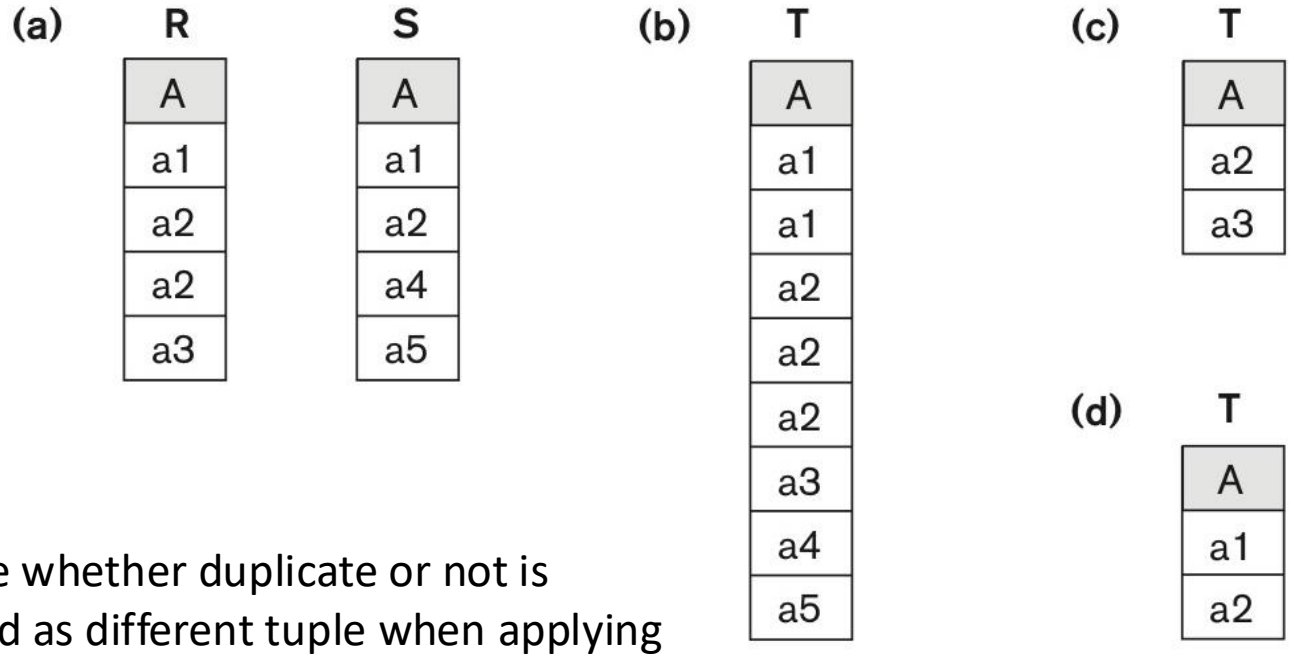
CS4.301 Data & Applications

Ponnurangam Kumaraguru (“PK”)
#ProfGiri @ IIIT Hyderabad

 pk.profgiri  /in/ponguru  @ponguru  Ponnurangam.kumaraguru

Figure 6.5 The results of SQL multiset operations. (a) Two tables, R(A) and S(A). (b) R(A)UNION ALL S(A). (c) R(A) EXCEPT ALL S(A). (d) R(A) INTERSECT ALL S(A).



Each tuple whether duplicate or not is considered as different tuple when applying these operations

Substring Pattern Matching and Arithmetic Operators

LIKE comparison operator

Used for string **pattern matching**

% replaces an arbitrary number of zero or more characters

underscore (_) replaces a single character

Examples: **WHERE** Address **LIKE** '%Houston,TX%';

WHERE Ssn **LIKE** '_ _ 1_ _ 8901'

WHERE text **LIKE** 'ka_ _ _ ka%'

BETWEEN comparison operator

WHERE (Salary **BETWEEN** 30000 **AND** 40000)

AND Dno = 5;

Arithmetic Operations

Standard arithmetic operators:

Addition (+), subtraction (–), multiplication (*), and division (/) may be included as a part of **SELECT**

Query 13. Show the resulting salaries if every employee working on the ‘ProductX’ project is given a 10 percent raise.

Arithmetic Operations

Standard arithmetic operators:

Addition (+), subtraction (–), multiplication (*), and division (/) may be included as a part of **SELECT**

Query 13. Show the resulting salaries if every employee working on the 'ProductX' project is given a 10 percent raise.

```
SELECT  E.Fname, E.Lname, 1.1 * E.Salary AS Increased_sal
FROM  EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
WHERE  E.Ssn=W.Essn AND W.Pno=P.Pnumber AND P.Pname='ProductX';
```

Ordering of Query Results

Use **`ORDER BY`** clause

- Keyword **`DESC`** to see result in a descending order of values
- Keyword **`ASC`** to specify ascending order explicitly
- Typically placed at the end of the query

```
ORDER BY D.Dname DESC, E.Lname ASC, E.Fname ASC
```

How did the quiz go?

This Lecture

Order by

Query 15. Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

Order by

Query 15. Retrieve a list of employees and the projects they are working on, ordered by department and, within each department, ordered alphabetically by last name, then first name.

```
Q15:    SELECT      D.Dname, E.Lname, E.Fname, P.Pname
        FROM        DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W,
                    PROJECT AS P
        WHERE       D.Dnumber = E.Dno AND E.Ssn = W.Essn AND W.Pno =
                    P.Pnumber
        ORDER BY    D.Dname, E.Lname, E.Fname;
```

```
mysql> SELECT
    -> D.Dname, E.Lname, E.Fname, P.Pname
    -> FROM
    -> DEPARTMENT AS D, EMPLOYEE AS E, WORKS_ON AS W, PROJECT AS P
    -> WHERE
    -> D.Dnumber = E.Dno AND E.Ssn = W.Essn AND W.Pno = P.Pnumber
    -> ORDER BY
    -> D.Dname, E.Lname, E.Fname;
+----------------+---------+----------+-----------------+
| Dname          | Lname   | Fname    | Pname           |
+----------------+---------+----------+-----------------+
| Administration | Jabbar  | Ahmad    | Computerization |
| Administration | Jabbar  | Ahmad    | Newbenefits     |
| Administration | Wallace | Jennifer | Reorganization  |
| Administration | Wallace | Jennifer | Newbenefits     |
| Administration | Zelaya  | Alicia   | Computerization |
| Administration | Zelaya  | Alicia   | Newbenefits     |
| Headquarters   | Borg    | James    | Reorganization  |
| Research       | English | Joyce    | ProductX        |
| Research       | English | Joyce    | ProductY        |
| Research       | Narayan | Ramesh   | ProductZ        |
| Research       | Smith   | John     | ProductX        |
| Research       | Smith   | John     | ProductY        |
| Research       | Wong    | Franklin | ProductY        |
| Research       | Wong    | Franklin | ProductZ        |
| Research       | Wong    | Franklin | Computerization |
| Research       | Wong    | Franklin | Reorganization  |
+----------------+---------+----------+-----------------+
16 rows in set (0.02 sec)
```

Basic SQL Retrieval Query Block

```
SELECT      <attribute list>
FROM        <table list>
[ WHERE     <condition> ]
[ ORDER BY <attribute list> ];
```

INSERT, DELETE, and UPDATE Statements in SQL

Three commands used to modify the database:

INSERT, DELETE, and UPDATE

INSERT typically inserts a tuple (row) in a relation (table)

UPDATE may update a number of tuples (rows) in a relation (table) that satisfy the condition

DELETE may also update a number of tuples (rows) in a relation (table) that satisfy the condition

INSERT

In its simplest form, it is used to add one or more tuples to a relation

Attribute values should be listed in the same order as the attributes were specified in the **CREATE TABLE** command

Constraints on data types are observed automatically

Any integrity constraints as a part of the DDL specification are enforced

The INSERT Command

Specify the relation name and a list of values for the tuple. All values including nulls are supplied.

```
U1:   INSERT INTO   EMPLOYEE
      VALUES        ( 'Richard', 'K', 'Marini', '653298653', '1962-12-30', '98
                      Oak Forest, Katy, TX', 'M', 37000, '653298653', 4 );
```

The INSERT Command

Specify the relation name and a list of values for the tuple. All values including nulls are supplied.

U1: **INSERT INTO** EMPLOYEE
VALUES ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98 Oak Forest, Katy, TX', 'M', 37000, '653298653', 4);

```
mysql> INSERT INTO EMPLOYEE
    -> VALUES ('Richard', 'K', 'Marini', '653298653', '1962-12-30', '98 Oak Forest, Katy, TX', 'M', 37000, '653298653', 4);
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM EMPLOYEE;
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
| Fname    | Minit | Lname   | Ssn       | Bdate      | Address                 | Sex  | Salary | Super_ssn | Dno |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
| John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston TX | M    |  30000 | 333445555 |   5 |
| Franklin | T     | Wong    | 333445555 | 1965-12-08 | 638 Voss, Houston TX    | M    |  40000 | 888665555 |   5 |
| Joyce    | A     | English | 453453453 | 1972-07-31 | 5631 Rice, Houston TX   | F    |  25000 | 333445555 |   5 |
| Richard  | K     | Marini  | 653298653 | 1962-12-30 | 98 Oak Forest, Katy, TX | M    |  37000 | 653298653 |   4 |
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX | M    |  38000 | 333445555 |   5 |
| James    | E     | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston TX   | M    |  55000 | NULL      |   1 |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire TX  | F    |  43000 | 888665555 |   4 |
| Ahmad    | V     | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston TX  | M    |  25000 | 987654321 |   4 |
| Alicia   | J     | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring TX  | F    |  25000 | 987654321 |   4 |
+----------+-------+---------+-----------+------------+-------------------------+------+--------+-----------+-----+
9 rows in set (0.00 sec)
```

The INSERT Command

The variation below inserts multiple tuples where a new table is loaded values from the result of a query.

```
CREATE TABLE WORKS_ON_INFO (
    Emp_name VARCHAR(150),
    Proj_name VARCHAR(150),
    Hours_per_week DECIMAL(3, 1)
);
```

```
U3B:  INSERT INTO   WORKS_ON_INFO ( Emp_name, Proj_name,
                    Hours_per_week )
      SELECT        E.Lname, P.Pname, W.Hours
      FROM          PROJECT P, WORKS_ON W, EMPLOYEE E
      WHERE         P.Pnumber=W.Pno AND W.Essn=E.Ssn;
```

```
mysql> CREATE TABLE WORKS_ON_INFO (
    ->     Emp_name VARCHAR(150),
    ->     Proj_name VARCHAR(150),
    ->     Hours_per_week DECIMAL(3, 1)
    -> );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO WORKS_ON_INFO ( Emp_name, Proj_name, Hours_per_week )
    -> SELECT E.Lname, P.Pname, W.Hours
    -> FROM PROJECT P, WORKS_ON W, EMPLOYEE E
    -> WHERE P.Pnumber = W.Pno AND W.Essn = E.Ssn;
Query OK, 16 rows affected (0.02 sec)
Records: 16  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM WORKS_ON_INFO;
+----------+-----------------+----------------+
| Emp_name | Proj_name       | Hours_per_week |
+----------+-----------------+----------------+
| Wong     | Computerization |           10.0 |
| Jabbar   | Computerization |           35.0 |
| Zelaya   | Computerization |           10.0 |
| Wallace  | Newbenefits     |           20.0 |
| Jabbar   | Newbenefits     |            5.0 |
| Zelaya   | Newbenefits     |           30.0 |
| Smith    | ProductX        |           32.5 |
| English  | ProductX        |           20.0 |
| Smith    | ProductY        |            7.5 |
| Wong     | ProductY        |           10.0 |
| English  | ProductY        |           20.0 |
| Wong     | ProductZ        |           10.0 |
| Narayan  | ProductZ        |           40.0 |
| Wong     | Reorganization  |           10.0 |
| Borg     | Reorganization  |           16.0 |
| Wallace  | Reorganization  |           15.0 |
+----------+-----------------+----------------+
16 rows in set (0.00 sec)
```

BULK LOADING OF TABLES

Another variation of **INSERT** is used for bulk-loading of several tuples into tables

A new table TNEW can be created with the same attributes as T and using LIKE and DATA in the syntax, it can be loaded with entire data.

EXAMPLE:

```
CREATE TABLE D5EMPS  LIKE  EMPLOYEE
                    (SELECT   E.*
                     FROM       EMPLOYEE AS E
                    WHERE     E.Dno=5)
 WITH DATA;
```

WITH DATA specifies that the table will be created & loaded with the data specified in the query

DELETE

Removes tuples from a relation

- Includes a WHERE-clause to select the tuples to be deleted
- Referential integrity should be enforced
- Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
- A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause

The DELETE Command

Removes tuples from a relation

Includes a `WHERE` clause to select the tuples to be deleted. The number of tuples deleted will vary.

```
U4A:  DELETE FROM  EMPLOYEE
      WHERE        Lname='Brown';

U4B:  DELETE FROM  EMPLOYEE
      WHERE        Ssn='123456789';

U4C:  DELETE FROM  EMPLOYEE
      WHERE        Dno=5;

U4D:  DELETE FROM  EMPLOYEE;
```

UPDATE

Used to modify attribute values of one or more selected tuples

A WHERE-clause selects the tuples to be modified

An additional SET-clause specifies the attributes to be modified and their new values

Each command modifies tuples *in the same relation*

Referential integrity specified as part of DDL specification is enforced

UPDATE (contd.)

Example: Change the location and controlling department number of project number 10 to 'Bellaire' and 5, respectively

```
U5: UPDATE      PROJECT
    SET         PLOCATION = 'Bellaire', DNUM = 5
    WHERE       PNUMBER=10
```

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

```
mysql> UPDATE PROJECT
    -> SET PLOCATION = 'Bellaire', DNUM = 5
    -> WHERE PNUMBER = 10;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM PROJECT;
+-----------------+---------+-----------+------+
| Pname           | Pnumber | Plocation | Dnum |
+-----------------+---------+-----------+------+
| ProductX        |       1 | Bellaire  |    5 |
| ProductY        |       2 | Sugarland |    5 |
| ProductZ        |       3 | Houston   |    5 |
| Computerization |      10 | Bellaire  |    5 |
| Reorganization  |      20 | Houston   |    1 |
| Newbenefits     |      30 | Stafford  |    4 |
+-----------------+---------+-----------+------+
6 rows in set (0.00 sec)
```

UPDATE (contd.)

Example: Give all employees in the 'Research' department a 10% raise in salary.

UPDATE (contd.)

Example: Give all employees in the 'Research' department a 10% raise in salary.

```
U6: UPDATE          EMPLOYEE
    SET             SALARY = SALARY *1.1
    WHERE DNO  IN (SELECT        DNUMBER
                   FROM          DEPARTMENT
                   WHERE         DNAME='Research')
```

In this request, the modified SALARY value depends on the original SALARY value in each tuple

- The reference to the SALARY attribute on the right of = refers to the old SALARY value before modification
- The reference to the SALARY attribute on the left of = refers to the new SALARY value after modification

```
mysql> UPDATE EMPLOYEE
    -> SET SALARY = SALARY * 1.1
    -> WHERE DNO IN (
    ->     SELECT DNUMBER
    ->     FROM DEPARTMENT
    ->     WHERE DNAME = 'Research'
    -> );
Query OK, 4 rows affected (0.02 sec)
Rows matched: 4  Changed: 4  Warnings: 0

mysql> SELECT * FROM EMPLOYEE;
+----------+-------+---------+-----------+------------+--------------------------+------+--------+-----------+-----+
| Fname    | Minit | Lname   | Ssn       | Bdate      | Address                  | Sex  | Salary | Super_ssn | Dno |
+----------+-------+---------+-----------+------------+--------------------------+------+--------+-----------+-----+
| John     | B     | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston TX  | M    |  33000 | 333445555 |   5 |
| Franklin | T     | Wong    | 333445555 | 1965-12-08 | 638 Voss, Houston TX     | M    |  44000 | 888665555 |   5 |
| Joyce    | A     | English | 453453453 | 1972-07-31 | 5631 Rice, Houston TX    | F    |  27500 | 333445555 |   5 |
| Richard  | K     | Marini  | 653298653 | 1962-12-30 | 98 Oak Forest, Katy, TX  | M    |  37000 | 653298653 |   4 |
| Ramesh   | K     | Narayan | 666884444 | 1962-09-15 | 975 Fire Oak, Humble TX  | M    |  41800 | 333445555 |   5 |
| James    | E     | Borg    | 888665555 | 1937-11-10 | 450 Stone, Houston TX    | M    |  55000 | NULL      |   1 |
| Jennifer | S     | Wallace | 987654321 | 1941-06-20 | 291 Berry, Bellaire TX   | F    |  43000 | 888665555 |   4 |
| Ahmad    | V     | Jabbar  | 987987987 | 1969-03-29 | 980 Dallas, Houston TX   | M    |  25000 | 987654321 |   4 |
| Alicia   | J     | Zelaya  | 999887777 | 1968-01-19 | 3321 Castle, Spring TX   | F    |  25000 | 987654321 |   4 |
+----------+-------+---------+-----------+------------+--------------------------+------+--------+-----------+-----+
9 rows in set (0.00 sec)
```

Specifying Joined Tables in the FROM Clause of SQL

Joined table

Permits users to specify a table resulting from a join operation in the FROM clause of a query

The FROM clause in Q1A

Contains a single joined table. JOIN may also be called INNER JOIN

```
SELECT fname, lname, address
FROM employee, department
WHERE dno = dnumber
  AND dname = 'Research';
```

Specifying Joined Tables in the FROM Clause of SQL

Joined table

Permits users to specify a table resulting from a join operation in the FROM clause of a query

The FROM clause in Q1A

Contains a single joined table. JOIN may also be called INNER JOIN

```
SELECT fname, lname, address
FROM employee, department
WHERE dno = dnumber
 AND dname = 'Research';
```

```
Select fname, lname, address
from (employee join department
on dno=dnumber) where
dname='research';
```

Specifying Joined Tables in the FROM Clause of SQL

Joined table

Permits users to specify a table resulting from a join operation in the FROM clause of a query

The FROM clause in Q1A

Contains a single joined table. JOIN may also be called INNER JOIN

Select fname, lname, address
from (employee join department
on dno=dnumber) where
dname='research';

```
mysql> Select fname, lname, address from (employee
join department on dno=dnumber) where dname='resear
ch';
+----------+---------+-------------------------+
| fname    | lname   | address                 |
+----------+---------+-------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX    |
| Joyce    | English | 5631 Rice, Houston TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
+----------+---------+-------------------------+
4 rows in set (0.04 sec)
```

Different Types of JOINed Tables in SQL

Specify different types of join

- NATURAL JOIN
- Various types of OUTER JOIN (LEFT, RIGHT, FULL)

NATURAL JOIN on two relations R and S

- Automatically joins two tables based on all columns that have the same name in both tables.
- It eliminates duplicate columns in the result. So, you don't have to specify the join condition manually — SQL does it for you based on matching column names.
- The columns must be the same data type

EMPLOYEE

emp_id	fname	lname	dno
1	John	Doe	10
2	Alice	Lee	20
3	Bob	Kim	10

DEPARTMENT

dno	dname	location
10	Research	Delhi
20	HR	Mumbai
30	Finance	Chennai

```
SELECT fname, lname, dname, location
FROM employee NATURAL JOIN department;
```

EMPLOYEE

emp_id	fname	lname	dno
1	John	Doe	10
2	Alice	Lee	20
3	Bob	Kim	10

DEPARTMENT

dno	dname	location
10	Research	Delhi
20	HR	Mumbai
30	Finance	Chennai

```
SELECT fname, lname, dname, location
FROM employee NATURAL JOIN department;
```

fname	lname	dname	location
John	Doe	Research	Delhi
Bob	Kim	Research	Delhi
Alice	Lee	HR	Mumbai

NATURAL JOIN

```
mysql> select Fname, Lname, Address FROM (EMPLOYEE NATURAL JOIN DEPARTMEN
T) WHERE Dname='Research';
```

NATURAL JOIN

```
mysql> select Fname, Lname, Address FROM (EMPLOYEE NATURAL JOIN DEPARTMEN
T) WHERE Dname='Research';
+----------+---------+-------------------------+
| Fname    | Lname   | Address                 |
+----------+---------+-------------------------+
| John     | Smith   | 731 Fondren, Houston TX |
| Franklin | Wong    | 638 Voss, Houston TX    |
| Joyce    | English | 5631 Rice, Houston TX   |
| Ramesh   | Narayan | 975 Fire Oak, Humble TX |
| James    | Borg    | 450 Stone, Houston TX   |
| Jennifer | Wallace | 291 Berry, Bellaire TX  |
| Ahmad    | Jabbar  | 980 Dallas, Houston TX  |
| Alicia   | Zelaya  | 3321 Castle, Spring TX  |
+----------+---------+-------------------------+
8 rows in set (0.01 sec)
```

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

- 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

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

Natural join & Inner join difference

S.No.	NATURAL JOIN	INNER JOIN
1.	Natural join is a join operation that merges two tables based on matching column names and data types.	Inner join operates with a specific join condition, forming a new table by pairing column values of two tables according to the join-predicate.
2.	The final table resulting from a natural join will contain all the attributes of both the tables without duplicating the column.	The final table resulting from an inner join will contain all the attributes of both the tables, including duplicate columns.

```sql
SELECT *
FROM company
INNER JOIN foods
ON company.company_id = foods.company_id;
```

Output:

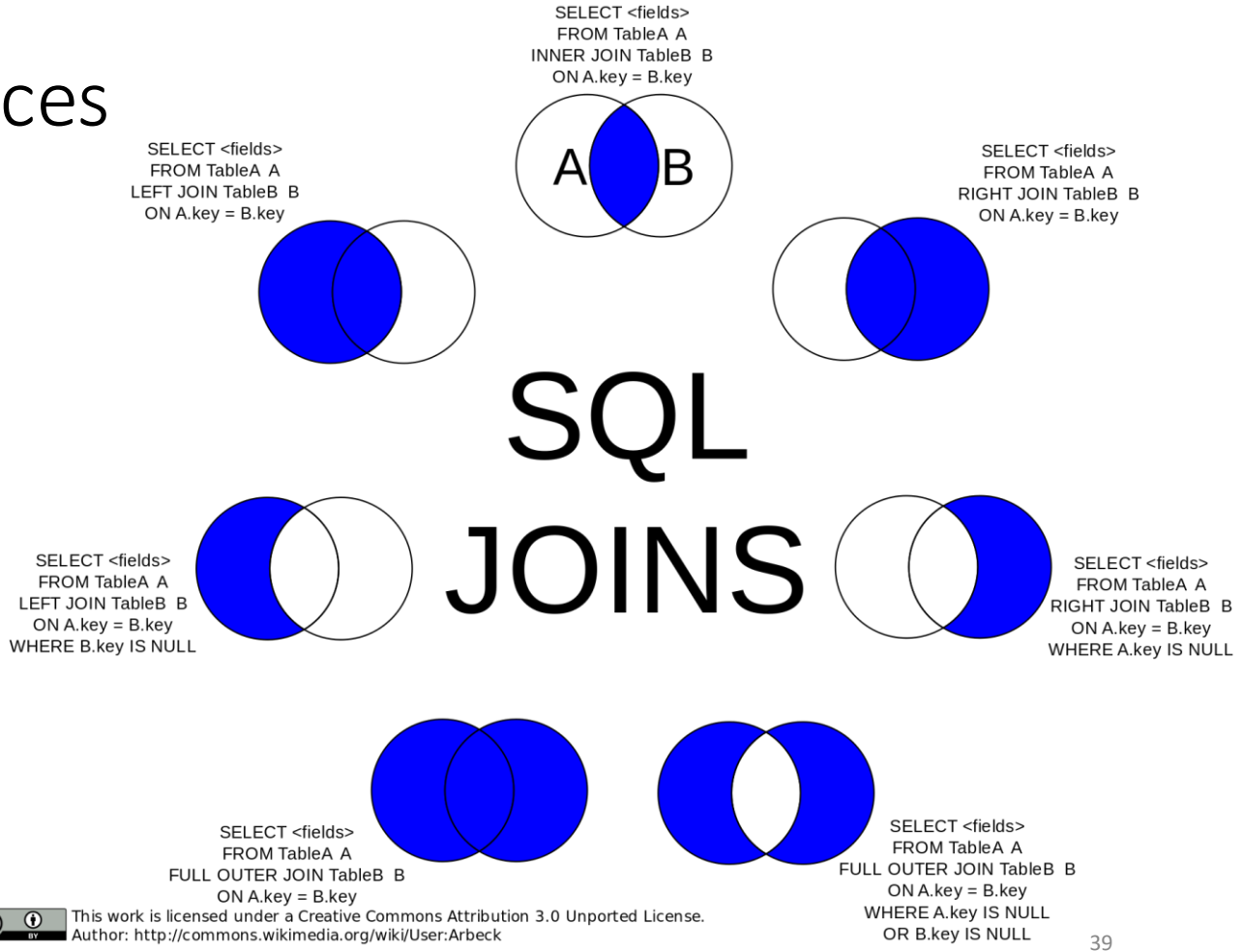
COMPANY_ID	COMPANY_NAME	COMPANY_CITY	ITEM_ID	ITEM_NAME	ITEM_UNIT	COMPANY_ID
16	Akas Foods	Delhi	1	Chex Mix	Pcs	16
15	Jack Hill Ltd	London	6	Cheez-It	Pcs	15
15	Jack Hill Ltd	London	2	BN Biscuit	Pcs	15
17	Foodies.	London	3	Mighty Munch	Pcs	17
15	Jack Hill Ltd	London	4	Pot Rice	Pcs	15
18	Order All	Boston	5	Jaffa Cakes	Pcs	18

```sql
SELECT *
FROM company
NATURAL JOIN foods;
```

Output:

COMPANY_ID	COMPANY_NAME	COMPANY_CITY	ITEM_ID	ITEM_NAME	ITEM_UNIT
16	Akas Foods	Delhi	1	Chex Mix	Pcs
15	Jack Hill Ltd	London	6	Cheez-It	Pcs
15	Jack Hill Ltd	London	2	BN Biscuit	Pcs
17	Foodies.	London	3	Mighty Munch	Pcs
15	Jack Hill Ltd	London	4	Pot Rice	Pcs
18	Order All	Boston	5	Jaffa Cakes	Pcs

Joins differences



https://i.stack.imgur.com/3bs7C.png

Multiway JOIN in the FROM clause

Can nest JOIN specifications for a multiway join:

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

Multiway JOIN in the FROM clause

Can nest JOIN specifications for a multiway join:

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

```
mysql> SELECT Pnumber, Dnum, Lname, Address, Bdate FROM ((PROJECT JOIN DE
PARTMENT ON Dnum=Dnumber)  JOIN EMPLOYEE ON Mgr_ssn=Ssn) WHERE   Plocatio
n='Stafford';
+---------+------+---------+-----------------------+------------+
| Pnumber | Dnum | Lname   | Address               | Bdate      |
+---------+------+---------+-----------------------+------------+
|      10 |    4 | Wallace | 291 Berry, Bellaire TX | 1941-06-20 |
|      30 |    4 | Wallace | 291 Berry, Bellaire TX | 1941-06-20 |
+---------+------+---------+-----------------------+------------+
2 rows in set (0.02 sec)
```

Activity

Try all the queries all yourself if you have not tried it

Write 3 join statements using any of the Company DB
Submit the query and results

Try the Multiway JOIN statement with any of the Company DB

Submit the query and results

Bibliography / Acknowledgements

Instructor materials from Elmasri & Navathe 7e


pk.profgiri


Ponnurangam.kumaraguru


/in/ponguru


ponguru

Thank you for attending the class!!!


pk.guru@iiit.ac.in