CC42A: Bases de datos

Auxiliar: SQL-92

Profesores: Claudio Gutiérrez, Gonzalo Navarro
Prof. Auxiliar: Rodrigo Paredes

Parte 1. SQL

El efecto producido por los conceptos de modelo relacional los lenguajes relacionales (álgebra y cálculo) introducidos por E. Codd provocó tal impacto que significó que comunidades comerciales y de investigación desarrollaran implementaciones de los lenguajes relacionales. Tal vez el mayor esfuerzo se lo llevó *SQL* (Structured Query Languaje), que junto con permitir consultas en la base de datos, contiene primitivas de definición de tablas, actualización de la base de datos, definición de vistas otorgamientos de privilegios, etc. A continuación se mostrarán aspectos del estándar ANSI de 1992, conocido como SQL-92.

Aún cuando las distintos administradores de bases de datos (MySQL, posgreSQL, ORACLE, etc.) siguen la definición ANSI SQL-92, es necesario verificar con el manual del administración de la base de datos específica definiciones particulares tales como el tipo de los datos predefinidos, las funciones, etc.

Definición de tablas y esquemas

Definición de Esquemas

La definición de un esquema es simple. Sólo se necesita identificar el comienzo de la definición con una instrucción CREATE SCHEMA y una cláusula adicional AUTHORIZATION y a continuación definir cada dominio, tabla, vista y demás en el esquema, por ejemplo:

```
CREATE SCHEMA EMPRESA_CL
  AUTHORIZATION DUEÑO
    definición de dominios
    definición de tablas
    definición de vistas
    etc.
```

El dueño del esquema, o propietario del esquema puede otorgar privilegios de acceso y actualización de la base de datos definida en el esquema a otros usuarios del sistema.

Tipos de datos y dominios

Un dominio es un conjunto del cual toma sus valores una columna de una relación. Según este concepto, los tipos de datos predefinidos son dominios. Adicionalmente SQL-92 permite la definición de dominios por parte de los usuarios.

Numéricos exactos:

Integer	enteros
Small Integer	enteros pequeños
Numeric(p, e)	p: precisión: total de números o dígitos en el número e: escala: cuantos números están a la derecha del punto decimal

Decimal(p,e)	p: precisión e: escala

Numéricos aproximados:

Real	
Double precision	doble precisión
float	flotante

Estos tipos de datos se utilizan normalmente para cálculos científicos y de ingeniería.

Cadenas de caracteres:

Character(n)	carácter
Character varying(n)	carácter variable

Los campos de character siempre almacenan *n* caracteres, aún cuando tengan que rellenar con blancos a la derecha para completar la longitud *n*. Los campos character varying sólo almacenan el número real de caracteres que se introdujeron (hasta un máximo de *n*).

Cadenas de bits:

Bit(n)
Bit varying(n)

Estos campos se usan para banderas u otras máscaras de bits para el control.

Fechas y horas:

Date	fecha
Time	hora
Timestamp	sello de tiempo
Time con tiempo zonal	
Timestamp con tiempo zonal	

El tipo Date se da en el orden año, mes, día con cuatro dígitos para el año. El Time se da en horas (0 a 23), minutos, segundos y décimas de segundos. El Timestamp es la fecha más la hora.

Intervalos:

Year-month	año-mes
Day-time	día-hora

Un intervalo es la diferencia entre dos fechas (año-mes) o entre dos horas (día-hora).

Definición de dominios:

Los tipos de datos con **restricciones** (constrains) y **valores por defecto** (default values) se pueden combinar en la definición de dominios. Una definición de dominio es un tipo de datos especializado que puede estar definido dentro de un esquema y utilizado en la definición de columnas. Por ejemplo:

```
CREATE DOMAIN IDENTIFICADOR NUMERIC(4) DEFAULT 0
  CHECK (VALUE IS NOT NULL)
```

Esta definición dice que un dominio llamado IDENTIFICADOR tiene las siguientes propiedades:

1. Su tipo de datos es numérico de cuatro dígitos.
2. Su valor por defecto es 0.
3. Nunca puede ser nulo.

Definición de Tablas

Las tablas se definen en tres pasos:

1. Dar el nombre de la tabla.
2. Definir cada columna, posiblemente incluyendo restricciones de columna.
3. Definir las restricciones de la tabla.

Siguiendo el ejemplo de EMPRESA_CL y del dominio IDENTIFICADOR:

```
CREATE SCHEMA EMPRESA_CL
  AUTHORIZATION DUEÑO

CREATE DOMAIN IDENTIFICADOR NUMERIC(4) DEFAULT 0
  CHECK (VALUE IS NOT NULL)

CREATE TABLE TRABAJADOR (
  TRA_ID         IDENTIFICADOR    PRIMARY KEY,
  TRA_NOMBRE     CHARACTER(12),
  TRA_TARIFA_HR  NUMERIC(5,2),
  TRA_OFICIO     CHARACTER(8),
  TRA_SUP        NUMERIC(4),
  FOREIGN KEY ID_SUPV REFERENCES TRABAJADOR
                 ON DELETE SET NULL
)

CREATE TABLE ASIGNACION (
  ASG_ID_TRABAJADOR    IDENTIFICADOR,
  ASG_ID_EDIFICIO      IDENTIFICADOR,
  ASG_FECHA_INICIO     DATE,
  ASG_NUM_DIAS         INTERVAL DAY (3),
  PRIMARY KEY (ASG_ID_TRABAJADOR, ASG_ID_EDIFICIO),
  FOREIGN KEY ASG_ID_TRABAJADOR REFERENCES TRABAJADOR
                       ON DELETE CASCADE,
  FOREIGN KEY ASG_ID_EDIFICIO REFERENCES EDIFICIO
                       ON DELETE CASCADE
)

CREATE TABLE EDIFICIO (
  EDI_ID             IDENTIFICADOR    PRIMARY KEY,
  EDI_DIRECCION      CHARACTER(12),
  EDI_TIPO           CHARACTER(9) DEFAULT 'Oficina'
```

```
                    CHECK (TIPO IN ('Oficina', 'Almacén', 'Comercio',
'Residencia')),
  EDI_NIVEL_CALIDAD  NUMERIC(1),
  EDI_CALIDAD        NUMERIC(1) DEFAULT 1
                     CHECK (CATEGORIA > 0 AND CATEGORIA < 4)
)
```

Después del CREATE SCHEMA y de la definición de dominios (CREATE DOMAIN) van las instrucciones de creación de tablas. La instrucción CREATE TABLE identifica el nombre de la tabla, que debe ser única dentro del esquema. Después del CREATE TABLE van encerradas entre paréntesis y separadas por coma las instrucciones de definición de columnas y restricciones sobre la tabla.

Las definiciones de columnas de la tabla (que son los atributos del modelo relacional) están compuestas por:

NombreColumna TipoDeDatos [ValorPorDefecto] [RestriccionesEspecíficas]

El [ValorPorDefecto] y las [RestriccionesEspecíficas] pueden no especificarse, por ejemplo:

```
  EDI_DIRECCION      CHARACTER(12),
```

especificar solamente restricciones específicas:

```
  EDI_ID             IDENTIFICADOR    PRIMARY KEY,
```

especificar solamente el valor por defecto:

```
  EDI_TIPO           CHARACTER(9) DEFAULT 'Oficina',
```

o especificar valor por defecto y restricciones:

```
  EDI_CALIDAD        NUMERIC(1) DEFAULT 1
                     CHECK (CATEGORIA > 0 AND CATEGORIA < 4)
```

Las restricciones de la tabla son por ejemplo:

```
  FOREIGN KEY ID_SUPV REFERENCES TRABAJADOR
                ON DELETE SET NULL

  PRIMARY KEY (ASG_ID_TRABAJADOR, ASG_ID_EDIFICIO),
  FOREIGN KEY ASG_ID_TRABAJADOR REFERENCES TRABAJADOR
                      ON DELETE CASCADE,
  FOREIGN KEY ASG_ID_EDIFICIO REFERENCES EDIFICIO
                      ON DELETE CASCADE
```

Analicemos la definición de las llaves:

```
  PRIMARY KEY (ASG_ID_TRABAJADOR, ASG_ID_EDIFICIO),
```

significa que ASG_ID_TRABAJADOR y ASG_IS_EDIFICIO son las llaves de la tabla, por lo tanto los valores combinados de estas dos columnas deben ser únicos para la tabla. Por otro lado dado que son FOREIGN KEY sabemos que son llaves externas.

```
  FOREIGN KEY ASG_ID_TRABAJADOR REFERENCES TRABAJADOR
                      ON DELETE CASCADE,
```

es una clave externa recursiva (referencia a otra fila de su propia relación).

```
  FOREIGN KEY ASG_ID_EDIFICIO REFERENCES EDIFICIO
```

```
                    ON DELETE CASCADE
```

Es una clave externa normal (referencia a otra relación).

Las palabras ON DELETE indican que hacer en caso de borrar al referido. La cláusula ON DELETE SET NULL le dice al sistema que si se borra la tupla a la que apunta la clave externa entonces el valor de ésta se debe poner a cero, esto sirve para mantener la integridad de referencias (si borro al referido, y quedan referencias apuntando a una tupla borrada tenemos una falla en la integridad de referencia). La cláusula ON DELETE CASCADE significa que si se borra la tupla referida en la relación, se realiza un borrado en "cascada" de todas las tuplas que hacían referencia a ésta.

Similar a ON DELETE tenemos a ON UPDATE y ambas cláusulas tienen las opciones siguientes:

- CASCADE
- SET NULL
- SET DEFAULT

Por último se debe decir que a diferencia de las relaciones en el modelo relaciona, no se requiere que cada tabla SQL tenga una clave primaria. En otras palabras, si no se indica ninguna clave, entonces dos filas de la tabla pueden tener valores idénticos. Lo malo es que una tabla que no tenga clave primaria no puede ser referida mediante clave externa desde otra tabla.

Adicionalmente SQL-92 define instrucciones para cambiar las definiciones de las tablas (ALTER TABLE) o para borrar las tablas (DROP TABLE). Para borrar un esquema completo se usa DROP SCHEMA, pero como borrar un esquema es una cosa seria, se tiene que especificar que opción de borrado se usa:

- DROP SCHEMA NombreEsquema CASCADE: significa eliminar el esquema con ese nombre al igual que todas las tablas, datos y otros esquemas que aún existan.
- DROP SCHEMA NombreEsquema RESTRICT: significa eliminar el esquema sólo si todos los restantes objetos del esquema ya han sido borrados.

Manipulación de datos

Consultas simples

Consultas simples: Una consulta que involucra una sola tabla de la base de datos. Ejemplo: ¿Quiénes son los fontanero?

```
SELECT TRA_NOMBRE
  FROM TRABAJADOR
 WHERE TRA_OFICIO = 'Fontanero'
```

Acá se muestran las tres cláusulas más usadas en SQL: la cláusula SELECT, la cláusula FROM y la cláusula WHERE. Una buena regla es escribir cada cláusula en una línea aparte y con sangrías, aunque se puede escribir todo en la misma línea.

Cláusula SELECT: Señala las columnas que se desean en la consulta (equivalente a la proyección del álgebra relacional).

Cláusula FROM: Lista las tablas que son referidas por la consulta.

Clausula WHERE: Nos da la condición para seleccionar las filas de las tablas indicadas.

La sentencia SQL anterior se procesa por el sistema en el orden FROM, WHERE, SELECT.

Otro ejemplo: ¿Cuál es la tarifa semanal de cada electricista?, además entregue la salida ordenada

alfabéticamente.

```
SELECT TRA_NOMBRE, 'Tarifa semanal = ', 48 * TRA_TARIFA_HR
FROM TRABAJADOR
WHERE TIPO = 'ELECTRICISTA'
ORDER BY TRA_NOMBRE
```

Noten la inclusion de la cadena de caracteres 'Tarifa semanal = ' al cálculo de la consulta. Además noten que con la cláusula ORDER BY podemos ordenar la salida, si quisieramos order descendente, utilizamos "DESC".

Consulta: ¿Quiénes tienen una tarifa por hora entre $7000 y $9000, entregue toda la información de la tabla?

```
SELECT *
FROM TRABAJADOR
WHERE TRA_TARIFA_HR >= 7000 AND TRA_TARIFA_HR <= 9000
```

Se pueden usar los siguientes operadores de comparación: =, <> (distinto), <, >, <=, >=.
Se pueden usar los siguientes conectores booleanos: AND, OR, NOT.

Consulta: Indique los fontaneros, albañiles y electricistas.

```
SELECT *
FROM TRABAJADOR
WHERE TRA_OFICIO IN ('Fontaneros', 'Albañiles', 'Electricistas')
```

Consulta: Encontrar todos los trabajadores cuyos oficios comiencen con "Elec". Para esto necesitamos usar caracteres comodines. es decir, símboles especiales que valen por cualquier cadena de caracteres.

```
SELECT *
FROM TRABAJADOR
WHERE TRA_OFICIO LIKE 'Elec%'
```

SQL tiene dos caracteres comodines, el % que vale por cero o cualquier cantidad de caracteres, y _ que vale por exactamente un caracter cualquiera. El operador LIKE se usa para comparar variables de caracteres literales cuando se utilizan comodines.

Consulta: Encuentre todas las asignaciones que comiencen en las dos próximas semanas.

```
SELECT *
FROM ASIGNACION
WHERE ASG_FECHA_INICIO BETWEEN CURRENT_DATE AND CURRENT_DATE + INTERVAL '14' DAY
```

Acá se introduce el operador BETWEEN, que devuelve VERDADERO si el valor está dentro del intervalo [CURRENT_DAY, CURRENT_DATE + INTERVAL '14' DAY] y FALSO si no.
CURRENT_DAY (fecha actual) es una función que siempre devuelve la fecha de hoy.

Consultas multi-tablas

Consulta: ¿Cuáles son los oficios de los trabajadores asignados al edificio 435?

```
SELECT TRA_OFICIO
FROM TRABAJADORES, ASIGNACION
WHERE TRA_ID = ASG_ID_TRABAJADOR
  AND ASG_ID_EDIFICIO = 435
```

En este caso, la cláusula FROM calcula el producto cartesiano de las tablas indicadas y luego con la cláusula WHERE filtramos las filas interesantes (aquellas en que TRA_ID es igual a ASG_ID_TRABAJADOR) y luego las que nos interesan (las asignaciones al edificio 435), y por último SELECT toma la columna que nos interesa. Ojo que cuando hay problemas de que se repite un nombre de columna le ponemos como prefijo el nombre de la tabla seguida por un punto, ejemplo: TRABAJADORES.TRA_ID, o ASIGNACION.ASG_ID_EDIFICIO.

Consulta: Indicar los trabajadores con los nombres de sus supervisores

```
SELECT A.TRA_NOMBRE, B.TRA_NOMBRE
FROM TRABAJADORES A, TRABAJADORES B
WHERE A.TRA_SUP = B.TRA_ID
```

En este caso, para resolver la consulta necesitamos dos copias de la tabla, para ello usamos *alias* (un nombre alternativo que se le da a una relación) de las tablas A y B para referirnos a cada copia.

Subconsultas

Subconsulta: Una consulta dentro de una consulta.

Consulta: ¿Cuáles son los oficios de los trabajadores asignados al edificio 435?

```
SELECT TRA_OFICIO
FROM TRABAJADOR
WHERE TRA_ID IN (
  SELECT ASG_ID_TRABAJADOR
  FROM ASIGNACION
  WHERE ASG_ID_EDIFICIO = 435
)
```

En el ejemplo la subconsulta es:

```
(
  SELECT ASG_ID_TRABAJADOR
  FROM ASIGNACION
  WHERE ASG_ID_EDIFICIO = 435
)
```

A veces es posible realizar una sola consulta, sin subconsulta, pero más compleja:

```
SELECT OFICIO
FROM TRABAJADORES, ASIGNACIONES
WHERE ASG_ID_EDIFICIO = 435
  AND TRA_ID = ASG_ID_TRABAJADOR
```

EXISTS y NOT EXISTS

Operador EXISTS: Evalúa verdadero si el conjunto resultante es no vacío.
Operador NOT EXISTS: Evalúa verdadero si el conjunto resultante es vacío.

Consulta: ¿Quienes son los trabajadores que no están asignados al edificio 435?

```
SELECT TRA_ID, TRA_NOMBRE
FROM TRABAJADOR
```

```
WHERE NO EXISTS (
  SELECT *
  FROM ASIGNACION
  WHERE ASG_ID_TRABAJADOR = TRA_ID
    AND ASG_ID_EDIFICIO = 435
)
```

Los operadores EXISTS y NOT EXISTS siempre preceden a una subconsulta.

Además esta subconsulta tiene apellido, es una *subconsulta correlacionada*, es decir, es una subconsulta cuyos resultados dependen de la fila que se está examinando por una consulta más externa.

Funciones integradas

Consulta: ¿Cuáles son la tarifa por hora mayor y menor?

SELECT MAX(TRA_TARIFA_HR), MIN(TRA_TARIFA_HR)
FROM TRABAJADOR

Consulta: ¿Cuál es el promedio de días que los trabajadores están asignados al edificio 435?

```
SELECT AVG(ASG_NUM_DIAS)
FROM ASIGNACION
WHERE ASG_ID_EDIFICIO = 435
```

Consulta: ¿Cuál es el número total de días asignados a fontanería en el edificio 312?

```
SELECT SUM(ASG_NUM_DIAS)
FROM ASIGNACION, TRABAJADOR
WHERE TRA_ID = ASG_ID_TRABAJADOR
  AND TRA_OFICIO = 'Fontanero'
  AND ASG_ID_EDIFICIO = 312
```

Consulta: ¿Cuántos tipos de oficios diferentes hay?

```
SELECT COUNT(DISTINC TRA_OFICIO)
FROM TRABAJADOR
```

La palabra clave DISTINCT se usa para que el sistema no cuente el mismo tipo de oficio mas de una vez.

Como muestran todos estos ejemplos, si una función integrada aparece en una cláusula SELECT, entonces nada más que funciones integradas pueden aparecer en dicha cláusula SELECT. La una excepción ocurre en combinación con la cláusula GROUP BY.

GROUP BY y HAVING

Cláusula GROUP BY: Indica cuáles filas deben agruparse sobre un valor común de las columna(s) especificada(s)
Cláusula HAVING: Una cláusula que impone condiciones a los grupos.

A diferencia de ORDER BY que se ejecutan sólo para ordenar la salida, GROUP BY y HAVING permiten generar particiones (grupos de datos) para realizar operaciones sobre las particiones.

Consulta: Para cada supervisor, ¿Cuál es la tarifa por horas más alta que se le paga a un trabajador que informe a este supervisor?

```
SELECT TRA_SUP, MAX(TRA_TARIFA_HR)
FROM TRABAJADOR
GROUP BY TRA_SUP
```

Consulta: Para cada supervisor que dirige a más de un trabajador, ¿cuál es la tarifa por horas más alta que se le paga a un trabajador que informe a dicho supervisor?

```
SELECT TRA_SUP, MAX(TRA_TARIFA_HR)
FROM TRABAJADOR
GROUP BY TRA_SUP
HAVING COUNT(*) > 1
```

Noten que la cláusula WHERE aplica a las filas, en cambio la cláusula HAVING a grupos !!, aunque pueden trabajar juntos WHERE y HAVING.

Consulta: Para cada supervisor que dirige a más de un trabajador, ¿cuál es la tarifa por horas más alta que se le paga a un electricista que informe a dicho supervisor?

```
SELECT TRA_SUP, MAX(TRA_TARIFA_HR)
FROM TRABAJADOR
WHERE TRA_OFICIO = 'Electricista'
GROUP BY TRA_SUP
HAVING COUNT(*) > 1
```

Operaciones de modificación de la base de datos

NSERT: Operación que causa que se añadan filas a una relación.
UPDATE: Operación que cambia los valores de las columnas en las filas.
DELETE: Operación que quita filas de una relación.

Inserción:

```
INSERT INTO ASIGNACION (ASG_ID_TRABAJADOR, ASG_ID_EDIFICIO, ASG_FECHA_INICIO)
VALUES (1284, 485, '2002-05-13')
```

Actualización:

```
UPDATE TRABAJADOR
SET TRA_TARIFA_HR = 1.05 * TRA_TARIFA_HR
WHERE ID_SUPV = 1520
```

Borrado:

```
DELETE FROM TRABAJADOR
WHERE ID_SUPV 1520
```

Definición de vistas

Una vista es una porción restringida de la base de datos, y se obtiene de una tabla que contiene información básica o real. Por ejemplo si quisieramos crear una vista de la tabla TRABAJADOR pero que no muestre su tarifa por hora:

```
CREATE VIEW TRABAJADOR_B
```

```
AS SELECT TRA_ID, TRA_NOMBRE., TRA_OFICIO, TRA_SUP
  FROM TRABAJADOR
```

En general para definir una vista se utiliza:

CREATE VIEW NombreVista
AS ConsultaSQL

Parte 2. Más ejemplos

Considere el siguiente esquema:

CLIENTE{CLI_ID, CLI_NOMB, CLI_RENTA_ANUAL}
EMBARQUE{EMB_ID, EMB_ID_CLI, EMB_PESO, EMB_ID_CAMION, EMB_DESTINO}
Clave Foránea: EMB_CLI_ID referencia a CLI_ID en CLIENTE
Clave Foránea: EMB_DESTINO referencia a CIU_NOMBRE en CIUDAD
Clave Foránea: EMB_ID_CAMION referencia a CAM_ID en CAMION
CAMION{CAM_ID, CAM_NOMBRE_CHOFER}
CIUDAD{CIU_NOMBRE, CIU_POBLACION}

¿Cómo se llaman los clientes que han enviado paquetes a Sioux City?

```
SELECT CLI_NOMBRE
FROM CLIENTE, EMBARQUE
WHERE CLI_ID = EMB_ID_CLI
  AND EMB_DESTINO = 'Sioux'
```

¿Quiénes son los choferes que han conducido embarques de clientes que tienen renta anual sobre los $20 millones a ciudades con población por encima del millón?

```
SELECT CAM_NOMBRE_CHOFER
FROM CLIENTE, EMBARQUE, CAMION, CIUDAD
WHERE CLI_RENTA_ANUAL > 20000000
  CIU_POBLACION > 1000000
  AND CLI_ID = EMB_ID_CLI
  AND CAM_ID = EMB_ID_CAMION
  AND CIU_NOMBRE = EMB_DESTINO
```

Indique el nombre y la renta anual de los clientes que han enviado embarques que pesan más de 100 libras

```
SELECT CLI_NOMBRE, CLI_RENTA_ANUAL
FROM CLIENTE
WHERE CLI_ID IN (
  SELECT EMB_ID_CLI
  FROM EMBARQUE
  WHERE EMB_PESO > 100
)
```

Indique los choferes que han transportado embarques a cada una de las ciudades.
Esto es equivalente a determinar cuáles son los choferes tales que NO hay una ciudad a la cual NO hayan llevado un embarque.

```
SELECT C1.CAM_NOMBRE_CHOFER
FROM CAMION C1
```

```
WHERE NOT EXISTS (
  SELECT CIU_NOMBRE
  FROM CIUDADES
  WHERE NOT EXISTS (
    SELECT *
    FROM EMBARQUE, CAMION C2
    WHERE EMB_DESTINO = CIU_NOMBRE
      AND EMB_ID_CAMION = CAM_ID
      AND C2.CAM_NOMBRE_CHOFER = C1.CAM_NOMBRE_CHOFER
  )
)
```

De una lista de los clientes que hacen todos sus envíos a una sola ciudad. (Note que la ciudad no tiene que ser la misma para cada cliente)

```
SELECT CLI_NOMB
FROM CLIENTE
WHERE 1 IN (
    SELECT COUNT(DISTINC EMB_DESTINO)
    FROM EMBARQUE
    WHERE EMB_ID_CLIENTE = CLI_ID
  )
```

Para cada ciudad que haya recibido al menos diez paquetes, ¿cuál es el peso medio de los paquetes enviado a dicha ciudad?

```
SELECT EMB_DESTINO, AVG(EMB_PESO)
FROM EMBARQUE
GROUP BY EMB_DESTINO
HAVING COUNT(*) > 10
```