SISTEMAS DE BASES DE DATOS – INGENIERÍA INFORMÁTICA

Manipulación de datos en el modelo relacional

Alcance:

Abarcaremos en este tema, desde la base conceptual del tratamiento de datos en el modelo relacional, revisando aspectos del álgebra relacional y el cálculo relacional orientado a tuplas, base de la implementación tecnológica del lenguaje SQL, estándar de todos los sistemas de bases de datos relacionales del mercado, a través del cual se realiza el manejo de los datos almacenados en las mismas.

Conocimientos previos:

Asumimos aquí que los conceptos básicos de dominio, atributo, relación, tupla, clave candidata, clave primaria, clave foránea, etc. los cuales corresponden a definiciones del modelo relacional son conocidos y comprendidos.
También asumiremos conocimientos teóricos acerca de álgebra y cálculo relacional y de modelización y normalización.
NOTA: Estos últimos serán revisados en la segunda unidad de este módulo, pero estamos asumiendo aquí que los conceptos teóricos ya han sido adquiridos por Ud. antes de iniciar esta asignatura.

Forma de trabajo:

Se abordará este tema desde un punto de vista práctico, brindando diferentes técnicas de aplicación de los conceptos adquiridos.

Desarrollo:

ÁLGEBRA RELACIONAL:

Vamos a considerar aquí las ocho operaciones básicas del álgebra relacional y dos operaciones adicionales que no forman parte del AR, pero que nos facilitarán la comprensión y aplicación de SQL en la resolución de algunos problemas.

Definiciones iniciales:

- El álgebra relacional está constituida por un conjunto de operadores de alto nivel no procedurales, basados en dos conceptos matemáticos, la teoría de conjuntos y la lógica proposicional.
- Los operadores del álgebra relacional se aplican a relaciones (operandos).

- El álgebra relacional es cerrada, es decir, el resultado de una expresión del AR, la cual es aplicada a una o más relaciones, es también una relación. Es decir, los operadores del álgebra relacional se aplican a una o varias relaciones y como resultado de esto se obtiene siempre otra relación. Esto permite aplicar a este resultado un nuevo operador del AR.

- La solución a un requerimiento determinado, expresado en álgebra relacional, debe ser siempre una expresión única. Es decir, los resultados parciales no se pueden "grabar" en una relación temporal y luego utilizarlos en otra expresión. Una solución con SQL y, fundamentalmente con SQL-extendido, puede permitir la obtención y utilización posterior de resultados parciales utilizando más de una expresión SQL. De todas maneras, se deberá tener en cuenta la performance de esa solución a la hora de elegirla.

- Expresión escalar: expresión que devuelve como resultado un único valor de algún tipo determinado.

- Expresión relacional: expresión que devuelve como resultado una relación. Por ejemplo, las expresiones del AR son expresiones relacionales

- Expresión tabular: expresión que devuelve como resultado una tabla. Es el equivalente a expresión relacional en la implementación en SQL de una expresión en AR.

- Valor nulo: un valor nulo no forma parte de ningún dominio y su significado es: ausencia de información o información no aplicable. No es un cero, ni un string vacío, ni ningún otro valor. Simplemente no existe un valor para el atributo.
  En cualquier expresión escalar, si uno de los operandos tiene valor nulo, el resultado de la expresión será nulo (salvo algunas excepciones que se analizarán específicamente).

Dejemos establecida ahora la sintaxis que utilizaremos y algunas consideraciones acerca de las operaciones del álgebra relacional. Los resultados que producen estos operadores podrán estudiarse directamente de la bibliografía recomendada.

Proyección:

Sintaxis: ***relación [lista de atributos]***

NOTA: "relación" puede ser una relación propiamente dicha o una expresión del AR cerrada entre paréntesis, que, como dijimos, devuelve como resultado siempre

una relación. Aplicar un operador a una expresión del AR cerrada entre paréntesis permite anidar expresiones del AR.

Restricción:

Sintaxis: ***relación WHERE condición***

NOTA1: condición es una expresión lógica simple (comparación) o compuesta (expresiones lógicas simples combinadas con operadores AND, OR y NOT). Las tuplas seleccionadas serán aquellas para las cuales la "condición" es verdadera.

NOTA2: en AR los operadores de comparación tales como =, <>, !=, >, <, >=, <=, etc. solo se aplican a atributos (operandos escalares) y no se pueden aplicar a relaciones (operandos relacionales) por más que las mismas sean de tamaño (1,1) (una tupla de un atributo). En SQL, se extiende la aplicación de estos operadores a expresiones relacionales de tamaño (1,1). Es decir, las tablas de una fila por una columna se pueden utilizar como valores escalares.

NOTA3: Además de los operadores de comparación clásicos en AR, SQL incorpora otros que facilitan la escritura de expresiones lógicas. Para más información ver referencia de expresiones lógicas o booleanas en cualquier libro o manual de SQL o esperar hasta el desarrollo del próximo módulo. Aquellos que tengan algún conocimiento de SQL, podrán utilizarlas en la ejercitación planteada en este módulo.

Producto cartesiano (CROSS JOIN):

Sintaxis: ***relación X relación***

Requiere que los operandos (expresiones relacionales) tengan encabezamientos disjuntos (no tienen ningún atributo común)

Reunión natural (NATURAL JOIN, INNER JOIN o simplemente JOIN):

Sintaxis: ***relación JOIN relación***

Es la operación más importante del AR. Requiere que los operandos (expresiones relacionales) tengan encabezamientos que se intersecten (tienen algún atributo común).

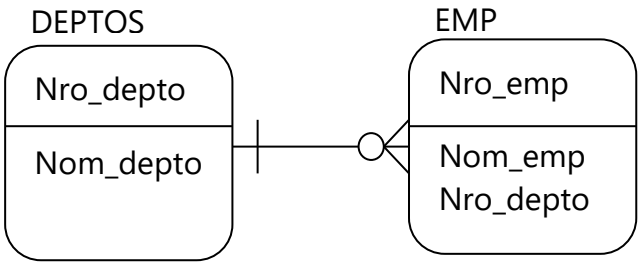
Es una operación no primaria. Se puede obtener aplicando una proyección, de una restricción de un producto cartesiano.

En SQL la operación JOIN (o INNER JOIN), es en realidad un EQUIJOIN, el cual es similar al JOIN, pero sin la proyección final.

Si recordamos el proceso de normalización, podremos deducir que en la mayoría de los casos esta operación JOIN se utiliza para "reunir" información acerca de alguna entidad, la cual posiblemente fue "distribuida" por el proceso de normalización (a través de la operación de proyección) en diferentes relaciones.

Ejemplo:

Dadas las siguientes entidades y su asociación.



Podemos ver que, en cualquier caso (deptos JOIN emp o emp JOIN deptos), la operación JOIN lo que hace es ampliar la información de los empleados incorporándole el nombre del departamento en el cual trabajan.
Si lo pensamos desde el punto de vista del proceso de normalización, podríamos ver que estas dos entidades o relaciones normalizadas surgieron de una primera relación no normalizada:

deptos (nro_depto, nom_depto, {nro_emp, nom_emp})

o de una relación plana en 1FN:

emp (nro_emp, nom_emp, nro_depto, nom_depto)

que al normalizarlas produjeron las relaciones deptos y emp.

Ahora bien, la operación JOIN "asocia" cada tupla de una de las relaciones con aquellas tuplas de la otra relación que tienen el mismo valor (=) para los mismos atributos.
Por lo tanto, habrá tuplas en la primera relación, que no tienen correspondencia en la segunda y, por lo tanto, no serán parte del resultado. Lo mismo se puede decir para la segunda relación.

NOTA: Si Ud. aún no conoce las características del proceso de normalización, no se preocupe, cuando esté estudiando el tema, vuelva a revisar la operación JOIN. Esto le ayudará también a afianzar ambos conceptos.

Unión:

Sintaxis: ***relación UNION relación***

Esta operación requiere que los encabezamientos de ambas relaciones sean iguales (los mismos atributos).

Intersección:

Sintaxis: ***relación INTERSECT relación***

Esta operación, al igual que la operación UNION, requiere que los encabezamientos de ambas relaciones sean iguales (los mismos atributos).

Diferencia:

Sintaxis: ***relación – relación***

Esta operación, al igual que la operación UNION, requiere que los encabezamientos de ambas relaciones sean iguales (los mismos atributos).

NOTA1: Esta operación no es conmutativa, es decir A – B es diferente a B – A.

NOTA2: No es equivalente a la operación escalar matemática. Esta operación descarta o elimina las tuplas de la primera relación (minuendo) que aparecen en la segunda relación (sustraendo). Es decir, se "sustraen" las tuplas de la primera relación que están en la segunda, quedando solo aquellas que NO están en la segunda. La comparación se realiza atributo por atributo con todos los atributos de cada relación, es por eso que los encabezamientos deben ser iguales.

División:

Sintaxis: ***relación / relación***

Se requiere que el encabezamiento de la relación divisor esté incluido en la relación dividendo.

El resultado contiene "todas" las combinaciones diferentes de valores de los atributos propios del dividendo (atributos que no están en la relación divisor), que están combinados o asociados a todas las tuplas de la relación divisor.

Operadores adicionales (no incluidos dentro de los 8 operadores originales);

Extensión:

Permite obtener un nuevo atributo aplicando una operación escalar sobre los atributos de la relación original.
Para facilitar la construcción de expresiones del AR, se utilizará una sintaxis similar a la de la operación proyección.

Sintaxis: ***relación [atributo AS expresión escalar, ...]***

En la práctica, las operaciones de proyección y extensión, cuando se deban aplicar sobre la misma relación, se combinarán aprovechando su sintaxis similar.

Ejemplo: utilizando la relación emp, obtener el nombre de cada empleado y el 10% de su salario:

EMP

Nro_emp	**Nom_emp**	**Salario**
1	Empleado 1	1500.00
2	Empleado 2	2000.00
3	Empleado 3	3000.00
4	Empleado 4	800.00
5	Empleado 5	500.00

La expresión combinada (proyección y extensión) será:

emp [nom_emp, diez_por_ciento_salario AS salario * 10.00 / 100.00]

Resumen (SUMMARYZE):

Sintaxis: ***relación GROUP BY lista de atributos ADD atributo AS funcion_grupal, ...***

Esta operación se utiliza para obtener un resumen de información, agrupando tuplas de la relación original que tienen los mismos valores para los atributos de

la lista y aplicando las funciones grupales especificadas a cada grupo, para obtener nuevos atributos.

El resultado consta de un encabezamiento con los atributos por los que se agrupa y los nuevos atributos obtenidos a partir de las funciones.

NOTA: la cláusula GROUP BY es opcional. Si no se incluye se construye un único grupo con todas las tuplas de la relación original, al cual se aplicarán las funciones de grupo programadas.

Funciones grupales más utilizadas:

- COUNT: tiene varias posibilidades.

  COUNT(*): cuenta la cantidad de tuplas en el grupo sin interesar el contenido de los atributos.
  COUNT(expresión escalar): cuenta la cantidad de tuplas en el grupo para las cuales el resultado de la expresión escalar es NO NULO.
  COUNT (DISTINCT expresión escalar): cuenta la cantidad de resultados DISTINTOS Y NO NULOS para la expresión escalar en cada grupo.

- SUM: tiene varias posibilidades.

  SUM(expresión escalar): obtiene la suma del resultado de la expresión escalar en cada grupo, descartando los resultados nulos de la expresión.

  SUM (DISTINCT expresión escalar): obtiene la suma de los resultados DISTINTOS para la expresión escalar en cada grupo, descartando los resultados nulos de la expresión.

- AVG: tiene varias posibilidades.

  AVG(expresión escalar): obtiene el promedio del resultado de la expresión escalar en cada grupo, descartando los resultados nulos de la expresión.

  AVG (DISTINCT expresión escalar): obtiene el promedio de los resultados DISTINTOS para la expresión escalar en cada grupo, descartando los resultados nulos de la expresión.

- MAX:

MAX(expresión escalar): obtiene el máximo resultado de la expresión escalar en cada grupo. No considerando los resultados nulos de la expresión.

- MIN:

  MIN(expresión escalar): obtiene el mínimo resultado de la expresión escalar en cada grupo. No considerando los resultados nulos de la expresión.

Reunión exterior (OUTER JOIN):

Corresponde a una ampliación del concepto de join (inner join). Agrega al resultado de JOIN, las tuplas sin correspondencia.

Hay tres tipos:

- **LEFT OUTER JOIN (LEFT JOIN):**

  Sintaxis: ***relación LEFT OUTER JOIN relación***

  Produce una nueva relación con las tuplas obtenidas a través de una operación JOIN entre las relaciones, más las tuplas de la relación de la izquierda que no tienen correspondencia en la relación de la derecha, completadas con valores nulos para los atributos correspondientes a esta relación.

  Ejemplo: dadas las siguientes relaciones:

EMP

Nro_emp	**Nom_emp**	**Nro_depto**
1	Empleado 1	10
2	Empleado 2	20
3	Empleado 3	10
4	Empleado 4	30
5	Empleado 5	20

DEPTOS

Nro_depto	**Nom_depto**
10	Departamento 10
20	Departamento 20
30	Departamento 30
40	Departamento 40
50	Departamento 50

La operación “deptos LEFT JOIN emp”, produce como resultado la siguiente relación:

Nro_depto	Nom_depto	Nro_emp	Nom_emp
10	Departamento 10	1	Empleado 1
10	Departamento 10	3	Empleado 3
20	Departamento 20	2	Empleado 2
20	Departamento 20	5	Empleado 5
30	Departamento 30	4	Empleado 4
40	Departamento 40	Null	Null
50	Departamento 50	Null	Null

- **RIGHT OUTER JOIN (RIGHT JOIN):**

  Sintaxis: ***relación RIGHT OUTER JOIN relación***

  Produce una nueva relación con las tuplas obtenidas a través de una operación JOIN entre las relaciones, más las tuplas de la relación de la derecha que no tienen correspondencia en la relación de la izquierda, completadas con valores nulos para los atributos correspondientes a esta relación.

  Por lo tanto, la operación “deptos RIGHT JOIN emp”, produce como resultado la siguiente relación:

Nro_depto	Nom_depto	Nro_emp	Nom_emp
10	Departamento 10	1	Empleado 1
10	Departamento 10	3	Empleado 3
20	Departamento 20	2	Empleado 2
20	Departamento 20	5	Empleado 5
30	Departamento 30	4	Empleado 4

- **FULL OUTER JOIN (FULL JOIN):**

  Sintaxis: ***relación FULL OUTER JOIN relación***

  Produce una nueva relación con las tuplas obtenidas a través de una operación JOIN entre las relaciones, más las tuplas de la relación de la izquierda que no tienen correspondencia en la relación de la derecha, completadas con valores nulos para los atributos correspondientes a esta relación, más las tuplas de la relación de la derecha que no tienen

correpondencia en la relación de la izquierda, completadas con valores nulos para los atributos correspondientes a esta relación.

Como se ve, las operaciones OUTER JOIN, son equivalentes a una combinación de operaciones básicas del AR.

CÁLCULO RELACIONAL:

Es una alternativa al álgebra relacional. Mientras en el álgebra, la relación resultado se obtiene a partir de una expresión procedural de alto nivel que utiliza los operadores mencionados, en el cálculo relacional se obtiene a partir de una expresión no procedural que "describe" dicha relación resultado a partir de las relaciones originales.

En este material abordaremos el tema desde el punto de vista de su aplicación práctica y su aporte a la implementación de SQL. Por esa razón solo trataremos el cálculo relacional orientado a tuplas.

Asumimos que la declaración de las variables de tupla fueron definidas previamente, que están basadas en las relaciones originales y no como resultado de expresiones y que su nombre se construye con la primera letra del nombre de la relación a la que se agregará una o más letras significativas para evitar la ambigüedad en el caso que la hubiera y números correlativos para representar diferentes variables basadas en las mismas relaciones.

La sintaxis que utilizaremos es:

Lista_de_expresiones_escalares_resultado [WHERE fórmula_bien_formada]

Donde, una expresión_escalar_resultado puede ser:

- variable de tupla, con lo cual el resultado tiene un encabezamiento igual al de la relación sobre la que se basa la variable de tupla
- variable.atributo
- expresión escalar basada en atributos

y fórmula_bien_formada es cualquier expresión lógica (expresada como una condición que utiliza los atributos de las relaciones sobre las que se basan las variables utilizadas). A las expresiones lógicas regulares se le agregan las siguientes:

- EXISTS variable (condición): cuyo significado es ¿existe alguna tupla en la relación sobre la que se define la "variable" que cumpla con la "condición"?

- FORALL variable (condición): cuyo significado es ¿todas las tuplas de la relación sobre la que se define la "variable" cumplen con la "condición"?

NOTA: las variables de tupla especificadas en estas dos expresiones se dice que son limitadas ya que solo pueden ser utilizadas (tienen validez) dentro de su contexto. "Su contexto" es el que está limitado por los paréntesis entre los que está cerrada la "condición" planteada.

Ejemplos y comparación con el álgebra relación:

Dadas las siguientes relaciones:

EMP

Nro_emp	**Nom_emp**	**Nro_depto**
1	Empleado 1	10
2	Empleado 2	20
3	Empleado 3	10
4	Empleado 4	30
5	Empleado 5	20

DEPTOS

Nro_depto	**Nom_depto**
10	Departamento 10
20	Departamento 20
30	Departamento 30
40	Departamento 40
50	Departamento 50

Suponemos las siguientes variables de tupla:

e: basada en la relación emp
d: basada en la relación deptos

Consulta	**Expresión AR**	**Expresión CR**
Todos los datos de los empleados	emp	e
Nro. y nombre de todos los empleados	emp [nro_emp, nom_emp]	e.nro_emp, e.nom_emp
Combinación de todos los empleados con todos los departamentos	emp X deptos	e, d
Nro., nombre y nombre del departamento de	emp JOIN deptos [nro_emp, nom_emp,	e.nro_emp, e.nom_emp, d.nom_depto WHERE

cada empleado	nom_depto]	e.nro_depto = d.nro_depto
Departamentos que tienen empleados	emp JOIN deptos [nom_depto]	d.nom_depto WHERE EXISTS e (e.nro_depto = d.nro_depto)

Como aclaración final es necesario destacar que no trabajaremos en el cálculo relacional las alternativas a la operación resumen del AR ya que su implementación en SQL está basada en la utilización de AR.

Si estos conceptos y su aplicación han sido claramente interpretados y asimilados, estamos en condiciones de dar nuestro próximo paso y aplicarlos a la programación con SQL, paso que daremos en el módulo 3 correspondiente a la segunda parte del programa de la materia.