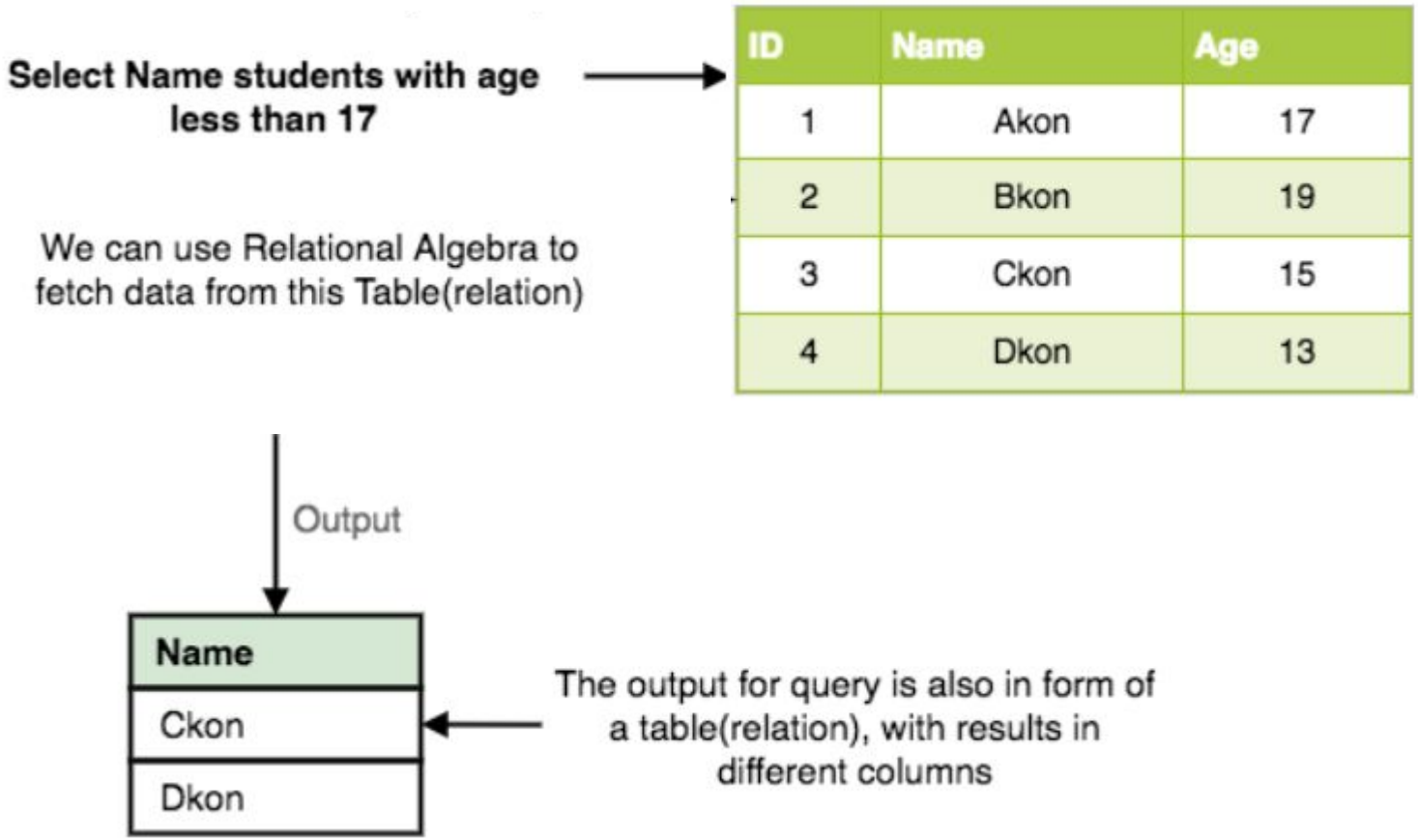
# Relational Algebra

# Relational Algebra

- Relational Algebra is a **procedural query language**. Relational algebra mainly provides a **theoretical foundation** for relational databases and SQL.
- The main purpose of using Relational Algebra is to define operators that transform **one or more input relations** into an **output relation**.
- Given that these operators accept **relations** as **input** and produce **relations** as **output.**
- As it is **pure mathematics**, there is **no use of English Keywords** in Relational Algebra and **operators** are represented using **symbols.**

# Relational Algebra

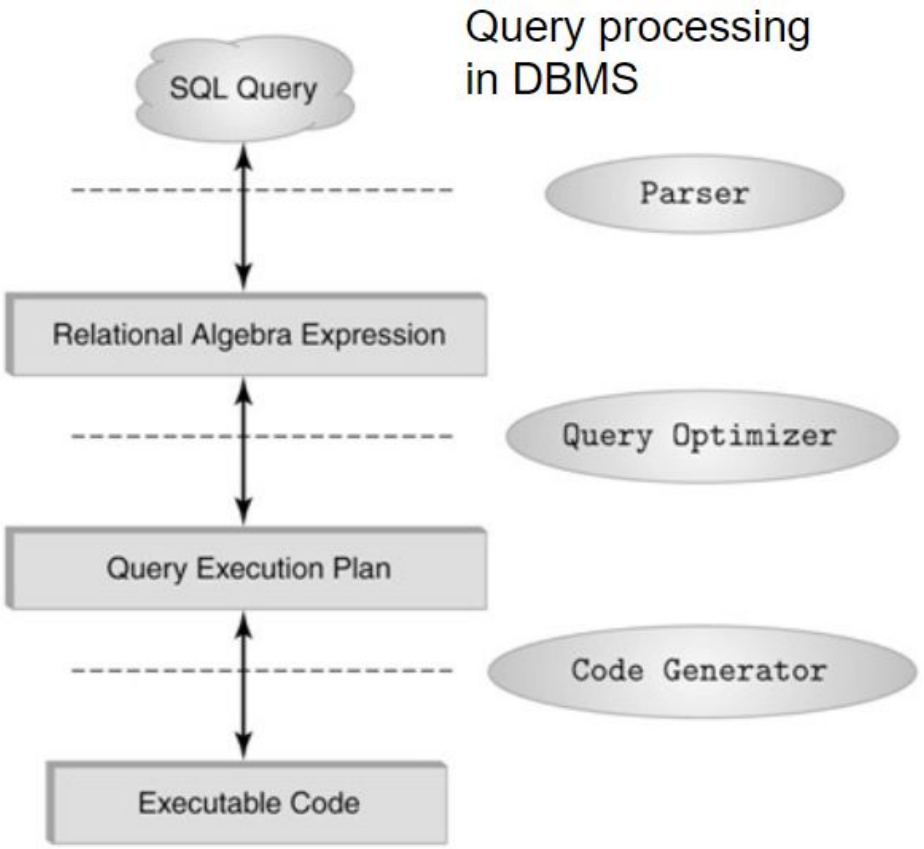


# Relational Algebra

SELECT A.Name, B.Grade
FROM A, B
WHERE A.Id = B.Id

↓

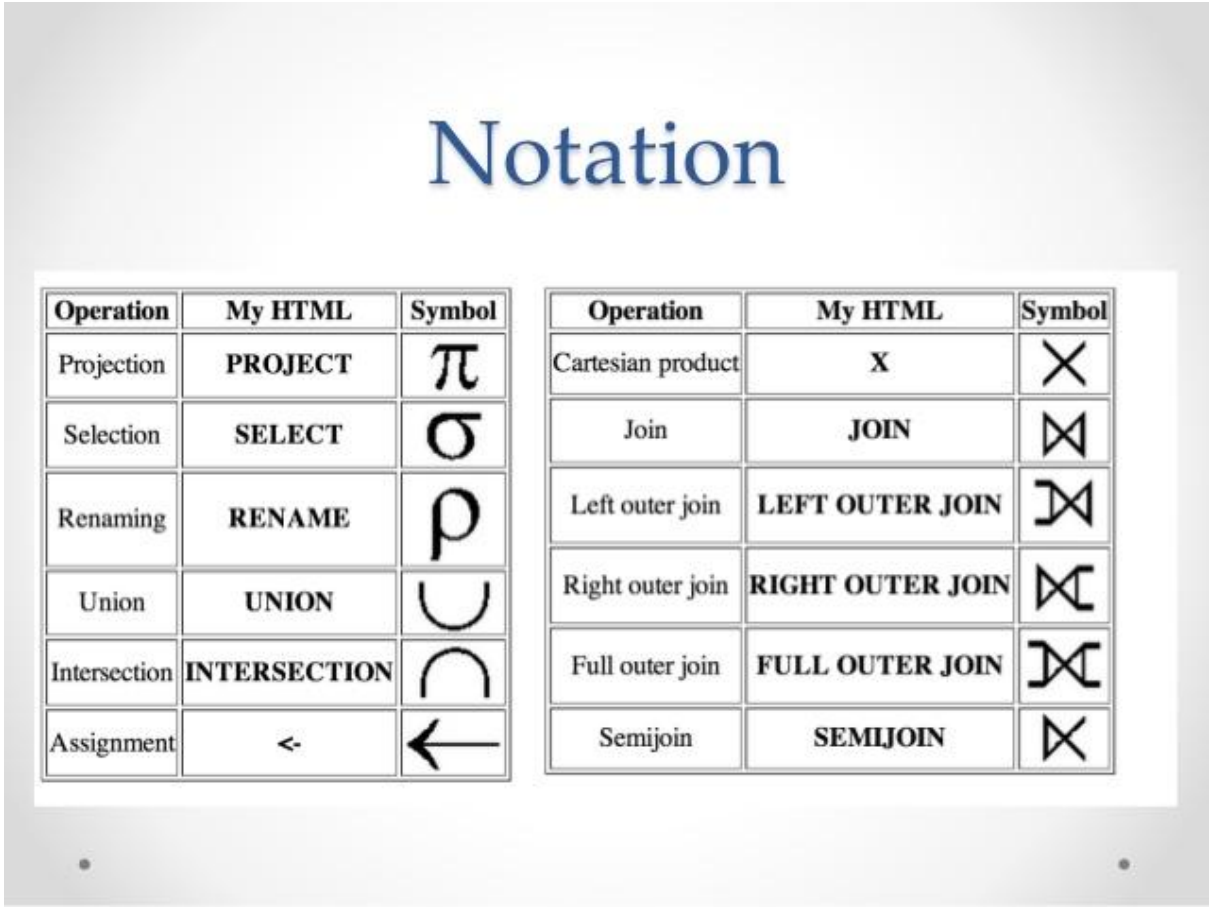
$\pi_{\text{Name, Grade}}(\sigma_{\text{Id,Name}}(A \times B))$



# Relational Algebra - Operations

- Select
- Project
- Rename
- Union
- Intersect
- Set Difference
- Cartesian Product
- Join

# Relational Algebra - Operations



# Select

- Used to **select the required tuples** of data from a relation.
- Denoted by **sigma (σ)**
- During selection, we can specify certain conditions that the data must satisfy
- **Syntax :**

$$\sigma_p(r)$$

# Select Example

**Member**

| Member ID | Name | Date of Birth |
|---|---|---|
| 1 | Alice | 03/03/1995 |
| 2 | Bob | 11/07/1993 |
| 3 | Charlie | 21/10/1997 |
| 4 | Mike | 16/09/1992 |
| 5 | Katie | 21/10/1997 |

***Query***:

*Details of the members who were born on 21/10/1997.*

$$\sigma_{Date\ of\ Birth=21/10/1997}(Member)$$

| Member ID | Name | Date of Birth |
|---|---|---|
| 3 | Charlie | 21/10/1997 |
| 5 | Katie | 21/10/1997 |

## Select Quiz :

***Query***: *Details of loan in branch 'Perryride'.*

- **TABLE NAME : LOAN**

| BRANCH_NAME | LOAN_NO | AMOUNT |
|---|---|---|
| Downtown | L-17 | 1000 |
| Redwood | L-23 | 2000 |
| Perryride | L-15 | 1500 |
| Downtown | L-14 | 1500 |
| Mianus | L-13 | 500 |
| Roundhill | L-11 | 900 |
| Perryride | L-16 | 1300 |

$$\sigma \text{ BRANCH\_NAME}=\text{"perryride"} \ (\text{LOAN})$$

# Project

- Used to **select the required columns** of data from a relation
- Projection **removes duplicate data**
- Denoted by **Π**
- **Syntax:**

$$\Pi_{A1,\ A2...}(r)$$

- A1, A2 etc are attribute names

# Project Example

- **TABLE NAME: BORROW**

| Member ID | Book ID | Book Name |
|---|---|---|
| 1 | 1 | OOPS |
| 3 | 5 | DBMS |
| 4 | 3 | DS |
| 5 | 2 | Java |

***Query:***

*Member IDs of members who have borrowed books.*

$$\pi_{Member\ ID}(Borrow)$$

| Member ID |
|---|
| 1 |
| 3 |
| 4 |
| 5 |

# Project

***Query:***

*Member IDs of members and the Book IDs of the books they have borrowed books.*

$$\pi_{Member\ ID, Book\ ID}(Borrow)$$

| Member ID | Book ID | Book Name |
|---|---|---|
| 1 | 1 | OOPS |
| 3 | 5 | AI |
| 3 | 3 | DBMS |
| 4 | 2 | DS |
| 5 | 4 | Java |

| Member ID | Book ID |
|---|---|
| 1 | 1 |
| 3 | 5 |
| 3 | 3 |
| 4 | 2 |
| 5 | 4 |

# Project

- **TABLE NAME : CUSTOMERS**

| CustomerID | CustomerName | Status |
|---|---|---|
| 1 | Google | Active |
| 2 | Amazon | Active |
| 3 | Apple | Inactive |
| 4 | Alibaba | Active |

***Query:***

*Select the columns customer Name and status from the table Customers*

$$\Pi_{\text{CustomerName, Status}}(\text{Customers})$$

# Rename

- **Rename** operation allows renaming a certain output relation
- Denoted by ρ

**Member**

| Member ID | Name | Date of Birth |
|---|---|---|
| 1 | Alice | 03/03/1995 |
| 2 | Bob | 11/07/1993 |
| 3 | Charlie | 21/10/1997 |
| 4 | Mike | 16/09/1992 |
| 5 | Katie | 21/10/1997 |

***Query:***

*Rename the Member relation as LibraryMember.*

$$\rho_{LibraryMember}(Member)$$

**LibraryMember**

| Member ID | Name | Date of Birth |
|---|---|---|
| 1 | Alice | 03/03/1995 |
| 2 | Bob | 11/07/1993 |
| 3 | Charlie | 21/10/1997 |
| 4 | Mike | 16/09/1992 |
| 5 | Katie | 21/10/1997 |

# Union

- The **UNION** operator could be used to find the **result set or combination of two or more tables.**
  - Each **table** used within UNION must have the same **number of columns**.
  - The **columns** must have the **same data types**.
  - The **columns in each table** must be in the **same order**.
- Denoted by ∪ symbol
- **Syntax:**

```
table_name1 ∪ table_name2
```

| Table A | | Table B | |
|---|---|---|---|
| column 1 | column 2 | column 1 | column 2 |
| 1 | 1 | 1 | 1 |
| 1 | 2 | 1 | 3 |

A ∪ B gives

| Table A ∪ B | |
|---|---|
| column 1 | column 2 |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |

# Union – Perform union on student_name

Table 1: COURSE

| Course_Id | Student_Name | Student_Id |
|---|---|---|
| C101 | Aditya | S901 |
| C104 | Aditya | S901 |
| C106 | Steve | S911 |
| C109 | Paul | S921 |
| C115 | Lucy | S931 |

Table 2: STUDENT

| Student_Id | Student_Name | Student_Age |
|---|---|---|
| S901 | Aditya | 19 |
| S911 | Steve | 18 |
| S921 | Paul | 19 |
| S931 | Lucy | 17 |
| S941 | Carl | 16 |
| S951 | Rick | 18 |

$$\prod \text{Student\_Name}(\text{COURSE}) \cup \prod \text{Student\_Name}(\text{STUDENT})$$

# Union - Perform union on customer_name

**DEPOSITOR RELATION**

| CUSTOMER_NAME | ACCOUNT_NO |
|---|---|
| Johnson | A-101 |
| Smith | A-121 |
| Mayes | A-321 |
| Turner | A-176 |
| Johnson | A-273 |
| Jones | A-472 |
| Lindsay | A-284 |

**BORROW RELATION**

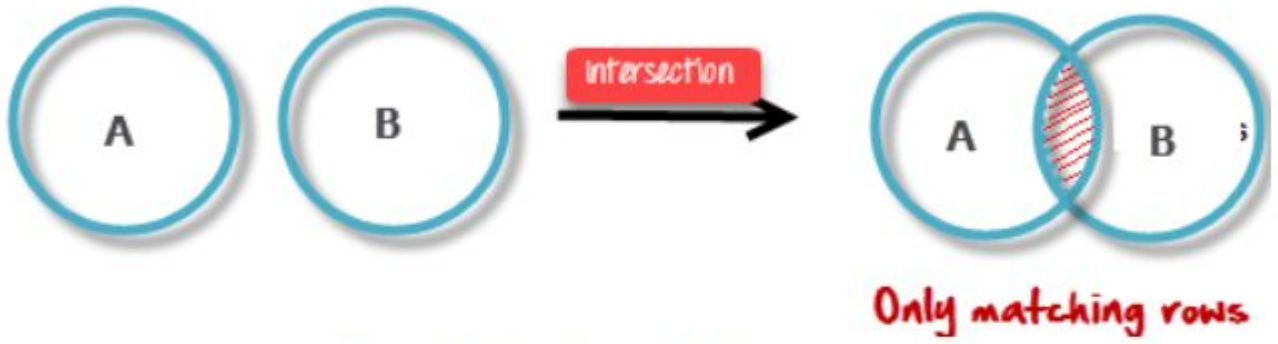
| CUSTOMER_NAME | LOAN_NO |
|---|---|
| Jones | L-17 |
| Smith | L-23 |
| Hayes | L-15 |
| Jackson | L-14 |
| Curry | L-93 |
| Smith | L-11 |
| Williams | L-17 |

$$\Pi \text{ CUSTOMER\_NAME (BORROW)} \cup \Pi \text{ CUSTOMER\_NAME (DEPOSITOR)}$$

# Intersection

- Defined by the symbol ∩
- Suppose there are two tables A and B. The set intersection operation will return only those rows which will be **common to both of the tables**
- **Syntax:**

A ∩ B




Visual Definition of Intersection

# Intersection

| Table A | | Table B | |
|---|---|---|---|
| column 1 | column 2 | column 1 | column 2 |
| 1 | 1 | 1 | 1 |
| 1 | 2 | 1 | 3 |

| Table A ∩ B | |
|---|---|
| column 1 | column 2 |
| 1 | 1 |

# Intersection

**DEPOSITOR RELATION**

| CUSTOMER_NAME | ACCOUNT_NO |
|---|---|
| Johnson | A-101 |
| Smith | A-121 |
| Mayes | A-321 |
| Turner | A-176 |
| Johnson | A-273 |
| Jones | A-472 |
| Lindsay | A-284 |

**BORROW RELATION**

| CUSTOMER_NAME | LOAN_NO |
|---|---|
| Jones | L-17 |
| Smith | L-23 |
| Hayes | L-15 |
| Jackson | L-14 |
| Curry | L-93 |
| Smith | L-11 |
| Williams | L-17 |

$\prod$ CUSTOMER_NAME (BORROW) $\cap$ $\prod$ CUSTOMER_NAME (DEPOSITOR)

| CUSTOMER_NAME |
|---|
| Smith |
| Jones |

# Set difference

- We have two relations R1 and R2 and selects all those tuples(rows) that are **present in relation R1 but not present in relation R2**
- Denoted by – symbol
- Both the relations must have the same set of attributes
- **Syntax:**

```
table_name1 - table_name2
```

# Set difference

| Table A | | Table B | |
|---|---|---|---|
| column 1 | column 2 | column 1 | column 2 |
| 1 | 1 | 1 | 1 |
| 1 | 2 | 1 | 3 |

A - B

| Table A - B | |
|---|---|
| column 1 | column 2 |
| 1 | 2 |

$$\Pi_{\text{author}}(\text{Books}) - \Pi_{\text{author}}(\text{Articles})$$

**Provides the name of authors who have written books but not articles**

# SET DIFFERENCE

Table 1: COURSE

| Course_Id | Student_Name | Student_Id |
|---|---|---|
| C101 | Aditya | S901 |
| C104 | Aditya | S901 |
| C106 | Steve | S911 |
| C109 | Paul | S921 |
| C115 | Lucy | S931 |

Table 2: STUDENT

| Student_Id | Student_Name | Student_Age |
|---|---|---|
| S901 | Aditya | 19 |
| S911 | Steve | 18 |
| S921 | Paul | 19 |
| S931 | Lucy | 17 |
| S941 | Carl | 16 |
| S951 | Rick | 18 |

**write a query to select those students who have not enrolled their courses**

$$\prod \text{Student\_Name}(\text{STUDENT}) - \prod \text{Student\_Name}(\text{COURSE})$$

**Output:**

| Student_Name |
|---|
| Carl |
| Rick |

# SET DIFFERENCE

**DEPOSITOR RELATION**

| CUSTOMER_NAME | ACCOUNT_NO |
|---|---|
| Johnson | A-101 |
| Smith | A-121 |
| Mayes | A-321 |
| Turner | A-176 |
| Johnson | A-273 |
| Jones | A-472 |
| Lindsay | A-284 |

**BORROW RELATION**

| CUSTOMER_NAME | LOAN_NO |
|---|---|
| Jones | L-17 |
| Smith | L-23 |
| Hayes | L-15 |
| Jackson | L-14 |
| Curry | L-93 |
| Smith | L-11 |
| Williams | L-17 |

**write a query to select customers who have loan but does not maintain a deposit in the bank**

**Output:**

| CUSTOMER_NAME |
|---|
| Jackson |
| Hayes |
| Willians |
| Curry |

$$\prod \text{CUSTOMER\_NAME (BORROW)} - \prod \text{CUSTOMER\_NAME (DEPOSITOR)}$$

# CARTESIAN PRODUCT

- Operation used to **merge columns** from **two relations**
- Combines information of two different relations into one
- Denoted by **X** symbol
- A X B will results all the attributes of A followed by each attribute of B
- **Each record of A will pairs with every record of B**
- It is also called **cross product or cross join**
- **Syntax:**

A X B

meaningful operation when it is followed by other operations

# CARTESIAN PRODUCT

A

| Name | Age | Sex |
|---|---|---|
| Ram | 14 | M |
| Sona | 15 | F |
| kim | 20 | M |

B

| Id | Course |
|---|---|
| 1 | DS |
| 2 | DBMS |

**A X B**

| Name | Age | Sex | Id | Course |
|---|---|---|---|---|
| Ram | 14 | M | 1 | DS |
| Ram | 14 | M | 2 | DBMS |
| Sona | 15 | F | 1 | DS |
| Sona | 15 | F | 2 | DBMS |
| Kim | 20 | M | 1 | DS |
| Kim | 20 | M | 2 | DBMS |

# CARTESIAN PRODUCT

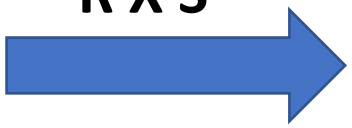
Table 1: R

| Col_A | Col_B |
|---|---|
| AA | 100 |
| BB | 200 |
| CC | 300 |

Table 2: S

| Col_X | Col_Y |
|---|---|
| XX | 99 |
| YY | 11 |
| ZZ | 101 |

**R X S**



| Col_A | Col_B | Col_X | Col_Y |
|---|---|---|---|
| AA | 100 | XX | 99 |
| AA | 100 | YY | 11 |
| AA | 100 | ZZ | 101 |
| BB | 200 | XX | 99 |
| BB | 200 | YY | 11 |
| BB | 200 | ZZ | 101 |
| CC | 300 | XX | 99 |
| CC | 300 | YY | 11 |
| CC | 300 | ZZ | 101 |

Total rows in R X S = no of rows in R x no of rows in S
= 3 x 3
= 9

# CARTESIAN PRODUCT

## EMPLOYEE

| EMP_ID | EMP_NAME | EMP_DEPT |
|---|---|---|
| 1 | Smith | A |
| 2 | Harry | C |
| 3 | John | B |

## DEPARTMENT

| DEPT_NO | DEPT_NAME |
|---|---|
| A | Marketing |
| B | Sales |
| C | Legal |

## EMPLOYEE X DEPARTMENT

| EMP_ID | EMP_NAME | EMP_DEPT | DEPT_NO | DEPT_NAME |
|---|---|---|---|---|
| 1 | Smith | A | A | Marketing |
| 1 | Smith | A | B | Sales |
| 1 | Smith | A | C | Legal |
| 2 | Harry | C | A | Marketing |
| 2 | Harry | C | B | Sales |
| 2 | Harry | C | C | Legal |
| 3 | John | B | A | Marketing |
| 3 | John | B | B | Sales |
| 3 | John | B | C | Legal |

# CARTESIAN PRODUCT

$$\sigma_{author = 'tutorialspoint'}(Books \times Articles)$$

Yields a relation, which shows all the books and articles written by tutorialspoint.

# CARTESIAN PRODUCT

*Characters*

| name | house |
|---|---|
| Tyrion | Lannister |
| Daenerys | Targaryen |

*Episodes*

| season | num | title |
|---|---|---|
| 1 | 1 | Winter is Coming |
| 1 | 2 | The Kingsroad |

*Characters × Episodes*

| name | house | season | num | title |
|---|---|---|---|---|
| Tyrion | Lannister | 1 | 1 | Winter is Coming |
| Tyrion | Lannister | 1 | 2 | The Kingsroad |
| Daenery | Targaryen | 1 | 1 | Winter is Coming |
| Daenery | Targaryen | 1 | 2 | The Kingsroad |

# Joins

- **Selectively pairs up tuples from two relations**
- Join operation is essentially a cartesian product followed by a selection criterion.
- Denoted by ⋈
- **Combines** related tuples from different relations, if and **only if a given join condition is satisfied**
- **Syntax:**

$$\text{Relation1} \bowtie_{\text{condition}} \text{Relation2}$$

# Joins - Example

**EMPLOYEE**

| EMP_CODE | EMP_NAME |
|---|---|
| 101 | Stephan |
| 102 | Jack |
| 103 | Harry |

**SALARY**

| EMP_CODE | SALARY |
|---|---|
| 101 | 50000 |
| 102 | 30000 |
| 103 | 25000 |

**EMPLOYEE ⋈ SALARY**

| EMP_CODE | EMP_NAME | SALARY |
|---|---|---|
| 101 | Stephan | 50000 |
| 102 | Jack | 30000 |
| 103 | Harry | 25000 |

# JOINS - EXAMPLE

*Characters*

| name | house |
|---|---|
| Tyrion | Lannister |
| Daenerys | Targaryen |

*Appearances*

| name | season | num |
|---|---|---|
| Jon Snow | 2 | 1 |
| Tyrion | 1 | 1 |
| Tyrion | 2 | 2 |
| Daenerys | 1 | 2 |

$$Characters \triangleright\!\triangleleft_{name} Appearances$$

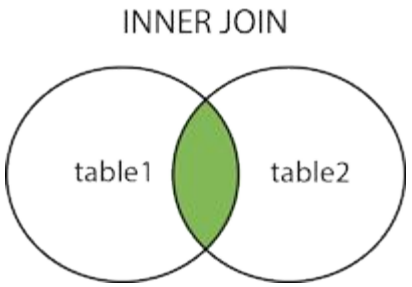
| name | house | name | season | num |
|---|---|---|---|---|
| Tyrion | Lannister | Tyrion | 1 | 1 |
| Tyrion | Lannister | Tyrion | 2 | 2 |
| Daenerys | Targaryen | Daenery | 1 | 2 |

# Joins - Types

Includes tuples that satisfy the matching criteria

Include tuples that satisfy the criteria, some or all tuples that do not match the criteria

- Joins
  - Inner Join
    - Theta Join
    - Equi Join
    - Natural Join
  - Outer Join
    - Left Outer
    - Right Outer
    - Full Outer



# Inner Join – Theta Join

- General case of JOIN operation
- Denoted by symbol **θ**
- **Combines tuples** from different relations provided they satisfy the **theta condition**
- **Syntax :**

$$A \bowtie_{\theta} B$$

# INNER JOIN – THETA JOIN

| Table A | | Table B | |
|---|---|---|---|
| column 1 | column 2 | column 1 | column 2 |
| 1 | 1 | 1 | 1 |
| 1 | 2 | 1 | 3 |

$$A \bowtie_{A.column\ 2 < B.column\ 2} (B)$$

| column 1 | column 2 |
|---|---|
| 1 | 2 |

# INNER JOIN – EQUI JOIN

- When theta join uses **only equality comparison operator**, it is said to be **equijoin**
- Special case of conditional join where only equality condition holds between a pair of attributes
- As values of two attributes will be equal in result of equijoin, only one attribute will be appeared in result

$$A \bowtie_{A.column\ 2 = B.column\ 2} (B)$$

# INNER JOIN – EQUI JOIN

| Table A | | Table B | |
|---|---|---|---|
| column 1 | column 2 | column 1 | column 2 |
| 1 | 1 | 1 | 1 |
| 1 | 2 | 1 | 3 |

$$A \bowtie_{A.column\ 2 = B.column\ 2} (B)$$

| column 1 | column 2 |
|---|---|
| 1 | 1 |

# INNER JOIN –EQUI JOIN

| Student | | |
|---|---|---|
| SID | Name | Std |
| 101 | Alex | 10 |
| 102 | Maria | 11 |

| Subjects | |
|---|---|
| Class | Subject |
| 10 | Math |
| 10 | English |
| 11 | Music |
| 11 | Sports |

$$STUDENT \bowtie_{Student.Std = Subject.Class} SUBJECTS$$

Used a = operator → **Equi Join**

| Student_detail | | | | |
|---|---|---|---|---|
| SID | Name | Std | Class | Subject |
| 101 | Alex | 10 | 10 | Math |
| 101 | Alex | 10 | 10 | English |
| 102 | Maria | 11 | 11 | Music |
| 102 | Maria | 11 | 11 | Sports |

# Inner Join – Natural Join

- Binary operator
- Can only be performed if there is a **common attribute** (column) between the relations.
- Set of tuples of **all combinations in r and s** that are equal on their common attribute names
- **Does not use any comparison operator**. It does not concatenate the way a cartesian product does
- **Name and type of the attribute** must be **same.**
- **Syntax:**

$$C \bowtie D$$

# INNER JOIN – NATURAL JOIN

| C | |
|---|---|
| **Num** | **Square** |
| 2 | 4 |
| 3 | 9 |

| D | |
|---|---|
| **Num** | **Cube** |
| 2 | 8 |
| 3 | 27 |

$C \bowtie D$

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

| C ⋈ D | | |
|---|---|---|
| **Num** | **Square** | **Cube** |
| 2 | 4 | 4 |
| 3 | 9 | 27 |

# INNER JOIN – NATURAL JOIN

| Courses | | |
|---|---|---|
| CID | Course | Dept |
| CS01 | Database | CS |
| ME01 | Mechanics | ME |
| EE01 | Electronics | EE |

| HoD | |
|---|---|
| Dept | Head |
| CS | Alex |
| ME | Maya |
| EE | Mira |

| Courses ⋈ HoD | | | |
|---|---|---|---|
| Dept | CID | Course | Head |
| CS | CS01 | Database | Alex |
| ME | ME01 | Mechanics | Maya |
| EE | EE01 | Electronics | Mira |

# INNER Join - Natural Join

Emp

| (Name | Id | Dept_name ) |
|---|---|---|
| A | 120 | IT |
| B | 125 | HR |
| C | 110 | Sale |
| D | 111 | IT |

Dep

| (Dept_name | Manager) |
|---|---|
| Sale | Y |
| Prod | Z |
| IT | A |

Emp ⋈ Dep

| Name | Id | Dept_name | Manager |
|---|---|---|---|
| A | 120 | IT | A |
| C | 110 | Sale | Y |
| D | 111 | IT | A |

# Joins

$R$

| sid | name | gpa |
|---|---|---|
| 1111 | Joe | 3.2 |
| 2222 | Ann | 4.0 |
| 3333 | Mike | 3.5 |

$S$

| sid | did | cid | term | grade |
|---|---|---|---|---|
| 1111 | 1 | 210 | Fall 2012 | A |
| 2222 | 1 | 220 | Winter 2013 | |

$R \bowtie S$

| R.sid | R.name | R.gpa | S.sid | S.did | S.cid | S.term | S.grade |
|---|---|---|---|---|---|---|---|
| 1111 | Joe | 3.2 | 1111 | 1 | 210 | Fall 2012 | A |
| 2222 | Ann | 4.0 | 2222 | 1 | 220 | Winter 2013 | |

**What are the names of students who got an A in any course?**

*Students*

| sid | name | gpa |
|---|---|---|
| 1111 | Joe | 3.2 |
| 2222 | Ann | 4.0 |
| 3333 | Mike | 3.5 |

*Enrollment*

| sid | did | cid | term | grade |
|---|---|---|---|---|
| 1111 | 1 | 210 | Fall 2015 | A |
| 2222 | 1 | 220 | Winter 2016 | |

$(\ Students \bowtie Enrollment)$

| R.sid | R.name | R.gpa | S.sid | S.did | S.cid | S.term | S.grade |
|---|---|---|---|---|---|---|---|
| 1111 | Joe | 3.2 | 1111 | 1 | 210 | Fall 2012 | A |
| 2222 | Ann | 4.0 | 2222 | 1 | 220 | Winter 2013 | |

$$(\ \sigma_{grade='A'}(\ Students \bowtie Enrollment))$$

$$\pi_{name}\ (\ \sigma_{grade='A'}(\ Students \bowtie Enrollment))$$

| name |
|---|
| Joe |

**What are the names of students who got an A in any course?**

*Students*

| sid | name | gpa |
|---|---|---|
| 1111 | Joe | 3.2 |
| 2222 | Ann | 4.0 |
| 3333 | Mike | 3.5 |

*Enrollment*

| sid | did | cid | term | grade |
|---|---|---|---|---|
| 1111 | 1 | 210 | Fall 2015 | A |
| 2222 | 1 | 220 | Winter 2016 | |

$$\pi_{name}\,(\,Students \bowtie (\,\sigma_{grade='A'}\,Enrollment))$$

| name |
|---|
| Joe |

# Outer Join – Left JOIN

- Select records from the first (left-most) table with matching right table records
- Join starting with the first (left-most) table.
- Then, any matched records from the second table (right-most) will be included
- There is no matching tuple is found in right relation, then the attributes of right relation in the join result are filled with null values.
- **Syntax :**

$$A \text{⟕} B$$

# Left JOIN - EXAMPLE

| A | |
|---|---|
| Num | Square |
| 2 | 4 |
| 3 | 9 |
| 4 | 16 |

| B | |
|---|---|
| Num | Cube |
| 2 | 8 |
| 3 | 27 |
| 5 | 125 |

| A ⋈ B | | |
|---|---|---|
| Num | Square | Cube |
| 2 | 4 | 4 |
| 3 | 9 | 27 |
| 4 | 16 | - |

# Right join

- Operation allows keeping all tuple in the right relation
- Join starting with the second (right-most) table and then any matching first (left-most) table records
- No matching tuple is found in the left relation, then the attributes of the left relation in the join result are filled with null values
- **Syntax:**


# Right JOIN - eXAMPLE

| A | |
|---|---|
| Num | Square |
| 2 | 4 |
| 3 | 9 |
| 4 | 16 |

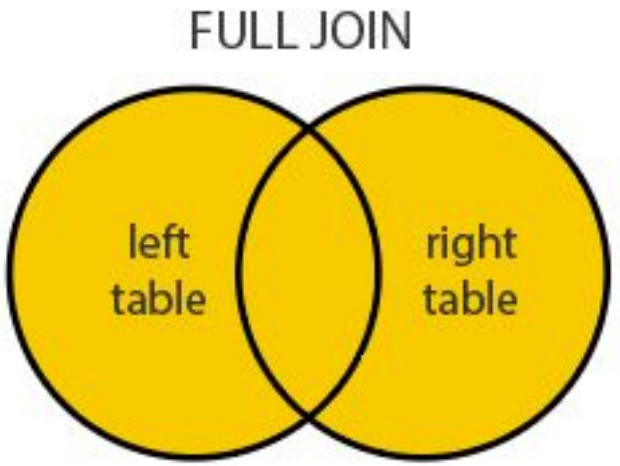
| B | |
|---|---|
| Num | Cube |
| 2 | 8 |
| 3 | 27 |
| 5 | 125 |

| A ⋈ B | | |
|---|---|---|
| Num | Cube | Square |
| 2 | 8 | 4 |
| 3 | 27 | 9 |
| 5 | 125 | - |

# Full join

- **All tuples** from **both relations** are included in the **result**, irrespective of the matching condition.
- **Syntax:**

$$A \bowtie B$$



# Full JOIN - eXAMPLE

| A | |
|---|---|
| Num | Square |
| 2 | 4 |
| 3 | 9 |
| 4 | 16 |

| B | |
|---|---|
| Num | Cube |
| 2 | 8 |
| 3 | 27 |
| 5 | 125 |

| A ⋈ B | | |
|---|---|---|
| Num | Square | Cube |
| 2 | 4 | 8 |
| 3 | 9 | 27 |
| 4 | 16 | - |
| 5 | - | 125 |

# Example Database

**Movies**

| title | director | myear | rating |
|---|---|---|---|
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

**Actors**

| actor | ayear |
|---|---|
| Cage | 1964 |
| Hanks | 1956 |
| Maguire | 1975 |
| McDormand | 1957 |

**Acts**

| actor | title |
|---|---|
| Cage | Raising Arizona |
| Maguire | Spiderman |
| Maguire | Wonder Boys |
| McDormand | Fargo |
| McDormand | Raising Arizona |
| McDormand | Wonder Boys |

**Directors**

| director | dyear |
|---|---|
| Coen | 1954 |
| Hanson | 1945 |
| Raimi | 1959 |

**Example**: Find actors who have acted in some Coen's movie

$e_1 = \textbf{Acts} \bowtie_{Acts.title\ =\ Movies.title} \textbf{Movies}$

| actor | title | director | myear | rating |
|---|---|---|---|---|
| Cage | Raising Arizona | Coen | 1987 | 7.6 |
| Maguire | Spiderman | Raimi | 2002 | 7.4 |
| Maguire | Wonder Boys | Hanson | 2000 | 7.6 |
| McDormand | Fargo | Coen | 1996 | 8.2 |
| McDormand | Raising Arizona | Coen | 1987 | 7.6 |
| McDormand | Wonder Boys | Hanson | 2000 | 7.6 |

$\pi_{actor}(\sigma_{director='Coen'}((e_1))$

| actor |
|---|
| Cage |
| McDormand |

*Sailors (sid, name, rating, age)*

| sid | name | rating | age |
|---|---|---|---|
| 1 | Dustin | 7 | 45 |
| 2 | Rusty | 10 | 35 |
| 3 | Horatio | 5 | 35 |
| 4 | Zorba | 8 | 18 |
| 5 | Julius | | 25 |

*Boats (bid, name, color)*

| bid | name | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

*Reserves (sid, bid, day)*

| sid | bid | day |
|---|---|---|
| 1 | 101 | 10/10/12 |
| 1 | 102 | 10/10/12 |
| 1 | 101 | 10/7/12 |
| 2 | 102 | 11/9/12 |
| 2 | 102 | 7/11/12 |
| 3 | 101 | 7/11/12 |
| 3 | 102 | 7/8/12 |
| 4 | 103 | 19/9/12 |

List names of sailors who reserved boat 102

$$\pi_{name}\ (Sailors \bowtie (\ \sigma_{bid=102}\ Reserves))$$

*Sailors (sid, name, rating, age)*

| sid | name | rating | age |
|---|---|---|---|
| 1 | Dustin | 7 | 45 |
| 2 | Rusty | 10 | 35 |
| 3 | Horatio | 5 | 35 |
| 4 | Zorba | 8 | 18 |
| 5 | Julius | | 25 |

*Boats (bid, name, color)*

| bid | name | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

*Reserves (sid, bid, day)*

| sid | bid | day |
|---|---|---|
| 1 | 101 | 10/10/12 |
| 1 | 102 | 10/10/12 |
| 1 | 101 | 10/7/12 |
| 2 | 102 | 11/9/12 |
| 2 | 102 | 7/11/12 |
| 3 | 101 | 7/11/12 |
| 3 | 102 | 7/8/12 |
| 4 | 103 | 19/9/12 |

List names of sailors who reserved the red Interlake.

$$\pi_{Sailors.name}\ (\ Sailors \bowtie (\ (\ \sigma_{name=Interlake\ and\ color=red}\ Boats) \bowtie Reserves))$$

*Sailors (sid, name, rating, age)*

| sid | name | rating | age |
|---|---|---|---|
| 1 | Dustin | 7 | 45 |
| 2 | Rusty | 10 | 35 |
| 3 | Horatio | 5 | 35 |
| 4 | Zorba | 8 | 18 |
| 5 | Julius | | 25 |

*Boats (bid, name, color)*

| bid | name | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

*Reserves (sid, bid, day)*

| sid | bid | day |
|---|---|---|
| 1 | 101 | 10/10/12 |
| 1 | 102 | 10/10/12 |
| 1 | 101 | 10/7/12 |
| 2 | 102 | 11/9/12 |
| 2 | 102 | 7/11/12 |
| 3 | 101 | 7/11/12 |
| 3 | 102 | 7/8/12 |
| 4 | 103 | 19/9/12 |

List names of boats that were reserved by Horatio.

$$\pi_{Boats.name}\,(\,(\sigma_{Sailors.name=Horatio}\,Sailors) \bowtie (\,Boats \bowtie Reserves))$$

*Sailors (sid, name, rating, age)*

| sid | name | rating | age |
|---|---|---|---|
| 1 | Dustin | 7 | 45 |
| 2 | Rusty | 10 | 35 |
| 3 | Horatio | 5 | 35 |
| 4 | Zorba | 8 | 18 |
| 5 | Julius | | 25 |

*Boats (bid, name, color)*

| bid | name | color |
|---|---|---|
| 101 | Interlake | blue |
| 102 | Interlake | red |
| 103 | Clipper | green |
| 104 | Marine | red |

*Reserves (sid, bid, day)*

| sid | bid | day |
|---|---|---|
| 1 | 101 | 10/10/12 |
| 1 | 102 | 10/10/12 |
| 1 | 101 | 10/7/12 |
| 2 | 102 | 11/9/12 |
| 2 | 102 | 7/11/12 |
| 3 | 101 | 7/11/12 |
| 3 | 102 | 7/8/12 |
| 4 | 103 | 19/9/12 |

List days on which some sailor with rating higher than 7 was at sea

$$\pi_{day}((\sigma_{rating>7} Sailors) \bowtie Reserves)$$

**Find movies made after 1997**

Movies

| title | director | myear | rating |
|---|---|---|---|
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

$$\sigma_{myear>1997}(\text{Movies})$$

| title | director | myear | rating |
|---|---|---|---|
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

**Find movies made by Hanson after 1997**

Movies

| title | director | myear | rating |
|---|---|---|---|
| Fargo | Coen | 1996 | 8.2 |
| Raising Arizona | Coen | 1987 | 7.6 |
| Spiderman | Raimi | 2002 | 7.4 |
| Wonder Boys | Hanson | 2000 | 7.6 |

$$\sigma_{myear>1997 \ \wedge \ director='Hanson'}(\textbf{Movies})$$

| title | director | myear | rating |
|---|---|---|---|
| Wonder Boys | Hanson | 2000 | 7.6 |