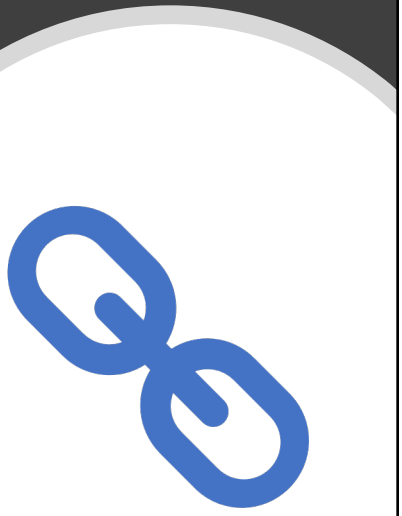
Different Types of JOINed Tables in SQL

10

Different Types of JOINed Tables in SQL

- Cross Reference
- Specify different types of join
  - NATURAL JOIN
  - Various types of OUTER JOIN (LEFT, RIGHT, FULL)
- NATURAL JOIN on two relations R and S
  - No join condition specified
  - Is equivalent to an implicit EQUIJOIN condition for each pair of attributes with same name from R and S

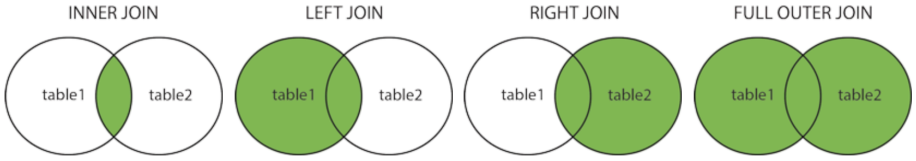


11

Different Types of SQL JOINs

Here are the different types of the JOINs in SQL:

- **(INNER) JOIN**: Returns records that have matching values in both tables ⋈ <join condition>
- **LEFT (OUTER) JOIN**: Return all records from the left table, and the matched records from the right table ⟕ <JOIN COND.>
- **RIGHT (OUTER) JOIN**: Return all records from the right table, and the matched records from the left table ⟖ <JOIN COND.>
- **FULL (OUTER) JOIN**: Return all records when there is a match in either left or right table ⟗ <JOIN COND.>



12

INNER and OUTER Joins

INNER JOIN (versus OUTER JOIN)

- Default type of join in a joined table
- Tuple is included in the result only if a matching tuple exists in the other relation

LEFT OUTER JOIN

⟕ <JOIN COND.>

- Every tuple in left table must appear in result
- If no matching tuple
  - Padded with NULL values for attributes of right table

RIGHT OUTER JOIN

⟖ <JOIN COND.>

- Every tuple in right table must appear in result
- If no matching tuple
  - Padded with NULL values for attributes of left table

Slide 7- 22

13

Join

Student

ID	Name
1	Patrik
2	Albert
3	Maria
4	Darwin
5	Elizabeth
6	

Likes

ID	Like
3	Stars
1	Climbing
1	Code
6	Rugby
4	Apples

Natural join

ID	Name	Like
3	Maria	Stars
1	Patrik	Climbing
1	Patrik	Code
4	Darwin	Apples

Full Outer Join Student.id= likes.id

Name	Like
Patrik	Climbing
Patrik	Code
Albert	NULL
Maria	Stars
Darwin	Apples
Elizabeth	NULL
NULL	Rugby

Left outer join for student.id=like.id

Name	Like
Patrik	Climbing
Patrik	Code
Albert	NULL
Maria	Stars
Darwin	Apples
Elizabeth	NULL

Right outer join for student.id=like.id

Name	Like
Maria	Stars
Patrik	Climbing
Patrik	Code
NULL	Rugby
Darwin	Apples

14

Example: LEFT OUTER JOIN

```
SELECT *
FROM table1 LEFT [ OUTER ] JOIN table2
ON table1.column_name=table2.column_name;
```

$\exists\!\!\bowtie$ <JOIN COND.>

```
SELECT E.Lname AS Employee_Name
       S.Lname AS Supervisor_Name
FROM Employee AS E LEFT OUTER JOIN EMPLOYEE AS S
                   ON E.Super_ssn = S.Ssn)
```

15

Binary Relational Operations: EQUIJOIN

$\bowtie$ <join condition>

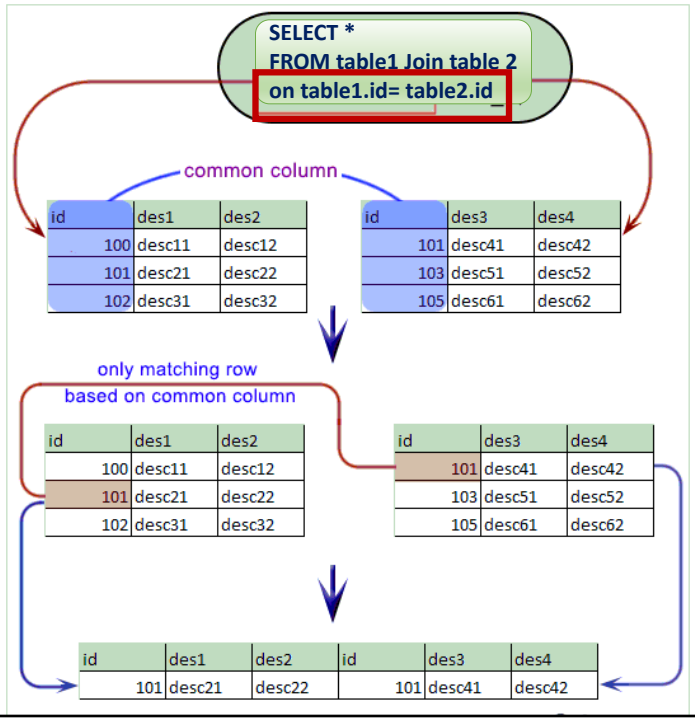
- EQUIJOIN Operation:The most common use of join involves join conditions with *equality comparisons* only
- Such a join, where the only comparison operator used is =, is called an EQUIJOIN.
  - In the result of an EQUIJOIN we always have one or more pairs of attributes (whose names need not be identical) that have identical values in every tuple.
  - The JOIN seen in the previous example was an EQUIJOIN.

Slide 8- 16

16

EQUIJOIN

SELECT *
FROM table1 JOIN table2
[ON (join_condition)]



id	des1	des2
100	desc11	desc12
101	desc21	desc22
102	desc31	desc32

id	des3	des4
101	desc41	desc42
103	desc51	desc52
105	desc61	desc62

id	des1	des2	id	des3	des4
101	desc21	desc22	101	desc41	desc42

17

Binary Relational Operations: NATURAL JOIN Operation

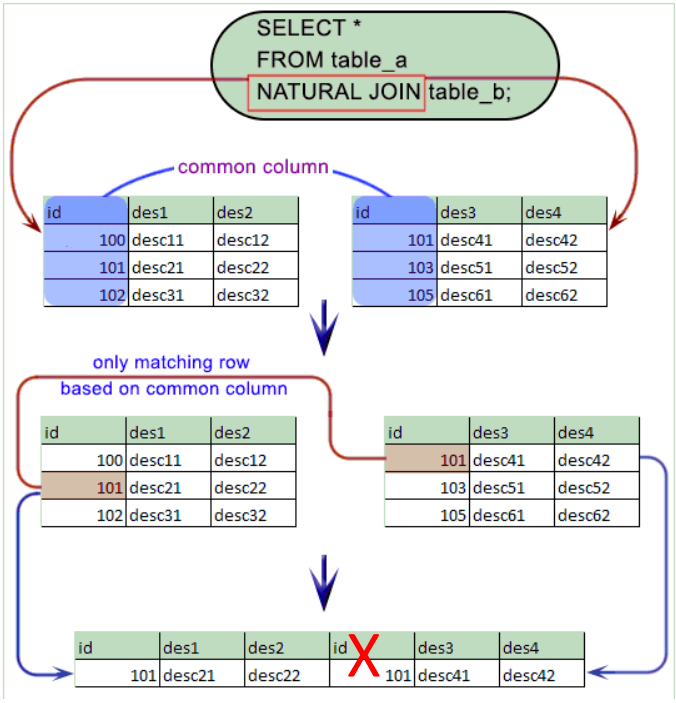
Donated by *

- NATURAL JOIN Operation
  - Another variation of JOIN called NATURAL JOIN — denoted by * — was created to get rid of the second (superfluous) attribute in an EQUIJOIN condition.
    - because one of each pair of attributes with identical values is superfluous
  - The standard definition of natural join requires that the two join attributes, or each pair of corresponding join attributes, *have the same name* in both relations
  - If this is not the case, a renaming operation is applied first.

18

Natural Join

```
SELECT *
FROM table1 NATURAL JOIN table2;
```



id	des1	des2
100	desc11	desc12
101	desc21	desc22
102	desc31	desc32

id	des3	des4
101	desc41	desc42
103	desc51	desc52
105	desc61	desc62

id	des1	des2	id	des3	des4
101	desc21	desc22	101	desc41	desc42

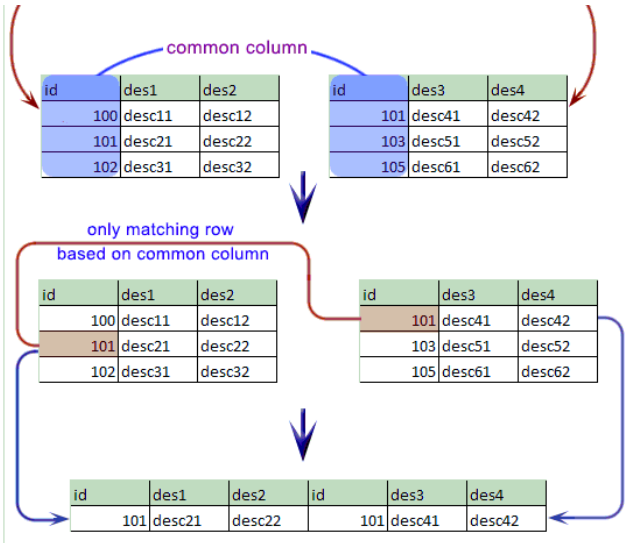
19

Binary Relational Operations NATURAL JOIN

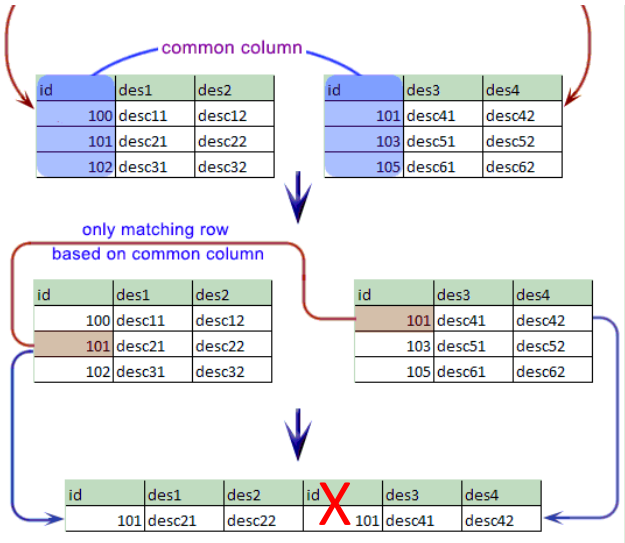
- Example: To apply a natural join on the DNUMBER attributes of DEPARTMENT and DEPT_LOCATIONS, it is sufficient to write:
  - DEPT_LOCS ← DEPARTMENT * DEPT_LOCATIONS
- Only attribute with the same name is DNUMBER
- An implicit join condition is created based on this attribute:
  DEPARTMENT.DNUMBER=DEPT_LOCATIONS.DNUMBER
- Another example: Q ← R(A,B,C,D) * S(C,D,E)
  - The implicit join condition includes *each pair* of attributes with the same name, "AND"ed together:
    - R.C=S.C AND R.D.S.D
  - Result keeps only one attribute of each such pair:
    - Q(A,B,C,D,E)

20

EQUIJOIN



Natural Join



21

NATURAL JOIN Example

- Rename attributes of one relation so it can be joined with another using NATURAL JOIN:

Q1B:

```
SELECT   Fname, Lname, Address
FROM     (EMPLOYEE NATURAL JOIN (DEPARTMENT
         AS DEPT (Dname, Dno, Mssn,Msdate)))
WHERE    Dname='Research';
```

The above works with EMPLOYEE.Dno = DEPT.Dno as an implicit join condition

22

How to Simulate FULL OUTER JOIN in MySQL

Apple

Variety	Price
Fuji	5.00
Gala	6.00

Orange

Variety	Price
Valencia	4.00
Navel	5.00

- **two JOINs and a UNION**

Union eliminate duplicates

```
select * from apples as a
left outer join oranges as o
on a.price = o.price
union
select * from apples as a
right outer join oranges as o
on a.price = o.price
```

- **Using UNION ALL**

Union All keeps duplicates

```
select * from apples as a
left outer join oranges as o
on a.price = o.price
union all
select * from apples as a
right outer join oranges as o
on a.price = o.price;
```

variety	price	variety	price
Fuji	5	Navel	5
Gala	6	NULL	NULL
NULL	NULL	Valencia	4

23

Multiway JOIN in the FROM clause

- FULL OUTER JOIN – combines result if LEFT and RIGHT OUTER JOIN
- Can nest JOIN specifications for a multiway join:

Q2A:

SELECT Pnumber, Dnum, Lname, Address, Bdate
FROM ((PROJECT **JOIN** DEPARTMENT **ON** Dnum=Dnumber)
JOIN EMPLOYEE **ON** Mgr_ssn=Ssn)
WHERE Plocation='Stafford';

24

Join Condition and Where

Input:

committee_id	name
1	John
2	Mary
3	Amelia
4	Joe

member_id	name
1	John
2	Jane
3	Mary
4	David
5	Amelia

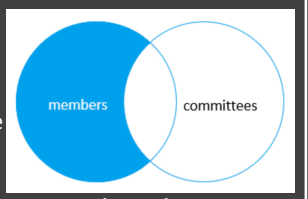
```
SELECT
   m.member_id,
   m.name member,
   c.committee_id,
   c.name committee
FROM
   members m
LEFT JOIN committees c USING(name);
```

member_id	member	committee_id	committee
1	John	1	John
3	Mary	2	Mary
5	Amelia	3	Amelia
2	Jane	NULL	NULL
4	David	NULL	NULL

Desire New Output:

member_id	member	committee_id	committee
2	Jane	NULL	NULL
4	David	NULL	NULL

```
SELECT
   m.member_id,
   m.name member,
   c.committee_id,
   c.name committee
FROM
   members m
LEFT JOIN committees c USING(name)
WHERE c.committee_id IS NULL;
```



25