Data Base Management System

Join

JOIN

- Generalized Projection
- Aggregate Functions(g)
- Join
  - Inner Join
  - Theta Join
  - Equi Join
  - Natural Join
- Outer Join
  - Left Outer Join
  - Right Outer Join
  - Full Outer Join
  - Natural Join
- Self Join

Generalized Projection

- The generalized-projection operation extends the projection operation by allowing arithmetic functions to be used in the projection list. The general form of generalized-projection is:

$$\pi_{F_1, F_2 \ldots F_n}(\mathrm{E})$$

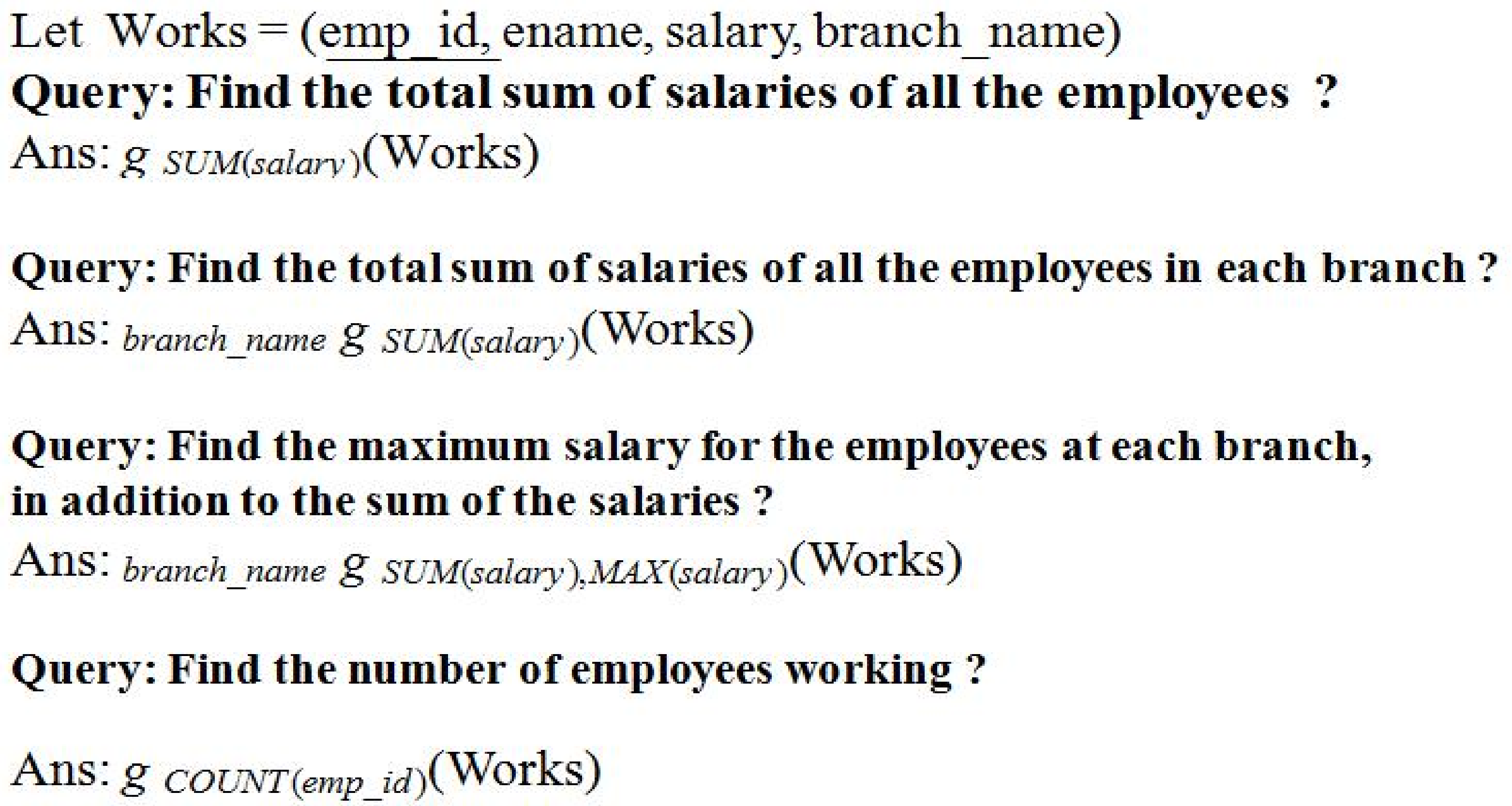
- Ex: Emp=(ssn, salary, deduction, years_service) be a relation. A report may be required to show net_salary=salary-deduction, bonus=2000*years_service and tax=0.25*salary

$$\mathrm{REPORT} \leftarrow \rho_{(ssn, net\_salary, bonus, tax)}(\pi_{ssn, salary - deduction, 2000 * years\_service, 0.25 * salary}(\mathrm{Emp}))$$

Aggregate Functions (g)

- Aggregate functions take a collection of values and return a single value as a result. NULL value will not participate in the aggregate functions. The general form of aggregate function is:

$$_{\textit{grouping\_attribute}}\ g\ _{\textit{aggregate\_functions}}\ (R)$$



Let Works = ($\underline{\text{emp\_id}}$, ename, salary, branch_name)

Query: Find the total sum of salaries of all the employees ?

Ans: $g_{SUM(salary)}(\text{Works})$

Query: Find the total sum of salaries of all the employees in each branch ?

Ans: $_{branch\_name}\ g_{SUM(salary)}(\text{Works})$

Query: Find the maximum salary for the employees at each branch, in addition to the sum of the salaries ?

Ans: $_{branch\_name}\ g_{SUM(salary),MAX(salary)}(\text{Works})$

Query: Find the number of employees working ?

Ans: $g_{COUNT(emp\_id)}(\text{Works})$

Join

- ➢ The join operation is used to connect data across relations. **Tables are joined on columns that have the same data type and data width in the tables.**
- ➢ Join operation joins two relations by merging those tuples from two relations that satisfy a given condition. The condition is defined on attributes belonging to relations to be joined.
- ➢ Different categories of join are:
    - ✓ **Inner Join**
    - ✓ **Outer Join**
    - ✓ **Self Join**
- ➢ **Inner Join:** In the inner join, tuples with NULL valued join attributes do not appear in the result. Tuples with NULL values in the join attributes are also eliminated. The different types of inner join are:
    - ✓ **Theta Join**
    - ✓ **Equi Join**
    - ✓ **Natural Join**

Theta Join(⋈θ)

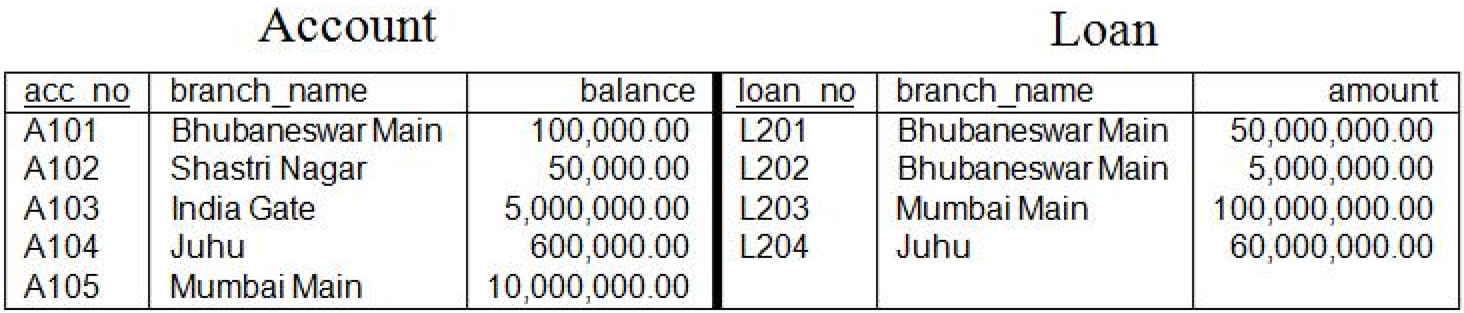
- The theta join is a **join with a specified condition** involving a column from each relation. This condition specifies that the two columns should be compared in some way.
- The comparison operator can be any of the six: <, ≤, >, ≥, = and ^ =
- Theta join is denoted by (⋈ θ) symbol. The general form of theta join is:

$$R \bowtie \theta \;\; S = \pi \text{ all } \; (\sigma \; \theta \;\; (R \times S))$$

✓ Degree (Result) = Degree (R) + Degree (S)

✓ Cardinality (Result) ≤ Cardinality(R) × Cardinality(S)

Theta Join(⋈θ)…



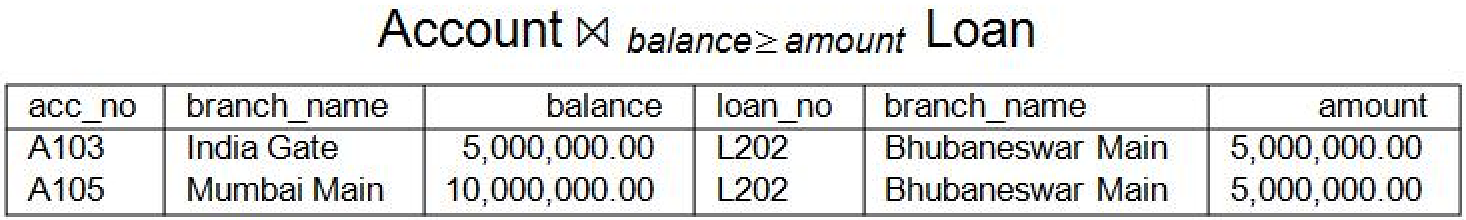
Account

acc_no	branch_name	balance
A101	Bhubaneswar Main	100,000.00
A102	Shastri Nagar	50,000.00
A103	India Gate	5,000,000.00
A104	Juhu	600,000.00
A105	Mumbai Main	10,000,000.00

Loan

loan_no	branch_name	amount
L201	Bhubaneswar Main	50,000,000.00
L202	Bhubaneswar Main	5,000,000.00
L203	Mumbai Main	100,000,000.00
L204	Juhu	60,000,000.00

Q: Find the account details as well as loan details for the situations where depositing balance is greater than or equal to the borrowing amount ?

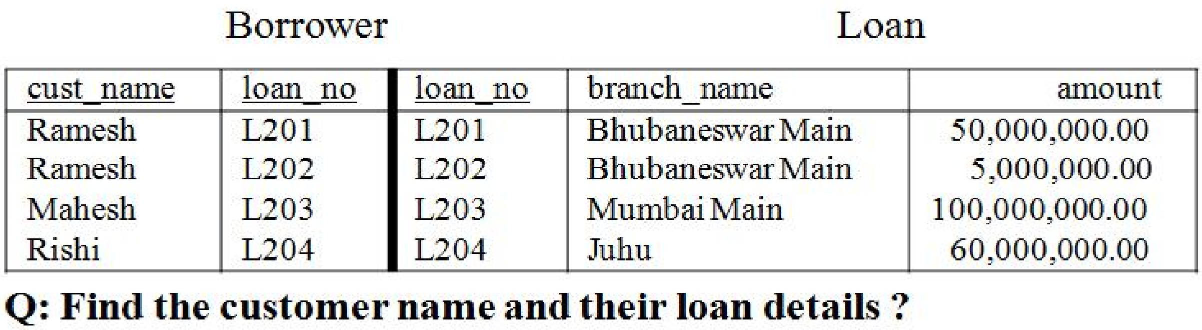


Account $\bowtie_{balance \geq amount}$ Loan

acc_no	branch_name	balance	loan_no	branch_name	amount
A103	India Gate	5,000,000.00	L202	Bhubaneswar Main	5,000,000.00
A105	Mumbai Main	10,000,000.00	L202	Bhubaneswar Main	5,000,000.00

Equi Join($\bowtie_{=}$)

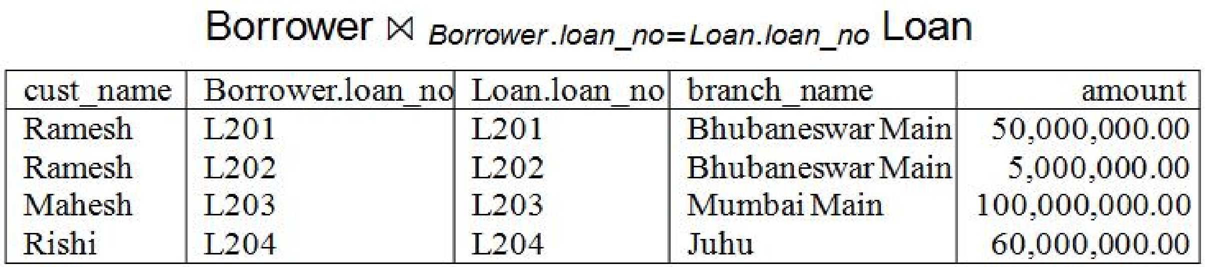
- The equi join is the theta **join based on equality of specified columns.** That means the equi join is the special type of theta join where the comparison operator is =.
- The general form of theta join is: $R \bowtie_{=} S = \pi_{all} (\sigma_{=} (R \times S))$
  - ✓Degree (Result) = Degree (R) + Degree (S)
  - ✓Cardinality (Result) ≤ Cardinality(R) × Cardinality(S)



Borrower | Loan

cust_name	loan_no	loan_no	branch_name	amount
Ramesh	L201	L201	Bhubaneswar Main	50,000,000.00
Ramesh	L202	L202	Bhubaneswar Main	5,000,000.00
Mahesh	L203	L203	Mumbai Main	100,000,000.00
Rishi	L204	L204	Juhu	60,000,000.00

Q: Find the customer name and their loan details ?



$$\text{Borrower} \bowtie_{\textit{Borrower.loan\_no=Loan.loan\_no}} \text{Loan}$$

cust_name	Borrower.loan_no	Loan.loan_no	branch_name	amount
Ramesh	L201	L201	Bhubaneswar Main	50,000,000.00
Ramesh	L202	L202	Bhubaneswar Main	5,000,000.00
Mahesh	L203	L203	Mumbai Main	100,000,000.00
Rishi	L204	L204	Juhu	60,000,000.00

Natural Join (⋈)

- To perform natural join on two relations, **they should contain at least one common attributes.** It is just like the equi join with the elimination of the common attributes. The natural join is denoted by (⋈) symbol.
- The general form of theta join is:

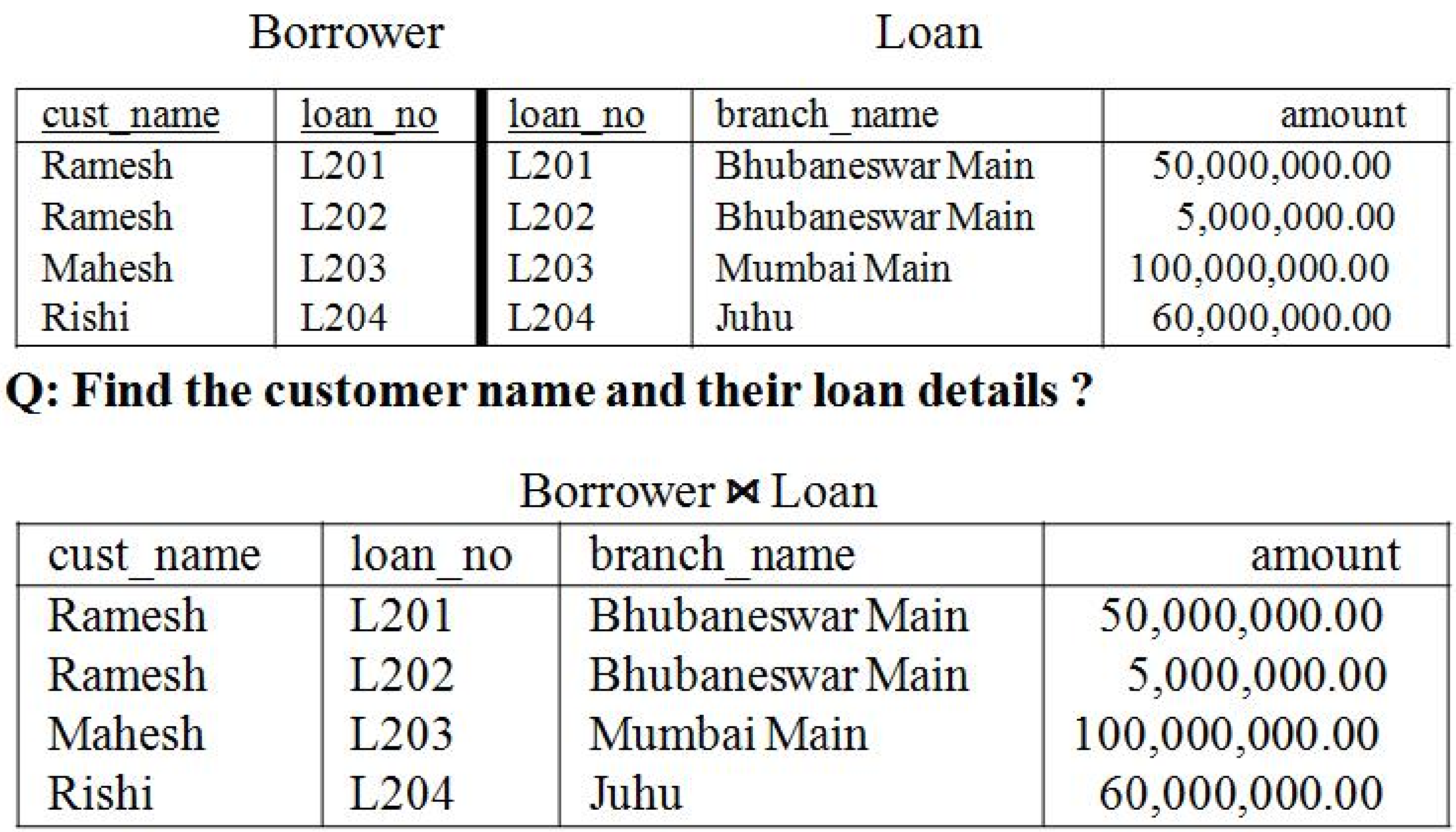
$$R \bowtie S = \Pi_{\text{all}-\text{common\_attributes}} \ (\sigma = (R \times S))$$

✓Degree (Result) = Degree (R) + Degree (S) - Degree (R ∩S)
✓Cardinality (Result) ≤ Cardinality(R) × Cardinality(S)

- The general form of the natural join can also be represented as:

$$R \bowtie S = \Pi_{\text{all}} \ (R \bowtie S)$$

Natural Join (⋈)



Borrower

cust_name	loan_no
Ramesh	L201
Ramesh	L202
Mahesh	L203
Rishi	L204

Loan

loan_no	branch_name	amount
L201	Bhubaneswar Main	50,000,000.00
L202	Bhubaneswar Main	5,000,000.00
L203	Mumbai Main	100,000,000.00
L204	Juhu	60,000,000.00

Q: Find the customer name and their loan details ?

Borrower ⋈ Loan

cust_name	loan_no	branch_name	amount
Ramesh	L201	Bhubaneswar Main	50,000,000.00
Ramesh	L202	Bhubaneswar Main	5,000,000.00
Mahesh	L203	Mumbai Main	100,000,000.00
Rishi	L204	Juhu	60,000,000.00

Outer Join

- It is an extension of the natural join operation to **deal with the missing information.** The outer join consists of two steps:
  - ✓ *First, a natural join is executed*
  - ✓ *Then if any record in one relation does not match a record from the other relation in the natural join, that unmatched record is added to the join relation, and the additional columns are filled with NULLs*
- The different types of outer join are:
  - ✓ **Left Outer Join**
  - ✓ **Right Outer Join**
  - ✓ **Full Outer Join**

Left Outer Join (⟕)

- ✓ Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- ✓ In the left outer join, tuples in R have no matching tuples in S.
- ✓ It is denoted by ⟕.
- ✓ Example: Using the above EMPLOYEE table and FACT_WORKERS table

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	**NULL**	**NULL**

Left Outer Join

Customer

cust_name	cust_street	cust_city
Rishi	India Gate	New Delhi
Sarthak	M. G. Road	Bangalore
Manas	Shastri Nagar	Bhubaneswar
Ramesh	M. G. Road	Bhubaneswar
Mahesh	Juhu	Mumbai

Borrower

cust_name	loan_no
Ramesh	L201
Ramesh	L202
Mahesh	L203
Rishi	L204

Q: Find out the customer details who have taken loans as well as who have not taken loans ?

Customer ⟕Borrower

cust_name	cust_street	cust_city	loan_no
Rishi	India Gate	New Delhi	L204
Ramesh	M. G. Road	Bhubaneswar	L201
Ramesh	M. G. Road	Bhubaneswar	L202
Mahesh	Juhu	Mumbai	L203
Sarthak	M. G. Road	Bangalore	NULL
Manas	Shastri Nagar	Bhubaneswar	NULL

Right Outer Join (⟖)

- Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names
- In right outer join, tuples in S have no matching tuples in R.
- It is denoted by ⟖.

Example: Using the above EMPLOYEE table and FACT_WORKERS Relation

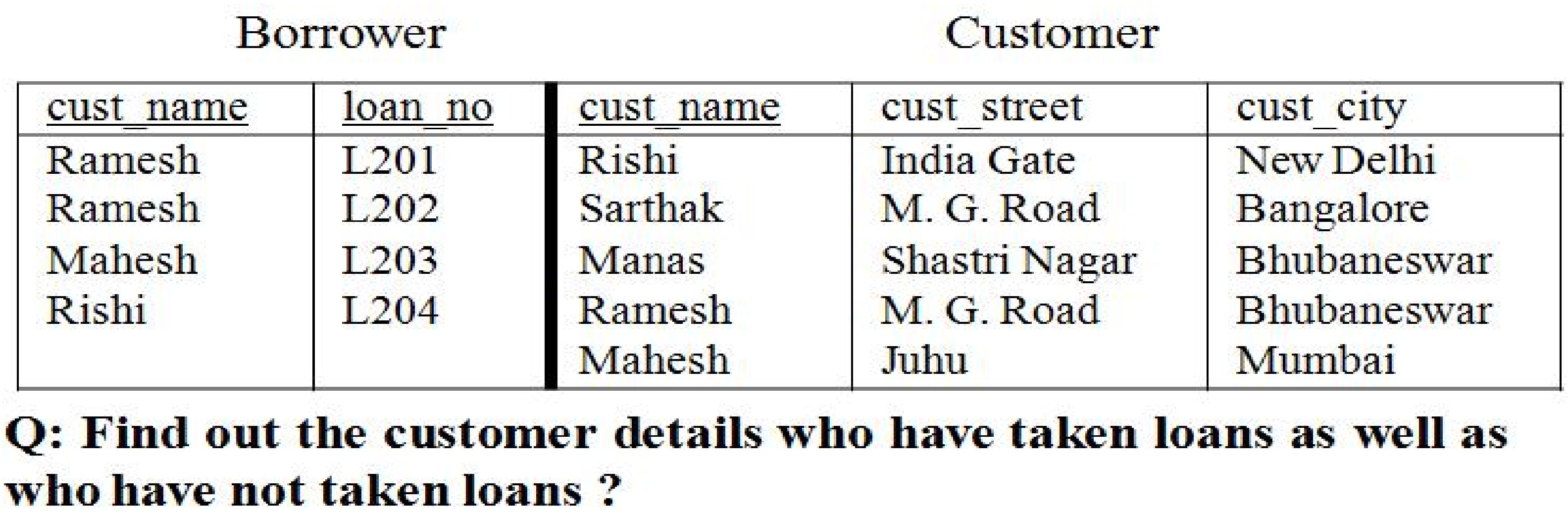
EMPLOYEE ⟖ FACT_WORKERS

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	**NULL**	**NULL**

Right Outer Join (⟖)



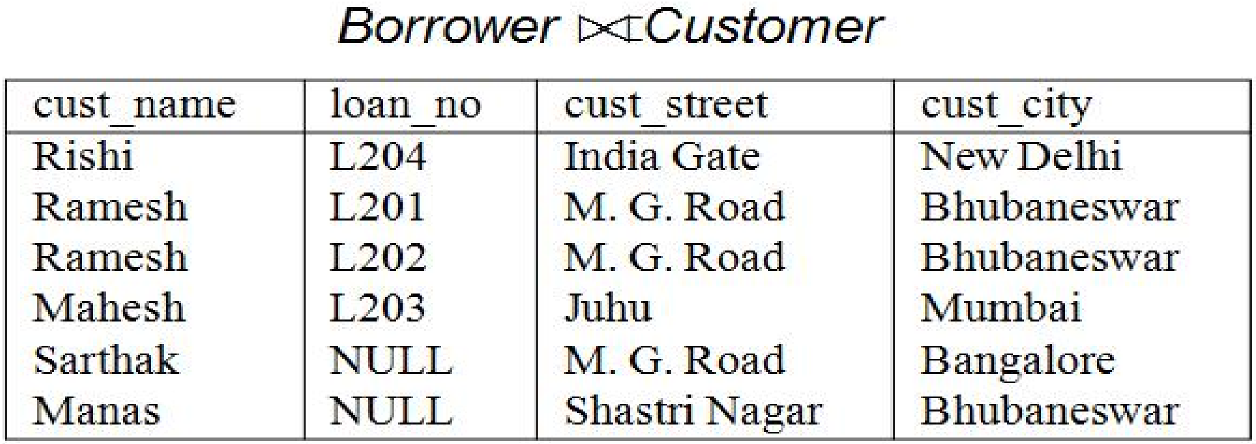
Borrower

cust_name	loan_no
Ramesh	L201
Ramesh	L202
Mahesh	L203
Rishi	L204

Customer

cust_name	cust_street	cust_city
Rishi	India Gate	New Delhi
Sarthak	M. G. Road	Bangalore
Manas	Shastri Nagar	Bhubaneswar
Ramesh	M. G. Road	Bhubaneswar
Mahesh	Juhu	Mumbai

Q: Find out the customer details who have taken loans as well as who have not taken loans ?



Borrower ⟖Customer

cust_name	loan_no	cust_street	cust_city
Rishi	L204	India Gate	New Delhi
Ramesh	L201	M. G. Road	Bhubaneswar
Ramesh	L202	M. G. Road	Bhubaneswar
Mahesh	L203	Juhu	Mumbai
Sarthak	NULL	M. G. Road	Bangalore
Manas	NULL	Shastri Nagar	Bhubaneswar

Full Outer Join (⟗)

- Full outer join is like a left or right join except that it contains all rows from both tables.
- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.
- Example: Using the above EMPLOYEE table and FACT_WORKERS table
- Input: **EMPLOYEE ⟗ FACT_WORKERS**

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

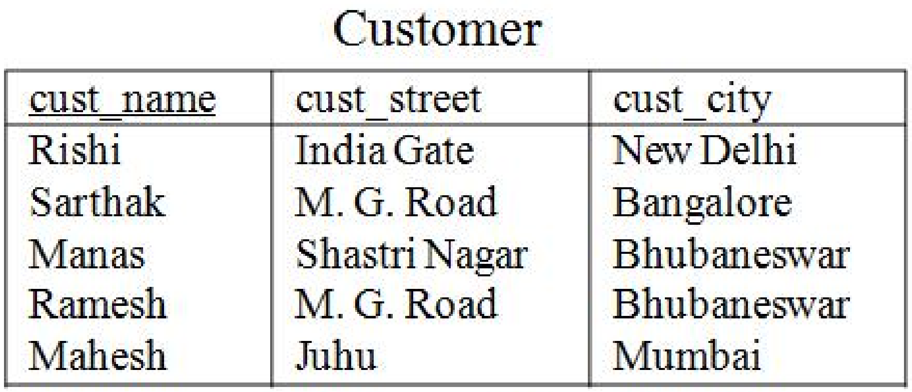
Self Join

- The self join is similar to the theta join. It joins a relation to itself by a condition. The self join can be viewed as a join of two copies of the same relation.
- The general form of self join is:

$$R \bowtie_{\theta} R = \pi_{all} \, (\sigma_{\theta} \, (R \times R))$$

- Thus, the self join creates two alias or copies of the same relation; then performs the theta join by a condition based on the attributes of these two copies.

Self Join

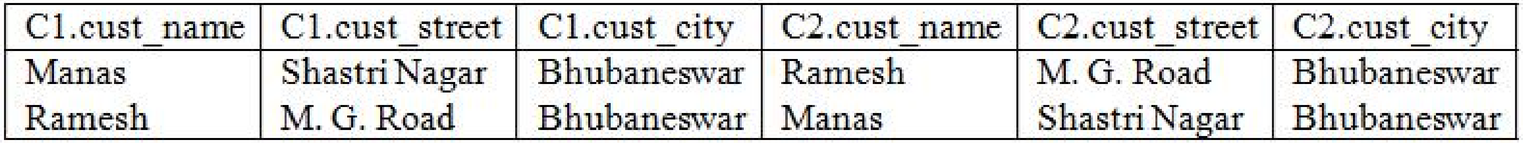


Customer

cust_name	cust_street	cust_city
Rishi	India Gate	New Delhi
Sarthak	M. G. Road	Bangalore
Manas	Shastri Nagar	Bhubaneswar
Ramesh	M. G. Road	Bhubaneswar
Mahesh	Juhu	Mumbai

Q: Find out the customer details as well as the others' staying in the same cust_city ?

$$C1 \bowtie_{C1.cust\_city = C2.cust\_city} C2$$



C1.cust_name	C1.cust_street	C1.cust_city	C2.cust_name	C2.cust_street	C2.cust_city
Manas	Shastri Nagar	Bhubaneswar	Ramesh	M. G. Road	Bhubaneswar
Ramesh	M. G. Road	Bhubaneswar	Manas	Shastri Nagar	Bhubaneswar

THANK YOU!