

Relations - Querying



pm jat @ daiict



“Data Manipulation”

Operations on Relations (/Tables)

eno [PK] int	ename character var	dob date	gender character	salary numeric	super integer	dno smallint
105	Jayesh	1967-11-10	M	55000	[null]	1
102	Kirit	1985-12-08	M	40000	105	5
106	Vijay	1971-06-20	M	53000	105	4
101	Sanjay	1955-11-09	M	70000	102	[null]
108	Priya	1978-07-19	F	45000	106	4
104	Ramesh	1972-09-15	M	38000	102	5
103	Kalpesh	1982-07-31	F	25000	102	5
107	Ahmad	1979-03-29	M	25000	106	4
111	Sanna	1982-07-23	F	35000	102	5

- **INSERT** – add tuple
- **UPDATE** – update tuple
- **DELETE** – delete tuple
- **SELECT** – **Query** Relations

- First three modifies relation; where as,
- Last one just reads relations



What is a Query

- Suppose you want to know “list of students having cpi > 7.5”
- You submit a query to DBMS; DBMS executes that query and returns you the result of query execution
- That result is “answer of the query”
- Relational Model originally suggested two languages for expressing the queries
 - Relational Algebra
 - Relational Calculus
- Later SQL was developed at IBM for querying (and other purposes too: Data Definitions and Updates)



What is a Query

- **Relational Algebra**

$$\sigma_{DNO=4}(EMPLOYEE)$$

- **Relational Calculus**

$$\{t \mid STUDENT(t) \text{ AND } t.cpi > 7.5\}$$

- **SQL**

`SELECT * FROM STUDENTS WHERE CPI > 7.5;`

- Note that first two are pure mathematical expressions.
- Being its ability to express queries mathematically has been foremost strength of relational model



What is a Query

- **Relational Algebra**

$$\sigma_{DNO=4}(EMPLOYEE)$$

- **Relational Calculus**

$$\{t \mid STUDENT(t) \text{ AND } t.cpi > 7.5\}$$

- **SQL**

`SELECT * FROM STUDENTS WHERE CPI > 7.5;`

- Note that first two are pure mathematical expressions.
- Being its ability to express queries mathematically has been foremost strength of relational model



Querying in this course

- Relational Algebra and SQL side by side
- Write SQL queries through Algebraic route
 - Find Algebraic solution and then
 - Translate into SQL
- Note: SQL was initially influenced by tuple relation calculus, however later updates incorporated most algebraic concepts as well.



Basic Querying operations

- Here is list of first set of operation Let us first discuss what is called as “the original **operators**” –
 - SELECT (σ)
 - PROJECT (π)
 - Join (\bowtie)
 - Cross JOIN (\times)
 - Set Operations: UNION, INTERSECTION, and DIFFERENCE
 - Aggregate operations
- Note that all these operations are READ only operations. They do not change any tuple data.



SELECT operation

- It is used to get subset of tuples from a relation for specified tuple selection criteria.
- SELECT is unary Operation; represented by sigma (σ); Expressed as -

$$\sigma_{\langle \text{tuple selection criteria} \rangle}(r)$$

- For example: to select the EMPLOYEE tuples whose department number is four, we express as following-

$$\sigma_{DNO=4}(EMPLOYEE)$$

- Interpretation of above statement should be read as following

-

“produce a new relation by reading EMPLOYEE relation, and selecting only the tuple that meet the specified condition”

EMPLOYEE

eno [PK] int	ename character var	dob date	gender character	salary numeric	super integer	dno smallint
105	Jayesh	1967-11-10	M	55000	[null]	1
102	Kirit	1985-12-08	M	40000	105	5
106	Vijay	1971-06-20	M	53000	105	4
101	Sanjay	1955-11-09	M	70000	102	[null]
108	Priya	1978-07-19	F	45000	106	4
104	Ramesh	1972-09-15	M	38000	102	5
103	Kalpesh	1982-07-31	F	25000	102	5
107	Ahmad	1979-03-29	M	25000	106	4
111	Sanna	1982-07-23	F	35000	102	5

$\sigma_{DNO=4}(EMPLOYEE)$

eno [PK] int	ename character var	dob date	gender character	salary numeric	super integer	dno smallint
106	Vijay	1971-06-20	M	53000	105	4
108	Priya	1978-07-19	F	45000	106	4
107	Ahmad	1979-03-29	M	25000	106	4

EMPLOYEE

eno [PK] int	ename character var	dob date	gender character	salary numeric	super integer	dno smallint
105	Jayesh	1967-11-10	M	55000	[null]	1
102	Kirit	1985-12-08	M	40000	105	5
106	Vijay	1971-06-20	M	53000	105	4
101	Sanjay	1955-11-09	M	70000	102	[null]
108	Priya	1978-07-19	F	45000	106	4
104	Ramesh	1972-09-15	M	38000	102	5
103	Kalpesh	1982-07-31	F	25000	102	5
107	Ahmad	1979-03-29	M	25000	106	4
111	Sanna	1982-07-23	F	35000	102	5

SELECT * FROM EMPLOYEE WHERE DNO=4;

$\sigma_{DNO=4}(EMPLOYEE)$



eno [PK] int	ename character var	dob date	gender character	salary numeric	super integer	dno smallint
106	Vijay	1971-06-20	M	53000	105	4
108	Priya	1978-07-19	F	45000	106	4
107	Ahmad	1979-03-29	M	25000	106	4



SELECT operation

- Same example with little more additional condition in “tuple selection criteria”

$\sigma_{DNO=4 \text{ AND } SALARY > 30000}(EMPLOYEE)$

- This shall produce a relation from Employee relation by selecting tuples where DNO=4 and Salary > 30000
- Same is written as following in SQL:

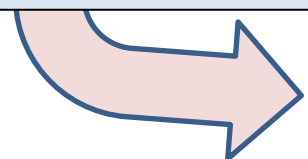
```
SELECT * FROM EMPLOYEE  
WHERE DNO=4 AND SALARY > 30000;
```

EMPLOYEE

eno [PK] int	ename character var	dob date	gender character	salary numeric	super integer	dno smallint
105	Jayesh	1967-11-10	M	55000	[null]	1
102	Kirit	1985-12-08	M	40000	105	5
106	Vijay	1971-06-20	M	53000	105	4
101	Sanjay	1955-11-09	M	70000	102	[null]
108	Priya	1978-07-19	F	45000	106	4
104	Ramesh	1972-09-15	M	38000	102	5
103	Kalpesh	1982-07-31	F	25000	102	5
107	Ahmad	1979-03-29	M	25000	106	4
111	Sanna	1982-07-23	F	35000	102	5

$\sigma_{DNO=4 \text{ AND } SALARY>30000}(EMPLOYEE)$

SELECT * FROM EMPLOYEE WHERE DNO=4 AND SALARY > 30000;



eno [PK] int	ename character var	dob date	gender character	salary numeric	super integer	dno smallint
106	Vijay	1971-06-20	M	53000	105	4
108	Priya	1978-07-19	F	45000	106	4



SELECT operation

- Note that this operation does not truncate the original relation stored in table in database.
- It only takes operand relation as input and produces the resultant relation in temporary storage
- The Resultant relation of SELECTION operation has-
 - Schema same as the operand relation
 - Degree* is same as the operand relation
 - Cardinality+ can be anything from 0 to N. Where N is cardinality of operand relation.

*Degree means – “Number of Attributes”

+Cardinality means - number of tuples



PROJECT Operation

- PROJECT operation selects certain “columns” of operand relation
- It can be viewed as getting vertical subset of a relation.
- Relational Algebra uses pi (π) operator for this operation.
- The general form of the PROJECT operation is

$$\pi_{\langle attribute-list \rangle}(r)$$

- Where <attribute list> is the desired list of attributes (columns) from the attributes of relation R.
- Example: $\pi_{ename,dno,salary}(EMPLOYEE)$



Projection Operation

EMPLOYEE

eno [PK]	ename	dob	gender	salary	supervisor	dept
105	Jayesh	1967-11-10	M	55000	[null]	1
102	Kirit	1985-12-08	M	40000	105	5
106	Vijay	1971-06-20	M	53000	105	4
101	Sanjay	1955-11-09	M	70000	102	[null]
108	Priya	1978-07-19	F	45000	106	4
104	Ramesh	1972-09-15	M	38000	102	5
103	Kalpesh	1982-07-31	F	25000	102	5
107	Ahmad	1979-03-29	M	25000	106	4
111	Sanna	1982-07-23	F	35000	102	5

$\pi_{ename,dno,salary}(EMPLOYEE)$

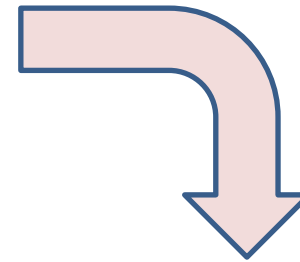
ename	dno	salary
Jayesh	1	55000
Kirit	5	40000
Vijay	4	53000
Sanjay	[null]	70000
Priya	4	45000
Ramesh	5	38000
Kalpesh	5	25000
Ahmad	4	25000
Sanna	5	35000

SELECT ename, dno, salary FROM EMPLOYEE;

EMPLOYEE

eno [PK] int	ename character var	dob date	gender character	salary numeric	super integer	dno smallint
105	Jayesh	1967-11-10	M	55000	[null]	1
102	Kirit	1985-12-08	M	40000	105	5
106	Vijay	1971-06-20	M	53000	105	4
101	Sanjay	1955-11-09	M	70000	102	[null]
108	Priya	1978-07-19	F	45000	106	4
104	Ramesh	1972-09-15	M	38000	102	5
103	Kalpesh	1982-07-31	F	25000	102	5
107	Ahmad	1979-03-29	M	25000	106	4
111	Sanna	1982-07-23	F	35000	102	5

$\pi_{dno}(EMPLOYEE)$



dno smallint
[null]
5
4
1

SELECT DISTINCT dno FROM EMPLOYEE;



- Another example:
List of Supervisors
- `SELECT distinct super_eno from employee;`



PROJECT Operation

- Result of PROJECT operation –
 - is a valid relation – no duplicates
 - has cardinality same as of the instance of EMPLOYEE (and less, if projection operation has to drop some duplicate tuples in result set)
 - has degree equal to number of attributes specified in the list



SQL Note

- If do not use DISTINCT in previous query; it produces duplicate tuples.

```
SELECT dno FROM EMPLOYEE;
```

```
SELECT DISTINCT dno  
FROM EMPLOYEE;
```

- DISTINCT is used for producing unique tuples!

dno smallint
[null]
5
4
1

dno smallint
1
5
4
[null]
4
5
5
4
5



Basic Querying operations

- Algebraic operations are immutable
- SELECT and PROJECT are unary operators.
- Result of any relational algebra expression is a “valid relation”.
Result has-
 - its own schema
 - its own instance
 - However NOT stored in database, just in working memory
 - Tuple in result relation are unique



Sequencing of Selection and Projection

- Often queries require sequencing multiple operations
- Projection followed by Selection is most basic sequencing
- For example, following query
 - List (eno, name) of employees having salary > 50000
- In algebra, it is answered as following

$$\pi_{eno,ename}(\sigma_{SALARY>50000}(EMPLOYEE))$$



Sequencing of Selection and Projection

- In algebra, it is answered as following

$$\pi_{eno,ename}(\sigma_{SALARY>50000}(EMPLOYEE))$$

- It can also be expressed as following:

$$r1 \leftarrow \sigma_{SALARY>50000}(EMPLOYEE)$$

$$\text{result} \leftarrow \pi_{eno,ename}(r1)$$

- In SQL, it is expressed as:

```
SELECT eno, ename FROM EMPLOYEE
WHERE salary > 50000;
```



Sequencing of Selection and Projection

- In algebra, it is answered as following

$$\pi_{eno,ename}(\sigma_{SALARY>50000}(EMPLOYEE))$$

- Following shall not be accepted

$$\sigma_{SALARY>50000}(\pi_{eno,ename}(EMPLOYEE))$$

Its exact translation to SQL is:

```
select * from  
(SELECT eno, ename from employee ) as r  
where salary > 50000;
```



Sequencing of Selection and Projection

- Here it is to be noted that in some cases order of sequencing of operations may not matter while in other cases it can!



SELECT statement in SQL

- SELECT statement of SQL is not only SELECT operation of relational algebra.
- It is a SQL command for executing all “Query Answering” operations.
- Following is most basic form of SELECT statement of SQL

SELECT __<attribute-list>__ FROM __<relational-expr>__
[WHERE __<tuple selection criteria>__];

- Above expression is like a combination of SELECT and PROJECT of relational algebra; where you specify “tuple selection criteria”, and also specify “attribute list”.



- Do not confuse with SQL's SELECT keyword with Selection as operations



Examples #1

- List (ID, Name) of B Tech CS Students having cpi > 7.5
 - Apply Selection on Student (prodid = 'bcs' and cpi > 7.5)
 - Apply projection on result (id, name)

$\pi_{studid, name} (\sigma_{progid='BCS' \text{ AND } CPI > 7.5}(Student))$

```
SELECT studid, name FROM Student  
WHERE progid='BCS' AND CPI > 7.5;
```



Examples #2

- List (ID, Name) of B Tech CS and B Tech IT students having cpi > 7.5

$\pi_{studid, name} (\sigma_{(progid='BCS' \text{ OR } progid='BIT') \text{ AND } CPI > 7.5} (Student))$

```
SELECT studid, name FROM Student
      WHERE (progid='BCS' OR progid='BIT')
            AND CPI > 7.5;
```

```
SELECT studid, name FROM Student
      WHERE progid IN ('BCS', 'BIT')
            AND CPI > 7.5;
```



Examples #3

- List eno, name of employees who are supervised by eno=105

$\pi_{\text{eno}, \text{ename}} (\sigma_{\text{supe_eno}=105}(\text{Employee}))$

- SQL:

```
select eno, ename from employee
where super_eno = 105;
```



Recap:

SELECT and PROJECT operations

- SELECT operation
 - General form: $\sigma_{\langle \text{tuple selection criteria} \rangle}(r)$
 - Results a subset of tuples meeting the selection criteria from the operand relation r
- PROJECT operation
 - General form: $\pi_{\langle \text{attribute-list} \rangle}(r)$
 - Results a relation that is a projection of specified attributes from the operand relation r
 - Removes duplicates tuple if project results



Recap:

SELECT and PROJECT in SQL

- SQL provides “SELECT statement” for writing all queries; be it SELECT, PROJECT, JOIN, and others
- General form of SELECT statement is (“**for now**”, we shall expand it as we progress)

```
SELECT __<attribute-list>__  
FROM __<relation-expression>__  
[WHERE ____<tuple selection criteria>____]  
[ORDER BY __<ordering attributes>_[DESC] ]_;
```



Recap:

SELECT and PROJECT in SQL

- Order of Evaluation
 - (1) FROM clause
 - (2) WHERE CLAUSE – produce the selected tuples from relation specified in FROM
 - (3) PROJECT as per attribute list



SELECT and PROJECT operation SQL

- SQL allows produce tuples in desired order and provides **ORDER BY** clause in SELECT statement.
- Example below-

```
SELECT ssn, fname, lname, salary  
      FROM employee  
      WHERE dno = 5  
      ORDER BY SALARY DESC;
```

- Note: There is no ordering operation in Relational Algebra.



SQL features

- For specifying the WHERE clause
 - WHERE salary BETWEEN 30000 AND 50000
 - WHERE DID IN ('CS', 'EC')
 - WHERE "LIKE" varchar type are
 - WHERE $(\text{mark1} + \text{marks2} + \text{marks3}) / 3 \geq 50$



Evaluation of WHERE clause

- Each tuple of operand relation is evaluated for specified condition, if the tuple meets the condition, then it is added to the result set
- “PREDICATE” in where clause



Properties of SELECT operation

- The SELECT operation $\sigma \langle \text{selection condition} \rangle (R)$ produces a relation $r(S)$ that has the same schema as R
- The **SELECT operation σ is commutative**; i.e.,
$$\sigma \langle \text{cond1} \rangle (\sigma \langle \text{cond2} \rangle (R)) = \sigma \langle \text{cond2} \rangle (\sigma \langle \text{cond1} \rangle (R))$$
- A cascaded SELECT operation may be applied in any order; i.e.,
$$\begin{aligned} &\sigma \langle \text{cond1} \rangle (\sigma \langle \text{cond2} \rangle (\sigma \langle \text{cond3} \rangle (R))) \\ &= \sigma \langle \text{cond2} \rangle (\sigma \langle \text{cond3} \rangle (\sigma \langle \text{cond1} \rangle (R))) \end{aligned}$$
- A **cascaded SELECT operation may be replaced by a single selection** with a conjunction of all the conditions; i.e.,
$$\begin{aligned} &\sigma \langle \text{cond1} \rangle (\sigma \langle \text{cond2} \rangle (\sigma \langle \text{cond3} \rangle (R))) \\ &= \sigma \langle \text{cond1} \rangle \text{ AND } \langle \text{cond2} \rangle \text{ AND } \langle \text{cond3} \rangle (R) \end{aligned}$$



- $\sigma_{\text{dno}=4}(\sigma_{\text{gender}='F'}(\sigma_{\text{salary} > 30000}(\text{EMPLOYEE})))$
= $\sigma_{\text{dno}=4 \text{ AND gender}='F' \text{ AND salary} > 30000}(\text{R})$



Properties of PROJECT Operation

- The number of tuples in the result of projection $\pi_{\langle \text{list} \rangle} (R)$ is always less or equal to the number of tuples in R .
- If the **list of attributes includes a key of R** , then the number of tuples is **equal to** the number of tuples in R .
- $\pi_{\langle \text{list1} \rangle} (\pi_{\langle \text{list2} \rangle} (R)) = \pi_{\langle \text{list1} \rangle} (R)$
as long as $\langle \text{list1} \rangle$ contains the attributes from $\langle \text{list2} \rangle$



JOIN operations



JOIN

- Here is an motivating example. Consider following query
 - List of B Tech Students who are studying in ‘CS’ department?



Understand JOIN

StudentID	Name	ProgID	CPI
101	Rahul	BCS	7.5
102	Vikash	BIT	8.6
103	Shally	BEE	5.4
104	Alka	BIT	6.8
105	Ravi	BCS	6.5

- Considering generating a relation SP from Student S, and Program P

PID	ProgName	Intake	DID
BCS	BTech(CS)	40	CS
BIT	BTech(IT)	30	CS
BEE	BTech(EE)	40	EE
BME	BTech(ME)	40	ME

by linking through FK and PK

And now applying necessary SELECTION operation?

studid character	name character varying(20)	progid character	cpi numerical	pid character	pname character varying	intake smallint	did character
101	Rahul	BCS	8.70	BCS	BTech (CS)	30	CS
102	Vikash	BEC	6.80	BEC	BTech (ECE)	40	EE
103	Shally	BEE	7.40	BEE	BTech (EE)	40	EE
104	Alka	BEC	7.90	BEC	BTech (ECE)	40	EE
105	Ravi	BCS	9.30	BCS	BTech (CS)	30	CS



Understand JOIN

StudentID	Name	ProgID	CPI
101	Rahul	BCS	7.5
102	Vikash	BIT	8.6
103	Shally	BEE	5.4
104	Alka	BIT	6.8
105	Ravi	BCS	6.5

- This generation of linked relation is what called JOIN, and

- Expressed as

student $\bowtie_{progid=pid}$ program

PID	ProgName	Intake	DID
BCS	BTech(CS)	40	CS
BIT	BTech(IT)	30	CS
BEE	BTech(EE)	40	EE
BME	BTech(ME)	40	ME

studid character	name character varying(20)	progid character	cpi numerical	pid character	pname character varying	intake smallint	did character
101	Rahul	BCS	8.70	BCS	BTech (CS)	30	CS
102	Vikash	BEC	6.80	BEC	BTech (ECE)	40	EE
103	Shally	BEE	7.40	BEE	BTech (EE)	40	EE
104	Alka	BEC	7.90	BEC	BTech (ECE)	40	EE
105	Ravi	BCS	9.30	BCS	BTech (CS)	30	CS



JOIN Operation

- JOIN is a Binary Operator and has join condition as a parameter.
- General Syntax: $r \xrightarrow{\langle \text{join-condition} \rangle} s$
- Here is how it is expressed for JOIN-ing STUDENT and PROGRAM relations

$\text{student} \xrightarrow{\text{progid}=\text{pid}} \text{program}$

eno	ename	dob	gender	salary	super_eno	dno
105	Jayesh	1967-11-10	M	55,000		1
102	Sanna	1982-07-23	F	35,000	105	5
106	Vijay	1971-06-20	M	53,000	105	4
101	Sanjay	1955-11-09	M	70,000		
108	Priya	1978-07-19	F	45,000	106	4
104	Ramesh	1972-09-15	M	38,000	106	5
103	Kalpesh	1982-07-31	F	25,000	102	5
107	Ahmad	1979-03-29	M	25,000	106	4
111	Kirit	1985-12-08	M	40,000	102	5

dno	dname	mgr_eno	mgrstartdate
5	Research	102	1998-05-22
4	Administration	106	2007-01-01
1	Headquater	104	2001-06-19



Understand JOIN

- Interpret tuples in results of following JOIN of EMPLOYEE and DEPARTMENT through dno

employee e $\bowtie_{e.dno=d.dno}$ department d

```
SELECT * FROM employee AS e JOIN department AS d ON (e.dno=d.dno);
```

eno	ename	dob	gender	salary	super_eno	dno	dno	dname	mgr_eno	mgrstartdate
105	Jayesh	1967-11-10	M	55,000		1	1	Headquater	104	2001-06-19
102	Sanna	1982-07-23	F	35,000	105	5	5	Research	102	1998-05-22
106	Vijay	1971-06-20	M	53,000	105	4	4	Administration	106	2007-01-01
108	Priya	1978-07-19	F	45,000	106	4	4	Administration	106	2007-01-01
104	Ramesh	1972-09-15	M	38,000	106	5	5	Research	102	1998-05-22
103	Kalpesh	1982-07-31	F	25,000	102	5	5	Research	102	1998-05-22
107	Ahmad	1979-03-29	M	25,000	106	4	4	Administration	106	2007-01-01
111	Kirit	1985-12-08	M	40,000	102	5	5	Research	102	1998-05-22

Note: employee having NULL in DNO is not included in the result



Understand JOIN

- Interpret tuples in results of following JOIN of EMPLOYEE and DEPARTMENT through mgr_eno

employee e \bowtie department d
 $e.eno=d.mgr_eno$

```
SELECT * FROM employee AS e JOIN department AS d ON (e.eno=d.mgr_eno);
```

eno	ename	dob	gender	salary	super_eno	dno	dno	dname	mgr_eno	mgrstartdate
102	Sanna	1982-07-23	F	35,000	105	5	5	Research	102	1998-05-22
106	Vijay	1971-06-20	M	53,000	105	4	4	Administration	106	2007-01-01
104	Ramesh	1972-09-15	M	38,000	106	5	1	Headquater	104	2001-06-19

Understand how two different join condition between employee and department reflects the result?



Understand JOIN

- Interpret tuples in results of following JOIN

employee e $\bowtie_{e.super_eno=s.eno}$ employee s

```
SELECT * FROM employee AS e JOIN employee AS s ON(e.super_eno=s.eno);
```

eno	ename	dob	gender	salary	super_eno	dno	eno	ename	dob	gender	salary	super_eno	dno
102	Sanna	1982-07-23	F	35,000	105	5	105	Jayesh	1967-11-10	M	55,000		1
106	Vijay	1971-06-20	M	53,000	105	4	105	Jayesh	1967-11-10	M	55,000		1
108	Priya	1978-07-19	F	45,000	106	4	106	Vijay	1971-06-20	M	53,000	105	4
104	Ramesh	1972-09-15	M	38,000	106	5	106	Vijay	1971-06-20	M	53,000	105	4
103	Kalpesh	1982-07-31	F	25,000	102	5	102	Sanna	1982-07-23	F	35,000	105	5
107	Ahmad	1979-03-29	M	25,000	106	4	106	Vijay	1971-06-20	M	53,000	105	4
111	Kirit	1985-12-08	M	40,000	102	5	102	Sanna	1982-07-23	F	35,000	105	5



Understand JOIN

- Interpret tuples in results of following JOIN

$$\text{employee } e \bowtie_{e.\text{super_eno}=s.\text{eno}} \text{employee } s$$

```
SELECT * FROM employee AS e JOIN employee AS s ON(e.super_eno=s.eno);
```

eno	ename	dob	gender	salary	super_eno	dno	eno	ename	dob	gender	salary	super_eno	dno
102	Sanna	1982-07-23	F	35,000	105	5	105	Jayesh	1967-11-10	M	55,000		1
106	Vijay	1971-06-20	M	53,000	105	4	105	Jayesh	1967-11-10	M	55,000		1
108	Priya	1978-07-19	F	45,000	106	4	106	Vijay	1971-06-20	M	53,000	105	4
104	Ramesh	1972-09-15	M	38,000	106	5	106	Vijay	1971-06-20	M	53,000	105	4
103	Kalpesh	1982-07-31	F	25,000	102	5	102	Sanna	1982-07-23	F	35,000	105	5
107	Ahmad	1979-03-29	M	25,000	106	4	106	Vijay	1971-06-20	M	53,000	105	4
111	Kirit	1985-12-08	M	40,000	102	5	102	Sanna	1982-07-23	F	35,000	105	5

student				
studid	name	progid	batch	cpi
1011	Shally	BCS	2010	8
1018	Jitendra	BEE	2010	8
1110	Rohit	BCS	2011	7.9
1113	Pharhan	BEE	2011	8
1118	Aesha	BEE	2011	8
1120	Manav	BEC	2011	7.9
1215	Mehul	BIT	2012	7.9
1218	Megh	BEE	2012	8
1219	Misri	MCS	2012	6.8
1312	Amita	MCS	2013	7
1401	Kirit	MCS	2014	1
1402	Mukesh	MCS	2014	

program			
pid	pname	intake	did
BCS	BTech(CS)	60	CS
MCS	MTech(CS)	20	CS
BEE	BTech(Ee)	40	EE
BME	BTech(ME)	40	ME
BIT	BTech(IT)	40	CS
BEC	BTech(EC)	40	EE

department	
did	dname
CS	Computer Engineering
EE	Electrical Engineering
ME	Mechanical Engineering
CE	Civil Engineering

Understand JOIN



student \bowtie program p \bowtie department
 $progid=pid$ $p.did=d.did$

```
select * from student JOIN program as p ON (progid=pid)
JOIN department as d ON p.did=d.did;
```

select * from student join program as p on (progid=pid) join department d on (p.did=d.did)										
studid	name	progid	batch	cpi	pid	pname	intake	did	did	dname
1011	Shally	BCS	2010	8	BCS	BTech(CS)	60	CS	CS	Computer Engineering
1018	Jitendra	BEE	2010	8	BEE	BTech(EE)	40	EE	EE	Electrical Engineering
1110	Rohit	BCS	2011	7.9	BCS	BTech(CS)	60	CS	CS	Computer Engineering
1113	Pharhan	BEE	2011	8	BEE	BTech(EE)	40	EE	EE	Electrical Engineering
1118	Aesha	BEE	2011	8	BEE	BTech(EE)	40	EE	EE	Electrical Engineering
1120	Manav	BEC	2011	7.9	BEC	BTech(EC)	40	EE	EE	Electrical Engineering
1215	Mehul	BIT	2012	7.9	BIT	BTech(IT)	40	CS	CS	Computer Engineering
1218	Megh	BEE	2012	8	BEE	BTech(EE)	40	EE	EE	Electrical Engineering
1219	Misri	MCS	2012	6.8	MCS	MTech(CS)	20	CS	CS	Computer Engineering
1312	Amita	MCS	2013	7	MCS	MTech(CS)	20	CS	CS	Computer Engineering
1401	Kirit	MCS	2014	1	MCS	MTech(CS)	20	CS	CS	Computer Engineering
1402	Mukesh	MCS	2014		MCS	MTech(CS)	20	CS	CS	Computer Engineering



Example

What is Name of Department in which student id=1120 studies?

$$r1 \leftarrow \text{student} \underset{\text{progid}=pid}{\bowtie} \text{program } p \underset{p.did=d.did}{\bowtie} \text{department}$$
$$\text{result} \leftarrow \sigma_{(\text{studid}=1120)} (r1)$$



JOIN operation

- Can be understood through following pseudo algorithm for computing $r \bowtie_{\langle join-cond \rangle} s$

result_set = NULL;

For each tuple t1 in relation r1

For each tuple t2 in relation r2

If join-cond met then

 derive new tuple t = t1 concatenate t2

 append t to result_set



JOIN Operation

- Result of a JOIN has
 - Schema = $R \cup S$ for a JOIN of R and S
 - Number of tuples in the result relation?



JOIN in SQL

- The join $r \bowtie_{\langle \text{join-cond} \rangle} s$ in relational algebra will be written in SQL as following-

SELECT * FROM r JOIN s ON (<join-cond>);

- Example JOIN **student** $\bowtie_{\langle \text{progid}=\text{pid} \rangle}$ **program** is expressed in SQL, as following-

SELECT * FROM STUDENT JOIN PROGRAM ON (progid=pid)



JOIN - Why do we need ?

- Normally database contains *normalized* relations, and in order to answer a query we may need to collect data from multiple relations
- In example on previous slide, we wanted to list program-name, its department-name. But these attributes are in two different relations (Why in two different relations?) for such situations, where we need to combine data from multiple relations, we need joins.



Exercises

- List (prog-name, deptt-name) of programs offered at XIT
- List student name, program name of students having cpi > 7.5
- List employees (eno, pname) who work on projects controlled by dno=5;
- List employees (ename, salary) who work on projects controlled by dno=5;



JOIN Operation

- In theory JOIN operation is also explained through CROSS JOIN.
- CROSS JOIN is cross product of operand relations

$$r \bowtie_{\langle \text{join-cond} \rangle} s$$

$$= \sigma_{\langle \text{join-cond} \rangle} (r \times s)$$



CROSS JOIN

student × program

studid character	name character varying(20)	progid character	cpi numerical	pid character	pname character varying	intake smallint	did character
101	Rahul	BCS	8.70	BCS	BTech (CS)	30	CS
102	Vikash	BEC	6.80	BCS	BTech (CS)	30	CS
103	Shally	BEE	7.40	BCS	BTech (CS)	30	CS
104	Alka	BEC	7.90	BCS	BTech (CS)	30	CS
105	Ravi	BCS	9.30	BCS	BTech (CS)	30	CS
101	Rahul	BCS	8.70	BEC	BTech (ECE)	40	EE
102	Vikash	BEC	6.80	BEC	BTech (ECE)	40	EE
103	Shally	BEE	7.40	BEC	BTech (ECE)	40	EE
104	Alka	BEC	7.90	BEC	BTech (ECE)	40	EE
105	Ravi	BCS	9.30	BEC	BTech (ECE)	40	EE
101	Rahul	BCS	8.70	BEE	BTech (EE)	40	EE
102	Vikash	BEC	6.80	BEE	BTech (EE)	40	EE
103	Shally	BEE	7.40	BEE	BTech (EE)	40	EE
104	Alka	BEC	7.90	BEE	BTech (EE)	40	EE
105	Ravi	BCS	9.30	BEE	BTech (EE)	40	EE
101	Rahul	BCS	8.70	BME	BTech (ME)	40	ME
102	Vikash	BEC	6.80	BME	BTech (ME)	40	ME
103	Shally	BEE	7.40	BME	BTech (ME)	40	ME
104	Alka	BEC	7.90	BME	BTech (ME)	40	ME
105	Ravi	BCS	9.30	BME	BTech (ME)	40	ME



Try interpreting content of each tuple of CROSS JOIN? And compare highlighted tuple with tuples in JOIN result

SELECT * FROM student
CROSS JOIN program

charact	character varying(20)	character	cpi numeri	pid charac	pname character vary	intake smallin	did chara
101	Rahul	BCS	8.70	BCS	BTech (CS)	30	CS
102	Vikash	BEC	6.80	BCS	BTech (CS)	30	CS
103	Shally	BEE	7.40	BCS	BTech (CS)	30	CS
104	Alka	BEC	7.90	BCS	BTech (CS)	30	CS
105	Ravi	BCS	9.30	BCS	BTech (CS)	30	CS
101	Rahul	BCS	8.70	BEC	BTech (ECE)	40	EE
102	Vikash	BEC	6.80	BEC	BTech (ECE)	40	EE
103	Shally	BEE	7.40	BEC	BTech (ECE)	40	EE
104	Alka	BEC	7.90	BEC	BTech (ECE)	40	EE
105	Ravi	BCS	9.30	BEC	BTech (ECE)	40	EE
101	Rahul	BCS	8.70	BEE	BTech (EE)	40	EE
102	Vikash	BEC	6.80	BEE	BTech (EE)	40	EE
103	Shally	BEE	7.40	BEE	BTech (EE)	40	EE
104	Alka	BEC	7.90	BEE	BTech (EE)	40	EE
105	Ravi	BCS	9.30	BEE	BTech (EE)	40	EE
101	Rahul	BCS	8.70	BME	BTech (ME)	40	ME
102	Vikash	BEC	6.80	BME	BTech (ME)	40	ME
103	Shally	BEE	7.40	BME	BTech (ME)	40	ME
104	Alka	BEC	7.90	BME	BTech (ME)	40	ME
105	Ravi	BCS	9.30	BME	BTech (ME)	40	ME



JOIN and CROSS JOIN (PRODUCT)

- Note that tuples where progid and pid match only represents correct fact set of values.
- It should be easy to establish following:

$\sigma_{\langle \text{progid}=\text{pid} \rangle} (\text{student} \times \text{program})$ is equal to
 $\text{student} \bowtie_{\langle \text{progid}=\text{pid} \rangle} \text{program}$



CROSS JOIN

- Following pseudo algorithm for CROSS PRODUCT or CROSS JOIN

$r \times s$ -

```
result_set = NULL;
for each tuple t1 in relation r1
    for each tuple t2 in relation r2
        form new tuple t = t1 + t2
        append t to result_set
```

- Below is how CROSS PRODUCT written in SQL-

```
SELECT * FROM student CROSS JOIN program; or
SELECT * FROM student, program;
```



Do Not perform JOIN through CROSS Product in SQL!

- When you want to perform JOIN in SQL, use JOIN keyword.
- Even though following two are algebraically same. You should be using second style.

```
SELECT * FROM program, department  
WHERE program.did = department.did;
```



```
SELECT * FROM program JOIN department  
ON (program.did = department.did);
```