CONCEPTOS INICIALES

SQL

El SQL es el lenguaje estándar ANSI/ISO de definición, manipulación y control de base de datos relacionales. Es un lenguaje declarativo: sólo hay que indicar qué se quiere hacer, a diferencia de los lenguajes procedimentales donde es necesario especificar cómo hay que hacer cualquier acción sobre la base de datos. Con SQL se puede definir, manipular y controlar una base de datos relacional. SQL posee los siguientes componentes:

- **Lenguaje de definición de datos (LDD):** El LDD de SQL proporciona órdenes para la definición de esquemas de relación, borrado de relaciones, creación de índices y modificación de esquemas de relación.
- **Lenguaje interactivo de manipulación de datos (LMD):** El LMD de SQL incluye un lenguaje de consultas, basado tanto en el álgebra relacional como en el cálculo relacional de tuplas. Incluye también órdenes para insertar, borrar y modificar tuplas de la base de datos.
- **Definición de vistas:** El LDD de SQL incluye órdenes para la definición de vistas.
- **Control de transacciones:** SQL incluye órdenes para la especificación del comienzo y final de transacciones.
- **SQL incorporado y SQL dinámico:** SQL dinámico e incorporado define cómo se pueden incorporar instrucciones SQL en lenguajes de programación de propósito general, tales como C, C++, Java, entre otros.
- **Integridad:** El LDD de SQL incluye órdenes para la especificación de las restricciones de integridad que deben satisfacer los datos almacenados en la base de datos. Las actualizaciones que violen las restricciones de integridad se rechazan.
- **Autorización:** El LDD de SQL incluye órdenes para especificar derechos de acceso para las relaciones y vistas.

Así mismo resalta que la estructura básica de una expresión SQL consiste en tres cláusulas: **select, from** y **where.**

- La cláusula **select** corresponde a la operación proyección del álgebra relacional. Se usa para listar los atributos deseados del resultado de una consulta.

- La cláusula **from** corresponde a la operación producto cartesiano del álgebra relacional. Lista las relaciones que deben ser analizadas en la evaluación de la expresión.
- La cláusula **where** corresponde a la operación selección del álgebra relacional. Es un predicado que engloba a los atributos de las relaciones que aparecen en la cláusula **from.**

Álgebra Relacional

El álgebra relacional es un lenguaje formal con una serie de operadores que trabajan sobre una o varias relaciones para obtener otra relación resultado, sin que cambien las relaciones originales. Tanto los operandos como los resultados son relaciones, por lo que la salida de una operación puede ser la entrada de otra operación. Esto permite anidar expresiones del álgebra. Algunos conceptos básicos relacionados:

- ***Relación:*** Es el elemento básico del modelo, está compuesta por un conjunto fijo de atributos y por un conjunto de tuplas.

- ***Una tupla*** de una relación o de una tabla corresponde a una fila de aquella tabla. Las tuplas no tienen un orden específico puesto que matemáticamente una relación se define como un conjunto y no como una lista.

- ***Un atributo*** de una relación o de una tabla corresponde a una columna de la tabla. Los atributos están desordenados y se referencian por nombres y no por la posición que ocupan. Esto significa que no se puede, por ejemplo, hacer referencia al tercer atributo de una relación. Todos los valores de los atributos son atómicos y una relación que satisfaga esta condición se llama relación normalizada. Un atributo extrae sus valores desde un dominio simple.

- ***Un dominio*** se define como un conjunto de valores atómicos. Por atómico se quiere decir que cada valor del dominio es invisible en lo que concierne al modelo relacional. Un método común de especificación de los dominios consiste en especificar un tipo de datos al cual pertenecen los valores que constituyen el dominio.

- ***Esquema Relacional*** Está compuesto por un nombre de relación, R, y una lista de atributos A1,A2,...,An, de tal forma que se puede denotar como R(A1,A2,...,An).
  Ejemplo:
  R=EMPLEADO
  Atributos: Cedula, Nombre, Edad, Departamento
  EMPLEADO(Cedula, Nombre, Edad, Departamento)

- ***No - repetición de las tuplas:*** En un archivo clásico puede ocurrir que dos de los registros sean exactamente iguales; es decir, que contengan los mismos datos. En el caso del modelo relacional no es posible que una relación contengan tuplas repetidas.

- ***No - ordenación de las tuplas:*** Los elementos de un conjunto no están ordenados; por lo tanto, las tuplas de una relación no tienen orden específico.

- ***Los atributos no están ordenados:*** Esta propiedad desprende el hecho de que la cabecera de una relación se define también como conjunto. Las columnas de una tabla tienen un orden evidente de izquierda a derecha, pero los atributos de una relación carecen de tal orden.

Las operaciones selección, proyección, y renombramiento se denominan operaciones *unarias* porque operan sobre una sola relación. Las otras tres operaciones unión, diferencia y producto cartesiano operan sobre pares de relaciones y se denominan por tanto, operaciones *binarias.*

Operación Selección: La operación **selección** selecciona tuplas que satisfacen un predicado dado. Se usa la letra griega sigma minúscula (***σ***) para denotar la selección. El predicado aparece como subíndice de ***σ*. La relación de argumentos se da entre paréntesis a continuación de *σ*. Por tanto, para seleccionar las tuplas de la relación *préstamo* en que la sucursal es <<Navacerrada>> hay que escribir** $\sigma_{nombre\_sucursal = "Navacerrada"}(prestamo)$**. Tomando la tabla 2 como la relación *préstamo*, la relación resultante es la que se muestra en la tabla 3.**

numero_prestamo	nombre_sucursal	importe
P-11	Collado Mediano	900
P-14	Centro	1500
P-15	Navacerrada	1500
P-16	Navacerrada	1300
P-17	Centro	1000
P-23	Moralzarzal	2000
P-93	Becerril	500

Tabla 2. La relación préstamo

numero_prestamo	nombre_sucursal	Importe
P-15	Navacerrada	1500
P-16	Navacerrada	1300

Tabla 3. Resultado de $\sigma_{nombre\_sucursal = "Navacerrada"}(prestamo)$

Operación Proyección: La operación **proyección** es una operación unaria que devuelve su relación de argumentos, excluyendo algunos argumentos. Dado que las relaciones son conjuntos, se eliminan todas las filas duplicadas. La proyección se denota por la letra griega mayúscula pi ($\prod$). Se crea una lista de los atributos que se desea que aparezcan en el resultado como subíndice de $\prod$. Su único argumento, una relación, se escribe a continuación entre paréntesis. Si se desea hacer una lista de todos los números de préstamos y del importe de los mismos pero sin que aparezca los nombres de sucursales se puede escribir la siguiente consulta: $\prod_{numero\_prestamo,importe}(prestamo)$, donde la relación préstamo sobre la cual se aplica la operación proyección se muestra en la tabla 2 y la relación que resulta de esta consulta se muestra en la tabla 4.

numero_prestamo	**importe**
P-11	900
P-14	1500
P-15	1500
P-16	1300
P-17	1000
P-23	2000
P-93	500

Tabla 4. Resultado de $\prod_{numero\_prestamo,importe}(prestamo)$

Operación Unión: Dos tablas se pueden unir si tienen el mismo número de columnas y dominios compatibles. El resultado de la unión es otra tabla con las filas de ambas tablas. Las filas repetidas aparecen una sola vez. Se representa de la siguiente manera: Tabla1 ∪ Tabla2.

Si se desea averiguar el nombre de todos los clientes del banco que tienen una cuenta, un préstamo o ambas cosas se debe averiguar: 1. los nombres de todos los clientes con préstamos en el banco, contenidos en la relación *prestatario* que se muestra en la tabla 5: $\prod_{nombre\_cliente}(prestatario)$, 2. el nombre de los clientes con cuentas en el banco, contenidos en la relación *impositor* que se muestra en la tabla 6: $\prod_{nombre\_cliente}(impositor)$. Y posterior a esto la unión de los dos conjuntos en una sola consulta: $\prod_{nombre\_cliente}(prestatario) \cup \prod_{nombre\_cliente}(impositor)$, el resultado de la nueva relación se muestra en la tabla 7.

nombre_cliente	numero_prestamo
Fernandez	P-16
Gomez	P-11
Gomez	P-23
Lopez	P-15
Perez	P-93
Santos	P-17
Sotoca	P-14
Valdivieso	P-17

Tabla 5. Relación prestatario

nombre_cliente	numero_cuenta
Abril	C-305
Gomez	C-215
Gonzales	C-101
Gonzales	C-201
Lopez	C-102
Ruperez	C-222
Santos	C-217

Tabla 6. Relación impositor

nombre_cliente
Fernandez
Gomez
Lopez
Perez
Santos
Sotoca
Valdivieso
Abril
Gonzales
Ruperez

Tabla 7. Resultado de $\prod_{nombre\_cliente}(prestatario) \cup \prod_{nombre\_cliente}(impositor)$

Operación Diferencia: La operación **diferencia** denotada por −, permite hallar las tuplas que están en una relación pero no en la otra. La expresión $r - s$ da como resultado una relación que contiene las tuplas que están en *r* pero no en *s*. La diferencia entre dos tablas solo es posible si tienen el mismo número de columnas y dominios compatibles.

Para ejemplificar la operación diferencia se pueden buscar todos los clientes del banco que tienen abierta una cuenta pero no tienen concedido ningún préstamo, se escribe la siguiente consulta: $\prod_{nombre\_cliente}(impositor) - \prod_{nombre\_cliente}(prestatario)$, donde la relación impositor se observa en la tabla 6, la relación prestatario en la tabla 5 y el resultado obtenido se muestra en la tabla 8.

nombre_cliente
Abril
Gonzales
Ruperez

Tabla 8. Resultado de $\prod_{nombre\_cliente}(impositor) - \prod_{nombre\_cliente}(prestatario)$

Operación Producto Cartesiano: La **operación producto cartesiano,** denotada por un aspa (X), permite combinar información de cualesquiera dos relaciones. El producto cartesiano de las relaciones *r1* y *r2* se escribe *r1* X *r2*. Las relaciones se definen como subconjuntos del producto cartesiano de un conjunto de dominios. A partir de esto se debe tener una intuición sobre la definición de la operación producto cartesiano. Se representa Tabla1 X Tabla2.

Para obtener todos los clientes con préstamos se construye el producto cartesiano entre la relación prestatario que se observa en la tabla 5 y la relación préstamo que se observa en la tabla 9. La consulta se construye de la siguiente manera: $prestatario \times prestamo$ y el resultado se observa en la tabla 10.

numero_prestamo	**nombre_sucursal**	**Importe**
P-11	Collado Mediano	900
P-14	Centro	1500
P-15	Navacerrada	1500
P-16	Navacerrada	1300
P-17	Centro	1000
P-23	Moralzarzal	2000
P-93	Becerril	500

Tabla 9. Relación préstamo

	Prestarario	Préstamo		
nombre_cliente	**numero_prestamo**	**numero_prestamo**	**nombre_sucursal**	**Importe**
Fernandez	P-16	P-11	Collado Mediano	900
Fernandez	P-16	P-14	Centro	1500
Fernandez	P-16	P-15	Navacerrada	1500
Fernandez	P-16	P-16	Navacerrada	1300
Fernandez	P-16	P-17	Centro	1000
Fernandez	P-16	P-23	Moralzarzal	2000
Fernandez	P-16	P-93	Becerril	500
Gomez	P-11	P-11	Collado Mediano	1500
Gomez	P-11	P-14	Centro	1500
Gomez	P-11	P-15	Navacerrada	1300
Gomez	P-11	P-16	Navacerrada	1000
Gomez	P-11	P-17	Centro	2000
Gomez	P-11	P-23	Moralzarzal	500
Gomez	P-11	P-93	Becerril	1500
Gomez	P-23	P-11	Collado Mediano	1500
Gomez	P-23	P-14	Centro	1300
Gomez	P-23	P-15	Navacerrada	1000
Gomez	P-23	P-16	Navacerrada	2000
Gomez	P-23	P-17	Centro	500
Gomez	P-23	P-23	Moralzarzal	2000
Gomez	P-23	P-93	Becerril	500
…	…	…	…	…
…	…	…	…	…
…	…	…	…	…
Sotoca	P-14	P-11	Collado Mediano	900
Sotoca	P-14	P-14	Centro	1500
Sotoca	P-14	P-15	Navacerrada	1500
Sotoca	P-14	P-16	Navacerrada	1300
Sotoca	P-14	P-17	Centro	1000
Sotoca	P-14	P-23	Moralzarzal	2000
Sotoca	P-14	P-93	Becerril	500
Valdivieso	P-17	P-11	Collado Mediano	900
Valdivieso	P-17	P-14	Centro	1500
Valdivieso	P-17	P-15	Navacerrada	1500
Valdivieso	P-17	P-16	Navacerrada	1300
Valdivieso	P-17	P-17	Centro	1000
Valdivieso	P-17	P-23	Moralzarzal	2000
Valdivieso	P-17	P-93	Becerril	500

Tabla 10. Resultado de $prestatario \times prestamo$

Otras operaciones de álgebra relacional

Las operaciones fundamentales del álgebra relacional son suficientes para expresar cualquier consulta del álgebra relacional. Sin embargo, limitándose exclusivamente a las operaciones fundamentales, algunas consultas habituales resultan complicadas de expresar. Por tanto (Ramez & Shamkant, 2002) definen otras operaciones que no añaden potencia al álgebra, pero que simplifican las consultas habituales. Para cada operación nueva se facilita una expresión equivalente usando sólo las operaciones fundamentales.

Operación Intersección: Es una operación derivada de la diferencia. La intersección de dos tablas es otra tabla formada por las filas que aparecen en ambas tablas y las columnas de una de las tablas. Las tablas han de tener el mismo número de columnas y dominios compatibles. Se representa de la siguiente manera: Tabla1 ∩ Tabla2

Si se desean conocer todos los clientes con un préstamo concedido y una cuenta abierta, se emplea la intersección de conjuntos de la siguiente manera: $\prod_{nombre\_cliente}(prestatario) \cap \prod_{nombre\_cliente}(impositor)$, donde la relación prestatario se observa en la tabla 5, la relación impositor en la tabla 6 y el resultado de la consulta intersección en la tabla 11.

nombre_cliente
Gomez
Lopez
Santos

Tabla 11. Resultado de $\prod_{nombre\_cliente}(prestatario) \cap \prod_{nombre\_cliente}(impositor)$

Operación División: La operación **división,** denotada por ÷, resulta adecuada para las consultas que incluyan la expresión para todos. Dada R1 y R2, donde R2 es subconjunto de R1, se construye una nueva relación formada por los atributos de R1 que no están en R2, donde los valores de los otros atributos concuerdan con todos los valores de la relación R2. Se representa R1 ÷ R2

Para hallar a todos los clientes que tengan abierta una cuenta en todas las sucursales ubicadas en Arganzuela, se pueden obtener todas las sucursales de Arganzuela mediante la expresión: $R2 = \prod_{nombre\_sucursal}(\sigma_{ciudad\_sucursal = "Arganzuela"}(sucursal))$ como se muestra en la tabla 13, luego se pueden encontrar todos los pares (nombre_cliente, nombre_sucursal) para los que el cliente tiene una cuenta en una sucursal escribiendo: $R1 = \prod_{nombre_cliente,\ nombre_sucursal}(impositor \bowtie \text{cuenta})$como se muestra en la tabla 14. Y por último hay que hallar los clientes que aparecen en R2 con los nombres de todas las sucursales de R1, la operación que proporciona exactamente esos clientes es la operación división y la consulta asociada es: R1 ÷ R2, el resultado total obtenido se muestra en la figura 15. Las relaciones impositor, cuenta y sucursal se muestran en las tablas 6, 1 y 12 respectivamente.

nombre_sucursal	ciudad_sucursal	activos
Becerril	Aluche	400.000
Centro	Arganzuela	9.000.000
Collado Mediano	Aluche	8.000.000
Galapagar	Arganzuela	7.100.000
Moralzarzal	La Granja	2.100.000
Navacerrada	Aluche	1.700.000
Navas de la Asunción	Alcalá de Henares	300.000
Segovia	Cerceda	3.700.000

Tabla 12. Relación Sucursal

nombre_cliente	nombre_sucursal
Abril	Collado Mediano
Gomez	Becerril
Gonzales	Centro
Gonzales	Galapagar
Lopez	Navacerrada
Ruperez	Moralzarzal
Santos	Galapagar

Tabla 13. Resultado de R2

nombre_sucursal
Centro
Galapagar

Tabla 14. Resultado de R1

nombre_cliente
Gonzales

Tabla 15. Resultado de $\boldsymbol{R1 \div R2}$

Nota: información obtenida de la fuente: **Fundamentos de Base de Datos.** (Silberschatz, Korth, & Sudarshan, 2002)

Operación Reunión Natural: La operación **reunión natural,** denotada por ⋈, hace un producto cartesiano de sus dos argumentos y realiza una selección forzando la igualdad de atributos que aparecen en ambas relaciones eliminando los repetidos. Se representa R1 ⋈ R2

Para determinar los nombres de todos los clientes que tienen concedido un préstamo en el banco y a su vez averiguar el importe se construye la siguiente consulta:$\prod_{nombre\_cliente,\ numero\_prestamo,\ importe}(prestatario \bowtie prestamo)$, la relación prestatario se observa en la tabla 5, la relación préstamo en la tabla 9 y el resultado de la consulta en la tabla 16.

Tabla 16. Resultado de:

nombre_cliente	numero_prestamo	Importe
Fernandez	P-16	1300
Gomez	P-11	2000
Gomez	P-23	900
Lopez	P-15	1500
Perez	P-93	500
Santos	P-17	1000
Sotoca	P-14	1500
Valdivieso	P-17	1000

$\prod_{nombre\_cliente,\ numero\_prestamo,\ importe}(prestatario \bowtie prestamo)$

GUAVA

Como indica (Guava-Libraries) GUAVA es la librería de Google para java, su idea es proveer bloques constructivos basados en buenas prácticas que aumenten la productividad de los desarrolladores. También integra a Google Collections, unificando las utilidades de entrada/salida, primitivas, concurrencia y colecciones en una única librería. Colecciones como:

- **Table:** Una Table es una colección de datos que asocia un par ordenado de claves, fila y columna a un único valor mapeado, o accedido desde cualquier ángulo dentro de un mismo contexto. Su estructura se define:

  Table<R,C,V>

  - R - Tipo de dato de las claves de las filas en la tabla
  - C - Tipo de dato de las claves de las columnas en la tabla
  - V - Tipo de dato de los valores mapeados

  Y sus métodos son:

  - **class.create():** creación de nuevas tablas vacías.
  - **cellSet():** Asignar el conjunto de todos los triples (R,C,V)
  - **clear():** Remover todos los valores
  - **column(c):** Devuelve la columna para la clave dada, como mapa<R,V>
  - **contains(R,C):** Devuelve verdadero si la tabla tiene una asignación para las claves dadas.
  - **containsRow(R):** Devuelve verdadero si la tabla tiene cualquier asignación para la clave fila dada.
  - **containsColumn(C):** Devuelve verdadero si la tabla contiene cualquier asignación para la clave columna dada.
  - **get(R,C):** Retorna el valor para la clave indicada, o nulo.
  - **isEmpty():** Retorna verdadero si la tabla está vacía.
  - **put(R,C,V):** Asignación del valor V en la posición de las claves (R,C)
  - **putAll(table):** Agrega todos los datos de una tabla dada a esta.
  - **remove(R,C):** Remueve el valor asignado en las claves dadas.
  - **row(R):** Devuelve la fila para la clave dada, como mapa<C,V>

        - **size():** Número de valores<R,C,V> existentes en la tabla.
        - **toString():** Devuelva la cadena como "{a={b=c, d=e},f={g=h}}"

- Multiset
- MultiMap
- BiMap
- entre otras.

µALGEBRA RELACIONAL (microALGEBRA RELACIONAL)

Especificación Léxica

Operadores de álgebra relacional:

Símbolo	Significado en álgebra relacional	Palabra Reservada
σ	Operación Selección	SEL
∏	Operación Proyección	PRO
U	Operación Unión	UNI
–	Operación Diferencia	DIF
X	Operación Producto Cartesiano	PROC
∩	Operación Intersección	INT

Operadores lógicos

Símbolo	Palabra Reservada
AND	AND
NOT	NOT
OR	OR

Operadores Comparación:

Símbolo	Palabra Reservada
=	EQ
!=	DIFERENTE
>	MAYOR
<	MENOR

extras:

Símbolo	Token
(	IPAREN
)	DPAREN
,	COMA
“ ”	COMILLAS

Operadores matemáticos:

Símbolo	Token
+	SUMA
-	RESTA
*	MULTI

Especificación Sintáctica

Se debe poder crear las siguientes expresiones en dicho lenguaje

Consulta 1: $\prod_{nombre\_cliente}(prestatario) \cap \prod_{nombre\_cliente}(impositor)$

Lo que se expresaría en el lenguaje µALGEBRA RELACIONAL asi:

PRO nombre_cliente (prestatario) INT PRO nombre_cliente (impositor)

Consulta 2: $(prestatario) \times (prestamo)$

Lo que se expresaría en el lenguaje µALGEBRA RELACIONAL asi:

(prestatario) PROC (prestamo)

Consulta 3: $(\prod_{nombre\_cliente,\ nombre\_sucursal}((impositor) \bowtie (\text{cuenta})))$

$\div (\prod_{nombre\_sucusal}(\sigma_{ciudad\_sucursal = "Arganzuela"}(sucursal)))$

Lo que se expresaría en el lenguaje µALGEBRA RELACIONAL asi:

```
(
    PRO nombre_cliente,
    nombre_sucursal (
        (impositor) REUN (cuenta)
    )
)
DIV
(
    PRO nombre_sucursal (
    SEL ciudad_susursal = “Arganzuela” (sucursal)
    )
)
```

Consulta 4:

$$(\prod_{nombre\_cliente}(impositor)) - (\prod_{nombre\_cliente}(prestatario))$$

Lo que se expresaría en el lenguaje µALGEBRA RELACIONAL asi:

```
(
  PRO nombre_cliente(impositor)
  DIF
  (
   PRO nombre_cliente (prestatario)
  )
)
```

Consulta 5:

$$(\prod_{patente,rut\_chofer}(Movil)$$
$$\div \prod_{rut}(\sigma_{(fecha\_licencia\_hasta < 18/07/1990\ AND\ fecha\_licencia\_hasta \geq 01/01/1990)}(Chofer)))$$
$$\cup\ (\prod_{a.rut\_chofer}(\sigma_{a.rut\_chofer = b.rut}\ (Movil\ a\ \times\ Chofer\ b))$$
$$- \prod_{rut\_chofer}(\sigma_{patente\ != 23452*2}(Movil\ \bowtie\ \text{Movil})))$$

Lo que se expresaría en el lenguaje µALGEBRA RELACIONAL asi:

```
(
PRO patente,
rut_chofer (Movil) DIV PRO rut (
SEL (
fecha_licencia_hasta < 18/07/1990
      AND fecha_licencia_hasta >= 01/01/1900
    ) (Chofer)
  )
) UNI (
  PRO a.rut_chofer (
    SEL a.rut_chofer = b.rut (Movil a PROC Chofer b)
  ) DIF PRO rut_chofer (
    SEL patente != 2345 * 2 Movil REUN Movil
  )
)
```

Consulta 6:

$(\prod_{nombre\_cliente}(\sigma_{NOT\ nombre\ =\text{"}Gomez\text{"}}\ (prestamo)))$

$\cap\ (\prod_{nombre\_cliente}(\sigma_{importe>2000/2\ OR\ importe\ \leq importe-500\ OR\ importe\ =importe\ +1000}(prestamo$

$\bowtie$ prestatario)))

Lo que se expresaría en el lenguaje µALGEBRA RELACIONAL asi:

```
(
PRO patente,
rut_chofer (Movil) DIV PRO rut (
SEL (
fecha_licencia_hasta < 18 / 07 / 1990
      AND fecha_licencia_hasta >= 01 / 01 / 1900
    ) (Chofer)
  )
) UNI (
  PRO a.rut_chofer (
    SEL a.rut_chofer = b.rut (Movil a PROC Chofer b)
  ) DIF PRO rut_chofer (
    SEL patente != 2345 * 2 Movil REUN Movil
  )
)
```

Donde las operaciones pueden ser definidas de la siguiente manera:

Operación Proyección: La operación proyección está encargada de generar una relación que contenga los argumentos especificados en la consulta dada, se pueden incluir en él operaciones matemáticas como suma, resta, multiplicación y división (+,-,*,/). La función que realiza ésta tarea opera sobre una relación proporcionada por el usuario, donde:

- Se buscan los argumentos o nombres de las columnas sobre la relación
- Se seleccionan todas las filas de cada una de las columnas especificadas en los argumentos.
- Se eliminan las filas repetidas
- Se devuelve la nueva relación resultante

Operación Selección: La operación selección devuelve una relación con las filas que cumplan las condiciones especificadas en su predicado, condiciones que se aplican sobre las columnas de la relación a cantidades numéricas, fechas (dd/mm/yyyy) y cadenas de caracteres. En las condiciones se pueden hacer operaciones de comparación (=, <, >, !=), o lógicas (AND, OR, NOT).

Operación Unión: La operación unión es la encargada de combinar dos relaciones y retornar una nueva relación. El proceso de la unión es:

- Validar que ambas relaciones tengan las mismas columnas
- Descartar las tuplas repetidas
- Guardar en una nueva relación las filas de ambas relaciones

Operación Diferencia: La operación diferencia es la que permite obtener las tuplas que están en una relación pero no en la otra, es decir la expresión ***r – s*** da como resultado una relación que contiene las tuplas que están en *r* pero no en *s*, se opera sobre dos relaciones dadas, y el proceso de obtención de las tuplas de la nueva relación es el siguiente:

- Verificar que ambas relaciones tengan el mismo número de columnas y dominios compatibles
- Eliminar los valores de la relación izquierda ***r*** que estén en la relación derecha ***s***
- Guardar en una nueva relación los valores resultantes de la relación izquierda

Operación Producto Cartesiano: La operación Producto Cartesiano se basa en combinar la información de dos relaciones, la nueva relación tendrá n1*n2 número de filas, donde n1 es el número de filas de la relación izquierda dada y n2 el número de filas de la relación derecha. En vista de que en ambas relaciones puede existir una misma columna se debe anteponer al nombre de la columna el nombre de la relación a la que corresponden en la nueva relación. Si las relaciones provienen de otras consultas se antepondrá la palabra relacionI y relacionD según sea el caso.

Operación Intersección: La operación intersección está formada por las filas que aparecen en las dos relaciones sobre las que opera, y las columnas de una de las relaciones. Es una operación derivada de la diferencia y por ende funciona en base a la diferencia, siendo *r* la relación izquierda y *s* la relación derecha la intersección se define $r \cap s = r - (r - s)$, es decir para obtener la relación resultante de una intersección solo es necesario aplicar dos veces la operación diferencia.

Operación División: La operación división manipula los datos de dos relaciones dadas: R1 y R2, R2 es subconjunto de R1. La relación generada está formada por las tuplas de R1 que no están en R2 y a su vez los valores de las otras tuplas concuerdan con todos los valores de la relación R2. Para lograr esta nueva relación en la operación división se realizaron los siguientes pasos:

- Verificar en que columna coinciden ambas relaciones
- Almacenar las columnas en las que no coinciden las relaciones, denominada *lista*
- Proyectar de la relación izquierda los campos que se encuentran en *lista,* por medio de la operación proyección y almacenar su resultado en *T1*. T1=proyeccion(relacionI, lista)
- Realizar un producto cartesiano de *T1* con la relación derecha y almacenarlo en *producto.* producto = productoCartesiano(T1,relacionD)
- Realizar una diferencia entre el producto y la relación izquierda, almacenarla en *diferencia.* diferencia = diferencia(producto, relacionI)
- Proyectar de la relación diferencia los campos que se encuentran en *lista,* y almacenarlo en *T2*. T2 = proyeccion(diferencia,lista)
- El resultado final será la diferencia entre *T1* y *T2.* Resultado=diferencia(T1,T2)

Operación Reunión Natural: La operación reunión natural se encarga de realizar un producto cartesiano entre dos relaciones y sobre él una selección forzando la igualdad de atributos que aparecen en ambas relaciones para la columna donde estas coinciden, exceptuando los repetidos.

Proyecto

Se desea cree un intérprete de consultas de algebra relacional el cual denominaremos µALGEBRA RELACIONAL. A continuación se procede a enumerar lo solicitado:

- Se debe proceder a resolver operaciones de algebra relacional sobre relaciones definidas en archivos de texto (1 archivo por cada relación).
- Se posee solo tres tipos de datos disponibles: entero, cadena y fecha los cuales son los únicos que podrán ser usados en las relaciones creadas y no existirán valores nulos.
- Existirá una carpeta en donde se incluirán todas las relaciones deseadas en archivos de texto con el formato siguiente:

Fila 1:	Nombres columnas separadas por ; con tipo de dato entre paréntesis (entero, cadena, fecha)
Fila 2 en adelante datos	Cada tupla existente en la relacion

Por ejemplo:

numero_prestamo	nombre_sucursal	Importe
P-11	Collado Mediano	900
P-14	Centro	1500
…	..	..

```
Numero_prestamo(cadena);nombre_sucural(cadena);importe(entero)
“P-11”;”Collado Mediano”;900
“P-14”;”Centro”;1500
```

- Cada uno de estos archivos será convertido en una relación en memoria que se almacenara mediante el uso de la estructura de datos table de la Liberia GUAVA de google.
- Las operaciones aritméticas producirán siempre como resultado un entero.
- Se hará uso de la precedencia y asociatividad acostumbrada en las operaciones matemáticas para +,-,*,/
- Las operaciones relacionales son: menor (<), menor o igual (<=), mayor (>), mayor o igual (>=), igual (=) y diferente (!=).
- Los operadores lógicos son los de conjunción lógica (AND), disyunción lógica (OR) y Negación Lógica (NOT).
- Se pueden definir alias a las columnas para su uso posterior como se observa en las consultas de ejemplo al inicio.

El verificador semántico debe realizar las siguientes comprobaciones:

- No permitir el uso de atributos no declarados en ninguna relación, o declarados en otra relación (diferente a la que se usa)-
- En cuanto a las variables (entero, cadena, fecha) se debe verificar la compatibilidad de tipos a la hora de llevar a cabo cualquier operación aritmética o asignación.
- Este código debe ser llevado a cabo de forma manual en java como un recorrido especial extra al AST luego de su construcción y no mediante añadidos al código de CUP generado (analizador sintáctico).

La evaluación del proyecto se realizará en 3 fases:

1. Análisis léxico (5 puntos del primer parcial).
2. Comprobador de Tipos (30 puntos de segundo parcial).
3. Análisis semántico y ejecución interpretada del código (40 puntos del tercer parcial) .

Nota:

- El trabajo debe ser presentado en equipos de máximo 5 personas, donde **TODOS** los integrantes **deben trabajar** en el proyecto y conocer como se hace cada parte del mismo, y debido a que la universidad además de enseñarles conocimientos técnicos, les debe enseñar algún tipo de valores personales a sus estudiantes, como por ejemplo una ética mínima sobretodo a los estudiantes que ya están tan cerca de culminar sus estudios y convertirse en futuros Ingenieros, el hecho de entregar un compilador cuyo funcionamiento interno EN **CUALQUIER ASPECTO** sea una incógnita para **UNO** o todos los integrantes de un equipo conllevara a la perdida de nota correspondiente al equipo (mínimo 50% de la nota del proyecto **A TODOS SUS INTEGRANTES**) y en caso de ser una falta considerada grave, se elevara el caso hasta las instancias correspondientes en la universidad para la aplicación del reglamento interno.
- Aunque suena redundante les recuerdo que se validara mediante una entrega en vivo con asistencia física de **TODOS** los integrantes del equipo, el trabajo de los mismos para verificar lo anteriormente expuesto, y en caso de no asistir el estudiante no tendrá nota en el proyecto final de la materia.

Cronograma actividades:

- **REVISION #1 18/07 punto a revisar: AST debe ser generado correctamente.**
- **REVISION #2 25/07 punto a revisar: debe funcionar el verificador de tipos y la lectura de las relaciones en tablas de guava en memoria y ejecución de consultas simples debe mostrar resultados.**
- ***TERCER PARCIAL MATERIA 28/07/2013***
- **FECHA ENTREGA FINAL PROYECTO: Jueves 31 de Julio de 2014 NO HAY PRORROGA.**