1. EVOLUCIÓN DE LOS LENGUAJES DE PROGRAMACIÓN

- **Lenguajes de 1ª Generación**: lenguajes en los que se programa directamente en código máquina, esto es en binario (0,1), con la dificultad y falta de potencia que ello supone.

- **Lenguajes de 2ª Generación**: en este grupo se incluye lo que usualmente suele denominarse como lenguaje ensamblador. Este lenguaje permite utilizar códigos alfabéticos mnemotécnicos para representar las instrucciones primarias, cosa que hace más sencilla la programación. Todo programa escrito en lenguaje ensamblador necesita que se le haga un proceso de compilación para traducir estas instrucciones a código binario que la máquina pueda ejecutar.

- **Lenguajes de 3ª Generación**: denominados procedimentales o procedurales ("Lenguajes de alto nivel"), ejecutan sus sentencias en un orden concreto especificado en el programa. Esos lenguajes tienen una estructura lógica más natural, en el sentido de que su sintaxis está más próxima al lenguaje oral humano. Algunos lenguajes de programación como C, C++, C#, Visual Basic, Ada, Fortrand, Cobol, Java y PHP, pertenecen a esta categoría. Su característica principal es que, en general, el código resulta independiente de la plataforma.

- **Lenguajes de 4ª Generación**: se suelen denominar como no procedimentales o procedurales, se alejan del concepto "procedimiento". A este tipo pertenece el SQL, se especifican los resultados que se quieren obtener, antes que cómo hacerlo. Se apoyan en unas herramientas de más alto nivel denominadas herramientas de cuarta generación. El usuario no debe definir los pasos a seguir en un programa para realizar una determinada tarea, tan sólo debe definir una serie de parámetros que estas herramientas utilizarán para generar un programa de aplicación. Esto a veces limita el tipo de problemas que pueden resolver. Los 4GL incluyen:
  - Lenguajes de presentación, de consultas y generadores de informes.
  - Lenguajes especializados, como lenguajes de bases de datos.
  - Generadores de gráficos a partir de datos de una Base de datos.
  - Generadores de aplicaciones que definen, insertan, actualizan y obtienen datos de la base de datos, con lenguajes de muy alto nivel que se utilizan para generar el código de la aplicación.

- **Lenguajes de 5ª Generación**: en ocasiones se llama así a los lenguajes de inteligencia artificial, aunque esta denominación ha caído en desuso.

SQL suele ser considerado un lenguaje de cuarta generación, pues al gestor de la base de datos se le reclaman los resultados que se quieren obtener y no el modo en como dicho gestor debe conseguirlos.

Muchas veces se hace coexistir y cooperar a SQL, con lenguajes de tercera generación, recordemos que existen algunas diferencias entre ambos:

- **SQL es un lenguaje con débil presencia de tipos (Weakly Typed).**

  En las sentencias de SQL normalmente no se especifican tipos de datos. Los tipos se encuentran intrínsecamente presentes en la definición de las columnas. Esta característica lo diferencia enormemente de con respecto a los lenguajes de tercera generación, cuyo chequeo de tipos es una de sus características principales.

- **SQL es un lenguaje orientado a conjuntos**.

  Cuando realizamos una consulta de datos mediante SQL esperamos como resultado un conjunto de datos homogéneo, sin que conozcamos quizás su estructura y volumen fijo de elementos. Esto hace que los resultados sean con frecuencia difícilmente previsibles, cosa que no sucede con los lenguajes de programación de tercera generación, lo que nos obligará en ocasiones a implementar mecanismos de programación adicionales para solventar esta disconformidad.

2. ANSI-SQL

Si bien el uso principal de ANSI-SQL es la realización de consultas y actualizaciones sobre datos almacenados en bases relacionales, el lenguaje también permite realizar otras tareas. Una clasificación de estas tareas permite hacer lo propio con el lenguaje ANSI-SQL en los subconjuntos siguientes:

- **DQL (Data Query Language)**: Es un sublenguaje que contiene todas las sentencias de SQL que permite obtener datos de las tablas y especificar la forma en que deseamos que se presenten.(select)

- **DML (Data Manipulation Language)**: Es un sublenguaje con sentencias para la inserción, modificación y borrado de datos. (insert, update, delete)

- **DDL (Data Definition Language)**: Es un subconjunto de sentencias que permite definir nuevos objetos, borrarlos, poner restricciones a los campos de las tablas, y establecer relaciones entre tablas, etc. El sistema genera objetos propios para el mantenimiento de las estructuras, estos pueden ser configurables. (create, drop, not null, check, constraint, primary key, …) Una tabla debe ser definida antes de poder usarla. Otros muchos objetos son manejados por el SGBD, espacios físicos, índices…

- **DCL (Data Control Language)**: Es un subconjunto de sentencias que permite controlar diversos aspectos de los datos, como la confidencialidad, conceder o denegar autorizaciones, permisos…(grant, revoke)

- **TPL (Transaction Procesing Language)**: Conjunto de instrucciones que permite validar o denegar una seria de sentencias DML, para que sean ejecutadas de manera coherente manteniendo la consistencia e integridad de la BBDD. (commit, rollback)

- **CCL (Cursor Control Language)**: Este sublenguaje permite operar sobre filas individuales de una tabla resultado de una consulta cuando consta de varios registros o filas de información. (declare cursor, fetch)

Existen diferentes ámbitos de trabajo y maneras en las que en la práctica podemos utilizar sentencias SQL. Métodos de uso:

- **Usos estáticos**
  - **SQL interactivo:** Consiste en indicar las sentencias SQL y obtener los resultados todos dentro de un entorno similar a una línea de comandos o en una pantalla visual. (sqlcmd, iSQL, SQLPlus; SQL Server Management Studio (SSMS).
  - **SQL inmerso en programas (Embedded SQL):** Permite ejecutar sentencias SQL en el interior de programas escritos en otros lenguajes, como por ejemplo, C, Java, etc., anteponiendo a las sentencias SQL una cláusula que la delimite. Para interpretar estos delimitadores y sustituir las sentencias suele ser preciso un precompilador de SQL que será específico al gestor de base de datos y al lenguaje que se trate.
  - **SQL modular:** Consiste en permitir compilar sentencias SQL de manera separada a las del propio código del lenguaje de programación y enlazarlas (link) después con el resto de módulos objeto para crear el programa ejecutable. Los módulos SQL pueden incluir declaraciones de variable, de tablas, que contengan resultados de las consultas, etc…
- **Usos dinámicos**
  - **SQL dinámico:** Se utiliza para crear sentencias SQL que no resultan predecibles en el instante en el que se escriben. Un ejemplo sería un programa que permitiese a un usuario escribir una sentencia SQL y enviarla al gestor.

3. Historia del SQL

SQL (**Structured Query Language**) es el lenguaje de computación más ampliamente usado para crear, modificar y consultar bases de datos relacionales.

Sus orígenes parten de un documento ("A Relational Model of Data for Large Shared Data Banks") del **Dr. Edgar F. Codd**, publicado en Junio de **1970** por la revista ACM (Association of Computing Machinery).

El modelo de Codd llegó a ser ampliamente aceptado como el modelo definitivo para Sistemas Gestores de Bases de Datos Relacionales (SGBDR).

IBM Research desarrolló un lenguaje llamado SEQUEL (Structured English Query Language) como interfaz para un sistema experimental de Bases de Datos relacional llamado SYSTEM R para aplicar el modelo de Codd. SEQUEL se convirtió posteriormente en SQL.

En 1979, Relational Software, Inc. (ahora Oracle Corporation) presentó la primera implementación comercial de SQL, y pronto, muchos vendedores desarrollaron dialectos del mismo.

SQL fue adoptado como un estándar por ANSI (American National Standards Institute) en 1986 y por ISO en 1987.

Los estándares de SQL han pasado por varias versiones:

Año	Nombre	Alias	Descripción
1986	SQL-86	SQL1	Primera publicación hecha por ANSI. Confirmada por ISO en 1987.
1989	SQL-89		Revisión menor
1992	SQL-92	SQL2	Revisión mayor
1999	SQL:1999	SQL3	Se añaden: expresiones regulares para concordancia de patrones, consultas recursivas, triggers, tipos no escalares y algunas características orientadas a objetos (las dos últimas han sido controvertidas y no son soportadas ampliamente).
2003	SQL:2003		Añade: características XML, secuencias estandarizadas y columnas con valores auto-generados.
2006	SQL:2006		Evolución en el desarrollo web. 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	SQL: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. (ISO/IEC 9075-1:2008)
2011	SQL:2011		Soporte mejorado para bases de datos temporales (PERIOD FOR). (ISO/IEC 9075:2011)
2016	SQL:2016		Permite búsqueda de patrones, funciones de tabla polimórficas y compatibilidad con los ficheros JSON. (ISO/IEC 9075:2016)

Aunque SQL está definido por ANSI e ISO, hay muchas extensiones y variaciones sobre la versión estándar. Muchas de estas extensiones son propietarias como por ejemplo PL/SQL (de Oracle Corporation) o Transact-SQL (de Sybase o Microsoft). Por tanto el código escrito en SQL raramente es portado entre sistemas de bases de datos sin tener que hacer importantes modificaciones.

SQL, en el modo interactivo, utiliza una sintaxis de "formato libre", dado que se puede escribir en cualquier punto de la pantalla, y los espacios, tabuladores, etc… no son tenidos en cuenta. Este modo hace referencia a sentencias de *SQL* que se envían al motor de SQL utilizando un procesador de línea de mandatos o utilizando herramientas de consulta.

4. Notación BNF/EBNF (Extended Backus-Naur Form)

Vamos a utilizar la notación **EBNF** para ver la sintaxis de las sentencias de SQL.

EBNF es una meta sintaxis usada para expresar gramáticas libres de contexto, es decir, una manera formal de describir lenguajes formales. Nosotros utilizaremos sólo algunos símbolos para representar las sentencias de forma global. Simbología:

- **[ ]** → Opcionalidad. Se repite 0 o 1 veces
- **{** } → Opcionalidad. Se repite entre 0 y muchas veces.
- **|** → Opcionalidad. Una de las dos opciones (X | Y). Es un OR
- **<elemento>** → define un objeto del SGBD (base de datos, tabla, columna, etc…)

EJEMPLO:

SELECT <nomb_columna> {, <nomb_columna>}

FROM <nombre_tabla> {, <nombre_tabla>}

5. TIPO DE DATOS

- Numéricos:
  - Enteros:
    1. **bit**(valores 1, 0 o NULL.)
    2. **int**(-2^31.. 2^31=2.147.483.647)
    3. **smallint**(-32.768..32.767)
    4. **tinyint**(0..255)

  - Coma fija:
    1. **decimal[** (***p*[** ,***s*]**)**]**
    2. **numeric[** (***p*[** ,***s*]**)**] (sinónimo)**
       p: precisión, nº total de dígitos; **s**: escala, nº de dígitos decimales.

    Coma flotante: float(-1.79E+308.. 1.79E+308), real(-3.40E+308..-3.40E+308)

- Carácter:

  De Longitud fija: CHAR [(tamaño)]) tamaño(1..8000 caracteres)

  De longitud variable: VARCHAR [(tamaño)], TEXT (hasta 2.147.483.647)

  *Varchar: Se utilizará solo cuando los datos no se prevea que varíen.
  *Text: Se utiliza para almacenar grandes cantidades de texto.

- Fechas y horas:

  DATETIME, SMALLDATETIME

ENLACES A TIPOS DE DATOS EN TRANSACT-SQL DE MS SQL SERVER 2017

- https://docs.microsoft.com/es-es/sql/t-sql/data-types/int-bigint-smallint-and-tinyint-transact-sql?view=sql-server-2017
- https://docs.microsoft.com/es-es/sql/t-sql/data-types/decimal-and-numeric-transact-sql?view=sql-server-2017
- https://docs.microsoft.com/es-es/sql/t-sql/data-types/float-and-real-transact-sql?view=sql-server-2017
- https://docs.microsoft.com/es-es/sql/t-sql/data-types/char-and-varchar-transact-sql?view=sql-server-2017
- https://docs.microsoft.com/es-es/sql/t-sql/data-types/ntext-text-and-image-transact-sql?view=sql-server-2017

6. Sentencias SQL

• DDL (DATA DEFINITION LANGUAGE)

- **Crear base de datos**

  CREATE DATABASE <nombre_bbdd>

- **Crear tablas**

  CREATE TABLE <nombre_tabla>

  (<nombre_columna> <tipodedato> [NOT NULL | NULL]

  {, <nombre_columna> <tipodedato> [NOT NULL | NULL]})

  * Por defecto toma NULL

- **Añadir columnas**

  ALTER TABLE <nombre_tabla>

  ADD <nombre_columna> <tipodedato> [NOT NULL | NULL]

  {, <nombre_columna> <tipodedato> [NOT NULL | NULL]}

- **Crear índices**: que permiten recuperar datos de la base de datos muy rápidamente. Los usuarios no pueden ver los índices, solo se utilizan para acelerar las búsquedas / consultas.

  CREATE [UNIQUE] [CLUSTERED | NON CLUSTERED] INDEX <nombre_indice>

  ON <nombre_tabla>

  (<nombre_columna> [ASC | DESC]

  {, <nombre_columna> [ASC | DESC]})

  *UNIQUE: Si se pone no permite valores duplicados

  *CLUSTERED: Obligar o no a almacenar según la ordenación impuesta por el índice

- **Creación de vistas**: tablas virtuales, con filas y columnas, basadas en resultados de una instrucción SQL. Los campos en una vista son campos de una o más tablas reales en la base de datos. Puede agregar funciones SQL, instrucciones WHERE, etc., a una vista y presentar los datos como si los datos procedieran de una sola tabla.

  CREATE VIEW <nombre_vista>

  [(<nombre_columna_vista> {, <nombre_columna_vista>})]

  AS <sentencia_SELECT>

- **Borrado de objetos**

  DROP DATABASE <nombre_base_datos>

  DROP TABLE <nombre_tabla>

  DROP INDEX <nombre_tabla> . <nombre_indice>

  DROP VIEW <nombre_vista>

- **DML (DATA MANIPULATION LANGUAGE)**

  - **Insertar datos en las tablas (filas)**

    INSERT INTO <nombre_tabla>

    [(<nombre_columna> {, <nombre_columna>})]

    VALUES (<dato> {, <dato>}) | <sentencia_SELECT>

  - **Modificar datos en las tablas (filas)**

    UPDATE <nombre_tabla>

    SET <nombre_columna> = <nuevo_valor>

    {, <nombre_columna> = <nuevo_valor>}

    [WHERE <condición> | partícula_where]

  - **Eliminar datos en las tablas (filas)**

    DELETE FROM <nombre_tabla>

    [WHERE <condición> | partícula_where]

- **Creación de consultas de datos**

SELECT [ALL | DISTINCT] <nomb_columna> {, <nomb_columna>} | * | TOP <valor>

FROM <nombre_tabla> {, <nombre_tabla>}

[WHERE <condición> | partícula_where]

[GROUP BY <nombre_columna> {, <nombre_columna>}]

[HAVING <condicion_simple_o_compuesta>]

[ORDER BY <nombre_columna> [ASC | DESC] {, <nombre_columna> [ASC | DESC]} | <ordinal> [ASC | DESC] {, <ordinal> [ASC | DESC]}]

1. **WHERE**: Sirve para restringir las filas de la tabla que se va a visualizar. Utilizaremos condiciones para que la consulta nos devuelva solo aquellas filas que cumplen la condición. (Operadores: <> = …(simple) and not or (compuesta))
[WHERE <condición>] → simple <nom_col> <operador> <nom_col2>

   [WHERE <condición>] → compuesta <c_simple> <operador> <c_simple2>

   - **Partícula IN**
     Sirve para establecer una relación de permanencia entre un campo y un conjunto de valores.

     WHERE <nombre_columna> [NOT] IN (<valor> {, <valor>})

   - **Partícula BETWEEN**
     Sirve para expresar un rango de valores (incluidos los de la expresión)

     WHERE <nombre_columna> [NOT] BETWEEN <valor1> AND <valor2>

   - **Partícula LIKE**
     Determina si una cadena de caracteres específica coincide con un patrón especificado, que puede contener caracteres normales y caracteres comodín (meta símbolos).

     WHERE <nombre_columna> [NOT] LIKE 'Expresión_con_metasimbolos'

METASÍMBOLOS

%: Cualquier cadena de cero o más caracteres.

Ej: WHERE title LIKE '%computer%' Títulos de libros que contengan la palabra 'computer' en el título.

_: Un único carácter.

Ej.: WHERE name LIKE '_ean' busca todos los nombres de cuatro letras que terminen en 'ean' (Dean, Sean, etc.)

[ ]: Cualquier carácter individual del intervalo ([a-f]) o del conjunto ([abcdef]) que se ha especificado.

Ej.: WHERE name LIKE '[C-P]arsen' busca apellidos de autores que terminen en arsen y empiecen por cualquier carácter individual entre C y P, como Carsen, Larsen, Karsen, etc. En las búsquedas de intervalos, los caracteres incluidos en el intervalo pueden variar, dependiendo de las reglas de ordenación de la intercalación.

[^]: Cualquier carácter individual que no se encuentre en el intervalo ([^a-f]) o el conjunto ([^abcdef]) que se ha especificado.

Ej.: WHERE name LIKE 'ma[^r]%' busca todos los apellidos de autores que empiecen por 'ma' y en los que la siguiente letra no sea 'r'.

- **Partículas ANY y ALL**

  Se usan en la cláusula WHERE, al utilizar operadores relacionales, para que no falle al comparar con varios valores.

  - **ANY**: Se cumple cuando coincide con alguno de los valores que devuelve una subconsulta (Ver apartado SUBCONSULTAS).

  - **ALL**: Se cumple cuando coincide con TODOS los valores que devuelve una subconsulta (Ver apartado SUBCONSULTAS).

2. GROUP BY

Sirve para agrupar filas con el valor de una columna en común.

[GROUP BY <nombre_columna> {, <nombre_columna>}]

3. HAVING

Sirve para discriminar las agrupaciones del GROUP BY mediante una condición.

[HAVING <condicion_simple_o_compuesta>]

4. ORDER BY

Sirve para ordenar las columnas.

[ORDER BY <nombre_columna> [ASC | DESC] {, <nombre_columna> [ASC | DESC]} | <ordinal> [ASC | DESC] {, <ordinal> [ASC | DESC]}]

OPERACIONES ENTRE CONSULTAS

- **UNION** (Operador **unión** de Álgebra Relacional)
  Sirve para recuperar mediante una sola consulta la información de varios SELECT, que tengan tipos de datos equivalentes y el mismo número de columnas.

  <sentencia_SELECT> UNION [ALL] <sentencia_SELECT>

  { UNION [ALL] <sentencia_SELECT>}

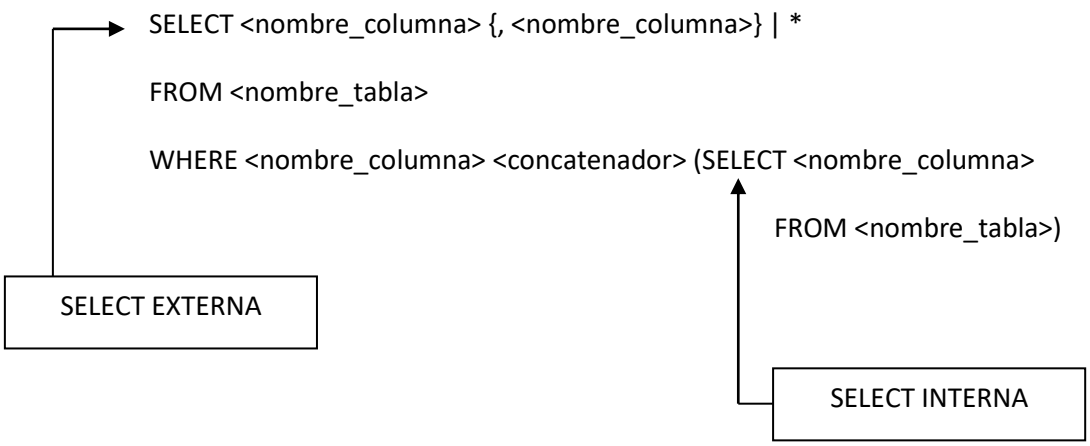
  [ORDER BY <nombre_columna> [ASC | DESC] {, <nombre_columna> [ASC | DESC]} | <ordinal> [ASC | DESC] {, <ordinal> [ASC | DESC]}]

- **EXCEPT** (Operador ***diferencia*** de Álgebra Relacional)
  Sirve para recuperar mediante una sola consulta la información diferente entre dos sentencias SELECT; deben tener tipos de datos equivalentes y el mismo número de columnas.

  <sentencia_SELECT> EXCEPT <sentencia_SELECT>

SUBCONSULTAS

Sirven para recuperar información de una tabla a partir de otra tabla. Es una sentencia SELECT que forma parte del WHERE de otra sentencia SELECT. Lo primero en ejecutarse es la SELECT interna.

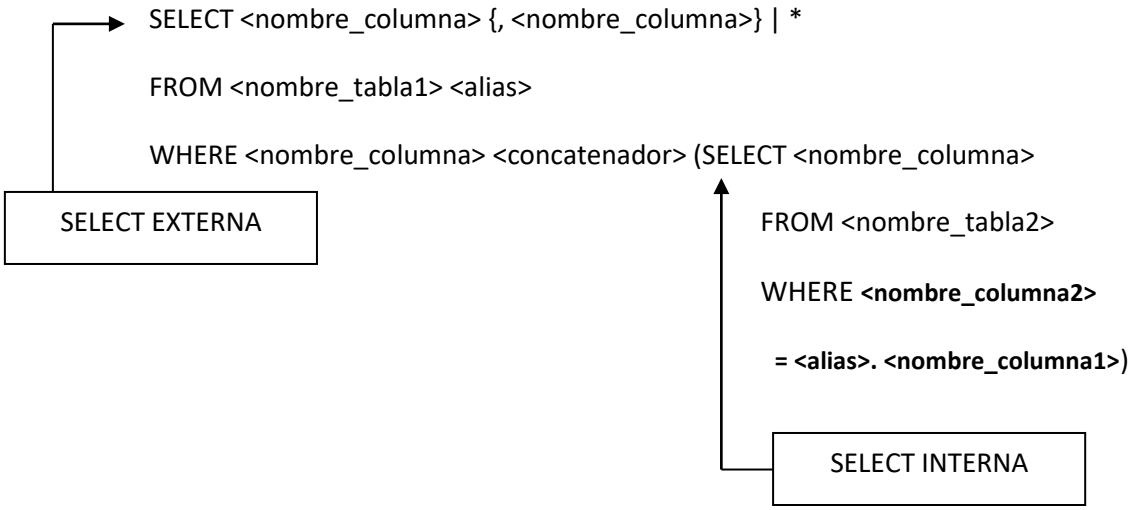


** Concatenador: Cualquier operador relacional o partícula valida de WHERE

SUBCONSULTAS CORRELACIONADAS

Son un tipo especial de subconsultas en la que **la SELECT interna se ejecuta 1 vez por cada fila candidata a ser mostrada en el resultado final de consulta en la SELECT externa**.

A veces son necesarias cuando en una sentencia SQL se necesitan condiciones complejas para seleccionar conjuntos específicos de datos de las tablas.



La consulta interna realiza una función de agregado, tal como una estadística y alimenta esta información a la consulta externa, que la utiliza como la base de una comparación.

Si las tablas son grandes, lleva más tiempo una correlacionada que una normal.

EJEMPLO 1: Vuelos que despegan más pronto para cada origen. (Sin GROUP BY)

(Con GROUP BY sería…: select origen, min(hora_salida) from vuelos group by origen)

```
select num_vuelo, origen, hora_salida
from vuelos v
where hora_salida = (select min(hora_salida)
                     from vuelos
                     where origen = v.origen)
```

EJEMPLO 2: Recuperar las reservas cuyo número de plazas libres es mayor que la media para ese mismo vuelo.

```
select *
from reservas a
where plazas_libres > (select avg(plazas_libres)
                       from reservas
                       where num_vuelo = a.num_vuelo)
```

SUBCONSULTAS CON EXISTS

Sirve para indicar si existe o no alguna fila que cumpla las condiciones de una subconsulta dentro del WHERE.

WHERE [NOT] EXISTS (<subconsulta>)

SUBCONSULTAS CON JOIN

Sirve para crear una macro tabla a través de dos tablas que tengan una o más columnas en común.

SELECT * FROM <nombre_tabla1> [alias] {, < nombre_tablaX> [alias]}

WHERE <nombre_tabla1>.<nombre_columna1> =

<nomObre_tablaX>.<nombre_columnaX>

{ [AND|OR]

<nombre_tabla1>.<nombre_columna1> =

<nomObre_tablaX>.<nombre_columnaX> }

ALGORITMOS DE FUNCIONAMIENTO:

- **CONSULTA BÁSICAS CON SELECT**

1. Where → Genera una tabla temporal con las filas que cumplen la condición
2. Group by → Genera tablas temporales, una por cada subgrupo
3. F(x) → Se aplican las funciones de columnas a cada subgrupo
4. Having → Elimina grupos que no queremos visualizar mediante la condición.

- **SUBCONSULTAS CORRELACIONADAS**

1. Se selecciona la primera fila de la tabla externa que cumple la condición de la WHERE.
2. Se ejecuta la SELECT interna.
3. Si la condición de la WHERE externa se cumple, la fila se visualizará.
4. Si existen más filas candidatas en la tabla externa se repite desde el primer punto.

- **DCL (DATA CONTROL LANGUAGE)**

  Es un lenguaje proporcionado por el Sistema de Gestión de Base de Datos que incluye una serie de comandos SQL que permiten al administrador controlar el acceso a los datos contenidos en la Base de Datos.

  - **Conceder autorizaciones a los usuarios**

    GRANT <operaciones> ON [TABLE] <nombre_tabla>

    TO <usuario> [WITH GRANT OPTIONS]

  - **Eliminar autorizaciones concedidas**

    REVOKE <operaciones> ON [TABLE] <nombre_tabla>

    FROM <usuario> [CASCADE]

    ** Operaciones: ALL | ALTER | DELETE | INSERT | SELECT | INDEX | UPDATE [<nombre_columna> {, <nombre_columna>}]

    ** Usuario: PUBLIC | <nombre_usuario> {, <nombre_usuario>}

AYUDA SOBRE LAS CONVENCIONES DE SINTAXIS DE TRANSACT-SQL (MS-SQL-SERVER 2017)

https://docs.microsoft.com/es-es/sql/t-sql/language-elements/transact-sql-syntax-conventions-transact-sql?view=sql-server-2017

7. Expresiones Aritméticas

Pueden aparecer:

- Detrás de la sentencia SELECT, operando esa expresión con alguna columna.

- Detrás de la sentencia WHERE, operando esa expresión con alguna columna.

8. Funciones predefinidas del SGBD

Son funciones ya creadas que nos proporcionan una ayuda. Pueden variar según el SGBD.
https://docs.microsoft.com/es-es/sql/t-sql/functions/functions?view=sql-server-2017

De manera general a los SGBD existen diferentes tipos de funciones, algunos son:

- **FUNCIONES DE COLUMNA O DE AGREGADO:** necesitan de una columna para poder operar. Realizan un cálculo sobre un conjunto de valores y devuelven un solo valor. Son comunes a cualquier sistema gestor de base de datos. Se pueden usar en la lista de selección o en la cláusula HAVING de una instrucción SELECT, también en combinación con la cláusula GROUP BY para calcular la agregación en las categorías de filas. No se utilizan en la cláusula WHERE.

    - **Min:** Devuelve el valor mínimo de la columna.
      SELECT MIN (<nombre_columna>) …

    - **Max:** Devuelve el valor máximo de la columna.
      SELECT MAX (<nombre_columna>) …

    - **Count:** Devuelve el número de filas de una columna.
      SELECT COUNT ([DISTINTC] <nombre_columna> | *) …

    - **Sum:** Devuelve la suma de todos los valores de esa columna
      SELECT SUM (<nombre_columna>) …

    - **AVG:** Devuelve la media aritmética de la columna
      SELECT AVG (<nombre_columna>) …

- **FUNCIONES ESCALARES**: suelen operar sobre un valor y después devuelven otro valor.

    - **Manejo de fechas y horas (GETDATE (), DATEPART (), MONTH (), etc.):**

        1. **GETDATE():** Devuelve un valor *datetime* que contiene la fecha y hora del equipo en el que se ejecuta la instancia de SQL Server.
        2. **DATEPART (datepart, date):** Devuelve un entero que representa el parámetro *datepart* especificado del parámetro *date* especificado.
        3. **MONTH (*date*):** Devuelve un entero que representa la parte del mes del parámetro *date* especificado.

      https://docs.microsoft.com/es-es/sql/t-sql/functions/date-and-time-data-types-and-functions-transact-sql?view=sql-server-2017#DateandTimeFunctions

- **Manejo de cadenas, tratamiento de caracteres**: funciones que necesitan de una columna para poder operar.
    1. **LOWER:** Devuelve la cadena en minúsculas
       LOWER (<cadena>)
    2. **UPPER:** Devuelve la cadena en mayúsculas
       UPPER (<cadena>)
    3. **SUBSTRING:** Devuelve una subcadena
       SUBSTRING (cadena, inicio, length)
    4. **LEN:** Devuelve el número de caracteres de la expresión de cadena especificada, excluidos los espacios en blanco finales.
       LEN (<cadena>)
    5. **REVERSE:** Devuelve la cadena invertida
       REVERSE (<cadena>)
    6. **LEFT:** Devuelve la parte izquierda de una cadena de caracteres con el número de caracteres especificado.
       LEFT (<cadena>, nº)
    7. **RIGHT:** Devuelve la parte derecha de una cadena de caracteres con el número de caracteres especificado.
       RIGHT (<cadena>, nº)

https://docs.microsoft.com/es-es/sql/t-sql/functions/string-functions-transact-sql?view=sql-server-2017

- **Conversión de tipos (CAST, CONVERT)**: convierten una expresión de un tipo de datos a otro.
https://docs.microsoft.com/es-es/sql/t-sql/functions/cast-and-convert-transact-sql?view=sql-server-2017#BKMK_examples

- **Otras** (específicas de cada sistema gestor de base de datos)