Database Management Systems

LECTURE 9

Relational algebra

IITU, ALMATY, 2019

Link to the Video

» https://youtu.be/kOIQHNZl2vk

Querying Data From Tables

- Query operations facilitate data retrieval from one or more tables.
- The result of any query is a table. The result can be further manipulated by other query operations.
- Syntax:

```
SELECT attribute(s)
FROM table(s)
WHERE selection condition(s);
```

Review of last lecture

The operations of Relational algebra:

- projection
- selection

- union
- difference
- intersection

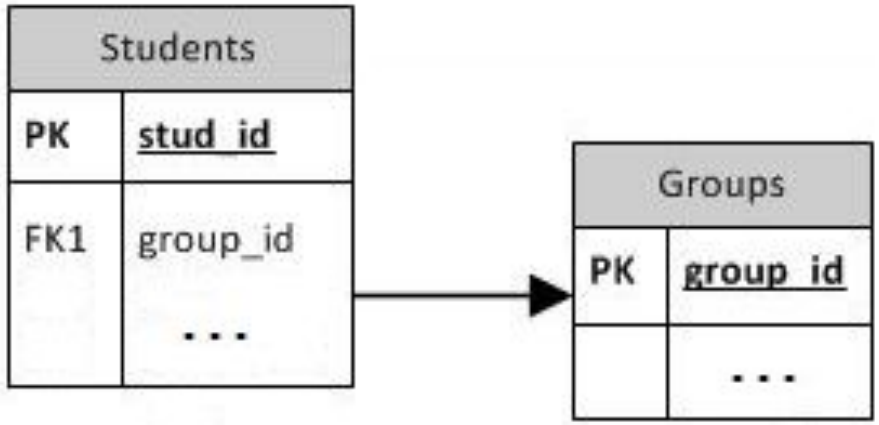
Join

- The **join** operation enables querying information from two or more related tables.
- It is similar to a selection condition except that values in two different tables are compared.
- The most common form of a join is an equi-join. An equi-join combines two or more tables based on the tables' primary and foreign keys.

Join: example 1

```
CREATE TABLE Groups (
group_id int PRIMARY KEY,
group_name varchar(15));

CREATE TABLE Students (
stud_id int PRIMARY KEY,
first_name varchar(20),
last_name varchar(20),
bdate date,
group_id int REFERENCES Groups(group_id));
```



Join: example 1

stud_id	*last_name*	*group_name*
…	*…*	*…*
…	*…*	*…*

```
SELECT stud_id, last_name, group_name
FROM Students, Groups
WHERE
    Students.group_id = Groups.group_id;
```

table.column format

- The table.column format used in the above selection condition.
- This syntax is used to resolve naming conflicts if fields in the tables have the same name.
- This syntax may be used in the SELECT clause or WHERE clause.

Join: example 2

```
CREATE TABLE Account (
   id int PRIMARY KEY,
   balance int);
CREATE TABLE Customer (
   id int PRIMARY KEY,
   name varchar (20),
   accountid int REFERENCES Account (id));
```

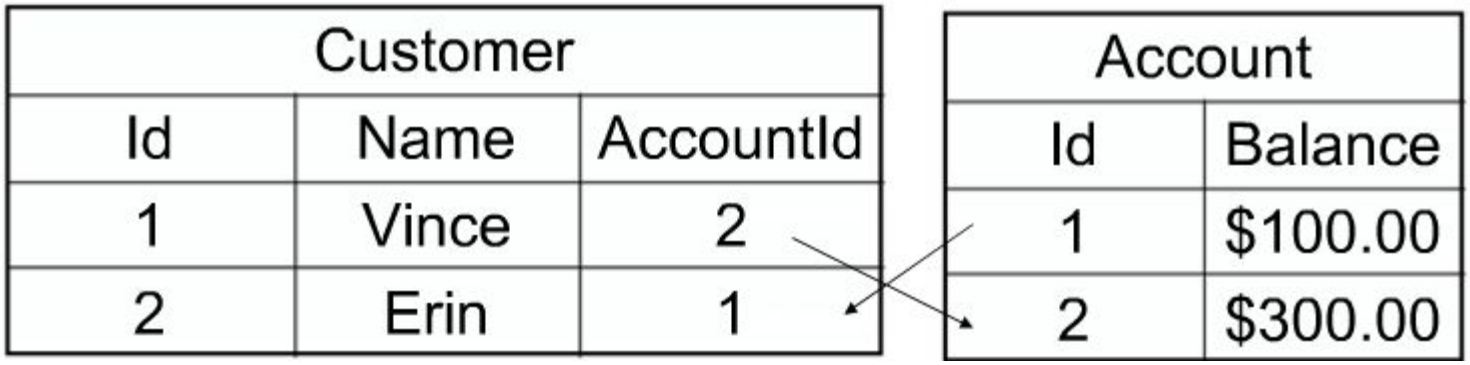
Customer		
Id	Name	AccountId
1	Vince	2
2	Erin	1

Account	
Id	Balance
1	100
2	300

Join: example 2

- Suppose we want to query the name of the Customer who has Balance = 100$.

- We can do this by joining the Account and Customer tables where they are equal – where the FK of Customer (AccountId) is equal to the PK of the Account (Id).



Customer		
Id	Name	AccountId
1	Vince	2
2	Erin	1

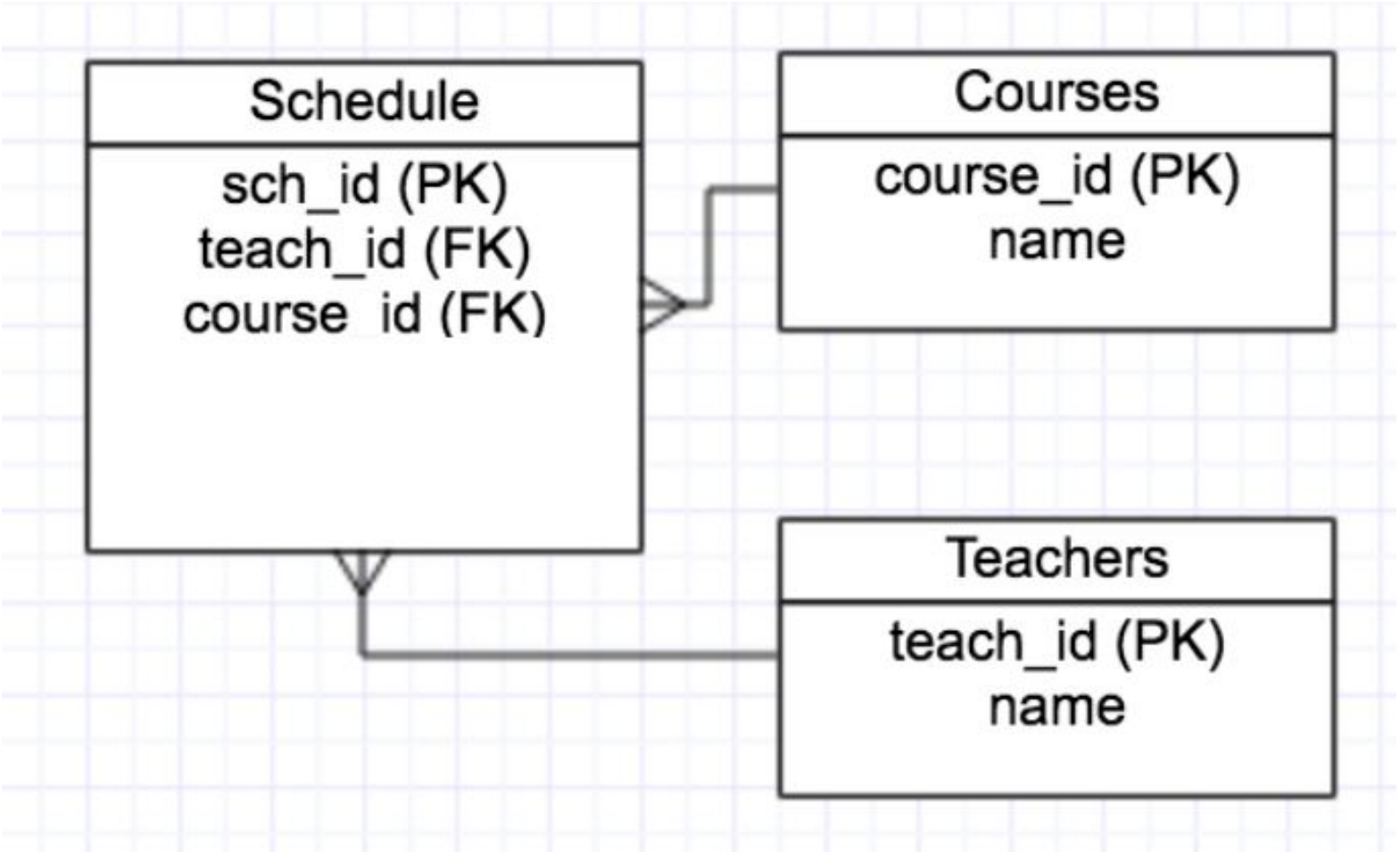
Account	
Id	Balance
1	$100.00
2	$300.00

Join: example 2

- The SQL query is:

```
SELECT name
FROM Customer, Account
WHERE
Customer.accountid = Account.id
AND Account.Balance =100;
```

Join: example 3



Join: example 3

```
CREATE TABLE Courses (
    course_id int PRIMARY KEY,
    name varchar(30));
CREATE TABLE Teachers (
    teach_id int PRIMARY KEY,
    name varchar (30));
 CREATE TABLE Schedule (
    sch_id int PRIMARY KEY,
    course_id int REFERENCES Courses (course_id),
    teach_id int REFERENCES Teachers (teach_id));
```

Join: example 3

course_name	*teach_name*
…	*…*
…	*…*

```
SELECT Courses.name, Teachers.name
FROM Courses, Teachers, Schedule
WHERE
Courses.course_id = Schedule.course_id
AND
Teachers.teach_id = Schedule.teach_id;
```

JOIN keyword

An SQL JOIN clause is used to combine rows from two or more tables.

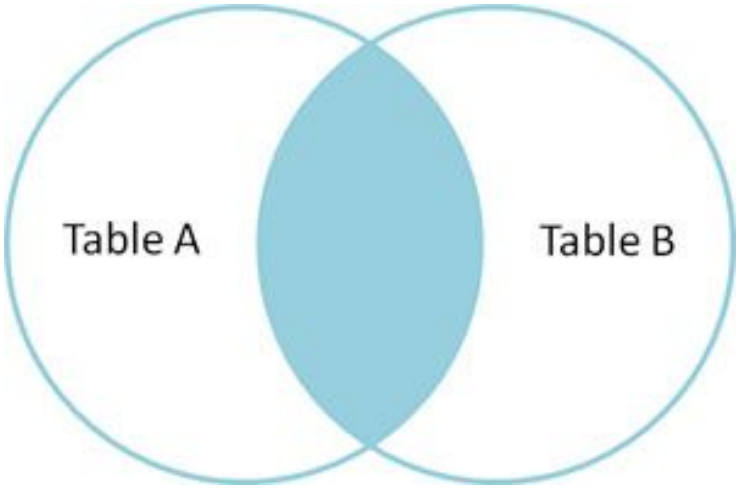
Types:

- INNER JOIN
- OUTER JOIN
  - LEFT JOIN
  - RIGHT JOIN
  - FULL JOIN
- CROSS JOIN

INNER JOIN

The most common type of join is SQL INNER JOIN (simple join).

An SQL INNER JOIN return all rows from multiple tables where the join condition is met.



Syntax:

```
SELECT column_name(s)
FROM tableA
INNER JOIN tableB
ON tableA.column_name = tableB.column_name;
```

INNER JOIN is the same as JOIN.

INNER JOIN: example

```
SELECT Students.stud_id, Students.fname,
    Groups.group_name
FROM Students
INNER JOIN Groups
ON Students.group_id = Groups.group_id;
```

The following example is equivalent to the previous one:

```
SELECT Students.stud_id, Students.fname,
    Groups.group_name
FROM Students, Groups
WHERE Students.group_id = Groups.group_id;
```

INNER JOIN: example

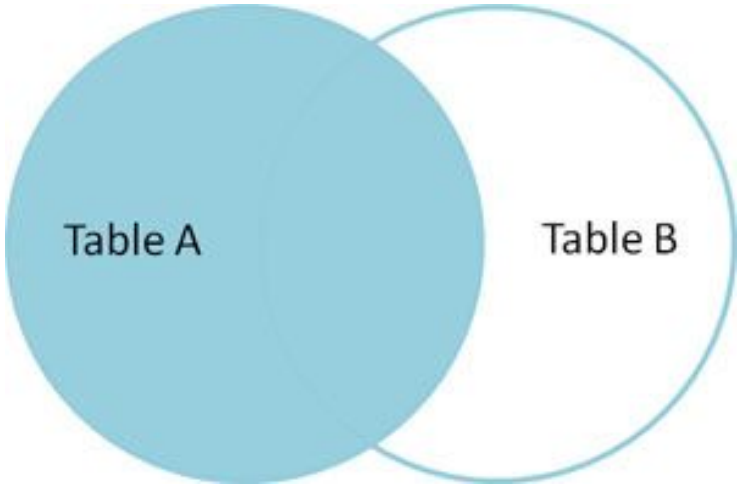
Students		
stud_id	***fname***	***group_id***
1	*Boris*	*2*
2	*Beksultan*	*2*
3	*Aynur*	

Groups	
group_id	***group_name***
1	*CSSE-122*
2	*CSSE-124*

Result table for INNER JOIN		
stud_id	***fname***	***group_name***
1	*Boris*	*CSSE-124*
2	*Beksultan*	*CSSE-124*

LEFT JOIN

The LEFT JOIN keyword returns all rows from the left table (tableA), with the matching rows in the right table (tableB). The result is NULL in the right side when there is no match.



Syntax:

```
SELECT column_name(s)
FROM tableA
LEFT JOIN tableB
ON tableA.column_name = tableB.column_name;
```

In some databases LEFT JOIN is used only like LEFT OUTER JOIN.

LEFT JOIN: example

The following SQL statement will return all students, and group they might have:

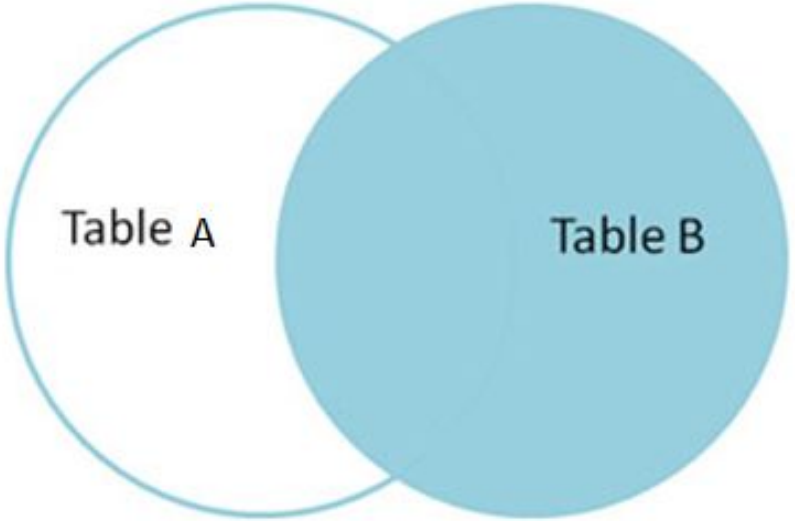
```
SELECT Students.stud_id, Students.fname,
Groups.group_name
FROM Students
LEFT JOIN Groups
ON Students.group_id = Groups.group_id;
```

The LEFT JOIN keyword returns all the rows from the left table (Students), even if there are no matches in the right table (Groups):

Result table for LEFT JOIN		
stud_id	***fname***	***group_name***
1	***Boris***	***CSSE-124***
2	***Beksultan***	***CSSE-124***
3	***Aynur***	

RIGHT JOIN

The RIGHT JOIN keyword returns all rows from the right table (tableB), with the matching rows in the left table (tableA). The result is NULL in the left side when there is no match.



Syntax:

```
SELECT column_name(s)
FROM tableA
RIGHT JOIN tableB
ON tableA.column_name=tableB.column_name;
```

In some databases RIGHT JOIN is used only like RIGHT OUTER JOIN.

RIGHT JOIN: example

The following SQL statement will return all groups, and students they might have:

```
SELECT Students.stud_id, Students.fname,
Groups.group_name
FROM Students
RIGHT JOIN Groups
ON Students.group_id = Groups.group_id;
```

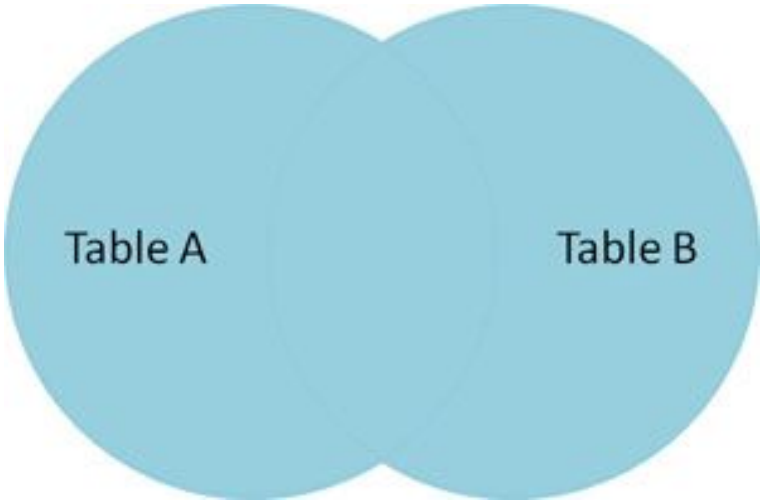
The RIGHT JOIN keyword returns all the rows from the right table (Groups), even if there are no matches in the left table (Students):

Result table for RIGHT JOIN		
stud_id	***fname***	***group_name***
1	***Boris***	***CSSE-124***
2	***Beksultan***	***CSSE-124***
		CSSE-122

FULL OUTER JOIN

The FULL OUTER JOIN keyword returns all rows from the left table (tableA) and from the right table (tableB).

The FULL OUTER JOIN keyword combines the result of both LEFT and RIGHT joins.



Syntax:

```
SELECT column_name(s)
FROM tableA
FULL OUTER JOIN tableB
ON tableA.column_name=tableB.column_name;
```

FULL JOIN: example

The following SQL statement selects all students and all groups:

```
SELECT Students.stud_id, Students.fname, Groups.group_name
FROM Students
FULL OUTER JOIN Groups
ON Students.group_id = Groups.group_id;
```

The FULL OUTER JOIN keyword returns all the rows from the left table (Students) and all the rows from the right table (Groups).
If there are rows in "Students" that do not have matches in “Groups", or if there are rows in “Groups" that do not have matches in “Students", those rows will be listed as well:

Result table for FULL OUTER JOIN

stud_id	*fname*	*group_name*
1	*Boris*	*CSSE-124*
2	*Beksultan*	*CSSE-124*
3	*Aynur*	
		CSSE-122

CROSS JOIN

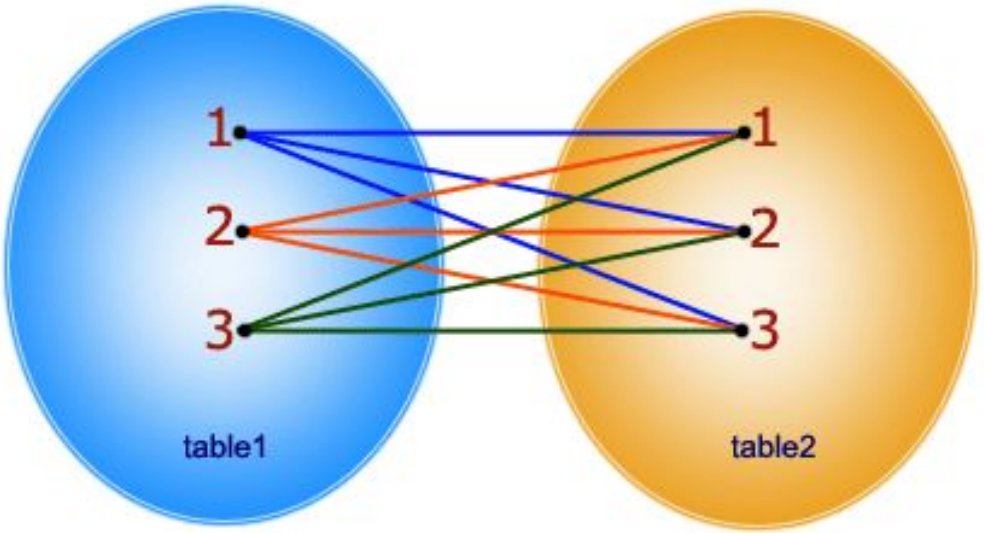
“All-to-All”. The SQL CROSS JOIN produces a result set which is the number of rows in the first table multiplied by the number of rows in the second table. WHERE clause is not used along with CROSS JOIN. This kind of result is called as **Cartesian Product**.

```
SELECT *
FROM  tableA
CROSS JOIN tableB;
```

or

```
SELECT *
FROM tableA, tableB
```

SELECT * FROM table1 CROSS JOIN table2;



In CROSS JOIN, each row from 1st table joins with all the rows of another table. If 1st table contain x rows and y rows in 2nd one the result set will be x * y rows.

CROSS JOIN: example

```
SELECT *
FROM  Students
CROSS JOIN Groups;
```

or

```
SELECT *
FROM Students, Groups;
```

CROSS JOIN: example

Result table for CROSS JOIN				
stud_id	***fname***	***group_id***	***group_id***	***Group_name***
1	***Boris***	***2***	***1***	***CSSE-122***
2	***Beksultan***	***2***	***1***	***CSSE-122***
3	***Aynur***		***1***	***CSSE-122***
1	***Boris***	***2***	***2***	***CSSE-124***
2	***Beksultan***	***2***	***2***	***CSSE-124***
3	***Aynur***		***2***	***CSSE-124***

The complete JOIN syntax

```
SELECT Attribute(s)
FROM TableA
{INNER | {LEFT | RIGHT | FULL}
OUTER | CROSS } JOIN TableB
ON <condition>
```

JOIN with USING

The USING clause is a shorthand that allows you to take advantage of the specific situation where both sides of the join use the same name for the joining column(s). It takes a comma-separated list of the shared column names and forms a join condition that includes an equality comparison for each one.

```
SELECT Attribute(s)
FROM TableA
{INNER | {LEFT | RIGHT | FULL} OUTER } JOIN TableB
USING (join column list)
```

JOIN with USING: example

```
SELECT *
FROM Students
INNER JOIN Groups
USING (group_id);
```

The output of JOIN USING suppresses redundant columns: there is no need to print both of the matched columns, since they must have equal values.

NATURAL JOIN

NATURAL is a shorthand form of USING: it forms a USING list consisting of all column names that appear in both input tables. As with USING, these columns appear only once in the output table.

```
SELECT Attribute(s)
FROM TableA
NATURAL
{INNER | {LEFT | RIGHT | FULL}
OUTER } JOIN TableB
```

NATURAL JOIN: example

```
SELECT *
FROM Students
NATURAL INNER JOIN Groups;
```

Notation

The operations have their own symbols.

Operation	*Symbol*
Union	$\cup$
Intersection	$\cap$
Set difference	-

Operation	*Symbol*
Projection	π
Selection	σ
Cartesian product	$\times$
Join	$\bowtie$
Left outer join	$⟕$
Right outer join	$⟖$
Full outer join	$⟗$

Books

- Connolly, Thomas M. Database Systems: A Practical Approach to Design, Implementation, and Management / Thomas M. Connolly, Carolyn E. Begg.- Fifth.- United States of America: Pearson Education, 2010
- Garcia-Molina, H. Database system: The Complete Book / Hector Garcia-Molina.- 2.- United States of America: Pearson Prentice Hall, 2009
- Sharma, N. Database Fundamentals: A book for the community by the community / Neeraj Sharma, Liviu Perniu.- First Edition.- Canada, 2010

- www.postgresql.org/docs/manuals/
- www.postgresql.org/docs/books/

Question

The SQL statement that queries or reads data from a table is _____ .

a) SELECT
b) READ
c) QUERY
d) None of these

Question

The result of a SQL SELECT statement is a(n) _____ .

a) Report
b) Form
c) File
d) Table

Question

Which of the following is the correct order of keywords for SQL SELECT statements?

a) SELECT, FROM, WHERE
b) FROM, WHERE, SELECT
c) WHERE, FROM,SELECT
d) SELECT,WHERE,FROM

Question

In an SQL SELECT statement querying a single table, the asterisk (*) means that:

a) all columns of the table are to be returned.
b) all records meeting the full criteria are to be returned.
c) all records with even partial criteria met are to be returned.
d) None of the above is correct.

Question

Which of the following SQL clauses specifies a search condition?

a) WHERE
b) SEARCH
c) WHILE
d) FROM

Question

Which of the following is used to denote the selection operation in relational algebra ?

a) Pi (Greek)
b) Sigma (Greek)
c) Lambda (Greek)
d) Omega (Greek)

Question

Which product is returned in a join query have no join condition:

a) Equijoins
b) Cartesian
c) Both
d) None