Manejo de Bases de Datos

Sesión # 2

Contenido

- Álgebra relacional
- El lenguaje de consulta SQL
- DDL
- DML
- Instalación de un DBMS

Álgebra relacional

Operaciones fundamentales

- Eliminadoras
  - Selección σ_F
  - Proyección π_{attr}
- Combinación
  - Producto cartesiano $\times$
- Operaciones de conjuntos
  - Unión $\cup$
  - Diferencia $-$
- Renombramiento ρ

- Operaciones unarias
- Operaciones binarias

Selección

$$\sigma_F \mathrm{R}$$

F es el predicado. Es una fórmula bien formada

Permite comparaciones de atributos

- $<, >, \leq, \geq, =, \neq$

Elaboración de predicados más grandes

- $\wedge, \vee, \neg$

Selección

Algunos ejemplos:

- Instructores del departamento de Física

$$\sigma_{\text{dept\_name}=\text{'Physics'}}(\text{instructor})$$

- Instructores que ganan más de 90000

$$\sigma_{\text{salary} < 90000}(\text{instructor})$$

- Instructores con el mismo nombre del departamento

$$\sigma_{\text{name} = \text{dept\_name}}(\text{instructor})$$

- Instructores del departamento de física que ganan más de 90000

$$\sigma_{\text{salary} < 90000 \wedge \text{dept\_name}=\text{'Physics'}}(\text{instructor})$$

instructor

instructor.ID	instructor.name	instructor.dept_name	instructor.salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

Proyección

$$\pi_{attr}\mathrm{R}$$

$attr$ es un subconjunto de atributos

Ejemplo:

- Nombre de edificio y capacidad de los salones

$\pi_{\text{building, capacity}}(\text{classroom})$

Composición de operaciones relacionales

En vez de dar el nombre de una relación, damos una expresión que al ser evaluada retorna una relación

Ejemplo:

- Nombre de los instructores del departamento de Física

$$\pi_{\text{name}}(\sigma_{\text{dept\_name}=\text{'Physics'}}(\text{instructor}))$$

Operaciones de conjuntos

Para que funcionen se requiere que las relaciones sean compatibles

Condiciones de compatibilidad:

- R y S deben tener la misma aridad (aridad = número de atributos)
- El dominio del *i*-ésimo atributo de R debe corresponder con el dominio del *i*-ésimo atributo de S

Unión

$$R \cup S$$

Crea una relación resultado de unir las tupas de R con las tuplas de S

Requiere que las relaciones R y S sean compatibles

Ejemplo:

$$\pi_{\text{course\_id}}(\sigma_{\text{semester}='\text{Fall}' \wedge \text{year}=2009}(\text{section}))$$
$$\cup$$
$$\pi_{\text{course\_id}}(\sigma_{\text{semester}='\text{Spring}' \wedge \text{year}=2010}(\text{section}))$$

Diferencia

$$R - S$$

Crea una relación con las tuplas que están en R, eliminando las tuplas de R que están en S

Requiere que las relaciones R y S sean compatibles

Ejemplo:

$$\pi_{\text{course\_id}}(\sigma_{\text{semester}='\text{Fall}' \wedge \text{year}=2009}(\text{section}))$$
$$-$$
$$\pi_{\text{course\_id}}(\sigma_{\text{semester}='\text{Spring}' \wedge \text{year}=2010}(\text{section}))$$

Producto cartesiano

$$R \times S$$

Combina las tuplas de las relaciones R y S

Asocia cada tupla de R con cada tupla de S

- La cardinalidad (número de tuplas) de la relación resultado es R*S
- La aridad de la relación resultado es la aridad de R + la aridad de S
- Para evitar atributos duplicados, cada atributo destino queda con el prefijo de la relación de la que proviene

Ejemplo:

$$\sigma_{\text{instructor.dept\_name}='\text{Physics}'}(\text{instructor} \times \text{teaches})$$

Renombramiento

$$\rho_d \mathrm{R}$$

Cambia el nombre de la relación o expresión R a d

$$\rho_{d(A_1,A_2,\ldots,A_n)} \mathrm{R}$$

Permite renombrar la relación o expresión y sus atributos, suponiendo que R tiene aridad n

Ejemplo:

$$\pi_{instructor.salary} \left(\sigma_{\mathrm{instructor.salary} < \mathrm{d.salary}} \left(\mathrm{instructor} \times \rho_d (instructor) \right) \right)$$

Actividad

Defina expresiones de álgebra relacional para calcular las siguientes relaciones:

- Salones de clase (el número) y capacidad donde la capacidad es menor a 50 estudiantes
- Nombres diferentes de edificios (tabla `department`)
- Cursos de Biología de 3 créditos o de Ciencias de la Computación de tres créditos
- Instructores que dictan clase en el edificio Taylor (tablas `department` e `instructor`)
- Edificio y número del salón con mayor capacidad

Más operaciones

- Eliminadoras
  - Selección σ_F
  - Proyección π_{attr}
- Combinación
  - Producto cartesiano $\times$
  - Inner join ∞, semijoins $\propto$, outerjoins
- Eliminación de duplicados δ
- Reordenamiento τ
- Operaciones de conjuntos
  - Unión $\cup$
  - Diferencia $-$
  - División $/$
- Renombramiento ρ
- Agrupación φ

El lenguaje de consulta SQL

Qué es SQL

IBM desarrolló la versión original de SQL, originalmente llamada Sequel, como parte del proyecto System R a principios de la década de 1970.

El lenguaje Sequel ha evolucionado desde entonces, y su nombre ha cambiado a SQL (Lenguaje de consulta estructurado)

En 1986, el American National Standards Institute (ANSI) y la Organización Internacional de Normalización (ISO) publicaron un estándar SQL, llamado SQL-86. Desde entonces ha evolucionado

Componentes de SQL

DDL (Lenguaje de Definición de Datos)

Permite hacer operaciones sobre las estructuras que contienen los datos

- Esquemas de relaciones
- Esquemas de bases de datos

DML (Lenguaje de Manipulación de Datos)

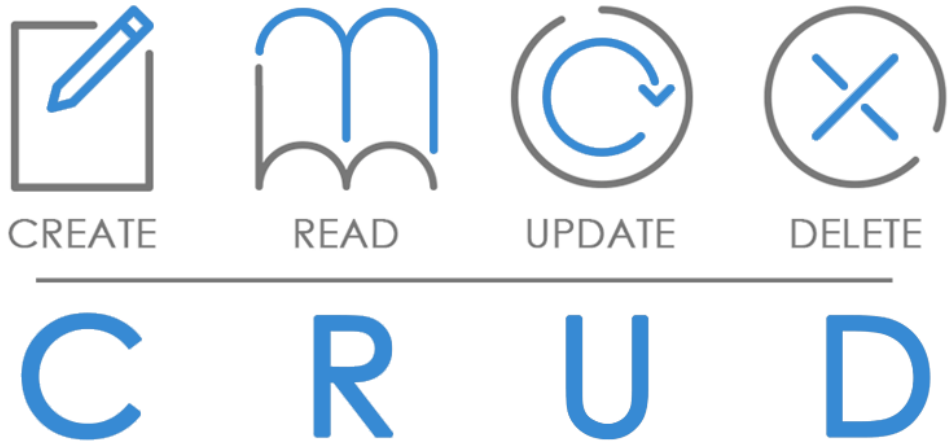
Permite hacer operaciones CRUD sobre los datos

DCL (Lenguaje de Control de Datos)

Permite la creación de roles y asignación de permisos sobre los objetos de la base de datos

CRUD

Conjunto de operaciones que podemos realizar



SQL cuenta con operaciones CRUD sobre los datos y CRUD sobre las estructuras que contienen los datos

DDL (Data Definition Language)

Tipos de datos nativos del SMBD

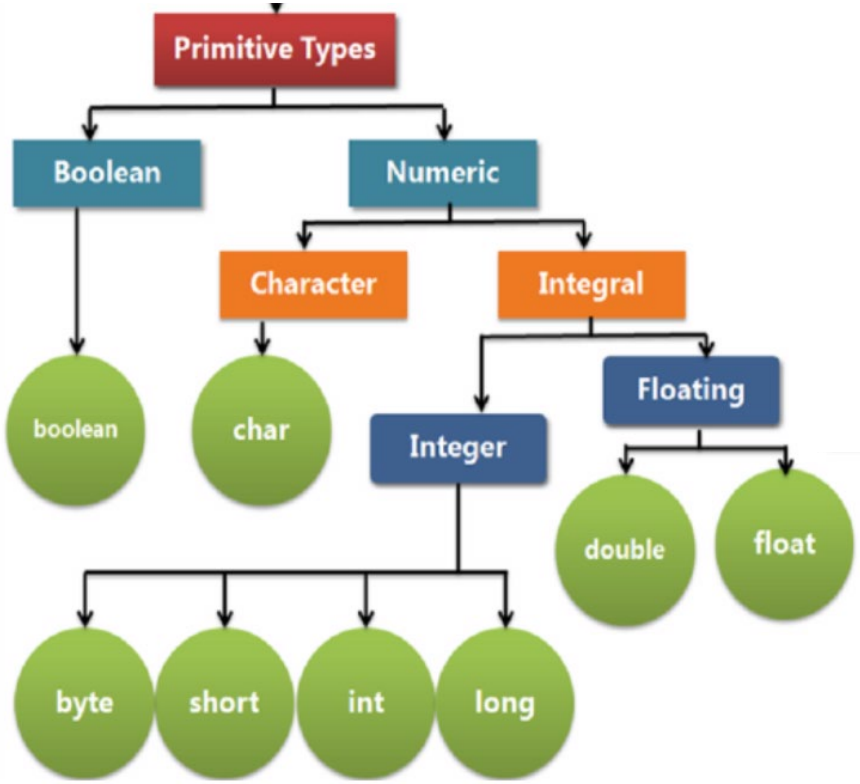
Seamos cuidadosos al escoger el tipo apropiado de dato

Factores:

- Extensibilidad (char-varchar)
- Tamaño del tipo de dato (small-big)
- Soporte (Compatibilidad con XML, JSON, hstore - drivers)

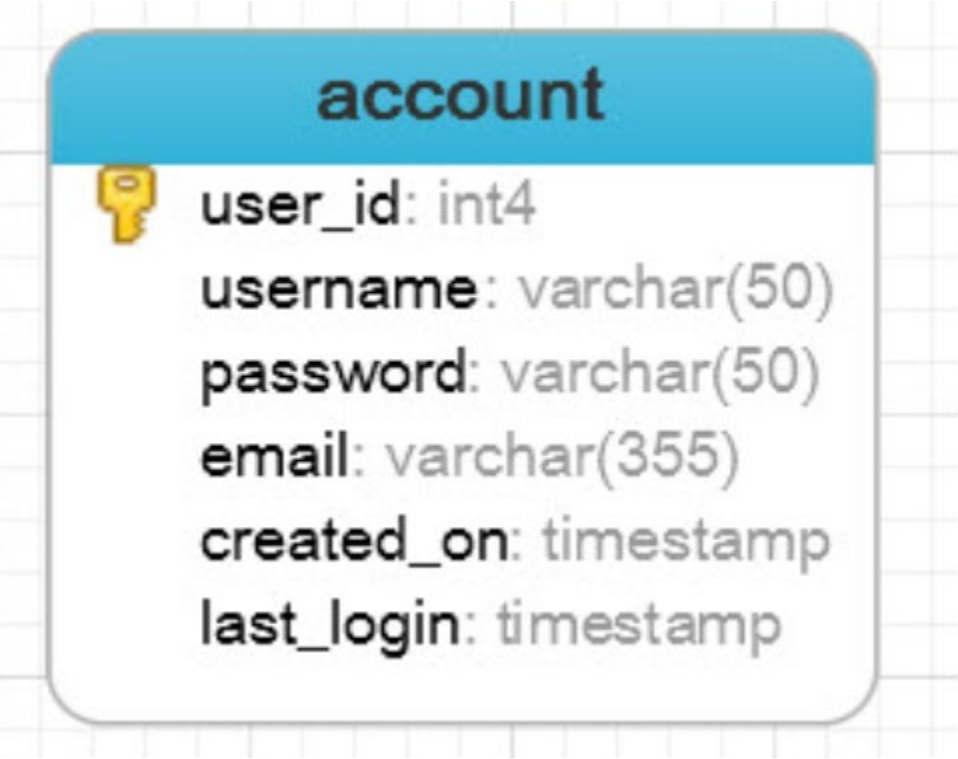
Tipos de datos

Toda columna debe tener un tipo de dato (atomicidad)



Lenguaje de Definición de Datos (DDL)

La forma más simple de declaración de una tabla consiste en las palabras clave `CREATE TABLE` seguidas por el nombre de la relación y una lista entre paréntesis, separadas por comas, de los nombres de atributos y sus tipos




```
CREATE TABLE account(
  user_id serial PRIMARY KEY,
  username VARCHAR (50) UNIQUE NOT NULL,
  password VARCHAR (50) NOT NULL,
  email VARCHAR (355) UNIQUE NOT NULL,
  created_on TIMESTAMP NOT NULL,
  last_login TIMESTAMP
);
```

Lenguaje de Definición de Datos (DDL)

Podemos eliminar toda la tabla, incluidas todas las tuplas actuales.

- La relación *movies* ya no hará parte del esquema de la base de datos
- No podremos acceder a las tupas de *movies*

```
DROP TABLE movies;
```

Lenguaje de Definición de Datos (DDL)

- Modificar el esquema de una tabla existente
- Estas modificaciones se realizan mediante una declaración que comienza con las palabras clave `ALTER TABLE` y el nombre de la relación
- Luego podemos hacer dos operaciones posibles:
  - `ADD` seguido por el nombre del atributo y su tipo de dato

```
ALTER TABLE movies ADD score INT;
```

  - `DROP` seguido por el nombre del atrubuto

```
ALTER TABLE movies DROP producerC;
```

Categorías de tipos de datos

- Numéricos
- De caracteres
- De fecha y tiempo

Tipos de datos numéricos

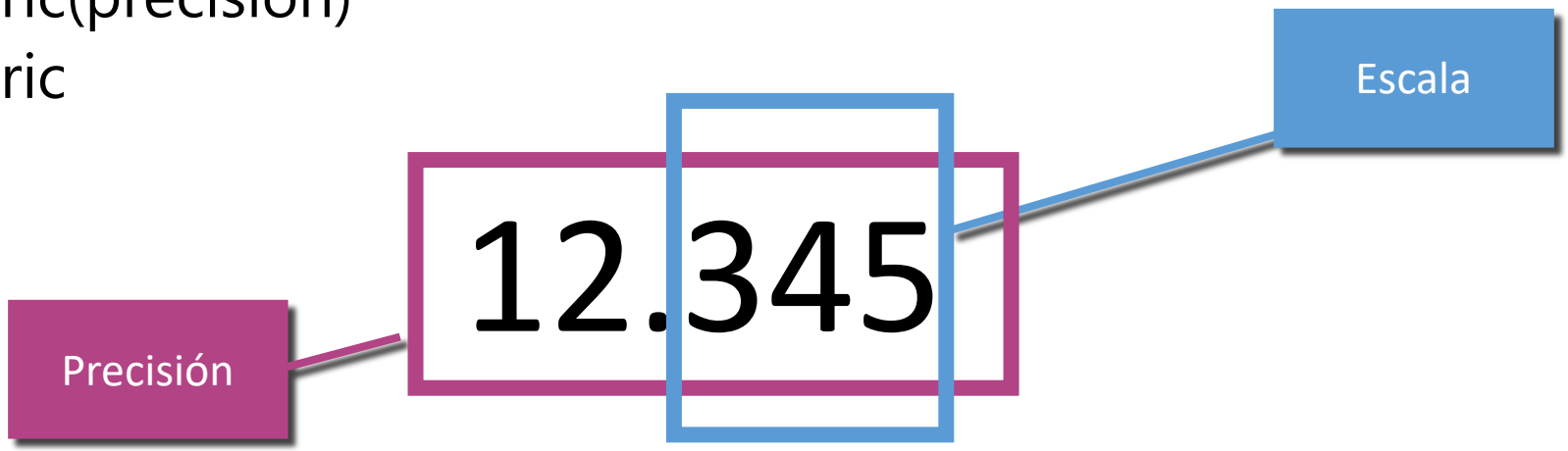
Name	Comments	Size	Range
`smallint`	SQL equivalent: `Int2`	2 bytes	-32,768 to +32,767.
`integer`	SQL equivalent: `Int4` Integer is an alias for `INT`.	4 bytes	-2,147,483,648 to +2,147,483,647.
`bigint`	SQL equivalent: `Int8` 8 bytes	8 bytes	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807.
`numeric` or `decimal`	No difference in PostgreSQL	Variable	Up to 131,072 digits before the decimal point; up to 16,383 digits after the decimal point.
`real`	Special values: Infinity, Infinity, `NaN`	4 bytes	Platform-dependent, at least six-digit precision. Often, the range is 1E-37 to 1E+37.
`double precision`	Special values: Infinity, Infinity, `NaN`	8 bytes	Platform dependent, at least 15-digit precision. Often, the range is 1E-307 to 1E+308.

```
SELECT
    CAST(5.9 AS INTEGER) AS "CAST (5.9 AS INT)",
    CAST(5.1 AS INTEGER) AS "CAST(5.1 AS INT)",
    CAST(-23.5 AS INTEGER) AS "CAST(-23.5 AS
INT)" ,
5.5::INT AS "5.5::INT";
```

Tipos de datos INTEGER (INT) y NUMERIC

NOTAS:

- smallint ahorra espacio
- numeric y decimal se recomiendan para guardar dinero o cifras donde la precisión es requerida
- Los tipo numeric se declaran de tres posibles formas
  - numeric(precisión, scale)
  - numeric(precisión)
  - numeric



Tipos de datos NUMERIC

¿Cuál es la precisión y escala del siguiente número?

1234.567

Tipos de datos NUMERIC

El valor NUMERIC puede tener hasta

- 131072 dígitos antes del punto decimal
- 16383 dígitos después del punto decimal

Ejemplo - Tipos de datos NUMERIC

- Cree una tabla productos con la siguiente sentencia DDL

```
CREATE TABLE IF NOT EXISTS products
(
    id    serial PRIMARY KEY,
    name  VARCHAR NOT NULL,
    price NUMERIC(5, 2)
);
```

```
ALTER TABLE products
ALTER COLUMN price new_data_type(size);
```

- Guarde dos productos, que tengan los precios 500.215 y 500.214
- Consulte la tabla
- Borre la tabla con `DROP TABLE products;`

Dato FLOAT

- Borre la tabla con `DROP TABLE products;`
- Cree la tabla nuevamente, cambiando el tipo de dato de PRICE

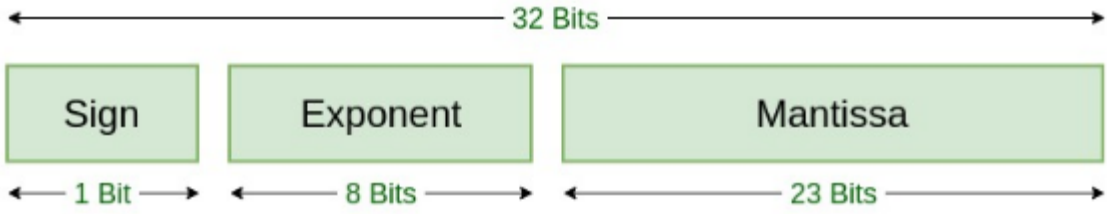
```
CREATE TABLE IF NOT EXISTS products
(
    id    serial PRIMARY KEY,
    name  VARCHAR NOT NULL,
    price FLOAT
);
```

```
ALTER TABLE products
ALTER COLUMN price TYPE new_data_type(size);
```

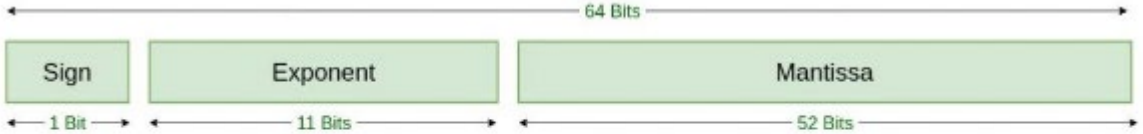
- Modifique el valor de un producto

```
UPDATE products
SET price = 'NaN'
WHERE
    id = 1;
```

Dato FLOAT



Single Precision
IEEE 754 Floating-Point Standard



Double Precision
IEEE 754 Floating-Point Standard

```
price FLOAT()
```

```
price2 FLOAT(p)
```

Precisión real y doble tienen exactamente 24 y 53 bits en la mantisa respectivamente

Recuperado de https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/

Tipos de datos serial

- Los tipos `serial`, concretamente `smallserial`, `serial` y `bigserial`, son definidos sobre `smallint`, `integer` y `bigint`, respectivamente.
- Los tipos `serial` no son tipos de datos verdaderos. Se usan como llaves sustitutas, no se les permite tener un valor nulo.
- El tipo `serial` utiliza las secuencias. Una secuencia es un objeto de base de datos que se utiliza para generar secuencias especificando los valores mínimo, máximo e incremental.

Ejemplo - Tipos de datos serial

```
CREATE TABLE somenumbers (
   a_number_id SERIAL
);
```

```
INSERT INTO somenumbers DEFAULT VALUES;
```

Tipos de datos serial

```
CREATE TABLE tablename (
    colname SERIAL
);
```

Equivalentes

```
CREATE SEQUENCE tablename_colname_seq;
CREATE TABLE tablename (
    colname integer NOT NULL DEFAULT nextval('tablename_colname_seq')
);
ALTER SEQUENCE tablename_colname_seq OWNED BY tablename.colname;
```

Lenguaje de Definición de Datos (DDL)


```
CREATE TABLE role(
  role_id serial PRIMARY KEY,
  role_name VARCHAR (255) UNIQUE NOT NULL
);
```

Tipos de datos de caracter

Name	Comments	Trailing spaces	Maximum length
`char`	Equivalent to `char(1)`, it must be quoted as shown in the name.	Semantically insignificant	1
`name`	Equivalent to `varchar(64)`. Used by Postgres for object names.	Semantically significant	64
`char(n)`	Alias: `character(n)`. Fixed-length character where the length is `n`. Internally called **blank padded character** (**bpchar**).	Semantically insignificant	1 to 10485760
`varchar(n)`	Alias: character `varying(n)`. Variable-length character where the maximum length is `n`.	Semantically significant	1 to 10485760
`text`	Variable-length character.	Semantically significant	Unlimited

Tipos de datos de tiempo

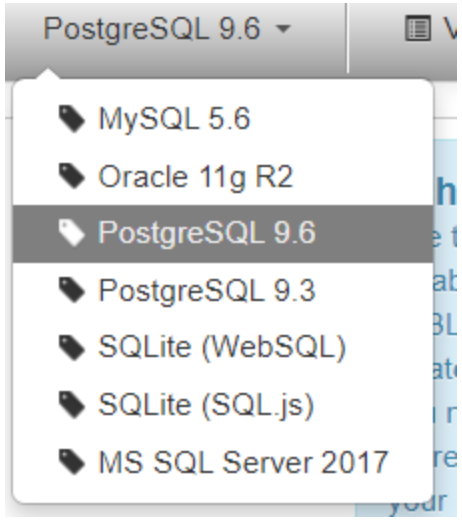
Name	Size in bytes	Description	Low value	High value
Timestamp without time zone	8	Date and time without time zone, equivalent to timestamp	4713 BC	294276 AD
Timestamp with time zone	8	Date and time with time zone, equivalent to timestamptz	4713 BC	294276 AD
Date	4	Date only	4713 BC	294276 AD
Time without time zone	8	Time of day	00:00:00	24:00:00
Time with time zone	12	Time of day with time zone	00:00:00+1459	24:00:00-1459
Interval	16	Time interval	-178,000,000 years	+178,000,000 years

Actividad

Cree una tabla en SQLFiddle usando los tipos de datos vistos anteriormente

http://sqlfiddle.com/

Antes de iniciar con la creación de la tabla, especifique el SMBD PostgreSQL



DML (Data Manipulation Language)

La sentencia

SELECT

Recuperar datos con SELECT

Recuperar todas las filas y todas las columnas

```
SELECT
  Code,
  Name,
  Continent,
  Region,
  SurfaceArea,
  IndepYear,
  Population,
  LifeExpectancy,
  GNP,
  GNPOld,
  LocalName,
  GovernmentForm,
  HeadOfState,
  Capital,
  Code2
FROM
  country;
```



```
SELECT *
FROM country;
```

Recuperar datos con SELECT

Recuperar todas las filas y columnas específicas

```
SELECT
  Code,
  Name,
  Continent,
  Population,
  LifeExpectancy
FROM
  country;
```

Recuperar datos con SELECT

Uso del `DISTINCT`

```
SELECT Continent
FROM country;
```

Entrega todos los resultados, así tengan columnas repetidas

Elimina los repetidos

```
SELECT DISTINCT Continent
FROM country;
```

Recuperar datos con SELECT

Columnas calculadas/derivadas

- Podemos realizar operaciones en una columna
- La columna toma el nombre de la operación (por defecto)

```
SELECT Name, SurfaceArea, Population, Population/SurfaceArea
FROM country;
```

- Podemos poner un alias al nombre de la columna

```
SELECT Name, SurfaceArea, Population, Population/SurfaceArea AS HabsPorKm2
FROM country;
```

La clausula WHERE

```
SELECT Name, Population
FROM city
WHERE CountryCode = 'COL';
```

Condiciones/Predicados básicos:

- Comparación: Compara el valor de una expresión con el valor de otra expresión
- Prueba de rango: si el valor de una expresión cae dentro de un determinado rango de valores
- Establecer pertenencia: Probar si el valor de una expresión es igual a uno de un conjunto de valores
- Coincidencia de patrón: Comprueba si una cadena coincide con un patrón específico.
- Null: Comprueba si una columna tiene un valor nulo (desconocido).

La clausula WHERE

Operadores de comparación:

COMPARISON OPERATORS	
SYMBOL	**MEANING**
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<> or !=	Not equal to

```
SELECT
  Name,
  Continent,
  Population
FROM
  country
WHERE
  Population < 10000;
```

La clausula WHERE

Se pueden generar predicados más complejos utilizando los operadores lógicos **AND**, **OR** y **NOT**, con paréntesis (si es necesario o se desea)

Las reglas para evaluar una expresión condicional son:

- Una expresión se evalúa de izquierda a derecha
- Las subexpresiones entre paréntesis se evalúan primero
- Los **NOT** se evalúan antes de los **AND** y los **OR**
- Los **AND** se evalúan antes que los **OR**

```
SELECT
  Name, Continent, Population
FROM
  country
WHERE
  Population < 10000 AND
  Continent <> 'Oceania' AND
  Continent <> 'Antarctica'  ;
```

Siempre se recomienda el uso de paréntesis, para eliminar cualquier posible ambigüedad.

La clausula WHERE

Rangos con **BETWEEN**/**NOT BETWEEN**

- Incluye los valores extremos

```
SELECT
  Name, Continent, SurfaceArea
FROM
  country
WHERE
  SurfaceArea NOT BETWEEN 15 AND 8000000;
```

La clausula WHERE

Comparación de un valor en una lista de valores con **IN**/**NOT IN**

```
SELECT
  Name, Continent
FROM
  country
WHERE
  Continent IN ('Oceania', 'Antarctica');
```

Crear información con

INSERT

Actualizar la base de datos con INSERT

Para insertar una columna se usa el commando `INSERT INTO`, de acuerdo a la siguiente síntaxis:

```
INSERT INTO nombreTabla (col1, col2, ... )
VALUES (valor1, valor2,...);
```

- **`nombreTabla`**: Tabla donde se agregará el registro
- **`col1, col2, ...`** : Nombre de las columnas de la table
- **`valor1, valor2, ...`** : Valores que irán en la nueva tupla

Actualizar la base de datos con INSERT

INSERT sencillo

```
INSERT INTO orders
   (order_id, order_date, amount, customer_id)
VALUES (34, '03/14/1760', 45.6, 1);
```

INSERT con tabla como parámetro

```
INSERT INTO orders
    (order_id,
     order_date,
     amount,
     customer_id)
VALUES
       (SELECT * FROM orders WHERE order_id = 1);
```

La sentencia

UPDATE

La sentencia UPDATE

Actualizar todas las filas

```
UPDATE orders
SET amount = amount * 1.10;
```

Actualizar todas las filas que correspondan con la clausula **WHERE**

```
UPDATE orders
SET amount = amount * 1.10
WHERE customer_id = 1;
```

Actualizar columnas múltiples

```
UPDATE orders
SET amount = amount * 1.10, order_date = REPLACE(order_date,'/','-')
WHERE customer_id = 1;
```

La sentencia

DELETE

La sentencia DELETE

Borra todas las filas

```
DELETE FROM orders;
```

Borra todas las filas que correspondan con la clausula **WHERE**

```
DELETE FROM orders
WHERE customer_id = 1;
```