UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE CIENCIAS
FUNDAMENTOS DE BASES DE DATOS

Structured Query Language

Gerardo Avilés Rosas
gar@ciencias.unam.mx

Un poco de historia

- En 1970 **Codd** propone el modelo relacional y asociado a éste, un *sublenguaje* de acceso a datos basado en cálculo de predicados.
- **SQL** fue desarrollado por **IBM** (*San Jose Research Laboratory*) a principios de la década de los **70s**.
- En **1974** se presenta en una conferencia de **ACM**. El lenguaje originalmente se llamaba **SEQUEL** (*Structured English Query Language*).
- La primera implementación de **SQL** fue proporcionada por **Oracle Corporation** (*Relational Software Inc.*)

…Un poco de historia

- **ANSI** publica el primer estándar **SQL** (**SQL-86**) en 1986
- En su primera revisión (**1989**) el estándar brindaba soporte para **modelo de datos orientado a objetos**. Se agregaron expresiones regulares, consultas recursivas y *triggers*.
  - Se permite que ***métodos/funciones/procedimientos*** puedan ser escritos en **SQL** o en otros lenguajes de programación: **C++**, **Java**.
- El estándar **SQL** ha sido actualizado cuatro veces más: **SQL:92**, **SQL:2003, SQL:2006** y **SQL:2008**, en las cuales se han ampliado el soporte orientado a objetos y se ha añadido y mejorado el soporte para **XML**
- Solo hasta **1996** los **SABD** estuvieron obligados a presentar sus productos al **NITS** (*National Institute for Standars and Technology*).

¿Qué es SQL?

- El **Lenguaje de Consulta Estructurado** (*Structured Query Language*) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones sobre éstas.
- Una de sus características principales es el manejo de **álgebra y cálculo relacional** que le permite realizar **consultas** con el fin de recuperar información de interés de una base de datos, así como hacer cambios sobre ella.

…¿Qué es SQL?

- **SQL** es el lenguaje estándar para trabajo con **BDR** ya que permite la definición, acceso y control de datos.
- Está basado principalmente en el **Álgebra Relacional**, sus componentes son:
  - **Lenguaje para definición de datos.** Permite la definición de esquemas, borrado de relaciones, creación de índices y modificación de esquemas.
  - **Control.** Permite definir vistas, especificar derechos de acceso a relaciones y especificar restricciones de integridad.
  - **Lenguaje para manipulación de datos.** Instrucciones para insertar, borrar y modificar tuplas, así como para consultar tablas.
  - **Control de transacciones.** Permiten especificar los límites de una transacción así como bloque explícito de datos para controlar la concurrencia.

…¿Qué es SQL?

- Uso directo, interactivo:

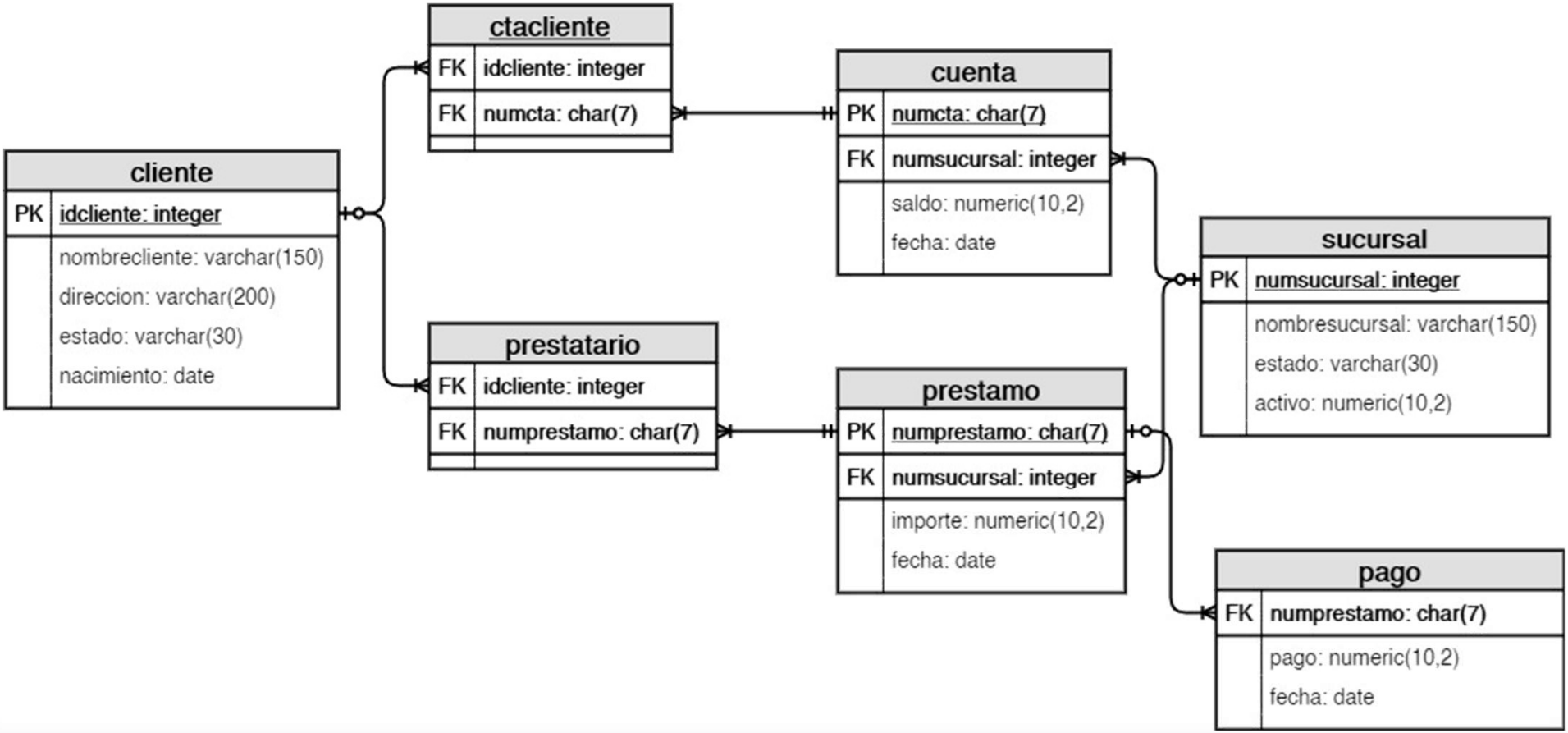
```
select numcta
from cuenta
where saldo > 10000;
```

- Uso desde un programa de aplicación (JAVA, SAS):

```
Statement s = c.createStatement()
for(int i = 0;i < cuentas.length;i++)
    s.executeUpdate("UPDATE cuentas" +
                    "SET balance = " + cuentas[i].getBalance() +
                    "WHERE id = " + cuentas[i].getId());
```

```
proc sql;
        create table lib.prueba as
        select *
        from bd.ctacliente natural join bd.prestatario;
quit;
```

Mi Banquito S.A. de C.V.



Consulta de datos

Para consultar una base de datos se usa la instrucción:

```
select a1,a2,a3,...,aN
from R1,R2,R3,...,RM
where condición
```

- La cláusula **FROM** indica las relaciones que serán consultadas.
- La cláusula **WHERE** especifica la condición que deben satisfacer las tuplas para ser seleccionadas.
- La cláusula **SELECT** se utiliza para describir los atributos que se desea formen parte de la respuesta.
- Condición:
  - **Operandos**: Constantes y atributos de las relaciones mencionadas en la cláusula **FROM**.
  - **Operadores:** =, <>, >, <, <=, >=, AND, OR, NOT

Comparación de cadenas

- Es posible comparar cadenas, aunque estas sean de diferente tipo (**VARCHAR o CHAR**). La comparación se hace usando el orden lexicográfico.
- La búsqueda de patrones implica usar el operador **LIKE,** una cadena y un patrón **s LIKE p.**
- Metacaracteres: **% y _:**
  - **%** indica que **p** puede coincidir con cualquier subcadena en **s**.
  - _ coincide con cualquier carácter en **s.**
- El valor de esta expresión es verdadero si y sólo si la cadena **s**, coincide con **p.**
- **s NOT LIKE p** es verdadera si y sólo si, la cadena **s** no coincide con el patrón **p**.
- A partir de la versión **7.4** de **PostgreSQL**, es posible utilizar la función ***REGEXP***
  - La información completa se encuentra en el siguiente vínculo:

https://www.postgresql.org/docs/current/functions-matching.html

Fechas y horas

- Para **SQL** una fecha constante se representa por la palabra **DATE** seguida de una fecha entre apóstrofes en formato **yyyy-mm-dd**.

Ejemplo: DATE '1810-09-15'

- Una hora constante es una cadena entre apóstrofes, en formato **hh:mm:ss** precedida de la palabra **TIME**.

Ejemplos: TIME '18:15:00' o bien TIME '10:05:10.5'

- Para combinar las fechas con las horas se utiliza la palabra **TIMESTAMP**.

Ejemplo: TIMESTAMP '1992-04-14 07:50:00'

Se comparan estos tipos de datos con los operadores de relación utilizados con cadenas y números

Proyección

- Para eliminar atributos de las tuplas elegidas, se puede hacer una proyección sobre algunos atributos.

```
select nombreSucursal
from prestamo;
```

- Para asegurar que no haya duplicados se debe usar la palabra **DISTINCT**.

```
select distinct nombreSucursal
from prestamo;
```

- Es posible cambiar de nombre a un atributo en la salida:

```
select distinct nombreSucursal as sucursal
from prestamo;
```

- Formula en lugar de atributo:

```
select nombreSucursal,numprestamo,importe*100
from prestamo;
```

…Proyección

- Constantes:

```
select nombreSucursal,numprestamo,importe,'pesos'
from prestamo;
```

- Operador de concatenación:

```
select concat('Sr.(a) ',nombreCliente) as cliente
from cliente;
```

```
select 'Sr.(a) ' || nombreCliente as cliente
from cliente;
```

- Combinación de selección, proyección y búsqueda de cadenas:

```
select direccion
from cliente
where regexp_like(direccion,'.+[(NUM. )|(NO. )]239');
```

Ordenar el resultado

Para presentar el resultado en orden ascendente (**ASC**) o descendente (**DESC**) se debe agregar a la instrucción **SELECT-FROM-WHERE** la cláusula **ORDER BY <lista de atributos>**.

Ejemplo. Obtener una lista ordenada de los clientes que viven en **HIDALGO**:

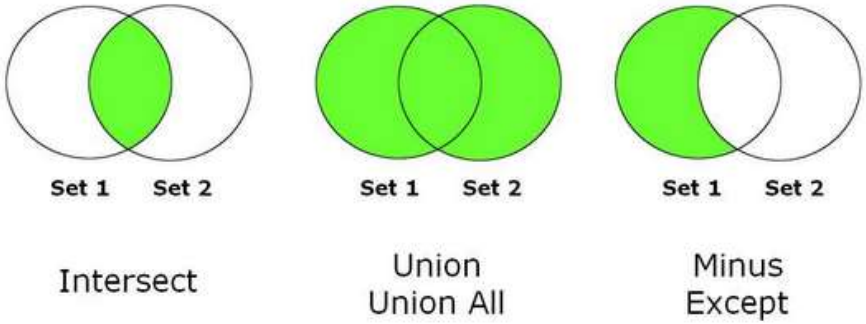
```
select distinct nombreCliente
from cliente
where lower(estado) = 'hidalgo'
order by nombrecliente;
```

Ejemplo. Ordenando de acuerdo a más de un atributo:

```
select *
from prestamo
order by nombreSucursal, importe desc;
```

Operaciones de conjunto

- SQL proporciona los operadores **UNION, INTERSECT y EXCEPT** para trabajar con relaciones compatibles, es decir que tengan el mismo conjunto de atributos.
- A diferencia del **SELECT**, las operaciones para manejo de conjuntos eliminan duplicados automáticamente.
- Para conservar los duplicados se debe utilizar **UNION ALL, EXCEPT ALL** o **INTERSECT ALL** según sea el caso.



Funciones de agregación

- Estos operadores toman una colección de valores y producen un único valor de salida. Los operadores de agregación son:
  - **SUM**, suma los valores en la columna indicada
  - **AVG**, promedia los valores en la columna indicada
  - **MIN**, el menor de los valores en la columna indicada
  - **MAX**, el mayor de los valores en la columna indicada
  - **COUNT**, la cantidad de los valores en la columna indicada.
- Los dos primeros operadores trabajan sobre números, los otros pueden operar con tipos no numéricos.
- Estos valores se aplican típicamente en la columna **SELECT**.

Agrupaciones

- Con frecuencia se requiere agrupar las tuplas antes de aplicar un operador de agregación, esto se hace a través de la instrucción: **GROUP BY atributos**.
- **HAVING** se utiliza para restringir las tuplas agrupadas. Su sintaxis es la palabra **HAVING** seguida de una condición acerca del grupo.
- Si hay **WHERE, HAVING** y **GROUP BY:**
  - Se aplica el predicado del **WHERE**
  - Las tuplas seleccionadas se agrupan por **GROUP BY**
  - Se aplica la cláusula **HAVING** a cada grupo
  - Los grupos que no satisfagan esta cláusula, se eliminan
  - La cláusula **SELECT** utiliza los grupos restantes para generar las tuplas resultado de la consulta.

Expresiones para JOIN: Cross Join

Estas expresiones pueden usarse como consultas como una alternativa a la instrucción **SELECT-FROM-WHERE** o bien como subconsultas en cláusulas **FROM**.

La forma más sencilla es el **Cross Join** o **producto cartesiano (RXS):**

$R \times S \quad \rightarrow \quad$ R **cross join** S

R:

A	B	C
1	2	3
6	7	8
9	7	8

S:

B	C	D
2	3	4
2	3	5
7	8	10

```
select R.*,S.*
from R cross join S;
```

R.A	R.B	R.C	S.B	S.C	S.D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	2	3	4
6	7	8	2	3	5
6	7	8	7	8	10
9	7	8	2	3	4
9	7	8	2	3	5
9	7	8	7	8	10

Expresiones para JOIN: Join Natural

$R \bowtie S \rightarrow$ R **natural join** S

R:

A	B	C
1	2	3
6	7	8
9	7	8

S:

B	C	D
2	3	4
2	3	5
7	8	10

```
select *
from R natural join S;
```

```
select A,R.B as B,S.C as C,D
from R join S on R.C = S.C and R.B = S.B;
```

A	B	C	D
1	2	3	4
1	2	3	5
6	7	8	10
9	7	8	10

Expresiones para JOIN: Theta Join

$R \bowtie S \rightarrow$ R **join** S **on** ***condición***

R:

A	B	C
1	2	3
6	7	8
9	7	8

S:

B	C	D
2	3	4
2	3	5
7	8	10

```
select R.*,S.*
from R join S on A < D;
```

A	R.B	R.C	S.B	S.C	D
1	2	3	2	3	4
1	2	3	2	3	5
1	2	3	7	8	10
6	7	8	7	8	10
9	7	8	7	8	10

Expresiones para JOIN: Join Externo

Agrega al resultado cada tupla que no se une con ninguna y completa con nulos los otros atributos. Existen tres variedades:

1. **Join externo completo**

$R \text{ =⋈= } S \quad \rightarrow \quad$ R **natural full outer join** S

R:

A	B	C
1	2	3
6	7	18
9	7	8

S:

B	C	D
2	3	4
2	3	5
7	8	10
2	5	15

```
select *
from R natural full outer join S;
```

A	B	C	D
1	2	3	4
1	2	3	5
6	7	18	NULL
9	7	8	10
NULL	2	5	15

…Expresiones para JOIN: Join Externo

2. Join externo por la izquierda

$R \; =\!\bowtie \; S$ → R **natural left outer join** S

R:

A	B	C
1	2	3
6	7	18
9	7	8

S:

B	C	D
2	3	4
2	3	5
7	8	10
2	5	15

```
select *
from R natural left outer join S;
```

A	B	C	D
1	2	3	4
1	2	3	5
6	7	18	NULL
9	7	8	10

…Expresiones para JOIN: Join Externo

3. Join externo por la derecha

$R \bowtie= S \rightarrow$ R **natural right outer join** S

R:

A	B	C
1	2	3
6	7	18
9	7	8

S:

B	C	D
2	3	4
2	3	5
7	8	10
2	5	15

```
select *
from R natural left outer join S;
```

A	B	C	D
1	2	3	4
1	2	3	5
9	7	8	10
NULL	2	5	15

En lugar de usar el **Join natural** puede especificarse cualquier condición:

- **R FULL OUTER JOIN S ON condición**
- **R LEFT OUTER JOIN S ON condición**
- **R RIGHT OUTER JOIN S ON condición**

Subconsultas

- Una **subconsulta** es una consulta que está incluida en otra.
- Formas de uso para las subconsultas son:
  - En consultas con operadores de conjuntos:

```
(select distinct nombreCliente from ctaCliente)
intersect
(select distinct nombreCliente from prestatario);
```

  - En la cláusula **WHERE**, si regresa un valor escalar, para comparar contra algún otro valor.
  - En la cláusula **WHERE**, si regresa una relación, para comparar contra un conjunto de valores (relación).
  - En la cláusula **FROM**, si devuelve una relación, como la relación sobre la que realizará la consulta.

Generar valores booleanos

Sea **s** un valor o una tupla, los operadores que pueden aplicarse al resultado de una subconsulta y producir un resultado **booleano** son:

- **EXISTS R**, devuelve verdadero si y sólo si R no está vacía
- **s IN R**, devuelve verdadero si y sólo si, **s** es igual a alguno de los valores de R
- **s > ALL R**, devuelve verdadero si y sólo si **s** es mayor que todos los valores en la relación R (el signo de mayor puede sustituirse por cualquier operador de comparación). R es una relación unaria.
- **s > ANY R**, devuelve verdadero si y sólo si **s** es mayor que al menos un valor en la relación R (el signo de mayor puede sustituirse por cualquier operador de comparación). Se puede usar **SOME** como sinónimo.

- Los operadores pueden ser negados precediéndolos de la palabra **NOT**: **NOT EXISTS R, NOT s > ALL R, NOT s > ANY R, s NOT IN R**.

Subconsultas como relaciones

Está permitido usar una subconsulta en la cláusula **FROM**, en cuyo caso es necesario dar nombre a la relación resultante de la subconsulta y posiblemente renombrar los atributos.

Ejemplo. Para cada sucursal con saldo promedio superior a $100,000 obtener el nombre y saldo promedio:

```
select sucursal,saldoProm
from (select nombreSucursal as sucursal,
             avg(saldo) as saldoProm
      from cuenta
      group by nombreSucursal) as resultado
where saldoProm > 100000;
```

¡Gracias!


No estés muy orgulloso de haber comprendido estas notas.
La habilidad para manejar SQL es insignificante comparado con el poder de la Fuerza.