**ST. XAVIER’S COLLEGE**

**Maitighar, Kathmandu**



DATABASE MANAGEMENT SYSTEM

theory Assignment #6

**Submitted by:**

Dibash Poudel

013BSCCSIT017

**Submitted to:**

|  |  |
| --- | --- |
| Er. Sanjay Kr. Yadav  Lecturer, St. Xavier’s college  Department of Computer Science |  |

**Date of Submission:**

September10, 2015

**Join**

**Join** is a combination of a Cartesian product followed by a selection process. A Join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.

We will briefly describe various join types in the following sections.

**Theta (θ) Join**

Theta join combines tuples from different relations provided they satisfy the theta condition. The join condition is denoted by the symbol **θ**.

**Notation**

R1 ⋈θ R2

## Equijoin

When Theta join uses only **equality** comparison operator, it is said to be equijoin. The above example corresponds to equijoin.

Natural join acts on those matching attributes where the values of attributes in both the relations are same.

**Explanation of various Joins with Example**

Assume 2 tables A and B has values with the 1st column of each table as the common column, in the following:

Table A:

|  |  |
| --- | --- |
| 2 |  |
| 3 |  |

Table B:

|  |  |  |
| --- | --- | --- |
| 1 | A | I |
| 2 | B | II |
| 4 | D | IV |

Concrete Example:

Table **Course**:

|  |  |
| --- | --- |
| **NAME** | **COURSE NUMBER** |
| John | ITEC 122 |
| Cindy | ITEC 120 |

Table **Student**:

|  |  |  |
| --- | --- | --- |
| **NAME** | **Phone Number** | **GPA** |
| John | 6391111 | 2.0 |
| David | 8311111 | 3.0 |
| Cindy | 7311111 | 4.0 |

1. **Cross Join**: append at the end of every row of Table A by every row of Table B.

SQL command: select \* from A cross join B;

Result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 |  | 1 | A | I |
| 2 |  | 2 | B | II |
| 2 |  | 4 | D | IV |
| 3 |  | 1 | A | I |
| 3 |  | 2 | B | II |
| 3 |  | 4 | D | IV |

1. **Natural Join**:

A NATURAL JOIN is a JOIN operation that creates an implicit join clause for you based on the common columns in the two tables being joined. Common columns are columns that have the same name in both tables.

A NATURAL JOIN can be an INNER join, a LEFT OUTER join, or a RIGHT OUTER join. The default is INNER join.

If the SELECT statement in which the NATURAL JOIN operation appears has an asterisk (\*) in the select list, the asterisk will be expanded to the following list of columns (in this order):

* All the common columns
* Every column in the first (left) table that is not a common column
* Every column in the second (right) table that is not a common column

An asterisk qualified by a table name (for example, COUNTRIES.\*) will be expanded to every column of that table that is not a common column.

If a common column is referenced without being qualified by a table name, the column reference points to the column in the first (left) table if the join is an INNER JOIN or a LEFT OUTER JOIN. If it is a RIGHT OUTER JOIN, unqualified references to a common column point to the column in the second (right) table.

after Cross Join, pick up only rows with the same (or matching) common column values. Show the common column value only once. Must have common names between 2 tables.

SQL command: select \* from A natural join B;

Result:

|  |  |  |  |
| --- | --- | --- | --- |
| 2 |  | B | II |

Concrete Example: SQL: select \* from Course natural join Student;

Result:

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **COURSE NUMBER** | **Phone Number** | **GPA** |
| John | ITEC 122 | 6391111 | 2.0 |
| Cindy | ITEC 120 | 7311111 | 4.0 |

1. **Inner Join**: after Cross Join, pick up only rows with the same (or matching) common column values. However, show the common matched column values for both tables. You must specify the common column(s).

SQL command: select \* from A inner join B

On A.commoncolumn=B.commoncolumn;

Result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 |  | 2 | B | II |

Concrete Example: SQL: select \* from Course inner join Student

On Course.Name= Student. Name;

Result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NAME** | **COURSE NUMBER** | **NAME** | **Phone Number** | **GPA** |
| John | ITEC 122 | John | 6391111 | 2.0 |
| Cindy | ITEC 120 | Cindy | 7311111 | 4.0 |

1. **Outer Join**:

(1) **Left Outer Join**: after Inner Join, add rows of the left Table (A) which is non-matching common column values from row of Cross Join with all fields of Table B blanked. You must specify the common column(s).

SQL command: select \* from A left outer join B

On A.commoncolumn=B.commoncolumn;

Result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 |  | 2 | B | II |
| 3 |  |  |  |  |

Concrete Example: If you want to find out who are new in the school (who don’t have any GPA yet)

SQL: select \* from Course left outer join Student

On Course.Name= Student. Name;

Result:

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **COURSE NUMBER** | **Phone Number** | **GPA** |
| John | ITEC 122 | 6391111 | 2.0 |
| Cindy | ITEC 120 | 7311111 | 4.0 |

(2) **Right Outer Join**: after Inner Join, add rows of the right Table (B) which is non-matching common column values from row of Cross Join with all fields of Table A blanked. You must specify the common column(s).

SQL command: select \* from A right outer join B

On A.commoncolumn=B.commoncolumn;

Result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 |  | 2 | B | II |
|  |  | 1 | A | I |
|  |  | 4 | D | IV |

Concrete Example: If you want to find out students who are not taking any class

SQL: select \* from Course right outer join Student

On Course.Name= Student. Name;

Result:

|  |  |  |  |
| --- | --- | --- | --- |
| **NAME** | **COURSE NUMBER** | **Phone Number** | **GPA** |
| John | ITEC 122 | 6391111 | 2.0 |
| David |  | 8311111 | 3.0 |
| Cindy | ITEC 120 | 7311111 | 4.0 |

(3) **Full Outer Join**: after Inner Join, add rows of the left Table (A) and the right Table (B) which are non-matching common column values from row of Cross Join with all fields of the other Table blanked. You must specify the common column(s).

SQL command: select \* from A full join B

On A.commoncolumn=B.commoncolumn;

Result:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 |  | 2 | B | II |
| 3 |  |  |  |  |
|  |  | 1 | A | I |
|  |  | 4 | D | IV |

[2]

**Rename Operation (ρ)**

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho** *ρ*.

**Notation** − *ρ* x (E)

Where the result of expression **E** is saved with name of **x**.

Additional operations are −

* Set intersection
* Assignment
* Natural join[3]

**Example**

Transcript ( StudId, CrsCode, Semester, Grade)

Teaching (ProfId, CrsCode, Semester)

*πStudId, CrsCode (Transcript) [StudId, CrsCode1] × πProfId, CrsCode (Teaching) [ProfId, CrsCode2]*

*[4]*

**Assignment Operator**

The assignment operator is one of the most intuitive to use. It assigns a value to a variable. The only confusion in using this operator could stem from its overloading. All  relational database management system overload this operator with an additional function — comparison — in the SQL.

The equals operator (=) is used as an assignment in the following SQL query that updates the price (PROD\_PRICE\_N) column in the PRODUCT table, raising the existing prices by 2 percent:

UPDATE product SET prod\_price\_n

= prod\_price\_n \* 1.02 (10 row(s) affected)

And the same operator would be used for comparing values when used, for example, in theWHERE clause of an SQL statement:

UPDATE product SET prod\_price\_n

= prod\_price\_n \* 1.02 WHERE prod\_id\_n = 1880 (1 row(s)

affected)

This statement assigns a 2 percent increase to a product whose ID is 1880; in the same query, the equals operator (=) is used in its assignment and comparison capacity at the same time. [5]

**The Division Operation**

Division, denoted http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1375.gif, is suited to queries that include the phrase ``for all''.

Suppose we want to find all the customers who have an account at **all** branches located in Brooklyn.

Strategy: think of it as three steps.

We can obtain the names of all branches located in Brooklyn by

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_displaymath1319.gif

Figure 3.19 in the textbook shows the result.

We can also find all cname, bname pairs for which the customer has an account by

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_displaymath1320.gif

Figure 3.20 in the textbook shows the result.

Now we need to find all customers who appear in http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1261.gif with **every** branch name in http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1251.gif.

The divide operation provides exactly those customers:

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_eqnarray492.gif

which is simply http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1381.gif.

Formally,

* + Let http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1383.gif and http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1385.gif be relations.
  + Let http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1387.gif.
  + The relation http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1409.gif is a relation on scheme http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1405.gif.
  + A tuple http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1447.gif is in http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1409.gif if for every tuple http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1397.gif in http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1531.gif there is a tuple http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1401.gif in http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1345.gif satisfying both of the following:

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_eqnarray502.gif

* + These conditions say that the http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1405.gif portion of a tuple http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1447.gif is in http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1409.gif if and only if there are tuples with the http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1291.gif portion **and** the http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1335.gif portion in http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1345.gif for **every** value of the http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1335.gif portion in relation http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1335.gif.

We will look at this explanation in class more closely.

The division operation can be defined in terms of the fundamental operations.

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_displaymath1321.gif [6]

**Division - Example**

List the Ids of students who have passed all courses that were taught in spring 2000

Numerator:

* StudId and CrsCode for every course passed by every student:
* π   StudId, CrsCode(  σGrade< > ‘F’ (Transcript) )

Denominator:

* CrsCode of all courses taught in spring 2000
* π CrsCode ( σ Semester=‘S2000’(Teaching) )

Result is numerator/denominator[4]

## 

## Additional Operations

Additional operations are defined in terms of the fundamental operations. They do not add power to the algebra, but are useful to simplify common queries.

**The Set Intersection Operation**

Set intersection is denoted by http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1329.gif, and returns a relation that contains tuples that are in **both** of its argument relations.

It does not add any power as

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_displaymath1315.gif

To find all customers having both a loan and an account at the SFU branch, we write

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_eqnarray430.gif

**The Natural Join Operation**

Often we want to simplify queries on a cartesian product.

For example, to find all customers having a loan at the bank and the cities in which they live, we need borrow and customer relations:

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_eqnarray439.gif

Our selection predicate obtains only those tuples pertaining to only one cname.

This type of operation is very common, so we have the **natural join**, denoted by a http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1365.gif sign. Natural join combines a cartesian product and a selection into one operation. It performs a selection forcing equality on those attributes that appear in both relation schemes. Duplicates are removed as in all relation operations.

To illustrate, we can rewrite the previous query as

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_displaymath1316.gif

The resulting relation is shown in Figure [3.7](http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/node10.html#fig317join).

 http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_figure447.gif   
**Figure 3.7:**   Joining borrow and customer relations.

We can now make a more formal definition of natural join.

* + Consider http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1181.gif and http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1335.gif to be **sets** of attributes.
  + We denote attributes appearing in **both** relations by http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1337.gif.
  + We denote attributes in **either or both** relations by http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1339.gif.
  + Consider two relations http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1383.gif and http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1385.gif.
  + The natural join of http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1345.gif and http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1531.gif, denoted by http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1349.gif is a relation on scheme http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1339.gif.
  + It is a projection onto http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1339.gif of a selection on http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1355.gif where the predicate requires http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1357.gif for each attribute http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1681.gif in http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1337.gif.

Formally,

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_displaymath1317.gif

wherehttp://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1363.gif.

To find the assets and names of all branches which have depositors living in Stamford, we need customer, deposit and branch relations:

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_eqnarray475.gif

Note that http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1365.gif is associative.

To find all customers who have both an account and a loan at the SFU branch:

http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_displaymath1318.gif

This is equivalent to the set intersection version we wrote earlier. We see now that there can be several ways to write a query in the relational algebra.

If two relations http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1383.gif and http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1385.gif have no attributes in common, then http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1371.gif, and http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/_7092_tex2html_wrap1373.gif.[6]

**Reference**

[1] <http://www.tutorialspoint.com/dbms/database_joins.htm>

[2] <https://docs.oracle.com/javadb/10.8.3.0/ref/rrefsqljnaturaljoin.html>

[3] <http://www.tutorialspoint.com/dbms/relational_algebra.htm>

[4] <http://jcsites.juniata.edu/faculty/rhodes/dbms/relalg.htm>

[5]<http://etutorials.org/SQL/SQL+Bible+Oracle/Part+IV+Retrieving+and+Transforming+Data/Chapter+11+SQL+Operators/Assignment+Operator/>

[6]<http://www.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/Chapter3/node10.html>