000 € del empleado que menos cobra. Aún no se sabe durante cuánto tiempo se le mantendrá el nuevo sueldo. Inserta en la tabla correspondiente el nuevo registro.
- ¿Qué valor tendrá la columna fecha_fin?

Actualización de registros

UPDATE ... SET ...

- Cuando queremos actualizar uno o varios registros utilizamos la sintaxis:

UPDATE <nombre_tabla>

SET <campo1> = <valor1> [, <campo2> = <valor2>, ...]

[WHERE <condiciones>]

- **Cuidado:** Si no se especifican condiciones, se actualizarán **todos** los registros de la tabla.
- En las condiciones se suelen especificar los campos de la **PK** de la tabla, así nos aseguramos de que no cambiamos más registros de los deseados.
- Si alguno de los campos que queremos actualizar forma parte de la clave primaria, sólo se podrá hacer si no se duplica una ya existente.

Actualización de registros

UPDATE ... SET ...

ILUSTRACIÓN

Con la query:

UPDATE Tprices **SET** Valor = 6.7

WHERE RefId = (**SELECT** RefId **FROM** Tstruments **WHERE** InstrumentId = 'REP:SM')

AND Fecha = **convert(datetime, '20/02/2018', 103)**

Obtenemos un cambio en el precio de Repsol a ese día.

- Al igual que antes, si alguno de los campos que queremos actualizar forma parte de una relación entre tablas y quebrantan la integridad referencial, no se llevará a cabo la actualización.
- Gracias a las relaciones creadas, conseguimos que los datos introducidos estén registrados en sus tablas correspondientes.

Actualización de registros

UPDATE ... SELECT ...

- Al igual que con el INSERT, podremos utilizar subconsultas dentro del **UPDATE**. Se pueden usar tanto en el **SET** como en el **WHERE**.
- En el **SET**: para conseguir el nuevo valor de un campo desde otra tabla. Dos formas:
 - **UPDATE** TInstrEquities
SET AmountLastDividend = (**SELECT** AmountLastDividend + 1 **FROM** TInstrEquities **WHERE** InstrumentId = 'TEF:SM' **AND** DateLastDividend = convert(datetime, '23/02/2018', 103) **WHERE** InstrumentId = 'REP:SM')
 - **UPDATE** Table_A
SET Table_A.col1 = Table_B.col1, Table_A.col2 = Table_B.col2
FROM Some_Table AS Table_A **INNER JOIN** Other_Table AS Table_B ON Table_A.id = Table_B.id
WHERE Table_A.col3 = 'cool'

Actualización de registros

UPDATE ... SELECT ...

- En el **WHERE**: para obtener una lista de los registros que queremos cambiar. En el siguiente ejemplo obtenemos la lista de acciones del mercado londinense.
 - **UPDATE** TInstrEquities
SET AmountLastDividend = 1.2
WHERE InstrumentId **IN**
(**SELECT** InstrumentId
FROM Tinstruments
WHERE MarketId = 'STOCKLON' **AND** InstrumentType = 'Equity')
- La subconsulta debe devolver únicamente el campo al que hace referencia el **WHERE**.

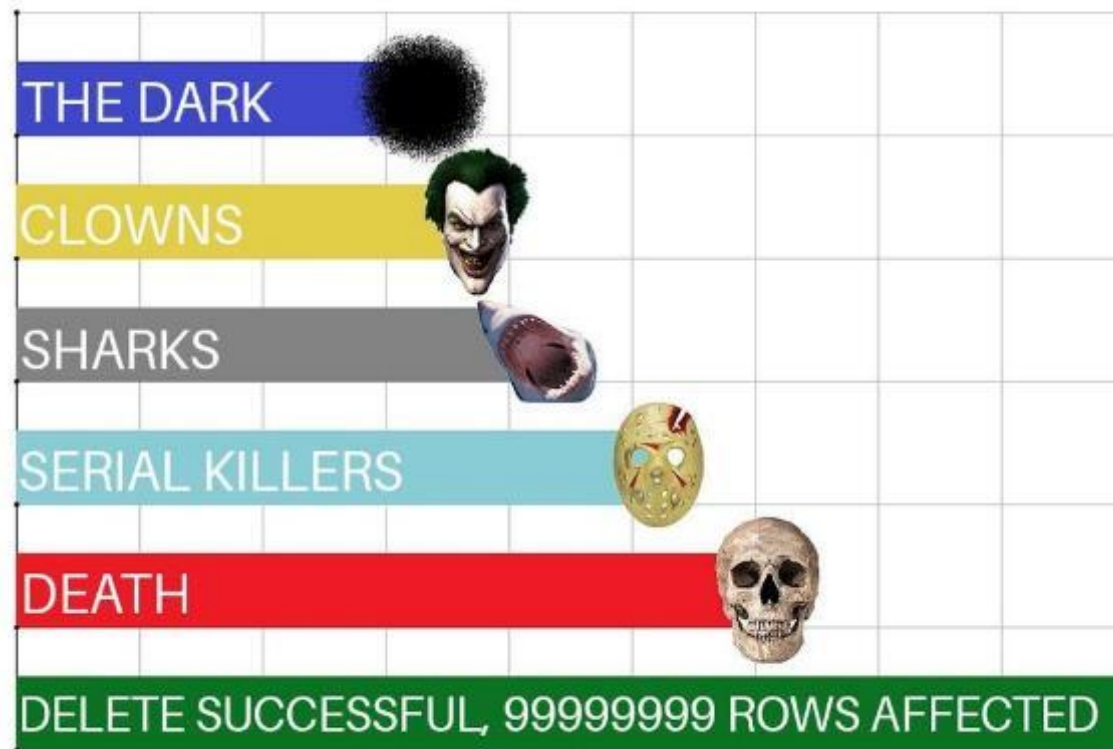
EJERCICIO

Vamos a aplicar lo aprendido y actualizar algunos datos

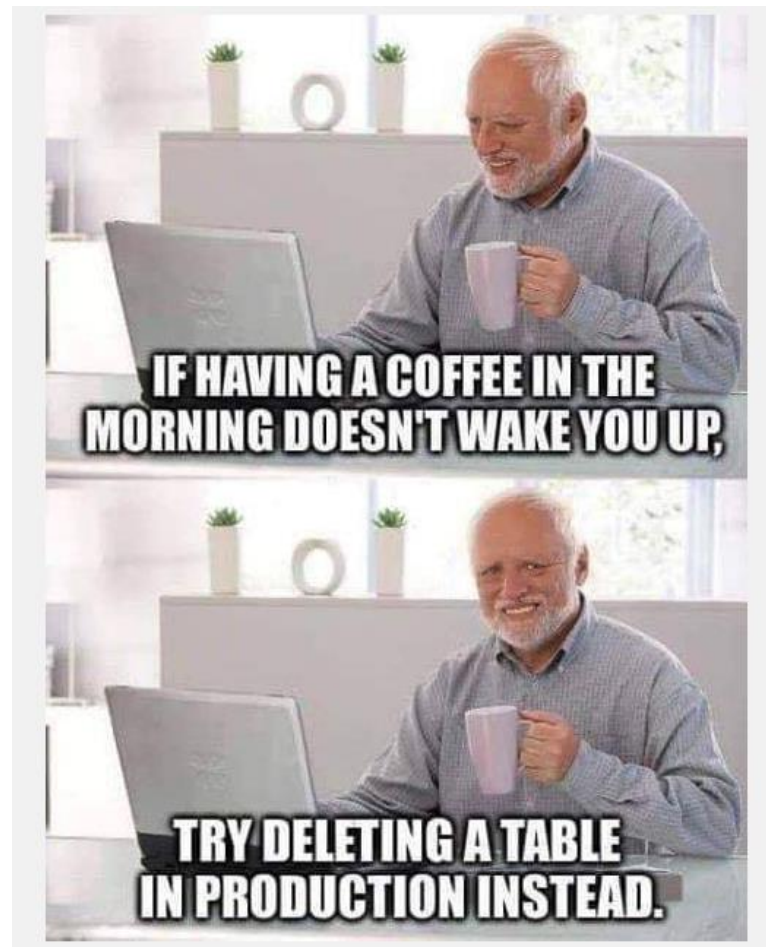
- Se ha decidido que hoy sea el último día de Pilar como responsable del departamento de “Administración”. Actualiza los datos para que se vea reflejado este cambio.
- El departamento de “Compras” pasa a llamarse “Departamento comercial”.
- Al aumentar el sueldo de Julia Rodríguez, no se actualizó la fecha_fin del salario anterior. Realice el cambio.

Borrado de registros

The Scariest Things on Earth



Cuidado con las tablas en producción



Borrado de registros

DELETE FROM ...

- Cuando queremos borrar uno o varios registros, utilizamos la sintaxis:

DELETE FROM <nombre_tabla>
[WHERE <condiciones>]

- **CUIDADO:** Si no se especifican condiciones se borrarán todos los registros de la tabla.
- Solo se pueden borrar registros de una tabla cada vez:

NOTA

Esto no es así en el caso de borrado en cascada. Si al borrar un dato se pierde la integridad referencial, se borran los datos necesarios (**sean de las tablas que sean**) para reestablecer dicha integridad.

- Si se intenta borrar un registro de una tabla que tiene relación con otra, el **DELETE** devolverá el error.

EJERCICIO

Vamos a aplicar lo aprendido y eliminar algunos datos

- Eliminar todos los registros de la tabla responsable_departamento que tengan asignada una fecha_fin.
- Nos hemos dado cuenta que el departamento "Contabilidad" no es necesario. Elimínelo.
- Elimine los empleados que tengan como titulo "Auxiliar administrativo".

Consulta Varias Tablas

Consulta Múltiples tablas

- La información de la BBDD está almacenada en registros, estructurados en tablas.
- Hemos visto operaciones (**SELECT, INSERT INTO, UPDATE**) que afectan a una única tabla. Pero esto no siempre es suficiente.
- En algunos casos se trata únicamente de juntar la información de distintas tablas (consultas **UNION**).
- Otras veces, agruparemos campos de unas y otras tablas cruzando la información (**JOIN**).
- En principio, cualquier número de tablas puede ser utilizado en una misma consulta.
- SQL no es un lenguaje de propósito general. Hay consultas de datos que resultan demasiado complejas o específicas, y es mejor afrontarlas en un lenguaje de propósito general.

Consulta Varias Tablas

Consulta Múltiples tablas - UNION

Se trata de reunir los registros de dos tablas en una sola consulta. Sólo se puede hacer si son tablas compatibles (mismo número de campos y mismo tipo de cada uno de ellos).

UNION estándar

```
SELECT T1.campos FROM T1  
UNION  
SELECT T2.campos FROM T2
```

Se eliminan registros duplicados, lo que supone un método relativamente lento.

UNION ALL

```
SELECT T1.campos FROM T1  
UNION ALL  
SELECT T2.campos FROM T2
```

No eliminan registros duplicados, de forma que la operación es muy rápida.

EJERCICIO

Vamos a aplicar lo aprendido y seleccionar algunos datos

- Juntar las tablas “salario” y “departamento_empleado”, teniendo en cuenta únicamente la columna “id_empleado”.
- Repetir el apartado anterior pero permitiendo duplicados.

Consulta Varias Tablas

Consulta Múltiples tablas – Producto Cartesiano

- En una consulta **SELECT** es posible enumerar varias tablas diferentes:

SELECT * **FROM** Tprices, Tmarkets

(o más genéricamente)

SELECT N1.campos1, N2.campos2 **FROM** T1 as N1, T2 as N2

- Se devuelven todas las combinaciones de registros: cada uno de los dos de la primera tabla con cada uno de los de la segunda (generalizable a **n** tablas).
- **Cuidado:** no confundir con la unión de las tuplas de una y otra tabla.
- Las columnas de la respuesta son la unión de las Tprices y Tmarkets, por tanto, en número, la suma.

Consulta Varias Tablas

Consulta Múltiples tablas – INNER JOIN

- El **INNER JOIN** cruza dos tablas haciendo el producto cartesiano y dejando solamente los registros en los que se cumple cierta condición.

SELECT * **FROM** Tinstruments I **INNER JOIN** Tprices P **ON** I.InstrumentId = P.InstrumentId **AND** I.InstrumentType = 'Equity'

- Esta consulta es eficiente y la forma general es:

SELECT T1.Campos, T2.Campos **FROM** T1 **INNER JOIN** T2 **ON** condicion

- La condición normalmente atañe a campos de T1 y T2. Para considerarse cierta, deben estar definidos todos los campos implicados (sin valores NULL) y tener valores que verifiquen la condición.

EJERCICIO

Vamos a aplicar lo aprendido y hacer algún join

- Crear una consulta que devuelva el nombre y apellidos de un empleado junto con su salario actual.
- Añadir al resultado de la consulta anterior el nombre del departamento del empleado.

Consulta Varias Tablas

Consulta Múltiples tablas – LEFT JOIN

- En ocasiones, queremos conservar todos los registros de una tabla y completar la información con valores correspondientes de otra tabla, si es que los hay.
- Si no los hay, esa información debe rellenarse con NULL.
- A diferencia del caso del **INNER JOIN**, el papel de las dos tablas es **asimétrico**: presenta cada registro de la primera tabla, posiblemente complementado con información de la segunda.
- La sintaxis general es:

SELECT T1.Campos, T2.Campos **FROM** T1 **LEFT JOIN** T2 **ON** condición

EJERCICIO

Vamos a aplicar lo aprendido y hacer algún join

- Crear una consulta que devuelva el código de un empleado y el código del departamento del que el empleado es responsable. La consulta debe devolver **todos** los empleados.
- Modificar la consulta anterior para quedarte únicamente con aquellos empleados que **no** son responsables.

Consulta Varias Tablas

Consulta Múltiples tablas – RIGHT JOIN

- El **RIGHT JOIN** es la idea dual del **LEFT JOIN**: toma todos los registros de la segunda tabla, complementando con la información que pueda de la primera tabla.
- Por tanto, los registros que se obtienen con un **RIGHT JOIN** son los que produce el **INNER JOIN** análogo, junto con los registros de la segunda tabla que no aparecieron en él, rellenos los valores correspondientes a la primera tabla con NULL.
- La sintaxis general es:

SELECT T1.Campos, T2.Campos **FROM** T1 **RIGHT JOIN** T2 **ON** condición

- Es equivalente a hacer un **LEFT JOIN** con las tablas en orden invertido.

EJERCICIO

Vamos a aplicar lo aprendido y hacer algún join

- Cambiar la consulta anterior para que funcione con un RIGHT JOIN.

Consulta Varias Tablas

Consulta Múltiples tablas – FULL JOIN

- Puede pensarse en ella en términos de unión de un **LEFT JOIN** y un **RIGHT JOIN**.
- Devuelve:
 - Los registros de **INNER JOIN** análogo.
 - Los registros de la primera tabla que no han sido incluidos en el **INNER JOIN**, con valores NULL para la segunda tabla.
 - Los registros de la segunda tabla que no han sido incluidos en el **INNER JOIN**, con valores NULL para la primera tabla.
- Por tanto, el rol de las tablas vuelve a ser simétrico.
- La sintaxis general es:

SELECT T1.Campos, T2.Campos **FROM** T1 **FULL JOIN** T2 **ON** condición



8. Otras acciones

Otras acciones

Transacciones

Mediante estas sentencias SQL podemos **crear una transacción** (operación ACID) sobre la información de la base de datos. Al finalizar la modificación de información, podemos confirmar los cambios o descartarlos.

BEGIN TRANSACTION;

<insert | update | delete statements>;

COMMIT | ROLLBACK;

- **BEGIN TRANSACTION:** comienza la transacción en base de datos.
- **COMMIT:** finaliza la transacción confirmando los cambios.
- **ROLLBACK:** finaliza la transacción descartando los cambios.

Otras acciones

Autocommit

Por defecto, SQL Server se ejecuta con el **modo autocommit habilitado**. Esto quiere decir, que tan pronto se ejecuta una actualización de información en base de datos, los cambios se convierten en permanentes. No se puede dar marcha atrás a los cambios. Se puede cambiar este modo de funcionamiento con la siguiente sentencia.

SET IMPLICIT_TRANSACTIONS ON;

Una vez ejecutada esa sentencia, después de cada acción habrá que ejecutar:

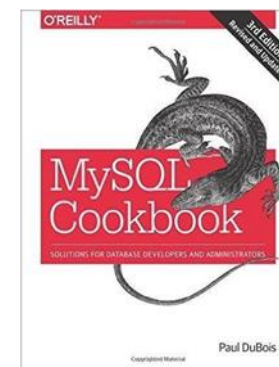
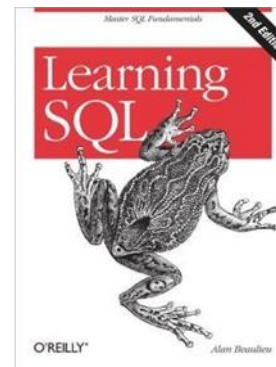
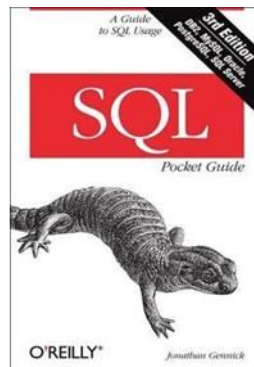
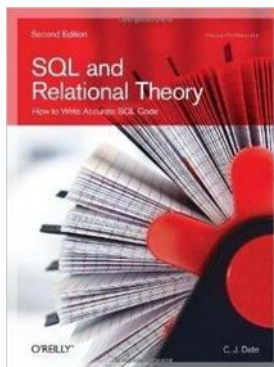
- **ROLLBACK TRANSACTION:** Para descartar los cambios.
- **COMMIT TRANSACTION:** Para conservar los cambios.



9. Materiales

Materiales

- Try SQL: <https://www.codeschool.com/courses/try-sql>
- SQL Tutorial: <http://www.w3schools.com/sql/>
- SQL and Relational Theory
- SQL Pocket Guide
- Learning SQL
- MySQL Cookbook





Afi Escuela

© 2023 Afi Escuela. Todos los derechos reservados.