Bases de Datos

Alumno: Luis Alfredo Cruz Gómez

Curso: Bases de Datos

Profesor: David Pineda Villagrán

Tarea 2
Álgebra Relacional

1. Objetivo.

Facilitar el aprendizaje de la escritura de las sentencias SQL, dado por la manipulación de datos de las relaciones con operaciones.

- Conocer las operaciones del álgebra relacional
- Saber utilizar las operaciones del álgebra relacional para consultar una base de datos
- Aprender a realizar su implementación en el programa SQL

2. Introducción.

El álgebra relacional permite entender el modelo relacional de bases de datos desde la perspectiva matemática. Aquí se introducen los fundamentos del álgebra relacional y se les relaciona con la estructura del lenguaje de consulta de bases de datos relacionales SQL, para conectar la teoría con la práctica.

3. ¿Qué es el Álgebra Relacional?

El álgebra relacional es un lenguaje de consulta procedimental. Consta de un conjunto de operaciones que toman como entrada una o dos relaciones y producen como resultado una nueva relación. Las operaciones fundamentales del álgebra relacional son selección, proyección, unión, diferencia de conjuntos, producto cartesiano y renombramiento. Además de las operaciones fundamentales hay otras operaciones, por ejemplo, intersección de conjuntos, reunión natural, división y

asignación. Estas operaciones se definirán en términos de las operaciones fundamentales.

Operaciones que describen paso a paso cómo computar una respuesta sobre las relaciones, tal y como éstas son definidas en el modelo relacional. Denominada de tipo procedimental, a diferencia del Cálculo relacional que es de tipo declarativo.

Describe el aspecto de la manipulación de datos. Estas operaciones se usan como una representación intermedia de una consulta a una base de datos y, debido a sus propiedades algebraicas, sirven para obtener una versión más optimizada y eficiente de dicha consulta.

4. Operadores

Operaciones fundamentales

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

La operación selección

La operación selección selecciona tuplas que satisfacen un predicado dado. Se utiliza la letra griega sigma minúscula (σ) para denotar la selección. El predicado aparece como subíndice de σ. La relación del argumento 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:

$$\sigma_{nombre\text{-}sucursal = \text{«Navacerrada»}} (préstamo)$$

Si la relación préstamo es como se muestra a continuación:

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

la relación que resulta de la consulta anterior es:

número-préstamo	*nombre-sucursal*	*importe*
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300

En general, se permiten las comparaciones que utilizan =, ≠, o ≥ en el predicado de selección. Además, se pueden combinar varios predicados en uno mayor utilizando las conectivas y (∧) y o (∨). Por tanto, para encontrar las tuplas correspondientes a préstamos de más de 1.200 concedidos por la sucursal de Navacerrada, se escribe:

$$\sigma_{nombre\text{-}sucursal = «Navacerrada» \wedge importe>1200}(préstamo)$$

El predicado de selección puede incluir comparaciones entre dos atributos. Para ilustrarlo, considérese la relación responsable-préstamo, que consta de tres atributos: nombre-cliente, nombre-banquero y número-préstamo, que especifica que un empleado concreto es el responsable del préstamo concedido a un cliente. Para hallar todos los clientes que se llaman igual que su responsable de préstamos se puede escribir:

La operación proyección

Supóngase que se desea hacer una lista de todos los números de préstamo y del importe de los mismos, pero sin que aparezcan los nombres de las sucursales. La operación **proyección** permite producir esta relación. 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 (Π). Se crea una lista de los atributos que se desea que aparezcan en el resultado como subíndice de Π. La relación de argumentos se escribe a continuación entre paréntesis. Por tanto, la consulta para crear una lista de todos los números de préstamo y del importe de los mismos puede escribirse como:

$$\Pi_{número\text{-}préstamo,\ importe}(préstamo)$$

La relación que resulta de esta consulta es la siguiente:

número-préstamo	importe
P-11	900
P-14	1.500
P-15	1.500
P-16	1.300
P-17	1.000
P-23	2.000
P-93	500

Composición de operaciones relacionales

Es importante el hecho de que el resultado de una operación relacional sea también una relación. Considérese la consulta más compleja «Encontrar los nombres de clientes que viven en Peguerinos». Hay que escribir:

$$\Pi_{nombre\text{-}cliente}\,(\,\sigma_{ciudad\text{-}cliente\,=\,\text{«Peguerinos»}}\,(cliente))$$

Téngase en cuenta que, en vez de dar en el argumento de la operación proyección el nombre de una relación, se da una expresión que se evalúa como una relación. En general, dado que el resultado de una operación del álgebra relacional es del mismo tipo (relación) que los datos de entrada, las operaciones del álgebra relacional pueden componerse para formar una **expresión del álgebra relacional**. La composición de operaciones del álgebra relacional para formar expresiones del álgebra relacional es igual que la composición de operaciones aritméticas (como +, −, * y ÷) para formar expresiones aritméticas.

La operación unión

Considérese una consulta para averiguar el nombre de todos los clientes del banco que tienen una cuenta, un préstamo o ambas cosas. Obsérvese que la relación cliente no contiene esa información:

nombre-cliente	*calle-cliente*	*ciudad-cliente*
Abril	Preciados	Valsaín
Amo	Embajadores	Arganzuela
Badorrey	Delicias	Valsaín
Fernández	Jazmín	León
Gómez	Carretas	Cerceda
González	Arenal	La Granja
López	Mayor	Peguerinos
Pérez	Carretas	Cerceda
Rodríguez	Yeserías	Cádiz
Rupérez	Ramblas	León
Santos	Mayor	Peguerinos
Valdivieso	Goya	Vigo

dado que los clientes no necesitan tener ni cuenta ni préstamo en el banco. Para contestar a esta consulta hace falta la información de la relación impositor:

nombre cliente	*número cuenta*
Abril	C-102
Gómez	C-101
González	C-201
González	C-217
López	C-222
Rupérez	C-215
Santos	C-305

y la de la relación prestatario

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

Se conoce la manera de averiguar los nombres de todos los clientes con préstamos en el banco:

$$\Pi_{nombre\text{-}cliente}\,(prestatario)$$

También se conoce la manera de averiguar el nombre de los clientes con cuenta en el banco:

$$\Pi_{nombre\text{-}cliente}\,(impositor)$$

Para contestar a la consulta hace falta la unión de estos dos conjuntos; es decir, hacen falta todos los nombres de clientes que aparecen en alguna de las dos relaciones o en ambas. Estos datos se pueden averiguar mediante la operación binaria unión, denotada, como en la teoría de conjuntos, por $\cup$. Por tanto, la expresión buscada es:

$$\Pi_{nombre\text{-}cliente}\,(prestatario) \cup \Pi_{nombre\text{-}cliente}\,(impositor)$$

La relación resultante de esta consulta es:

nombre-cliente
Abril
Fernández
Gómez
González
López
Pérez
Rupérez
Santos
Sotoca
Valdivieso

Téngase en cuenta que en el resultado hay diez tuplas, aunque hay siete prestatarios y seis impositores distintos. Esta discrepancia aparente se debe a que Gómez, Santos y López son a la vez prestatarios e impositores. Dado que las relaciones son conjuntos, se eliminan los valores duplicados.

Obsérvese que en este ejemplo se toma la unión de dos conjuntos, ambos consistentes en valores de *nombre-cliente*. En general, se debe asegurar que las uniones se realicen entre relaciones compatibles. Por ejemplo, no tendría sentido realizar la unión de las relaciones *préstamo* y *prestatario*. La primera es una relación con tres atributos, la segunda sólo tiene dos. Más aún, considérese la unión de un conjunto de nombres de clientes y de un conjunto de ciudades. Una unión así no tendría sentido en la mayor parte de los casos. Por tanto, para que una operación unión r ∪ s sea válida hay que exigir que se cumplan dos condiciones:

1. Las relaciones r y s deben ser de la misma aridad. Es decir, deben tener el mismo número de atributos.
2. Los dominios de los atributos i-ésimos de r y de s deben ser iguales para todo i.

Téngase en cuenta que r y s pueden ser, en general, relaciones temporales que sean resultado de expresiones del álgebra relacional.

La operación diferencia de conjuntos

La operación **diferencia de conjuntos**, denotada por –, permite buscar las tuplas que estén 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*. Se pueden buscar todos los clientes del banco que tienen abierta una cuenta pero no tienen concedido ningún préstamo escribiendo:

$$\Pi_{nombre\text{-}cliente}(impositor) - \Pi_{nombre\text{-}cliente}(prestatario)$$

La relación resultante de esta consulta es:

nombre-cliente
Abril
González
Rupérez

Como en el caso de la operación unión, hay que asegurarse de que las diferencias de conjuntos se realicen entre relaciones compatibles. Por tanto, para que una operación diferencia de conjuntos $r - s$ sea válida hay que exigir que las relaciones r y s sean de la misma aridad y que los dominios de los atributos *i-ésimos* de r y s sean iguales.

La operación producto cartesiano

La operación **producto cartesiano**, denotada por un aspa (×), permite combinar información de cualesquiera dos relaciones. El producto cartesiano de las relaciones *r1* y *r2* como $r1 \times r2$. Recuérdese que las relaciones se definen como subconjuntos del producto cartesiano de un conjunto de dominios. A partir de esta definición ya se debe tener una intuición sobre la definición de la operación producto cartesiano. Sin embargo, dado que el mismo nombre de atributo puede aparecer tanto en *r1* como en *r2*, hay que crear un esquema de denominaciones para distinguir entre ambos atributos. En este caso se logra adjuntando al atributo el nombre de la relación de la que proviene originalmente. Por ejemplo, el esquema de relación de $r = prestatario \times préstamo$ es:

(*prestatario.nombre-cliente, prestatario.número-préstamo, préstamo.nombre-sucursal, préstamo.número-préstamo, préstamo.importe*)

Con este esquema se puede distinguir entre *prestatario.número-préstamo* y *préstamo.número-préstamo*.

Para los atributos que sólo aparecen en uno de los dos esquemas se suele omitir el prefijo con el nombre de la relación. Esta simplificación no genera ambigüedad alguna. Por tanto, se puede escribir el esquema de relación de *r* como:

(nombre-cliente, prestatario.número-préstamo, nombre-sucursal, préstamo.número-préstamo, importe)

El acuerdo de denominaciones precedente exige que las relaciones que sean argumentos de la operación producto cartesiano tengan nombres diferentes. Esta exigencia causa problemas en algunos casos, como cuando se desea calcular el producto cartesiano de una relación consigo misma. Se produce un problema similar si se utiliza el resultado de una expresión del álgebra relacional en un producto cartesiano, dado que hará falta un nombre para la relación para poder hacer referencia a sus atributos. Para evitar estos problemas se utiliza la operación de renombramiento.

Ahora que se conoce el esquema de relación de *r = prestatario × préstamo* hay que averiguar las tuplas que aparecerán en *r*. Como se podía imaginar, se crea una tupla de *r* a partir de cada par de tuplas posible: una de la relación *prestatario* y otra de la relación *préstamo*.

Por tanto, *r* es una relación de gran tamaño, como se puede ver en la siguiente figura:

nombre-cliente	*prestatario.número-préstamo*	*préstamo.número-préstamo*	*nombre-sucursal*	*importe*
Santos	P-17	P-11	Collado Mediano	900
Santos	P-17	P-14	Centro	1.500
Santos	P-17	P-15	Navacerrada	1.500
Santos	P-17	P-16	Navacerrada	1.300
Santos	P-17	P-17	Centro	1.000
Santos	P-17	P-23	Moralzarzal	2.000
Santos	P-17	P-93	Becerril	500
Gómez	P-23	P-11	Collado Mediano	900
Gómez	P-23	P-14	Centro	1.500
Gómez	P-23	P-15	Navacerrada	1.500
Gómez	P-23	P-16	Navacerrada	1.300
Gómez	P-23	P-17	Centro	1.000
Gómez	P-23	P-23	Moralzarzal	2.000
Gómez	P-23	P-93	Becerril	500
López	P-15	P-11	Collado Mediano	900
López	P-15	P-14	Centro	1.500
López	P-15	P-15	Navacerrada	1.500
López	P-15	P-16	Navacerrada	1.300
López	P-15	P-17	Centro	1.000
López	P-15	P-23	Moralzarzal	2.000
...	...	...	...	...
...	...	...	...	...
Valdivieso	P-17	P-11	Collado Mediano	900
Valdivieso	P-17	P-14	Centro	1.500
Valdivieso	P-17	P-15	Navacerrada	1.500
Valdivieso	P-17	P-16	Navacerrada	1.300
Valdivieso	P-17	P-17	Centro	1.000
Valdivieso	P-17	P-23	Moralzarzal	2.000
Valdivieso	P-17	P-93	Becerril	500
Fernández	P-16	P-11	Collado Mediano	900
Fernández	P-16	P-14	Centro	1.500
Fernández	P-16	P-15	Navacerrada	1.500
Fernández	P-16	P-16	Navacerrada	1.300
Fernández	P-16	P-17	Centro	1.000
Fernández	P-16	P-23	Moralzarzal	2.000
Fernández	P-16	P-93	Becerril	500

donde sólo se ha incluido una parte de las tuplas que forman parte de *r*.

Supóngase que se tienen *n1* tuplas en *prestatario* y *n2* tuplas en *préstamo*. Por tanto, hay *n1 * n2* maneras de escoger un par de tuplas, una tupla de cada relación; por lo que hay *n1 * n2* tuplas en *r*. En concreto, obsérvese que para algunas tuplas *t* de *r* puede ocurrir que *t[prestatario.número-préstamo] ≠ r[préstamo.número-préstamo]*.

Supóngase que se desea averiguar los nombres de todos los clientes que tienen concedido un préstamo en la sucursal de *Navacerrada*. Se necesita para ello información de las relaciones *préstamo* y *prestatario*. Si se escribe

$$\sigma_{\text{nombre-sucursal} = \text{«Navacerrada»}} (prestatario \; x \; préstamo)$$

entonces el resultado es la relación será:

nombre-cliente	*prestatario.número-préstamo*	*préstamo.número-préstamo*	*nombre-sucursal*	*importe*
Santos	P-17	P-15	Navacerrada	1.500
Santos	P-17	P-16	Navacerrada	1.300
Gómez	P-23	P-15	Navacerrada	1.500
Gómez	P-23	P-16	Navacerrada	1.300
López	P-15	P-15	Navacerrada	1.500
López	P-15	P-16	Navacerrada	1.300
Sotoca	P-14	P-15	Navacerrada	1.500
Sotoca	P-14	P-16	Navacerrada	1.300
Pérez	P-93	P-15	Navacerrada	1.500
Pérez	P-93	P-16	Navacerrada	1.300
Gómez	P-11	P-15	Navacerrada	1.500
Gómez	P-11	P-16	Navacerrada	1.300
Valdivieso	P-17	P-15	Navacerrada	1.500
Valdivieso	P-17	P-16	Navacerrada	1.300
Fernández	P-16	P-15	Navacerrada	1.500
Fernández	P-16	P-16	Navacerrada	1.300

Se tiene una relación que sólo atañe a la sucursal de Navacerrada. Sin embargo, la columna *nombre-cliente* puede contener clientes que no tengan concedido ningún préstamo en la sucursal de Navacerrada. (Si no se ve el motivo por el que esto es cierto, recuérdese que el producto cartesiano toma todos los emparejamientos posibles de una tupla de prestatario con una tupla de préstamo.)

Dado que la operación producto cartesiano asocia todas las tuplas de *préstamo* con todas las tuplas de *prestatario*, se sabe que, si un cliente tiene concedido un préstamo en la sucursal de Navacerrada, hay alguna tupla de *prestatario x préstamo* que contiene su nombre y que *prestatario.número-préstamo = préstamo.número-préstamo*. Por tanto, si escribimos

$$\sigma_{prestatario.número\text{-}préstamo = préstamo.número\text{-}préstamo}$$

$$(\sigma_{nombre\text{-}sucursal = «Navacerrada»} (prestatario \times préstamo))$$

sólo se obtienen las tuplas de *prestatario x préstamo* que corresponden a los clientes que tienen concedido un préstamo en la sucursal de Navacerrada. Finalmente, dado que sólo se desea obtener *nombre-cliente*, se realiza una proyección:

$$\Pi_{nombre\text{-}cliente} (\sigma_{prestatario.número\text{-}préstamo = préstamo.número\text{-}préstamo}$$

$$(\sigma_{nombre\text{-}sucursal = «Navacerrada»} (prestatario \times préstamo)))$$

El resultado de esta expresión se muestra en la siguiente figura que muestra la respuesta correcta a la consulta formulada.

nombre-cliente
Férnandez
López

La operación renombramiento

A diferencia de las relaciones de la base de datos, los resultados de las expresiones de álgebra relacional no tienen un nombre que se pueda utilizar para referirse a ellas.- Resulta útil poder ponerles nombre; el operador **renombramiento**, denotado por la letra griega rho minúscula (ρ), permite realizar esta tarea.

Para ilustrar el uso del renombramiento de las relaciones, considérese la consulta «Buscar el máximo saldo de cuenta del banco». La estrategia empleada para obtener el resultado es:

1) Calcular una relación intermedia consistente en los saldos que no son el máximo y

2) realizar la diferencia entre la relación $\Pi_{saldo}(cuenta)$ y la relación intermedia recién calculada.

Paso 1: Para calcular la relación intermedia hay que comparar los valores de los saldos de todas las cuentas. Esta comparación se puede hacer calculando el producto cartesiano *cuenta x cuenta* y formando una selección para comparar el valor de cualesquiera dos saldos que aparezcan en una tupla. En primer lugar hay que crear un mecanismo para distinguir entre los dos atributos *saldo.* Se utilizará la operación **renombramiento** para cambiar el nombre de una referencia a la relación *cuenta*; así, se puede hacer referencia dos veces a la relación sin ambigüedad alguna.

La relación temporal que se compone de los saldos que no son el máximo puede escribirse ahora como

$$\Pi_{cuenta.saldo}\ (\ \sigma_{cuenta.saldo\ <\ d.saldo}\ (cuenta \cdot \rho d\ (cuenta)))$$

Esta expresión proporciona los *saldos* de la relación *cuenta* para los que aparece un saldo mayor en alguna parte de la relación *cuenta* (**cuyo nombre se ha cambiado a d**). El resultado contiene todos los saldos salvo el máximo. Esta relación se muestra en la siguiente figura:

saldo
500
400
700
750
350

Paso 2: La consulta para averiguar el máximo saldo de cuenta del banco puede escribirse de la manera siguiente:

$$\Pi_{saldo}(cuenta) - \Pi_{cuenta.saldo}(\sigma_{cuenta.saldo < d.saldo}(cuenta \cdot \rho d(cuenta)))$$

En la siguiente figura:

saldo
900

se muestra el resultado de esta consulta. Considérese la siguiente consulta como un nuevo ejemplo de la operación renombramiento: «**Averiguar los nombres de todos los clientes que viven en la misma calle y en la misma ciudad que Gómez**». Se puede obtener la calle y la ciudad en la que vive *Gómez* escribiendo:

$$\Pi_{calle\text{-}cliente,\ ciudad\text{-}cliente}(\sigma_{nombre\text{-}cliente\ =\ «Gómez»}(cliente))$$

Sin embargo, para hallar a otros *clientes* que vivan en esa calle y en esa ciudad hay que hacer referencia por segunda vez a la relación *cliente*. En la consulta siguiente se utiliza la operación **renombramiento** sobre la expresión anterior para darle al resultado el nombre *dirección-Gómez* y para cambiar el nombre de los atributos a *calle* y *ciudad* en lugar de *calle-cliente y ciudad-cliente*:

$$\Pi_{cliente.nombre\text{-}cliente}(\sigma_{cliente.calle\text{-}cliente\ =\ dirección\text{-}Gómez.calle} \wedge cliente.ciudad\text{-}cliente = dirección\text{-}Gómez.ciudad\ (cliente \cdot \rho_{dirección\text{-}Gómez\ (calle,\ ciudad)}(\Pi_{calle\text{-}cliente,\ ciudad\text{-}cliente}(\sigma_{nombre\text{-}cliente\ =\ «Gómez»}(cliente)))))$$

El resultado de esta consulta, cuando se aplica a la relación cliente es:

nombre-cliente
Gómez
Pérez

Otras operaciones

Las operaciones fundamentales del álgebra relacional son suficientes para expresar cualquier consulta del álgebra relacional. Sin embargo, si uno se limita únicamente a las operaciones fundamentales, algunas consultas habituales resultan de expresión intrincada. Por tanto, se definen otras operaciones que no añaden potencia al álgebra, pero que simplifican las consultas habituales.

La operación intersección de conjuntos

Supóngase que se desea averiguar todos los clientes que tienen un préstamo concedido y una cuenta abierta. Utilizando la intersección de conjuntos (∩) se puede escribir:

$$\Pi_{nombre\text{-}cliente}(prestatario) \cap \Pi_{nombre\text{-}cliente}(impositor)$$

nombre-cliente
Gómez
Pérez
Santos

Obsérvese que se puede volver a escribir cualquier expresión del álgebra relacional utilizando la intersección de conjuntos sustituyendo la operación intersección por un par de operaciones de diferencia de conjuntos, de la manera siguiente:

$$r \cap s = r - (r - s)$$

La operación reunión natural

Suele resultar deseable simplificar ciertas consultas que exigen un producto cartesiano. Generalmente, las consultas que implican un producto cartesiano incluyen un operador selección sobre el resultado del producto cartesiano.

Considérese la consulta «Hallar los nombres de todos los clientes que tienen concedido un préstamo en el banco y averiguar el importe del mismo». En primer lugar se calcula el producto cartesiano de las relaciones prestatario y préstamo. Luego, se seleccionan las tuplas que sólo atañen al mismo número-préstamo, seguidas por la proyección de nombre-cliente, número-préstamo e importe resultantes:

La operación división

La operación división, denotada por ÷, resulta adecuada para las consultas que incluyen la expresión «para todos». Supóngase que se desea hallar a todos los clientes que tengan abierta una cuenta en todas las sucursales ubicadas en la ciudad de *Arganzuela*. Se pueden obtener todas las sucursales de Arganzuela mediante la expresión:

$$r1 = \Pi_{nombre\text{-}sucursal}\ (\ \sigma_{ciudad\text{-}sucursal\ =\ «\text{Arganzuela}»}\ (sucursal))$$

La relación resultante de esta expresión es:

nombre-sucursal
Centro
Galapagar

5.Implementación en SQL.

El lenguaje SQL se puede considerar como una de las principales razones del éxito comercial de las bases de datos relacionales.

El nombre SQL significa Lenguaje de consulta estructurado (Structured Que/)I Langllage). Originalmente,SQL se denominaba SEQUEL (Stl'uctllred English QUE/)I Language) y fue diseñado e implementado por IBM Research a modo de interfaz para un sistema de base de datos relacional conocido como SYSTEM R. SQL es ahora el lenguaje estándar de los DBMSs relacionales comerciales.

SQL utiliza los términos tabla, fila y columna para los términos relación, tupla y atributo del modelo relacional formal, respectivamente.El principal comando de SQL para definir datos es la sentencia CREATE, que se utiliza para crear esquemas, tablas (relaciones) y dominios (así como otras estructuras, como vistas, aserciones y triggers).

Un esquema SQL se identifica con un nombre de esquema e incluye un identificador de autorización para indicar el usuario o la cuenta propietaria del esquema, así como unos descriptores para cada elemento. Los elementos del esquema son las tablas, las restricciones, las vistas, los dominios y otras estructuras

Los tipos de datos básicos disponibles para los atributos son numéricos, cadena de caracteres, cadena de bits, booleano, fecha y hora

El tipo de datos numéricos incluye los números enteros de varios tamaños (lNTEGER o INT, Y SMALLINT) así como los números en coma flotante (reales) de distintas precisiones (FLOAT o REAL, Y DOUBLE PRECISION

El tipo de datos cadena de caracteres puede ser de longitud fija [CHAR(U) o CHARACTER(n), donde n es la cantidad de caracteres] o de longitud variable [VARCHAR(n) oCHAR VARYING(n) o CHARACTER VARYING(n), donde 11 es la cantidad máxima de caracteres]. Al especificar un valor de cadena literal, se coloca entre comillas simples (apóstrofes) y disting e entre minúsculas y mayúsculas

El tipo de datos cadena de bits es de longitud fija n [BIT(I1)] o de longitud variable [BIT VARYING(ll)],donde 11 es la cantidad máxima de bits.

Un tipo de datos booleano tiene los valores tradicionales TRUE o FALSE. En SQL, debido a la presencia de los valores NULL, se utiliza una lógica de tres valores, que permite un tercer valor: UNKNOWN.

En SQL2 se añadieron dos nuevos tipos de datos, fecha y hora. El tipo de datos DATE tiene diez posiciones y sus componentes son AÑO, MES Y DíA según la formaAAAA-MM-DD.

Como en el caso de los más modernos lenguajes relacionales, SQL está basado en el cálculo relacional de tuplas. Como resultado, toda consulta formulada utilizando el cálculo relacional de tuplas (o su equivalente, el álgebra relacional) se pude formular también utilizando SQL. Hay, sin embargo, capacidades que van más allá del cálculo o del álgebra relaciona. Aquí tenemos una lista de algunas características proporcionadas por SQL que no forman parte del álgebra y del cálculo relacionales:

- Comandos para inserción, borrado o modificación de datos.
- Capacidades aritméticas: En SQL es posible incluir operaciones aritméticas así como comparaciones, por ejemplo A < B + 3. Nótese que ni + ni otros operadores aritméticos aparecían en el álgebra relacional ni en cálculo relacional.
- Asignación y comandos de impresión: es posible imprimir una relación construida por una consulta y asignar una relación calculada a un nombre de relación.
- Funciones agregadas: Operaciones tales como promedio (average), suma (sum), máximo (max), etc. se pueden aplicar a las columnas de una relación para obtener una cantidad única.

Selec

El co za para recup

```
SELECT [ALL|DISTINCT]
       { * | expr_1 [AS c_alias_1] [, ...
             [, expr_k [AS c_alias_k]]]}
FROM table_name_1 [t_alias_1]
     [, ... [, table_name_n [t_alias_n]]]
[WHERE condition]
[GROUP BY name_of_attr_i
          [,... [, name_of_attr_j]] [HAVING condition]]
[{UNION [ALL] | INTERSECT | EXCEPT} SELECT ...]
[ORDER BY name_of_attr_i [ASC|DESC]
          [, ... [, name_of_attr_j [ASC|DESC]]]];
```

Select sencillas

Aquí tenemos algunos ejemplos sencillos utilizando la instrucción SELECT: Para recuperar todas las tuplas de la tabla PART donde el atributo PRICE es mayor que 10, formularemos la siguiente consulta:

```
SELECT * FROM PART
 WHERE PRICE > 10;
```

y obtenemos la siguiente tabla:

```
 PNO |  PNAME      |  PRICE
-----+-------------+--------
  3  |  Cerrojos   |   15
  4  |  Levas      |   25
```

Utilizando "*" en la instrucción SELECT solicitaremos todos los atributos de la tabla. Si queremos recuperar sólo los atributos PNAME y PRICE de la tabla PART utilizaremos la instrucción:

```
SELECT PNAME, PRICE
FROM PART
WHERE PRICE > 10;
```

En este caso el resultado es:

```
 PNAME      |  PRICE
------------+--------
 Cerrojos   |   15
 Levas      |   25
```

Joins (Cruces)

El siguiente ejemplo muestra como las joins (cruces) se realizan en SQL.Para cruzar

tres tablas SUPPLIER, PART y SELLS a través de sus atributos comunes, formularemos la siguiente instrucción:

```
SELECT S.SNAME, P.PNAME
FROM SUPPLIER S, PART P, SELLS SE
WHERE S.SNO = SE.SNO AND
      P.PNO = SE.PNO;
```

y obtendremos la siguiente tabla como resultado:

```
 SNAME | PNAME
-------+-------
 Smith | Tornillos
 Smith | Tuercas
 Jones | Levas
 Adams | Tornillos
 Adams | Cerrojos
 Blake | Tuercas
 Blake | Cerrojos
 Blake | Levas
```

En la cláusula FROM hemos introducido un alias al nombre para cada relación porque hay atributos con nombre común (SNO y PNO) en las relaciones. Ahora podemos distinguir entre los atributos con nombre común simplificando la adicción de un prefijo al nombre del atributo con el nombre del alias seguido de un punto.
Primero el producto cartesiano: SUPPLIER × PART × SELLS Ahora seleccionamos únicamente aquellas tuplas que satisfagan las condiciones dadas en la cláusula WHERE (es decir, los atributos con nombre común deben ser iguales). Finalmente eliminamos las columnas repetidas (S.SNAME, P.PNAME).

Operadores Agregados

SQL proporciona operadores agregados (como son AVG, COUNT, SUM, MIN, MAX) que toman el nombre de un atributo como argumento. El valor del operador agregado se calcula sobre todos los valores de la columna especificada en la tabla completa. Si se especifican grupos en la consulta, el cálculo se hace sólo sobre los valores de cada grupo.

Agregación por Grupos

SQL nos permite particionar las tuplas de una tabla en grupos. En estas

condiciones, los operadores agregados descritos antes pueden aplicarse a los grupos (es decir, el valor del operador agregado no se calcula sobre todos los valores de la columna especificada, sino sobre todos los valores de un grupo. El operador agregado se calcula individualmente para cada grupo).
El particionamiento de las tuplas en grupos se hace utilizando las palabras clave GROUP BY seguidas de una lista de atributos que definen los grupos. Si tenemos GROUP BY A1, ···, Ak habremos particionado la relación en grupos, de tal modo que dos tuplas son del mismo grupo si y sólo si tienen el mismo valor en sus atributos A1, ···, Ak.

Having
La cláusula HAVING trabaja de forma muy parecida a la cláusula WHERE, y se utiliza para considerar sólo aquellos grupos que satisfagan la cualificación dada en la misma. Las expresiones permitidas en la cláusula HAVING deben involucrar funcionen agregadas. Cada expresión que utilice sólo atributos planos deberá recogerse en la cláusula WHERE. Por otro lado, toda expresión que involucre funciones agregadas debe aparecer en la clausula HAVING.

Subconsultas
En las cláusulas WHERE y HAVING se permite el uso de subconsultas (subselects) en cualquier lugar donde se espere un valor. En este caso, el valor debe derivar de la evaluación previa de la subconsulta. El uso de subconsultas amplía el poder expresivo de SQL.

6. Ejemplo

Si tuviésemos la siguiente extensión de la tabla PROFESORES

dni	apellidos	nombre	fecha-nacimiento	es-doctor	teléfono-móvil	D_nombre
32323	Pérez	Juan	30/10/1965	cierto	65454545	Matemáticas
4434343	Díez	José	30/10/1970	cierto	65789767	Historia
23423432	Sánchez	Juan	30/10/1980	falso	66789899	Matemáticas

La siguiente sentencia SELECT hace una proyección del nombre y los apellidos.

```
SELECT nombre, apellidos FROM PROFESORES
```

Es importante notar que la siguiente sentencia no devolvería una relación, ya que debería dos tuplas repetidas.

```
SELECT nombre FROM profesores
```

Para asegurar que el resultado es una relación en el sentido matemático, hay que utilizar el modificador DISTINCT.

```
SELECT DISTINCT nombre FROM profesores
```

La selección permite elegir algunas tuplas

La selección es una operación que elige algunas tuplas de una relación y elimina el resto. La nueva relación contiene por tanto solo tuplas seleccionadas que cumplen una determinada **condición de selección C** La condición de selección es una condición lógica que permite decidir qué incluir y qué no.

Los atributos de la selección son los mismos que los de la relación original, y todas las tuplas de la selección cumplen la condición C. Por ejemplo:

```
R := DEPARTAMENTOS(grado-experimentalidad >1.0)
```

```
Q := PROFESORES(no es-doctor y teléfono-móvil <> null)
```

En SQL, la selección se especifica mediante la cláusula WHERE. Por ejemplo:

```
SELECT * FROM DEPARTAMENTOS
```

```
WHERE grado-experimentalidad > 1.0
```

```
SELECT * FROM PROFESORES
```

```
WHERE not es doctor and teléfono-móvil IS NOT NULL.
```

La combinación o reunión (join) permite cruzar los valores de tablas relacionadas

La combinación toma dos relaciones y devuelve una relación con las tuplas que resultan de concatenar tuplas de la primera con tuplas de la segunda y después

seleccionar las que cumplen una **condición de combinación**C . Una combinación entre las relaciones R y Q mediante la condición C se denota como R[C]Q.

Por ejemplo, si queremos obtener la información de profesores y departamentos combinada, utilizaremos como combinación la igualdad de la clave ajena en PROFESORES con la clave primaria en DEPARTAMENTOS:

DPT:=DEPARTAMENTOS

PROFESORES[D_nombre = DPT.nombre]DPT

En SQL, la combinación se hace incluyendo más de una relación en la cláusula FROM, y la condición C se coloca como cláusula WHERE.

```
SELECT *

FROM PROFESORES AS P, DEPARTAMENTOS AS D

WHERE P.D_nombre = D.nombre
```

La equicombinación

La equicombinación (equireunión o equijoin) es una combinación que en la condición C solo tiene igualdades. Es especialmente importante porque permite obtener información relacionada con las claves ajenas de las tablas.

combinación natural

Cuando se hace un equicombinación, la relación resultante tiene los atributos de las dos relaciones originales. Dado que se está exigiendo igualdad de valores en al menos un atributo, la relación resultante tendrá atributos con información repetida (tantos como atributos hayan sido comparados en la condición C). La combinación natural es una variante del equicombinación donde se eliminan esos atributos superfluos.

En SQL, la combinación natural puede hacerse combinándola con una proyección de los atributos de las dos relaciones que excluya alguno de los

Operaciones de teoría de conjuntos

La unión de relaciones es la unión de conjuntos

La unión de dos relaciones devuelve una nueva relación que contiene todas las tuplas que aparecían en cualquiera de las relaciones originales, o en ambas.

Así, la operación de unión de dos relaciones R y T, que denominaremos UNION (R, T) sólo se puede aplicar el operador unión a dos relaciones que tengan los mismos atributos, o que sean compatibles (es decir, que se pueda establecer una biyección entre los atributos de las dos relaciones).

Es importante tener en cuenta lo siguiente:

- Se eliminarán las tuplas repetidas. Se entiende que en las dos relaciones no debe haber tuplas con la misma clave primaria y el resto de la información diferente.

Si tenemos dividida la información de los profesores antiguos y los nuevos en dos relaciones (por ejemplo, por motivos de rendimiento), pero queremos obtener un listado histórico de todos ellos, podemos utilizar el operador UNION de SQL.

```
SELECT * FROM PROFESORES

UNION

SELECT * FROM EX-PROFESORES
```

El operador UNION requiere que los atributos de las relaciones devueltas por las dos SELECT sean del mismo tipo según el orden de aparición. Si en una de las tablas este orden sería diferente, habría que utilizar una proyección que alterase el orden. Los nombres de atributos de la nueva relación serán los de la primera SELECT

La intersección de relaciones es la intersección de conjuntos

La intersección de relaciones toma dos relaciones y devuelve una relación con las tuplas que aparecían en ambas relaciones originales. Como la unión, la intersección solo puede actuar a relaciones con atributos compatibles.

La siguiente consulta en SQL por tanto debería devolver una relación vacía:

```
SELECT * FROM PROFESORES

INTERSECT

SELECT * FROM EX-PROFESORES
```

El producto cartesiano combina todas las tuplas de dos relaciones

El producto cartesiano es una operación que toma dos relaciones y obtiene una nueva relación con las tuplas son la concatenación de las tuplas de la primera relación con las tuplas de la segunda. En general, si la primera relación de entrada

tenía N tuplas y la segunda M tuplas, el producto cartesiano produce N × M tuplas, aunque en ocasiones serán menos dado que las tuplas repetidas habrán de eliminarse.

El producto cartesiano de dos relaciones en SQL se obtiene simplemente colocando más de una tabla en la cláusula FROM.

```
SELECT * FROM PROFESORES, DEPARTAMENTOS
```

El producto cartesiano raramente se utiliza en la práctica, pero es interesante conocerlo para diferenciarlo de la operación de combinación (unirse).

7. Conclusión

De forma personal puedo decir que Las operaciones del álgebra relacional sirven para hacer consultas a una base de datos. Es preciso conocer estas operaciones porque nos permiten saber qué servicios de consulta debe proporcionar un lenguaje relacional.

es muy importante por varias razones. La primera, porque proporciona un fundamento formal para las operaciones del modelo relacional. La segunda razón, y quizá la más importante, es que se utiliza como base para la implementación y optimización de consulta. Tercera, porque algunos de sus conceptos se han incorporado al lenguaje estándar de consultas SQL, lenguaje muy utilizado por las empresas para el manejo adecuado de sus bases de datos y su posterior análisis para toma de decisiones eficientes. Por ende podemos considerar el álgebra relacional como un precursor de la analítica y la toma de decisiones basadas en hechos

Fuentes:

1.http://fcays.ens.uabc.mx/anterior/BD/AlgebraRelacional

2. https://cnx.org/contents/74gU77-S@1/Introducci%C3%B3n-al%C3%A1lgebra-relacional-Con-ejemplos-en-SQL

3.http://openaccess.uoc.edu/webapps/o2/bitstream/10609/200/8/Bases%20de%20datos_M%C3%B3dulo2_El%20modelo%20relacional%20y%20el%20%C3%A1lgebra%20relacional.