Clase Nro. 11: Introducción a SQL (I)

Presentada por:
Amadís Martínez

Departamento de Computación, FaCyT
Versión: semestre 1-2025

Historia de SQL

- IBM desarrolló Sequel (la versión original de SQL), como parte del proyecto System R, a principios de la década de los 1970.
- El lenguaje Sequel evolucionó y pasó a denominarse SQL (*Structured Query Language*).
- SQL se ha establecido como el lenguaje estándar de bases de datos relacionales.
- SQL ha sido estandarizado desde 1986 por la *International Organization for Standardization* (ISO) y el *American National Standards Institute* (ANSI).

Historia de SQL

- Estándares de SQL:
  - SQL-86
  - SQL-89
  - SQL-92: versión soportada por la mayoría de los SGBDR.
  - SQL:1999
  - SQL:2003

Historia de SQL

- Estándares de SQL:
  - SQL:2006
  - SQL:2008
  - SQL:2011
  - SQL:2016
  - SQL:2023

Partes de SQL

- Lenguaje para la definición de datos: proporciona comandos para definir, eliminar y modificar esquemas de relaciones.
- Lenguaje para la manipulación de datos: permite consultar la información de la base de datos e insertar tuplas en ella, eliminarlas y modificarlas.
- Integridad: incluye comandos para especificar las restricciones de integridad.
- Definición de vistas: incluye comandos para definir las vistas de datos.

Partes de SQL

- Control de transacciones: incluye comandos para la especificación del punto inicial y final de las transacciones.
- *Embedded* SQL y SQL dinámico: definen cómo se pueden incluir las sentencias SQL dentro de lenguajes de programación de propósito general.
- Autorización: incluye comandos para especificar los derechos de acceso a relaciones y vistas.

Lenguaje para la definición de datos

- Permite la especificación de información sobre las relaciones, incluyendo:
  - Esquema de cada relación.
  - Tipo de valores asociados con cada atributo.
  - Restricciones de integridad
  - El conjunto de índices que deben mantenerse para cada relación.
  - La información de seguridad y autorización para cada relación.
  - La estructura de almacenamiento físico de cada relación en disco.

Tipos de dominio en SQL

- CHAR(n): Cadena de caracteres de longitud fija especificada por el usuario.
- VARCHAR(n): Cadenas de caracteres de longitud variable, con longitud máxima especificada por el usuario.
- INT o INTEGER: Entero (un subconjunto finito de los enteros que depende de la máquina).
- SMALLINT: Entero pequeño (un subconjunto dependiente de la máquina del tipo de dominio entero).

Tipos de dominio en SQL

- NUMERIC(p, d): Número de punto fijo, con una precisión especificada por el usuario de p dígitos, con d dígitos a la derecha del punto decimal.
- REAL, DOUBLE PRECISION: Números en punto flotante y en punto flotante de doble precisión, con precisión dependiente de la máquina.
- FLOAT(n): Número de punto flotante, con precisión especificada por el usuario de al menos n dígitos.
- NVARCHAR(n): Cadena de caracteres para datos multilingües que utilizan la representación Unicode.

Tipos de dominio en SQL

- DATE: Valor de fecha en el formato aaaa-MM-dd.
- TIME: Valor de hora en el formato hh:mm:ss.
- DATETIME: Valores de fecha y hora en el formato aaaa-MM-dd HH:mm:ss.
- BOOLEAN: Valores lógicos (TRUE o FALSE).

El constructor CREATE TABLE

- Una relación SQL se define utilizando el comando CREATE TABLE:

  CREATE TABLE r

  (A_1 D_1, A_2 D_2, ..., A_n D_n,

  ‹restriccion_de_integridad$_1$›,...,

  ‹restriccion_de_integridad$_k$›);

- Donde:
  - r es el nombre de la relación
  - A_i es un nombre de atributo en el esquema de la relación r
  - D_i es el tipo de dominio del atributo A_i

El constructor CREATE TABLE

- Ejemplo:

```
CREATE TABLE instructor (
    id VARCHAR(5),
    name VARCHAR(20),
    dept_name VARCHAR(20),
    salary NUMERIC(8,2));
```

El constructor CREATE TABLE

- Restricciones de integridad:
  - PRIMARY KEY (A_1, ..., A_n)
  - FOREIGN KEY (A_m, ..., A_n) REFERENCES s
  - NOT NULL
- Ejemplo:

```
CREATE TABLE instructor (
    id VARCHAR(5), name VARCHAR(20) NOT NULL,
    dept_name VARCHAR(20), salary NUMERIC(8,2),
    PRIMARY KEY (id),
    FOREIGN KEY (dept_name) REFERENCES department);
```

Modificaciones de tablas (I)

- DROP TABLE
  - DROP TABLE r
- ALTER TABLE
  - ALTER TABLE r ADD A D

donde A es el nombre del atributo que debe añadirse a la relación r y D es el dominio de A.

  - ALTER TABLE r DROP A

donde A es el nombre de un atributo de la relación r. La eliminación de atributos no es soportada por algunos SGBD relacionales.

Estructura básica de una consulta SQL

- Una consulta típica en SQL tiene la sintaxis:

  SELECT A_1, A_2, ..., A_n

  FROM r_1, r_2, ..., r_m

  WHERE P;

- Donde:
  - A_i representa un atributo.
  - r_j representa una relación.
  - P es un predicado.
- El resultado de una consulta SQL es una relación.

Cláusula SELECT

- Lista los atributos deseados en el resultado de una consulta.
- Corresponde a la operación de proyección del álgebra relacional.
- Ejemplo: retornar los nombres de todos los instructores

```
SELECT name
FROM instructor;
```

Cláusula SELECT

- Resultado:

name
Srinivasan
Wu
Mozart
Einstein
El Said
Gold
Katz
Califieri
Singh
Crick
Brandt
Kim

Cláusula SELECT

- SQL permite duplicados tanto en las relaciones como en los resultados de las consultas.
- Para forzar la eliminación de los duplicados, se inserta la palabra clave DISTINCT después de SELECT.
- Ejemplo: Encuentra los nombres de los departamentos de todos los instructores, y elimina los duplicados

```
SELECT DISTINCT dept_name
FROM instructor;
```

Cláusula SELECT

- La palabra clave ALL especifica que los duplicados no deben ser eliminados.
- Ejemplo:

```
SELECT ALL dept_name
FROM instructor;
```

dept_name
Comp. Sci.
Finance
Music
Physics
History
Physics
Comp. Sci.
History
Finance
Biology
Comp. Sci.
Elec. Eng.

Cláusula SELECT

- La cláusula SELECT puede contener expresiones aritméticas en las que intervengan los operadores +, -, * y / que operen sobre constantes o atributos de las tuplas.
- Ejemplo:

```
SELECT ID, name, dept_name, salary * 1.1
FROM instructor;
```

- Retorna una relación que es la misma que la relación instructor, excepto que el atributo salario es multiplicado por 1,1.

Cláusula WHERE

- Especifica las condiciones que debe satisfacer el resultado.
- Corresponde al predicado de selección del álgebra relacional.
- Ejemplo: Retornar todos los instructores del departamento 'Comp. Sci'.

```
SELECT name
FROM instructor
WHERE dept_name = 'Comp. Sci.';
```

Cláusula WHERE

- Ejemplo: Retornar todos los instructores del departamento 'Comp. Sci.' con sueldo mayor que 70000$.

```
SELECT name
FROM instructor
WHERE dept_name = 'Comp. Sci.' AND
        salary > 70000;
```

- Resultado:

name
Katz
Brandt

Cláusula WHERE

- SQL permite el uso de los conectores lógicos AND, OR, and NOT en la cláusula WHERE.
- Los operandos de los conectores lógicos pueden ser expresiones que incluyan los operadores de comparación <, <=, >, >=, = y <>.
- SQL permite comparar cadenas de caracteres, expresiones aritméticas, así como tipos especiales.

Cláusula FROM

- Enumera las relaciones implicadas en la consulta.
- Corresponde al producto cartesiano del álgebra relacional.
- Ejemplo: Encuentra el producto cartesiano entre instructor y teaches.

```
SELECT *
FROM instructor, teaches;
```

- Genera todos los pares posibles instructor - teaches, con todos los atributos de ambas relaciones.

Combinación de FROM y WHERE

- Ejemplo: Recupere los nombres de todos los instructores, junto con sus nombres de departamento y nombre del edificio del departamento.

```
SELECT name, instructor.dept_name, building
FROM instructor, department
WHERE instructor.dept_name = department.dept_name;
```

Combinación de FROM y WHERE

- Resultado:

name	*dept_name*	*building*
Srinivasan	Comp. Sci.	Taylor
Wu	Finance	Painter
Mozart	Music	Packard
Einstein	Physics	Watson
El Said	History	Painter
Gold	Physics	Watson
Katz	Comp. Sci.	Taylor
Califieri	History	Painter
Singh	Finance	Painter
Crick	Biology	Watson
Brandt	Comp. Sci.	Taylor
Kim	Elec. Eng.	Taylor

Combinación de FROM y WHERE

- Ejemplo: Encuentra los nombres de todos los instructores que han enseñado algún curso y el course_id.

```
SELECT name, course_id
FROM instructor, teaches
WHERE instructor.id = teaches.id;
```

Combinación de FROM y WHERE

- Resultado:

name	*course_id*
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

Combinación de FROM y WHERE

- Ejemplo: Encuentra los nombres de todos los instructores del departamento de Comp. Sci. que han enseñado algún curso y el course_id.

```
SELECT name, course_id
FROM instructor, teaches
WHERE instructor.id = teaches.id AND
        instructor. dept_name = 'Comp. Sci.';
```

La operación RENAME

- SQL permite renombrar relaciones y atributos usando la cláusula AS.
- Ejemplo: Encuentra los nombres de todos los instructores que tengan un salario más alto que algún instructor en 'Comp. Sci'.

```
SELECT DISTINCT T.name
FROM instructor AS T, instructor AS S
WHERE T.salary > S.salary AND
        S.dept_name = 'Comp. Sci.';
```

Operaciones sobre *strings*

- SQL especifica *strings* encerrándolos entre comillas simples.
- El estándar SQL especifica que la operación de igualdad entre *strings* distingue entre mayúsculas y minúsculas (*case sensitive*).
- SQL define diversas funciones sobre *strings*:
  - ||: concatenación de *strings*.
  - LENGTH(s): longitud del *string* s.
  - UPPER(s): conversión del *string* s a mayúsculas.
  - LOWER(s): conversión del *string* s a minúsculas.
  - TRIM(s): eliminación de espacios al final del *string* s.

Operaciones sobre *strings*

- El operador LIKE se utiliza para realizar coincidencia de patrones (*pattern matching*) sobre *strings*.
- Los patrones se describen usando dos caracteres especiales:
  - Porcentaje (%): El caracter % coincide con cualquier *substring*.
  - Underscore (_): El caracter _ coincide con cualquier caracter.
- Los patrones distinguen entre mayúsculas y minúsculas (*case sensitive*).

Operaciones sobre *strings*

- Ejemplo:
  - 'Intro%' coincide con cualquier *string* que empiece por 'Intro'.
  - '%Comp%' coincide con cualquier *string* que contenga 'Comp' como *substring*.
  - '___' coincide con cualquier *string* de exactamente tres caracteres.
  - '___%' coincide con cualquier *string* de al menos tres caracteres.

Operaciones sobre *strings*

- Ejemplo: Busque los nombres de todos los departamentos cuyo nombre de edificio incluya el *substring* 'Watson'.

```
SELECT dept_name
FROM department
WHERE building LIKE '%Watson%';
```

Ordenamiento de tuplas

- Ejemplo: Listar en orden alfabético los nombres de todos los instructores.

```
SELECT DISTINCT name
FROM instructor
ORDER BY name;
```

- Para cada atributo se puede especificar DESC para el orden descendente o ASC para el orden ascendente.

Ordenamiento de tuplas

- El orden ascendente es el predeterminado.
- Ejemplo:

```
SELECT name
FROM instructor
WHERE dept_name = 'Physics'
ORDER BY name;
```

- Se puede ordenar utilizando más de un atributo.
- Ejemplo: ORDER BY dept_name, name

Ordenamiento de tuplas

- Se puede ordenar utilizando más de un atributo.
- Ejemplo:

```
SELECT *
FROM instructor
ORDER BY salary DESC, name ASC;
```

Predicados en la cláusula WHERE

- Operador de comparación BETWEEN.

```
SELECT name
FROM instructor
WHERE salary BETWEEN 90000 AND 100000;
```

- Comparación de tuplas

```
SELECT name, course_id
FROM instructor, teaches
WHERE (instructor.id, dept_name) = (teaches.id, 'Biology');
```

Bibliografía

- Capítulo 3, Secciones 3.1 a 3.4 del libro Silberschatz, A., Korth, H. F. and Sudarshan, S. (2020). *Database System Concepts* (Seventh Edition). McGraw-Hill Education.