UNIVERSIDAD DE LOS ANDES VENEZUELA

SQL(1)

(...la guía ***básica*** de ***supervivencia*** de dmi)

Universidad de los Andes
Demián Gutierrez
Enero 2009

El estándar SQL

UNIVERSIDAD DE LOS ANDES VENEZUELA

Year	Name	Alias	Comments
1986	SQL-86	SQL-87	First published by ANSI. Ratified by ISO in 1987.
1989	SQL-89	FIPS	Minor revision, adopted as FIPS 127-1.
1992	SQL-92	SQL2, FIPS 127-2	Major revision (ISO 9075), Entry Level SQL-92 adopted as FIPS 127-2.
1999	SQL:1999	SQL3	Added regular expression matching, recursive queries, triggers, support for procedural and control-of-flow statements, non-scalar types, and some object-oriented features.
2003	SQL:2003		Introduced XML-related features, window functions, standardized sequences, and columns with auto-generated values (including identity-columns).
2006	SQL:2006		ISO/IEC 9075-14:2006 defines ways in which SQL can be used in conjunction with XML. It defines ways of importing and storing XML data in an SQL database, manipulating it within the database and publishing both XML and conventional SQL-data in XML form. In addition, it provides facilities that permit applications to integrate into their SQL code the use of XQuery, the XML Query Language published by the World Wide Web Consortium (W3C), to concurrently access ordinary SQL-data and XML documents.
2008	SQL:2008		Legalizes ORDER BY outside cursor definitions. Adds INSTEAD OF triggers. Adds the TRUNCATE statement.cite_ref-iablog.sybase.com-paulley_12-0cite_ref-iablog.sybase.com-paulley_12-0[13]

El estándar SQL

Hoy en día la mayoría de los sistemas de gestión de bases de datos soportan el estándar de SQL... y bueno... más o menos... al menos algo que está bastante cerca

Esta clase está basada en PostgreSQL

Pero en general el contenido aplica a otros SGBD

Conectarse a PostgreSQL

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
[dmi@cyrano ~]$ psql  -U postgres
Welcome to psql 8.3.1, the PostgreSQL interactive terminal.

Type:  \copyright for distribution terms
       \h for help with SQL commands
       \? for help with psql commands
       \g or terminate with semicolon to execute query
       \q to quit

postgres=#
```

En este punto ya tenemos un *prompt* (Línea de Comandos) en el que podemos ejecutar sentencias SQL

La ayuda interna... tanto de SQL como de los comandos propios de PostgreSQL **(¡ÚSELA!)**

Esto aplica solo para PostgreSQL, pero en general todos los manejadores de BD tienen una consola

Crear o Eliminar una Base de Datos

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# \l
        List of databases
   Name    |  Owner   | Encoding
-----------+----------+----------
 postgres  | postgres | UTF8
 template0 | postgres | UTF8
 template1 | postgres | UTF8
(3 rows)

postgres=# CREATE DATABASE prueba;
CREATE DATABASE
postgres=# \l
        List of databases
   Name    |  Owner   | Encoding
-----------+----------+----------
 postgres  | postgres | UTF8
 prueba    | postgres | UTF8
 template0 | postgres | UTF8
 template1 | postgres | UTF8
(4 rows)

postgres=#
```

\l se usa para listar las bases de datos existentes

Ojo, los comandos que comienzan con "\" no son de SQL, son de PostgreSQL

```
postgres=# DROP DATABASE prueba;
DROP DATABASE
postgres=# \l
        List of databases
   Name    |  Owner   | Encoding
-----------+----------+----------
 postgres  | postgres | UTF8
 template0 | postgres | UTF8
 template1 | postgres | UTF8
(3 rows)

postgres=#
```

Conectarse a una Base de Datos

```
postgres=# CREATE DATABASE prueba;
CREATE DATABASE
prueba=# \l
         List of databases
   Name    |  Owner   | Encoding
-----------+----------+----------
 postgres  | postgres | UTF8
 prueba    | postgres | UTF8
 template0 | postgres | UTF8
 template1 | postgres | UTF8
(4 rows)

postgres=# \c prueba
You are now connected to database "prueba".
prueba=#
```

\c se usa para conectarse (usar) una BD existente

Esto varía mucho a lo largo de los distintos manejadores de BD, por ejemplo, en MySQL es **USE prueba**

Crear una Tabla (Esquema de Relación) (1) (Lo Básico)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE persona (
prueba(# cedula    INT,
prueba(# nombre    VARCHAR(50),
prueba(# apellido  VARCHAR(50),
prueba(# fecha_nac DATE
prueba(# );
CREATE TABLE
prueba=#
```

```
prueba=# CREATE TABLE persona (
          cedula    INT,
          nombre    VARCHAR(50),
          apellido  VARCHAR(50),
          fecha_nac DATE
         );
CREATE TABLE
prueba=#
```

\dt se usa para listar las tablas existentes en la BD actual

```
prueba=# \dt
          List of relations
 Schema |  Name   | Type  |  Owner
--------+---------+-------+----------
 public | persona | table | postgres
(1 row)

prueba=#
```

Opps, pero se nos olvidó la clave primaria...

SQL (Tipos de Datos (1)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Nombre	Alias	Description	Descripción
bigint	int8	Exact numeric of selectable precision	Numero entero
bigserial	serial8	autoincrementing eight-byte integer	Entero autoincremental
bit [(n)]		fixed-length bit string	Cadena de bits de longitud fija
bit varying [(n)]	varbit	variable-length bit string	Cadena de bits de longitud fija
boolean	bool	logical Boolean (true/false)	Lógico (verdadero/falso)
box		rectangular box on a plane	Caja rectangular en un plano
bytea		binary data ("byte array")	Arreglo binario de datos
character varying [(n)]	varchar [(n)]	variable-length character string	Caracter de longitud variable
character [(n)]	char [(n)]	fixed-length character string	Caracter de longitud fija
cidr		IPv4 or IPv6 network address	Dirección IPv4 o IPv6

SQL (Tipos de Datos (2)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Nombre	Alias	Description	Descripción
circle		circle on a plane	Círculo en un plano
date		calendar date (year, month, day)	fecha calendario (año, mes, día)
double precision	float8	double precision floating-point number (8 bytes)	numero de punto flotante de doble precisión (8 bytes)
inet		IPv4 or IPv6 host address	Dirección IPv4 o IPv6
integer	int, int4	signed four-byte integer	Entero de cuatro bytes con signo
interval [fields] [(p)]		time span	Intervalo de tiempo
line		infinite line on a plane	Línea infinita en un plano
lseg		line segment on a plane	Segmento de línea en un plano
macaddr		MAC (Media Access Control) address	Dirección de red MAC (Dirección física)
money		currency amount	Cantidad de dinero
numeric [(p, s)]	decimal [(p, s)]	exact numeric of selectable precision	Numero de precisión exacta (seleccionable)

SQL (Tipos de Datos (3)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Nombre	Alias	Description	Descripción
path		geometric path on a plane	ruta o camino en un plano
point		geometric point on a plane	punto geométrico en un plano
polygon		closed geometric path on a plane	polígono cerrado en un plano
real	float4	single precision floating-point number (4 bytes)	numero de punto flotante de precisión simple (4 bytes)
smallint	int2	signed two-byte integer	Entero de dos bytes con signo
serial	serial4	autoincrementing four-byte integer	entero auto incremental (4 bytes)
text		variable-length character string	Cadena de caracteres de longitud variable
time [(p)] [without time zone]		time of day (no time zone)	hora del día (sin zona horaria)

SQL (Tipos de Datos (4)

UNIVERSIDAD DE LOS ANDES VENEZUELA

Nombre	Alias	Description	Descripción
time [(p)] with time zone	timetz	time of day, including time zone	hora del día (con zona horaria)
timestamp [(p)] [without time zone]		date and time (no time zone)	Fecha y hora (sin zona horaria)
timestamp [(p)] with time zone	timestamptz	date and time, including time zone	Fecha y hora (con zona horaria)
tsquery		text search query	Consulta de búsqueda en texto
tsvector		text search document	Consulta de búsqueda en documento
txid_snapshot		user-level transaction ID snapshot	instantanea de identificador de transacción a nivel de usuario
uuid		universally unique identifier	identificador universal único
xml		XML data	Datos en XML

Crear una Tabla (Esquema de Relación) (2) (Claves Primarias)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE prueba (
           a INT PRIMARY KEY,
           b INT,
           c INT
         );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "prueba_pkey" for table "prueba"
CREATE TABLE
prueba=#
```

```
prueba=# CREATE TABLE prueba (
           a INT,
           b INT,
           c INT,
           PRIMARY KEY(a)
         );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "prueba_pkey" for table "prueba"
CREATE TABLE
prueba=#
```

Se puede hacer de cualquiera de las dos maneras

Crear una Tabla (Esquema de Relación) (3) (Claves Primarias)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE prueba (
            a INT,
            b INT,
            c INT,
            PRIMARY KEY(a, b)
          );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "prueba_pkey" for table "prueba"
CREATE TABLE
prueba=#
```

En este caso, tenemos una clave compuesta por a y b

Crear una Tabla (Esquema de Relación) (4) (Claves Foráneas)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE prueba1 (a INT, b INT, c INT,
            PRIMARY KEY(a));
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "prueba_pkey" for table "prueba1"
CREATE TABLE
prueba=#
```

```
prueba=# CREATE TABLE prueba2 (
            x INT,
            y INT,
            a INT REFERENCES prueba1,
            PRIMARY KEY(x)
          );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "prueba2_pkey" for table "prueba2"
CREATE TABLE
prueba=#
```

Crear una Tabla (Esquema de Relación) (5) (Claves Foráneas)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE prueba1 (a INT, b INT, c INT,
            PRIMARY KEY(a));
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "prueba_pkey" for table "prueba1"
CREATE TABLE
prueba=#
```

```
prueba=# CREATE TABLE prueba2 (
            x INT,
            y INT,
            a INT,
            FOREIGN KEY (a) REFERENCES prueba1 (a),
            PRIMARY KEY(x)
          );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "prueba2_pkey" for table "prueba2"
CREATE TABLE
prueba=#
```

Crear una Tabla (Esquema de Relación) (6) (Claves Foráneas)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE prueba1 (a INT, b INT, c INT,
            PRIMARY KEY(a, b));
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "prueba_pkey" for table "prueba1"
CREATE TABLE
prueba=#


prueba=# CREATE TABLE prueba2 (
            x INT,
            y INT,
            a INT,
            b INT,
            FOREIGN KEY (a, b) REFERENCES prueba1 (a, b),
            PRIMARY KEY(x)
          );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "prueba2_pkey" for table "prueba2"
CREATE TABLE
prueba=#
```

Crear una Tabla (Esquema de Relación) (7) (Validaciones / Check Constraints)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE producto1 (
            codigo INT,
            nombre VARCHAR(100),
            precio NUMERIC CHECK (precio > 0),
            PRIMARY KEY(codigo)
          );
CREATE TABLE
prueba=#
```

Darle nombre facilita entender los mensajes de error y cambiar la restricción posteriormente

```
prueba=# CREATE TABLE producto2 (
            codigo INT,
            nombre VARCHAR(100),
            precio NUMERIC CONSTRAINT c1 CHECK (precio > 0),
            PRIMARY KEY(codigo)
          );
CREATE TABLE
prueba=#
```

Crear una Tabla (Esquema de Relación) (8) (Validaciones / Check Constraints)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE producto1 (
            codigo INT,
            nombre VARCHAR(100),
            precio_ini NUMERIC CHECK (precio_ini > 0),
            precio_des NUMERIC CHECK (precio_des > 0),
            CHECK (precio_des < precio_ini),
            PRIMARY KEY(codigo)
         );
CREATE TABLE
prueba=#
```

Estas son restricciones de columna

Esta es una restricción de tabla

```
prueba=# CREATE TABLE producto2 (
            codigo INT,
            nombre VARCHAR(100),
            precio_ini NUMERIC,
            precio_des NUMERIC,
            CONSTRAINT c1 CHECK (precio_ini > 0),
            CONSTRAINT c2 CHECK (precio_des > 0),
            CONSTRAINT c3 CHECK (precio_des < precio_ini),
            PRIMARY KEY(codigo)
         );
CREATE TABLE
prueba=#
```

Estas son restricciones de tabla

Crear una Tabla (Esquema de Relación) (9) (Validaciones: Not Null)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE persona (
            cedula    INT          PRIMARY KEY,
            nombre    VARCHAR(50) NOT NULL,
            apellido  VARCHAR(50) NOT NULL,
            fecha_nac DATE
          );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "persona_pkey" for table "persona"
CREATE TABLE
prueba=#
```

El nombre y el apellido de una persona no pueden ser nulos
(Es decir, son obligatorios)

Crear una Tabla (Esquema de Relación) (10) (Validaciones: Unique)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE carro (
           placa         VARCHAR(6)  PRIMARY KEY,
           serial_motor  VARCHAR(50) UNIQUE,
           serial_carro  VARCHAR(50) UNIQUE,
           color         VARCHAR(10)
         );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "carro_pkey" for table "carro"
NOTICE:  CREATE TABLE / UNIQUE will create
  implicit index "carro_serial_motor_key" for table "carro"
NOTICE:  CREATE TABLE / UNIQUE will create
  implicit index "carro_serial_carro_key" for table "carro"
CREATE TABLE
prueba=#
```

El serial del motor y el serial del carrocería son atributos únicos para cada carro

Crear una Tabla (Esquema de Relación) (11) (Validaciones: Not Null + Unique)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE carro (
            placa         VARCHAR(6)  PRIMARY KEY,
            serial_motor  VARCHAR(50) NOT NULL UNIQUE,
            serial_carro  VARCHAR(50) NOT NULL UNIQUE,
            color         VARCHAR(10)
          );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "carro_pkey" for table "carro"
NOTICE:  CREATE TABLE / UNIQUE will create
  implicit index "carro_serial_motor_key" for table "carro"
NOTICE:  CREATE TABLE / UNIQUE will create
  implicit index "carro_serial_carro_key" for table "carro"
CREATE TABLE
prueba=#
```

El serial del motor y el serial del carrocería son atributos únicos para cada carro, y además, no pueden ser nulos

Crear una Tabla (Esquema de Relación) (12) (Valores por Defecto)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# CREATE TABLE producto (
            codigo      INT           PRIMARY KEY,
            nombre      VARCHAR(50),
            descripcion TEXT          DEFAULT 'N/A',
            precio      NUMERIC       DEFAULT 100
          );
NOTICE:  CREATE TABLE / PRIMARY KEY will create
  implicit index "producto_pkey" for table "producto"
CREATE TABLE
prueba=#
```

Si no se especifica, el valor de descripción y de precio por defecto es 'N/A' y 100 respectivamente. Adicionalmente, es posible actualizar una fila y asignar nuevamente el valor por defecto sin conocerlo

Eliminar una Tabla

```
prueba=# DROP TABLE persona;
DROP TABLE
prueba=# DROP TABLE persona;
ERROR:  table "persona" does not exist
prueba=# DROP TABLE IF EXISTS persona;
NOTICE:  table "persona" does not exist, skipping
DROP TABLE
```

Tener en cuenta que el `IF EXISTS` no es parte del estándar

```
prueba=# DROP TABLE prueba1;
NOTICE: constraint prueba2_a_fkey on table prueba2
  depends on table prueba1
ERROR:  cannot drop table prueba1 because other objects
  depend on it
HINT:   Use DROP ... CASCADE to drop the dependent
  objects too.

prueba=# DROP TABLE prueba1 CASCADE;
NOTICE: drop cascades to constraint prueba2_a_fkey
  on table prueba2
DROP TABLE
```

Insertar Entidades en una Tabla (1)

```
prueba=# CREATE TABLE producto (
           codigo INT, nombre VARCHAR(100), precio NUMERIC,
           PRIMARY KEY(codigo)
         );
CREATE TABLE
prueba=#
```

```
prueba=# INSERT INTO producto VALUES (1, 'nombre1', 1);
INSERT 0 1
prueba=#
```

```
prueba=# INSERT INTO producto (codigo, nombre, precio)
           VALUES (2, 'nombre2', 2);
INSERT 0 1
prueba=#
```

```
prueba=# INSERT INTO producto (codigo, nombre)
           VALUES (3, 'nombre3');
INSERT 0 1
prueba=#
```

```
prueba=# INSERT INTO producto VALUES (4, 'nombre4');
INSERT 0 1
prueba=#
```

Insertar Entidades en una Tabla (2)

```
prueba=# CREATE TABLE producto (
           codigo INT, nombre VARCHAR(100), precio NUMERIC,
           PRIMARY KEY(codigo)
         );
CREATE TABLE
prueba=#
```

```
prueba=# INSERT INTO producto (codigo, nombre, precio)
           VALUES (5, 'nombre5', DEFAULT);
INSERT 0 1
prueba=#
```

```
prueba=# INSERT INTO producto DEFAULT VALUES;
ERROR:  null value in column "codigo" violates
  not-null constraint
prueba=#
```

```
prueba=# INSERT INTO producto (codigo, nombre, precio)
           VALUES
           (6, 'nombre6', 10),
           (7, 'nombre7', 20),
           (8, 'nombre8', 30);
INSERT 0 3
prueba=#
```

Consultar Entidades en una Tabla (Lo Básico)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
prueba=# SELECT * FROM producto;
 codigo | nombre  | precio
--------+---------+--------
      1 | nombre1 |      1
      2 | nombre2 |      2
      3 | nombre3 |
      4 | nombre4 |
      5 | nombre5 |
      6 | nombre6 |     10
      7 | nombre7 |     20
      8 | nombre8 |     30
(8 rows)

prueba=#
```

El "SELECT" más simple que existe. En este caso lo usamos para ver los registros que insertamos usando los "INSERTs" anteriores

El arreglo (layout) de los datos suele variar entre los distintos gestores de BD

Actualizar Entidades en una Tabla

UNIVERSIDAD DE LOS ANDES VENEZUELA

Siguiendo con la tabla anterior...

(1)

```
prueba=# UPDATE producto SET precio = 100
           WHERE codigo=1;
UPDATE 1
prueba=#
```

---

(2)

```
prueba=# UPDATE producto SET precio = 30
           WHERE precio > 0 AND precio < 50;
UPDATE 4
prueba=#
```

---

(3)

```
prueba=# UPDATE producto SET precio = precio * 1.1;
UPDATE 8
prueba=#
```

Actualizar Entidades en una Tabla (Resultados)

UNIVERSIDAD DE LOS ANDES VENEZUELA

(1)

```
prueba=# SELECT * FROM producto;
 codigo | nombre  | precio
--------+---------+--------
      1 | nombre1 |    100
      2 | nombre2 |      2
      3 | nombre3 |
      4 | nombre4 |
      5 | nombre5 |
      6 | nombre6 |     10
      7 | nombre7 |     20
      8 | nombre8 |     30
```

(2)

```
prueba=# SELECT * FROM producto;
 codigo | nombre  | precio
--------+---------+--------
      3 | nombre3 |
      4 | nombre4 |
      5 | nombre5 |
      1 | nombre1 |    100
      2 | nombre2 |     30
      6 | nombre6 |     30
      7 | nombre7 |     30
      8 | nombre8 |     30
```

(3)

```
prueba=# SELECT * FROM producto;
 codigo | nombre  | precio
--------+---------+--------
      3 | nombre3 |
      4 | nombre4 |
      5 | nombre5 |
      1 | nombre1 |  110.0
      2 | nombre2 |   33.0
      6 | nombre6 |   33.0
      7 | nombre7 |   33.0
      8 | nombre8 |   33.0
```

Eliminar Entidades en una Tabla

Siguiendo con la tabla anterior...

(1)
```
prueba=# DELETE FROM producto
           WHERE precio = 33;
DELETE 4
prueba=#
```

---

(2)
```
prueba=# DELETE FROM producto;
DELETE 4
prueba=#
```

Eliminar Entidades en una Tabla (Resultados)

UNIVERSIDAD DE LOS ANDES VENEZUELA

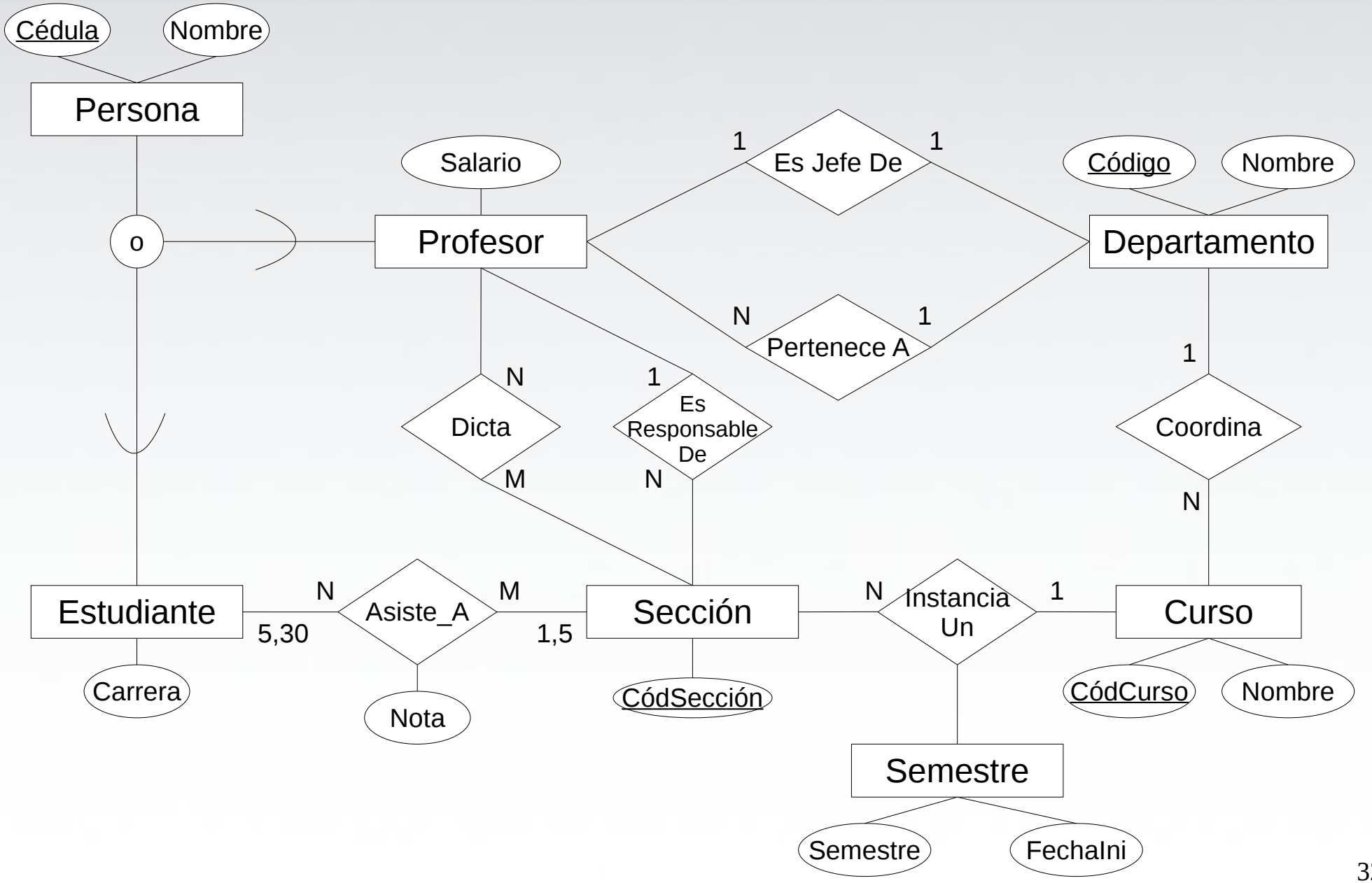
(1)

```
prueba=# SELECT * FROM producto;
 codigo | nombre  | precio
--------+---------+--------
      3 | nombre3 |
      4 | nombre4 |
      5 | nombre5 |
      1 | nombre1 |  110.0
```

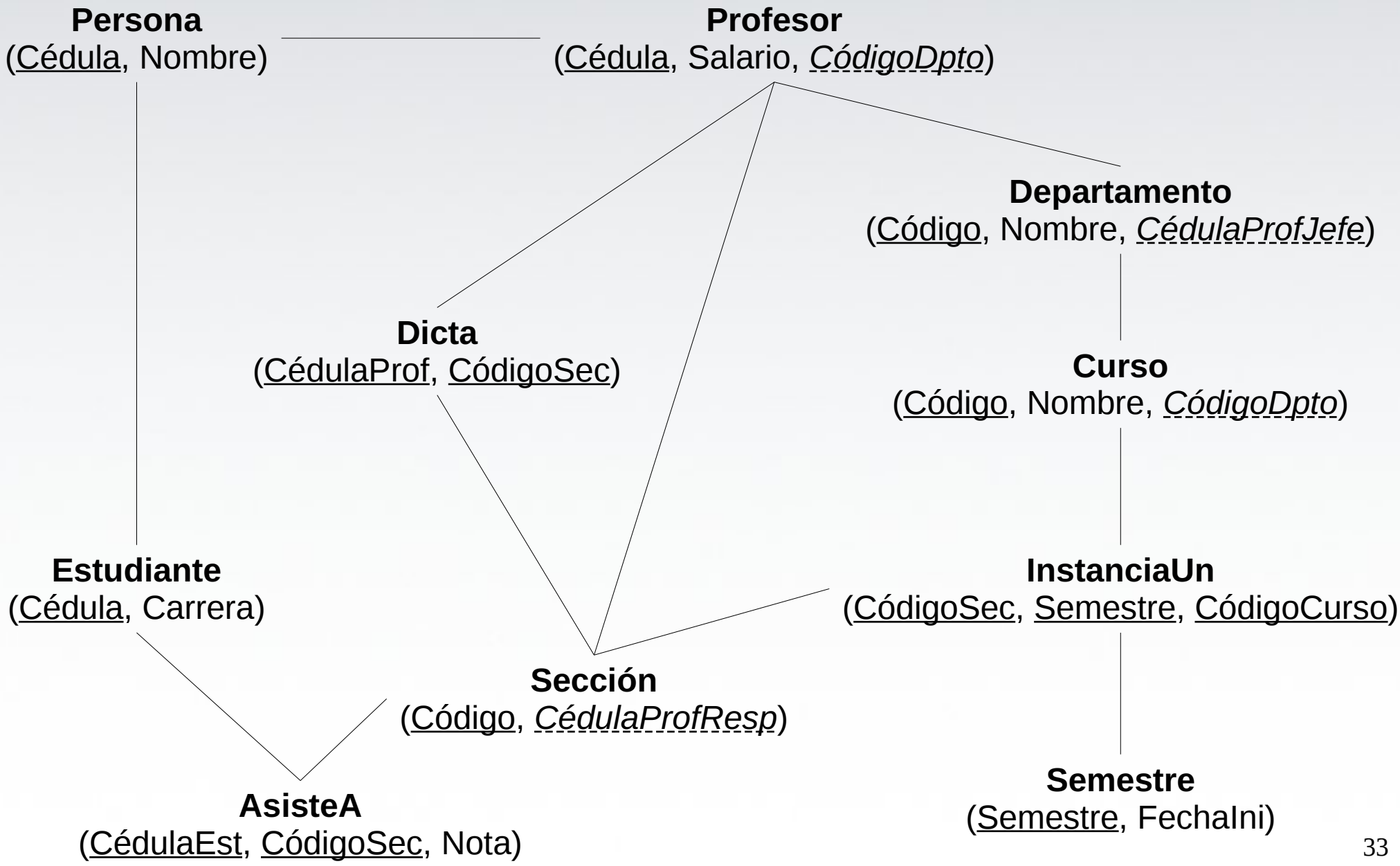
(2)

```
prueba=# SELECT * FROM producto;
 codigo | nombre | precio
--------+--------+--------
(0 rows)
```

Ejemplo (ERE)


Cédula
Nombre
Persona
Salario
Profesor
o
1
Es Jefe De
1
Código
Nombre
Departamento
N
Pertenece A
1
1
Coordina
N
N
Dicta
M
1
Es Responsable De
N
Estudiante
N
Asiste_A
M
5,30
1,5
Sección
N
Instancia Un
1
Curso
Carrera
Nota
CódSección
CódCurso
Nombre
Semestre
Semestre
FechaIni

Ejemplo (Esquema)



Ejemplo (Tablas / Filas)

cedula	nombre
12889883	Juan Guerra
11389998	Felipe Fernandez
14556334	Hector Loma
14443895	Jose Fuentes

codigo	nombre
2	Control
3	Investigacion
1	Computacion

codigo	nombre	codigodpto
1	Prog. Digital 1	1
2	Prog. Digital 2	1
3	Control	2
4	Instrumentacion	2
5	Astrofisica	

cedula	nombre	sueldo	codigodpto
12489778	Pedro Perez	2000	1
13449543	Juan Rojas	2500	1
15669442	Jose Fuentes	2100	2
11639337	Miguel Redondo	2000	2
10274448	Andres Silva	2000	3
12539445	Luis Cardenas	3000	3

cedula	codigo
12889883	1
12889883	2
14443895	2
14443895	3
14443895	4

Consultas (1) (Básico)

UNIVERSIDAD DE LOS ANDES VENEZUELA

SELECT <atributos> FROM <tablas> WHERE <condición>

<atributos> es una lista de nombres de los atributos cuyos valores va a obtener la consulta (operador ∏ álg. rel.)

<tablas> es una lista de los nombres de las relaciones requeridas para procesar la consulta (operador **x** álg. Rel.)

<condición> es una expresión condicional (booleana) de búsqueda para identificar las tuplas que obtendrá la consulta (operador σ del álg. rel.)

$$\Pi_{<atributos>}(\sigma_{<condición>}(<t_1> \times <t_2> \times \ldots \times <t_n>))$$

Consultas (2) (Básico)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT * FROM profesor;
  cedula  |      nombre      | sueldo | codigodpto
----------+------------------+--------+------------
 12489778 | Pedro Perez      |   2000 |          1
 13449543 | Juan Rojas       |   2500 |          1
 15669442 | Jose Fuentes     |   2100 |          2
 11639337 | Miguel Redondo   |   2000 |          2
 10274448 | Andres Silva     |   2000 |          3
 12539445 | Luis Cardenas    |   3000 |          3
```

```
postgres=# SELECT * FROM profesor WHERE sueldo > 2200;
  cedula  |     nombre      | sueldo | codigodpto
----------+-----------------+--------+------------
 13449543 | Juan Rojas      |   2500 |          1
 12539445 | Luis Cardenas   |   3000 |          3
```

Consultas (3) (Condiciones)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT nombre, sueldo FROM profesor WHERE sueldo > 2100;
    nombre     | sueldo
---------------+--------
 Juan Rojas    |   2500
 Luis Cardenas |   3000


postgres=# SELECT nombre, sueldo
             FROM profesor
             WHERE sueldo > 2200 AND sueldo < 2900;
   nombre   | sueldo
------------+--------
 Juan Rojas |   2500


postgres=# SELECT nombre, sueldo
             FROM profesor
             WHERE sueldo BETWEEN 2200 AND 2900;
   nombre   | sueldo
------------+--------
 Juan Rojas |   2500
```

Consultas (4) (Expresiones)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT nombre, sueldo, sueldo*1.1, (sueldo*1.1-sueldo)
             FROM profesor
             WHERE sueldo > 2200 AND sueldo < 2900;
   nombre    | sueldo | ?column? | ?column?
-------------+--------+----------+----------
 Juan Rojas |   2500 |   2750.0 |    250.0
```

```
postgres=# SELECT nombre, sueldo, sueldo*1.1, (sueldo*1.1–sueldo)
             FROM profesor
             WHERE (sueldo * 1.1 - sueldo) < 250;
     nombre      | sueldo | ?column? | ?column?
-----------------+--------+----------+----------
 Pedro Perez     |   2000 |   2200.0 |    200.0
 Jose Fuentes    |   2100 |   2310.0 |    210.0
 Miguel Redondo  |   2000 |   2200.0 |    200.0
 Andres Silva    |   2000 |   2200.0 |    200.0
```

Consultas (5)
(Cadenas y Subcadenas)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT * FROM profesor WHERE nombre='Pedro Perez';
  cedula  |   nombre     | sueldo | codigodpto
----------+--------------+--------+------------
 12489778 | Pedro Perez |   2000 |          1
(1 row)


postgres=# SELECT * FROM profesor WHERE nombre LIKE '%es%';
  cedula  |    nombre     | sueldo | codigodpto
----------+---------------+--------+------------
 15669442 | Jose Fuentes |   2100 |          2
 10274448 | Andres Silva |   2000 |          3
(2 rows)


postgres=# SELECT * FROM profesor WHERE nombre LIKE '%es';
  cedula  |    nombre     | sueldo | codigodpto
----------+---------------+--------+------------
 15669442 | Jose Fuentes |   2100 |          2
(1 row)
```

¿Por qué no encontró a Andres Silva?

Consultas (6)
(Cadenas y Subcadenas)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT * FROM profesor WHERE nombre LIKE '% _ _ _as';
  cedula  |   nombre    | sueldo | codigodpto
----------+-------------+--------+------------
 13449543 | Juan Rojas  |   2500 |          1
(1 row)
```

```
% ---as
```

```
postgres=# SELECT * FROM profesor WHERE nombre LIKE '%ue _  %';
  cedula  |     nombre      | sueldo | codigodpto
----------+-----------------+--------+------------
 11639337 | Miguel Redondo  |   2000 |          2
(1 row)
```

```
%ue- %
```

```
postgres=# SELECT * FROM profesor WHERE nombre < 'Luis';
  cedula  |    nombre     | sueldo | codigodpto
----------+---------------+--------+------------
 13449543 | Juan Rojas    |   2500 |          1
 15669442 | Jose Fuentes  |   2100 |          2
 10274448 | Andres Silva  |   2000 |          3
(3 rows)
```

¿Es esto posible realmente? ¿Qué significa? ¿Qué utilidad tiene?

Consultas (7) (Ordenación)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT * FROM profesor ORDER BY nombre;
  cedula  |     nombre      | sueldo | codigodpto
----------+-----------------+--------+------------
 10274448 | Andres Silva    |   2000 |          3
 15669442 | Jose Fuentes    |   2100 |          2
 13449543 | Juan Rojas      |   2500 |          1
 12539445 | Luis Cardenas   |   3000 |          3
 11639337 | Miguel Redondo  |   2000 |          2
 12489778 | Pedro Perez     |   2000 |          1
(6 rows)
```

```
postgres=# SELECT * FROM profesor ORDER BY sueldo;
  cedula  |     nombre      | sueldo | codigodpto
----------+-----------------+--------+------------
 12489778 | Pedro Perez     |   2000 |          1
 11639337 | Miguel Redondo  |   2000 |          2
 10274448 | Andres Silva    |   2000 |          3
 15669442 | Jose Fuentes    |   2100 |          2
 13449543 | Juan Rojas      |   2500 |          1
 12539445 | Luis Cardenas   |   3000 |          3
(6 rows)
```

Consultas (8) (Ordenación)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT * FROM profesor ORDER BY sueldo DESC;
  cedula   |     nombre      | sueldo | codigodpto
-----------+-----------------+--------+------------
 12539445 | Luis Cardenas   |   3000 |          3
 13449543 | Juan Rojas      |   2500 |          1
 15669442 | Jose Fuentes    |   2100 |          2
 12489778 | Pedro Perez     |   2000 |          1
 11639337 | Miguel Redondo  |   2000 |          2
 10274448 | Andres Silva    |   2000 |          3
(6 rows)

postgres=# SELECT * FROM profesor ORDER BY sueldo ASC;
  cedula   |     nombre      | sueldo | codigodpto
-----------+-----------------+--------+------------
 12489778 | Pedro Perez     |   2000 |          1
 11639337 | Miguel Redondo  |   2000 |          2
 10274448 | Andres Silva    |   2000 |          3
 15669442 | Jose Fuentes    |   2100 |          2
 13449543 | Juan Rojas      |   2500 |          1
 12539445 | Luis Cardenas   |   3000 |          3
(6 rows)
```

Consultas (9) (Ordenación)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT nombre, sueldo
             FROM profesor
             ORDER BY sueldo DESC, nombre;
     nombre      | sueldo
-----------------+--------
 Luis Cardenas   |   3000
 Juan Rojas      |   2500
 Jose Fuentes    |   2100
 Andres Silva    |   2000
 Miguel Redondo  |   2000
 Pedro Perez     |   2000
(6 rows)
```

Es posible ordenar por dos o más campos

Es posible ordenar en base a expresiones, Ej:
ORDER BY sueldo * 1.1 - sueldo

Consultas (10) (Funciones Agregadas)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT COUNT(*)    FROM profesor;
 count
-------
     6
```

```
postgres=# SELECT SUM(sueldo) FROM profesor;
  sum
-------
 13600
```

```
postgres=# SELECT AVG(sueldo) FROM profesor;
          avg
-----------------------
 2266.666666666666667
```

```
postgres=# SELECT MIN(sueldo) FROM profesor;
 min
------
 2000
```

```
postgres=# SELECT MAX(sueldo) FROM profesor;
 max
------
 3000
```

Consultas (11)
(Funciones Agregadas / Agrupación)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT
               codigodpto, COUNT(*),
               MAX(sueldo), AVG(sueldo), MIN(sueldo), SUM(sueldo)
             FROM profesor
             GROUP BY codigodpto
             ORDER BY codigodpto;
 codigodpto | count | max  |          avg           | min  | sum
------------+-------+------+------------------------+------+------
          1 |     2 | 2500 | 2250.0000000000000000 | 2000 | 4500
          2 |     2 | 2100 | 2050.0000000000000000 | 2000 | 4100
          3 |     2 | 3000 | 2500.0000000000000000 | 2000 | 5000
(3 rows)
```

```
postgres=# SELECT cedula, codigodpto, MAX(sueldo)
             FROM profesor GROUP BY codigodpto;
ERROR: column "profesor.cedula" must appear in the GROUP BY clause
  or be used in an aggregate function
postgres=#
```

Consultas (11)
(Funciones Agregadas / Agrupación / HAVING)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT
               codigodpto, COUNT(*),
               MAX(sueldo), AVG(sueldo), MIN(sueldo), SUM(sueldo)
           FROM profesor
           GROUP BY codigodpto HAVING AVG(sueldo) < 2400
           ORDER BY codigodpto;
 codigodpto | count | max  |          avg           | min  | sum
------------+-------+------+------------------------+------+------
          1 |     2 | 2500 | 2250.0000000000000000 | 2000 | 4500
          2 |     2 | 2100 | 2050.0000000000000000 | 2000 | 4100
(2 rows)
```

Esto no implica que aquí no pueda haber un [WHERE condición] opcional que me permite filtrar las filas ***antes*** de ser agrupadas

HAVING es similar a una clausula WHERE, pero me permite filtrar las filas ***luego*** de ser agrupadas

Consultas (12) (Producto / Reunión)

```
postgres=# SELECT
                cedula, profesor.nombre, sueldo, departamento.nombre
              FROM profesor, departamento
              WHERE codigo=codigoDpto;
  cedula   |      nombre      | sueldo |     nombre
-----------+------------------+--------+----------------
 12489778 | Pedro Perez      |   2000 | Computacion
 13449543 | Juan Rojas       |   2500 | Computacion
 15669442 | Jose Fuentes     |   2100 | Control
 11639337 | Miguel Redondo   |   2000 | Control
 10274448 | Andres Silva     |   2000 | Investigacion
 12539445 | Luis Cardenas    |   3000 | Investigacion
```

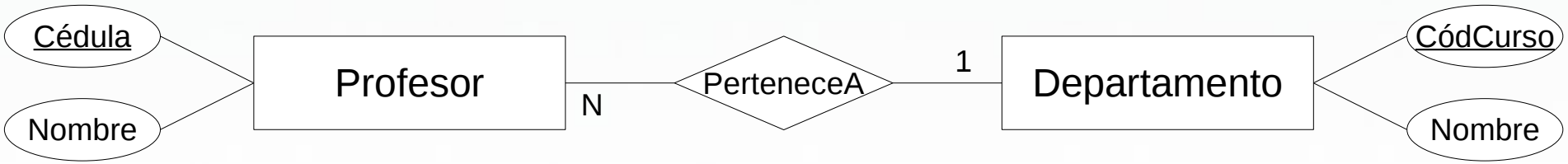
Hay más de dos tablas en el FROM

$$\Pi_{\text{cedula, profesor.nombre, sueldo, departamento.nombre}}(\sigma_{\text{codigo=codigoDpto}}(\text{profesor x departamento}))$$

```
postgres=# SELECT cedula, nombre, sueldo, nombre
              FROM profesor, departamento
              WHERE codigo=codigoDpto;
ERROR:  column reference "nombre" is ambiguous
LINE 1: SELECT cedula, nombre, sueldo, nombre FROM profesor...
```

Consultas (13)
(Producto / Reunión / Alias)

```
postgres=# SELECT
                cedula, profesor.nombre AS nombre_prof, sueldo,
                departamento.nombre AS nombre_dpto
              FROM profesor, departamento
              WHERE codigo=codigoDpto;
  cedula  |  nombre_prof    | sueldo |  nombre_dpto
----------+-----------------+--------+---------------
 12489778 | Pedro Perez     |   2000 | Computacion
 13449543 | Juan Rojas      |   2500 | Computacion
 15669442 | Jose Fuentes    |   2100 | Control
 11639337 | Miguel Redondo  |   2000 | Control
 10274448 | Andres Silva    |   2000 | Investigacion
 12539445 | Luis Cardenas   |   3000 | Investigacion
(6 rows)
```

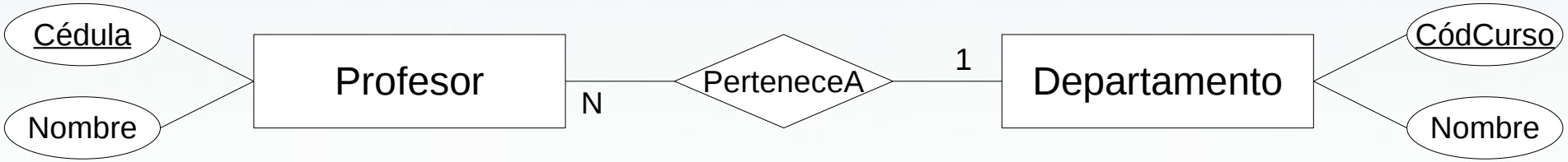


codigo=codigoDpto

Usualmente se le llama condición de reunión

Consultas (14)
(Producto / Reunión / Alias)

```
postgres=# SELECT
                cedula, p.nombre AS nombre_prof, sueldo,
                d.nombre AS nombre_dpto
              FROM profesor AS p, departamento AS d
              WHERE
                d.codigo=p.codigoDpto AND d.nombre='Computacion';
  cedula  | nombre_prof | sueldo | nombre_dpto
----------+-------------+--------+-------------
 12489778 | Pedro Perez |   2000 | Computacion
 13449543 | Juan Rojas  |   2500 | Computacion
(2 rows)
```



En lo que respecta a la consulta, la tabla persona fue renombrada como "***p***" y la tabla departamento fue renombrada como "***d***"

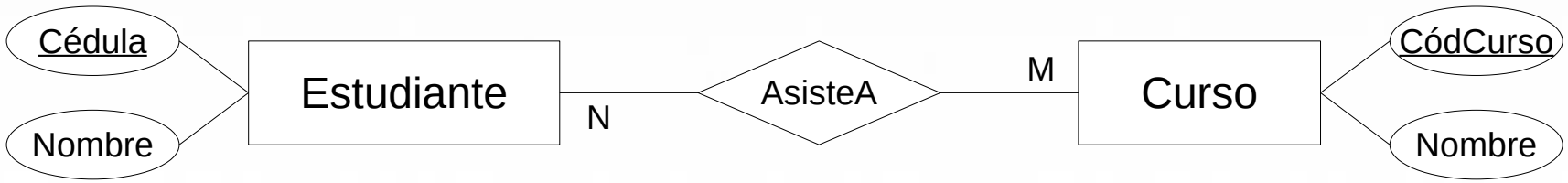
Los alias permiten acortar algunos nombres y darle legibilidad a las sentencias SQL... Sin embargo... no abuse en su uso, porque puede lograr el efecto contrario

Consultas (15) (Producto / Reunión)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT *
             FROM estudiante, asiste, curso
             WHERE estudiante.cedula = asiste.cedula
               AND asiste.codigo = curso.codigo;
```

```
 cedula |   nombre   | cedula |codigo|codigo|    nombre     |codigodpto
--------+------------+--------+------+------+---------------+----------
12889883|Juan Guerra |12889883|     1|     1|Prog. Digital 1|          1
12889883|Juan Guerra |12889883|     2|     2|Prog. Digital 2|          1
14443895|Jose Fuentes|14443895|     2|     2|Prog. Digital 2|          1
14443895|Jose Fuentes|14443895|     3|     3|Control        |          2
14443895|Jose Fuentes|14443895|     4|     4|Instrumentacion|          2
(5 rows)
```

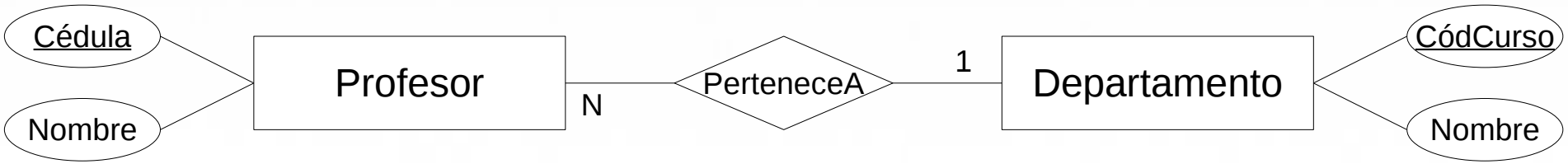


Consultas (16)
(Producto / Reunión usando JOIN)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT *
             FROM profesor
               JOIN departamento
               ON profesor.codigoDpto = departamento.codigo;
```

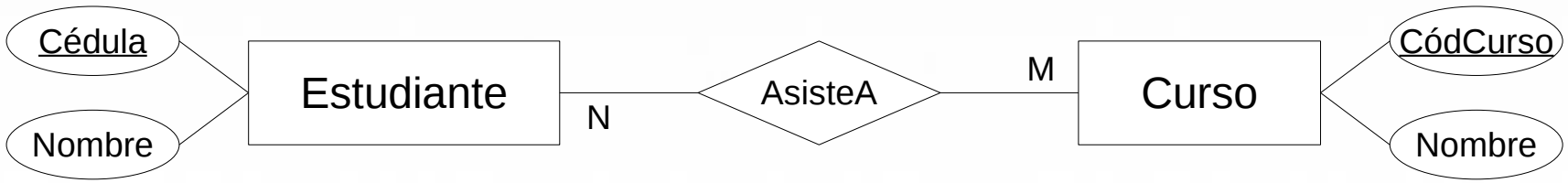
```
 cedula |    nombre    |sueldo|codigodpto|codigo|  nombre
--------+--------------+------+----------+------+-------------
12489778|Pedro Perez   |  2000|         1|     1|Computacion
13449543|Juan Rojas    |  2500|         1|     1|Computacion
15669442|Jose Fuentes  |  2100|         2|     2|Control
11639337|Miguel Redondo|  2000|         2|     2|Control
10274448|Andres Silva  |  2000|         3|     3|Investigacion
12539445|Luis Cardenas |  3000|         3|     3|Investigacion
(6 rows)
```



Consultas (17)
(Producto / Reunión usando JOIN N:M)

```
postgres=# SELECT *
             FROM estudiante
              JOIN asiste
              ON estudiante.cedula = asiste.cedula
              JOIN curso
              ON asiste.codigo = curso.codigo;
```

```
 cedula |   nombre    | cedula |codigo|codigo|    nombre     |codigodpto
--------+-------------+--------+------+------+---------------+----------
12889883|Juan Guerra  |12889883|     1|     1|Prog. Digital 1|          1
12889883|Juan Guerra  |12889883|     2|     2|Prog. Digital 2|          1
14443895|Jose Fuentes |14443895|     2|     2|Prog. Digital 2|          1
14443895|Jose Fuentes |14443895|     3|     3|Control        |          2
14443895|Jose Fuentes |14443895|     4|     4|Instrumentacion|          2
(5 rows)
```



Ejemplo (Tablas / Filas)

cedula	nombre
12889883	Juan Guerra
11389998	Felipe Fernandez
14556334	Hector Loma
14443895	Jose Fuentes

codigo	nombre
2	Control
3	Investigacion
1	Computacion

codigo	nombre	codigodpto
1	Prog. Digital 1	1
2	Prog. Digital 2	1
3	Control	2
4	Instrumentacion	2
5	Astrofisica	

cedula	nombre	sueldo	codigodpto
12489778	Pedro Perez	2000	1
13449543	Juan Rojas	2500	1
15669442	Jose Fuentes	2100	2
11639337	Miguel Redondo	2000	2
10274448	Andres Silva	2000	3
12539445	Luis Cardenas	3000	3

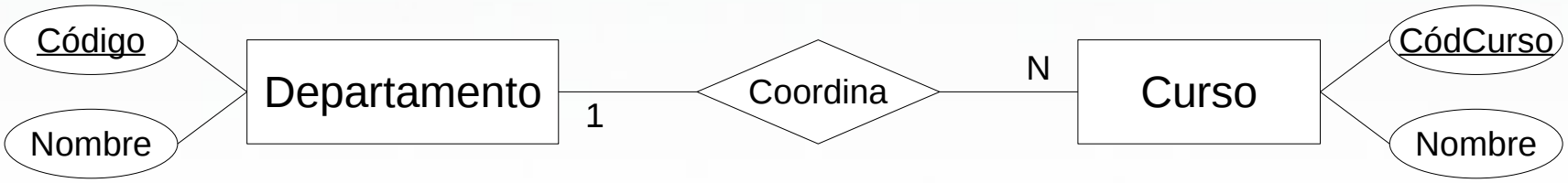
cedula	codigo
12889883	1
12889883	2
14443895	2
14443895	3
14443895	4

Consultas (18)
(Producto / Reunión usando LEFT JOIN)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT *
             FROM departamento
               JOIN curso
               ON departamento.codigo = curso.codigoDpto;

 codigo |   nombre    | codigo |     nombre       | codigodpto
--------+-------------+--------+------------------+------------
      1 | Computacion |      1 | Prog. Digital 1  |          1
      1 | Computacion |      2 | Prog. Digital 2  |          1
      2 | Control     |      3 | Control          |          2
      2 | Control     |      4 | Instrumentacion  |          2
(4 rows)
```

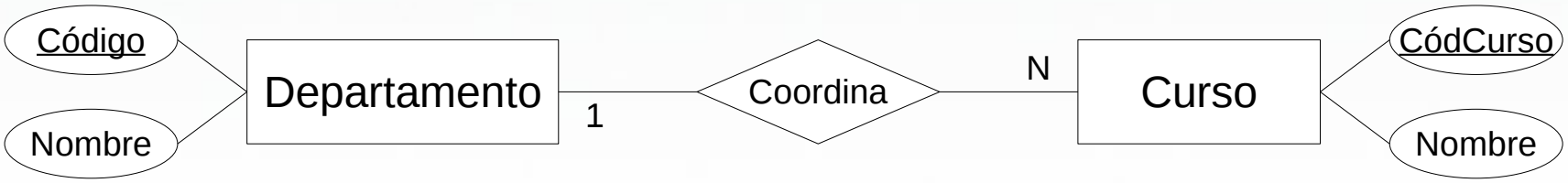


¿Y el departamento de investigación? ¿Qué pasa si pregunto por los departamentos que ***no*** tienen cursos?

Consultas (19)
(Producto / Reunión usando LEFT JOIN)

```
postgres=# SELECT *
             FROM departamento
               LEFT JOIN curso
               ON departamento.codigo = curso.codigoDpto;

 codigo |    nombre     | codigo |     nombre       | codigodpto
--------+---------------+--------+------------------+------------
      2 | Control       |      4 | Instrumentacion  |          2
      2 | Control       |      3 | Control          |          2
      3 | Investigacion |   NULL | NULL             |       NULL
      1 | Computacion   |      2 | Prog. Digital 2  |          1
      1 | Computacion   |      1 | Prog. Digital 1  |          1
(5 rows)
```

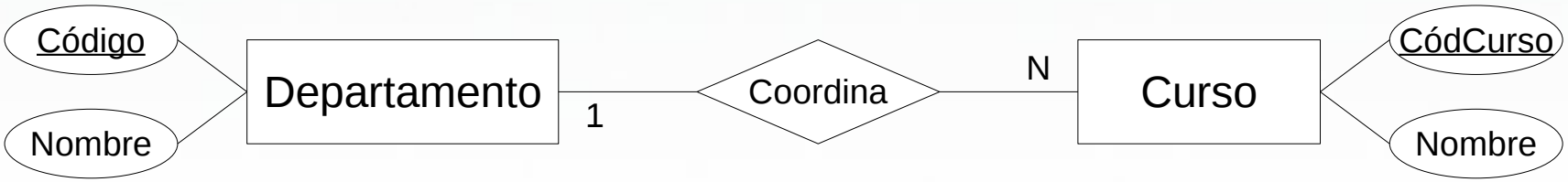


¡Aquí está!

Consultas (19)
(Producto / Reunión usando LEFT JOIN)

```
postgres=# SELECT departamento.codigo, departamento.nombre
             FROM departamento
               LEFT JOIN curso
               ON departamento.codigo = curso.codigoDpto
             WHERE curso.codigo IS NULL;
;
 codigo |    nombre
--------+---------------
      3 | Investigacion
(1 row)
```

IS NULL se usa para comparar atributos con valores nulos



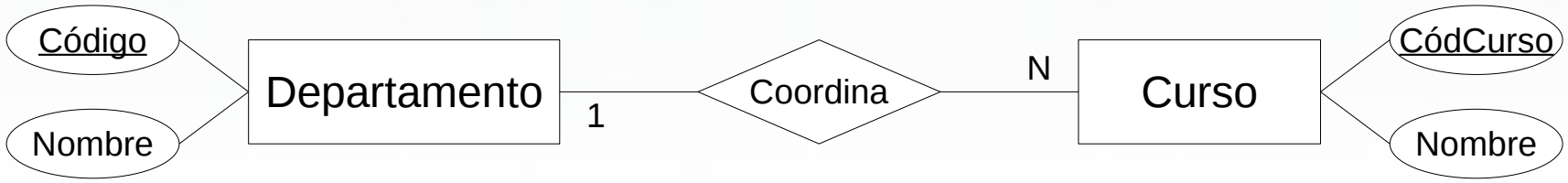
¿Qué pasa si pregunto por los departamentos que ***no*** tienen cursos? (***¡La Respuesta!***)

Consultas (20)
(Producto / Reunión usando RIGHT JOIN)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT *
             FROM curso
               JOIN departamento
               ON curso.codigoDpto = departamento.codigo;

 codigo |     nombre       | codigodpto | codigo |   nombre
--------+------------------+------------+--------+-------------
      1 | Prog. Digital 1 |          1 |      1 | Computacion
      2 | Prog. Digital 2 |          1 |      1 | Computacion
      3 | Control          |          2 |      2 | Control
      4 | Instrumentacion |          2 |      2 | Control
(4 rows)
```



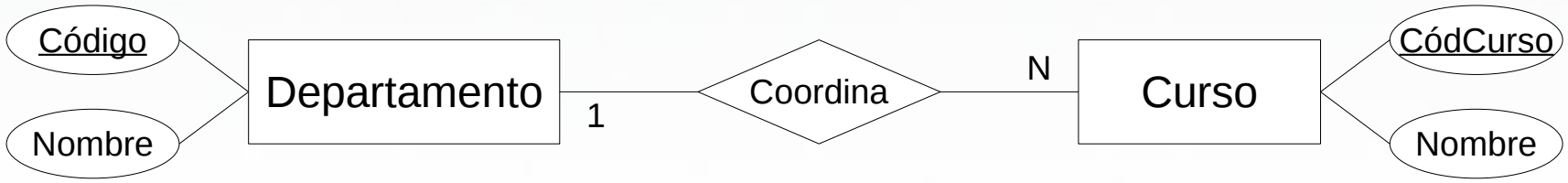
¿Y el departamento de investigación? (Nuevamente)

Consultas (21)
(Producto / Reunión usando RIGHT JOIN)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT *
             FROM curso
               RIGHT JOIN departamento
               ON curso.codigoDpto = departamento.codigo;

 codigo |     nombre       | codigodpto | codigo |    nombre
--------+------------------+------------+--------+---------------
      4 | Instrumentacion |          2 |      2 | Control
      3 | Control          |          2 |      2 | Control
   NULL | NULL             |       NULL |      3 | Investigacion
      2 | Prog. Digital 2 |          1 |      1 | Computacion
      1 | Prog. Digital 1 |          1 |      1 | Computacion
(5 rows)
```

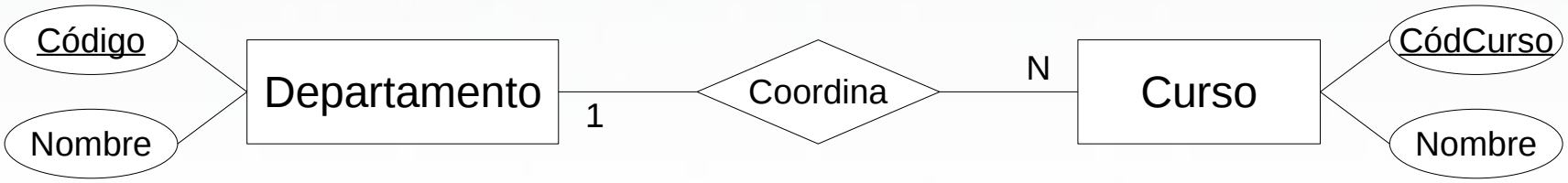


¡Aquí está! (Nuevamente)

Consultas (22)
(Producto / Reunión usando FULL JOIN)

```
postgres=# SELECT *
             FROM curso
               FULL JOIN departamento
               ON curso.codigoDpto = departamento.codigo;

 codigo |     nombre       | codigodpto | codigo |    nombre
--------+------------------+------------+--------+---------------
      1 | Prog. Digital 1  |          1 |      1 | Computacion
      2 | Prog. Digital 2  |          1 |      1 | Computacion
      3 | Control          |          2 |      2 | Control
      4 | Instrumentacion  |          2 |      2 | Control
      5 | Astrofisica      |       NULL |   NULL | NULL
   NULL | NULL             |       NULL |      3 | Investigacion
```



¡Aquí está! (Nuevamente)

Consultas (23) (Subconsultas)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT *
             FROM
               (SELECT cedula, profesor.nombre AS nombre_prof, sueldo,
                       departamento.nombre AS nombre_dpto
                  FROM profesor, departamento WHERE codigo=codigoDpto)
             AS x WHERE nombre_dpto='Computacion';

   cedula  | nombre_prof | sueldo | nombre_dpto
 ----------+-------------+--------+-------------
  12489778 | Pedro Perez |   2000 | Computacion
  13449543 | Juan Rojas  |   2500 | Computacion
```

```
postgres=# SELECT cedula, profesor.nombre AS nombre_prof, sueldo,
                  departamento.nombre AS nombre_dpto
             FROM profesor, departamento WHERE codigo=codigoDpto;

   cedula  |  nombre_prof    | sueldo |  nombre_dpto
 ----------+-----------------+--------+---------------
  12489778 | Pedro Perez     |   2000 | Computacion
  13449543 | Juan Rojas      |   2500 | Computacion
  15669442 | Jose Fuentes    |   2100 | Control
  11639337 | Miguel Redondo  |   2000 | Control
  10274448 | Andres Silva    |   2000 | Investigacion
  12539445 | Luis Cardenas   |   3000 | Investigacion
```

Ejemplo (Tablas / Filas)

cedula	nombre
12889883	Juan Guerra
11389998	Felipe Fernandez
14556334	Hector Loma
14443895	Jose Fuentes

codigo	nombre
2	Control
3	Investigacion
1	Computacion

codigo	nombre	codigodpto
1	Prog. Digital 1	1
2	Prog. Digital 2	1
3	Control	2
4	Instrumentacion	2
5	Astrofisica	

cedula	nombre	sueldo	codigodpto
12489778	Pedro Perez	2000	1
13449543	Juan Rojas	2500	1
15669442	Jose Fuentes	2100	2
11639337	Miguel Redondo	2000	2
10274448	Andres Silva	2000	3
12539445	Luis Cardenas	3000	3

cedula	codigo
12889883	1
12889883	2
14443895	2
14443895	3
14443895	4

Consultas (24) (Conjuntos)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# SELECT sueldo FROM profesor;
 sueldo
--------
   2000
   2500
   2100
   2000
   2000
   3000
(6 rows)

postgres=# SELECT DISTINCT sueldo FROM profesor;
 sueldo
--------
   2000
   2100
   2500
   3000
(4 rows)
```

DISTINCT simplemente elimina los valores repetidos del resultado de la consulta

Consultas (25) (Conjuntos UNION)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# (SELECT nombre FROM estudiante)
           UNION
           (SELECT nombre FROM profesor);
      nombre
------------------
 Andres Silva
 Felipe Fernandez
 Hector Loma
 Jose Fuentes
 Juan Guerra
 Juan Rojas
 Luis Cardenas
 Miguel Redondo
 Pedro Perez
(9 rows)
```

Estudiante ∪ Profesor

Las operaciones de conjuntos (unión intersección y diferencia automáticamente eliminan las filas duplicadas

Consultas (26) (Conjuntos UNION)

```
postgres=# (SELECT nombre FROM estudiante)
           UNION ALL
           (SELECT nombre FROM profesor);
      nombre
-----------------
 Juan Guerra
 Felipe Fernandez
 Hector Loma
 Jose Fuentes
 Pedro Perez
 Juan Rojas
 Jose Fuentes
 Miguel Redondo
 Andres Silva
 Luis Cardenas
(10 rows)
```

Para evitar la eliminación de los duplicados hay que utilizar la clausula ALL después de la operación de conjuntos

Consultas (26)
(Conjuntos INTERSECT / EXCEPT)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# (SELECT nombre FROM estudiante)
           INTERSECT
           (SELECT nombre FROM profesor);
    nombre
--------------
 Jose Fuentes
(1 row)
```

```
postgres=# (SELECT nombre FROM estudiante)
           EXCEPT
           (SELECT nombre FROM profesor);
      nombre
------------------
 Felipe Fernandez
 Hector Loma
 Juan Guerra
(3 rows)
```

Estudiante ∩ Profesor, Estudiante - Profesor

Aplican las misma reglas en relación los duplicados que las que aplican a la unión

Consultas (27)
(Conjuntos LIMIT / OFFSET)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# (SELECT cedula, nombre FROM estudiante)
           UNION ALL
           (SELECT cedula, nombre FROM profesor)
           ORDER BY cedula;
  cedula  |       nombre
----------+--------------------
 10274448 | Andres Silva
 11389998 | Felipe Fernandez
 11639337 | Miguel Redondo
 12489778 | Pedro Perez
 12539445 | Luis Cardenas
 12889883 | Juan Guerra
 13449543 | Juan Rojas
 14443895 | Jose Fuentes
 14556334 | Hector Loma
 15669442 | Jose Fuentes
(10 rows)
```

1er Grupo de 3 registros

2do Grupo de 3 registros

3er Grupo de 3 registros

4to Grupo de 3 registros
(Pero sólo hay 1)

LIMIT y OFFSET se utilizan para limitar la cantidad de registros que se muestran, y desplazar la posición de inicio a partir de la cual se comienzan a mostrar los registros. Muy útil para ***paginar*** registros.

Consultas (28)
(Conjuntos LIMIT / OFFSET)

UNIVERSIDAD DE LOS ANDES VENEZUELA

```
postgres=# (SELECT cedula, nombre FROM estudiante)
           UNION ALL
           (SELECT cedula, nombre FROM profesor)
           ORDER BY cedula LIMIT 3 OFFSET 0;
  cedula  |      nombre
----------+------------------
 10274448 | Andres Silva
 11389998 | Felipe Fernandez
 11639337 | Miguel Redondo
```

```
postgres=# (SELECT cedula, nombre FROM estudiante)
           UNION ALL
           (SELECT cedula, nombre FROM profesor)
           ORDER BY cedula LIMIT 3 OFFSET 3;
  cedula  |    nombre
----------+---------------
 12489778 | Pedro Perez
 12539445 | Luis Cardenas
 12889883 | Juan Guerra
```

La clausula ORDER BY es importante para predecir el orden en que aparecen los registros y mantener la consistencia de las distintas páginas

UNIVERSIDAD DE LOS ANDES VENEZUELA

¡Gracias!