Las nuevas caracteristicas de un estandar ISO de SQL-1992

Estrella Palacios, Katherine Lizbeth (2016056193))

Tacna, Perú

Abstract

SQL (structured query language), in addition to allowing us queries in the database, contains primitives for defining tables, updating the database, defining views granting privileges, etc. Aspects of the 1992 ANSI standard, known as SQL-92, are shown below.

1. Resumen

SQL (Structured Query Languaje) además de permitirnos 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.

2. Introducción

SQL-92 fue desarrollado por el comité técnico NCITS H2 sobre bases de datos. Este comité desarrolla estándares para la sintaxis y semántica de los lenguajes de bases de datos. SQL-92 fue diseñado para ser un estándar para los sistemas manejadores de bases de datos relacionales (RDBMS). Esta basado en SQL-89, cuya primera versión se conoce como SQL-86. En 1992 aparece SQL2 o SQL92, la versión hoy en día más difundida ([ISO/IEC 1992] [ANSI 1992] [ISO/IEC 1994]). Con la aparición de la segunda versión del estándar (SQL2) en 1992, prácticamente todos los RDBMS, incluso los no relacionales, incluían soporte a SQL. Hoy en día, SQL se ha convertido en el lenguaje de consulta más utilizado. Las nuevas características de SQL-92 son:

- Nuevos tipos de datos definidos: DATE, TIME, TIMESTAMP, INTERVAL, BIT string, VARCHAR strings, y NATIONAL CHARACTER strings.
- Soporte adicional codificación de caracteres más allá del requisito base para representar sentencias SQL.
- Nuevas operaciones escalares tales como concatenación de cadenas y extracción de subcadenas, matemáticas, de fecha y hora y declaraciones condicionales.
- Nuevas operaciones como UNION JOIN, NATURAL JOIN, establecer diferencias y establecer intersecciones.
- Expresiones condicionales con CASE.
- Soporte alternativo para el Lenguaje de Definición de Datos (DDL por sus siglas en inglés) a través de ALTER y DROP.
- Posibilidad de enlazar C (lenguaje de programación), Ada (lenguaje de programación) y MUMPS.
- Nueva funcionalidad para controlar los privilegios de usuario.
- Nueva funcionalidad de comprobación de la integridad, como la restricción de verificación.
- Un nuevo esquema de información de sólo lectura sobre metadatos de la base de datos como, por ejemplo, las tablas que contiene, etc.
- Ejecución dinámica de sentecias (en luegar de prepararlas).
- Mejor soporte para acceso remoto.
- Tablas temporales. Por ejemplo, CREATE TEMP TABLE etc.
- Transacciones.
- Nuevas operaciones para cambiar tipos de datos en el momento indicado a través de CAST (expr AS type).
- Cursores.

3. Marco Teórico

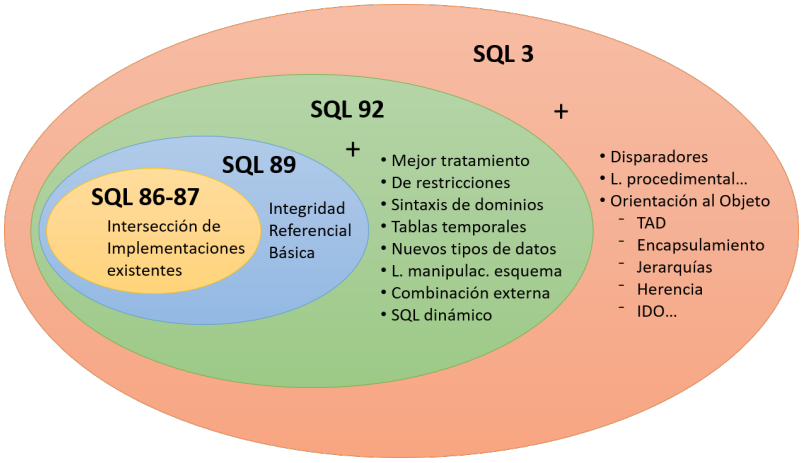
3.1. ***SQL***

3.1.1. ***Definición***

SQL. Structured Query Language (lenguaje de consulta estructurado), es un lenguaje surgido de un proyecto de investigación de IBM para el acceso a bases de datos relacionales. Actualmente se ha convertido en un estándar de lenguaje de bases de datos, y la mayoría de los sistemas de bases de datos lo soportan, desde sistemas para ordenadores personales, hasta grandes ordenadores. [2]

3.1.2. ***Orígenes y evolución***

Los orígenes del SQL están ligados a los de las bases de datos relacionales. En 1970 Codd propone el modelo relacional y asociado a este; un sublenguaje de acceso a los datos basado en el cálculo de predicados. Basándose en estas ideas los laboratorios de IBM, definen el lenguaje SEQUEL (Structured English Query Language) que más tarde sería ampliamente implementado por el SGBD experimental System R, desarrollado en 1977 también por IBM. Sin embargo, fue Oracle quien lo introdujo por primera vez en 1979 en un programa comercial. El SEQUEL terminaría siendo el predecesor de SQL, siendo éste una versión evolucionada del primero. El SQL pasa a ser el lenguaje por excelencia de los diversos SGBD relacionales surgidos en los años siguientes y es por fin estandarizado en 1986 por el ANSI, dando lugar a la primera versión estándar de este lenguaje, el SQL-86 o SQL1.



Al año siguiente este estándar es también adoptado por la ISO. Sin embargo este primer estándar no cubre todas las necesidades de los desarrolladores e incluye funcionalidades de definición de almacenamiento que se consideraron suprimir. Así que en 1992 se lanza un nuevo estándar ampliado y revisado del SQL llamado SQL-92 o SQL2. En la actualidad el SQL es el estándar de facto de la inmensa mayoría de los SGBD comerciales. Y, aunque la diversidad de añadidos particulares que incluyen las distintas implementaciones comerciales del lenguaje es amplia, el soporte al estándar SQL-92 es general y muy amplio. [2]
En la figura anterior se muestra la evolución del SQL-86 al SQL-3. [1]

3.1.3. ***Tipos de sentencias SQL***

Como su nombre indica, el SQL nos permite realizar consultas a la base de datos. Pero el nombre se queda corto ya que SQL además realiza funciones de definición, control y gestión de la base de datos. Las sentencias SQL se clasifican según su finalidad dando origen a tres 'lenguajes' o sublenguajes:

- El DDL (Data Definition Language), lenguaje de definición de datos, incluye órdenes para definir, modificar o borrar las tablas en las que se almacenan los datos y de las relaciones entre estas. Es el que más varia de un sistema a otro.

- El DCL(Data Control Language), lenguaje de control de datos, contiene elementos útiles para trabajar en un entorno multiusuario, en el que es importante la protección de los datos, la seguridad de las tablas y el establecimiento de restricciones en el acceso, así como elementos para coordinar la compartición de datos por parte de usuarios concurrentes, asegurando que no interfieren unos con otros.

- El DML (Data Manipulation Language), lenguaje de manipulación de datos, nos permite recuperar los datos almacenados en la base de datos y también incluye órdenes para permitir al usuario actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos o modificando datos previamente almacenados.

- PLSQL(SQL Programático: Define un cursor para una consulta. DECLARE: Abre un cursor para recuperar resultados de consulta OPEN Recupera una fila de resultados de consulta, FETCH, CLOSE: Cierra un cursor.)[2]

3.2. Nuevas Características de SQL-92

3.2.1. 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.

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

Real	
Double precision	doble precisión
float	flotante

- **Cadenas de caracteres:** 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).

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

- **Cadenas de bits:** Estos campos se usan para banderas u otras máscaras de bits para el control.

Bit(n)
Bit varying(n)

- **Fechas y horas:** 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.

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

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

3.2.3. ***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: 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.

3.2.4. ***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.

```
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 çascada"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.

3.2.5. ***Manipulación de datos:***

- **Consultas simples:**
  Una consulta que involucra una sola tabla de la base de datos.
  Aquí 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.

Ejemplo: ¿Quiénes son los fontaneros?

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

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.
Cláusula 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
```

Debe notarse la inclusión de la cadena de caracteres 'Tarifa semanal = ' al cálculo de la consulta. También nótese que con la cláusula ORDER BY podemos ordenar la salida, si se quisiera orden descendente, se utilizaría "DESC".
Consulta: ¿Quiénes tienen una tarifa por hora entre $ 7000 y $ 9000, entregue toda la información de la tabla? Se pueden usar los siguientes

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

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 .[El]ec". Para esto necesitamos usar caracteres comodines. Es decir, símbolos 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
```

Aquí 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:**
  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
```

3.2.6. ***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.

3.2.7. ***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 MAX(TRA_TARIFA_HR), MIN(TRA_TARIFA_HR)
FROM TRABAJADOR
```

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

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

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.

3.2.8. ***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
```

3.2.9. ***Operaciones de modificación de la base de datos:***

- INSERT: 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.

A continuación podemos ver ejemplos de inserción y actualización:

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

```
UPDATE TRABAJADOR
SET TRA_TARIFA_HR = 1.05 * TRA_TARIFA_HR
WHERE ID_SUPV = 1520
Borrado:
DELETE FROM TRABAJADOR
WHERE ID_SUPV 1520
```

3.2.10. ***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
```

4. Análisis

5. Conclusiones

- Conclusion 1 :
  El lenguaje de consulta estructurado de ANSI nos permite manejar nuestra base de datos de manera rápida y confiable, con la seguridad de recibir los informes tal y como sean requerids por los usuarios.

- Conclusion 2 :
  Con la ayuda de los sublenguajes que contienen las distintas versiones se nos hace más práctico el uso de esta herramienta. El uso de los comando DML, que conforman la base para una preparación adecuada, en la construcción de aplicaciones con BD. Las aplicaciones manejan comandos de inset, delete, update y select, contra la base de datos.

- Conclusion 3 :
  Finalmente, podemos concluir que SQL es un lenguaje que ha ido variando y mejorando su rendimiento con el tiempo, por lo que seguirá siendo una herramienta eficiente para el manejo de la base de datos.

Referencias

[1] De Miguel, A. y Piattini, M. (1999). Fundamentos y modelos de bases de datos. Editorial Rama.

[2] EncuRed (ne). Sql. Recuperado de https://www.ecured.cu/SQL. Accedido 15-08-2019.