Database Management Systems

LECTURE 10

Queries

IITU, ALMATY, 2019

Link to the Video

- https://youtu.be/i9uxOf5hddg

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);
```

Aliasing in SQL (1)

- Return the first name and the last name of student who has stud_id = 15.

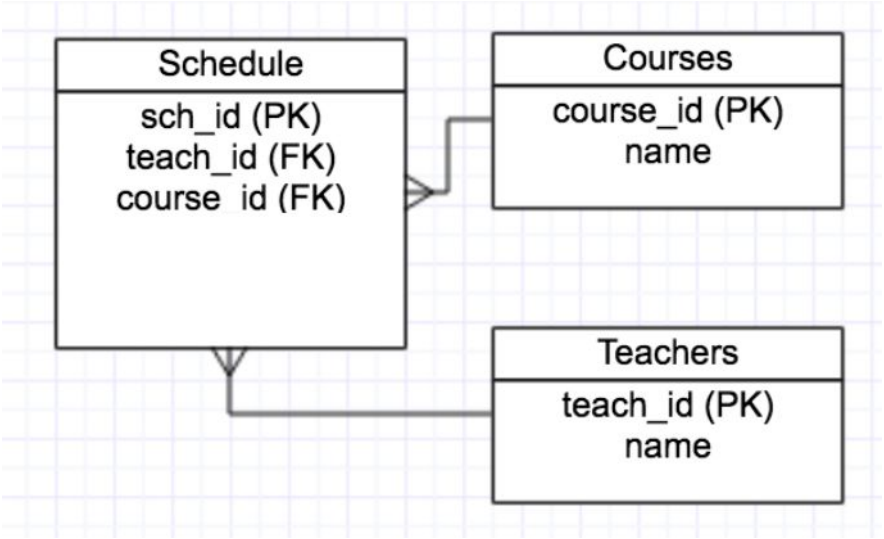
```
SELECT fname, lname
FROM Students s
WHERE s.stud_id=15;
```

- In this query, we rename the Students table to s.

Aliasing in SQL (1)

- Aliasing table names during Join operations makes them a lot more understandable.

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



Aliasing in SQL (2)

- Column names can be aliased to another in SQL using the AS operator.

- Example: Rename the fname column to First_Name in the following select statement:

```
SELECT fname AS First_name
FROM Students;
```

	first_name character varying(20)
1	John
2	Alis

String Concatenation

- In the Students table first and last names are stored as two attributes. For combining them into one column, use the '||' operator:

```
SELECT fname || lname
FROM Students;
```

	?column? text
1	JohnSmith
2	AlisBlack

- Notice that the names concatenated together without a space in between. We can add such a space using:

```
SELECT fname || ' ' || lname AS Full_name
FROM Students;
```

	full_name text
1	John Smith
2	Alis Black

Distinct Results

- Explicitly filtering of duplicates requires the DISTINCT keyword.
- Example: To select the distinct last names from the Students table we would write:

```
SELECT DISTINCT fname
FROM Students;
```

  - Instead of DISTINCT the key word ALL can be used to specify the default behavior of retaining all rows.

Distinct Results

If you specify multiple columns, the DISTINCT clause will evaluate the duplicate based on the combination of values of these columns.

```
SELECT DISTINCT column_1, column_2
FROM table_name;
```

In this case, the combination of both column_1 and column_2 will be used for evaluating duplicate.

NULL Values

- NULL indicates absence of a value in a column. It's a special value that is valid for all domains.

- Since NULL may appear in a column, we must be able to detect its presence.
- For this reason, SQL provides the IS NULL and IS NOT NULL operators.

NULL Values

- Consider the following query:

```
SELECT stud_id, fname
FROM Students
WHERE group_id IS NULL;
```

- This query returns record of each student where the group_id is null (is empty).

IS NULL and IS NOT NULL

Students table in the database

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

… WHERE group_id IS NULL;

stud_id	*fname*
3	*Aynur*

… WHERE group_id IS NOT NULL;

stud_id	*fname*
1	*Boris*
2	*Beksultan*

Comparison Operators

- One of the most common selection conditions is a range condition. **R**ange condition filters results where the values in a column are between one or two values.

- There are two ways to perform a range operation:
  - Using the <, <=, >, >= operators.
  - Using the BETWEEN operator.

Comparison Operators

Operator	***Description***
<	less than
>	***greater than***
<=	less than or equal to
>=	***greater than or equal to***
=	equal
<> *or* *!=*	***not equal***

- Comparison operators are available for all relevant data types.
- All comparison operators are binary operators that return values of type boolean
- expressions like 1 < 2 < 3 are not valid (because there is no < operator to compare a Boolean value with 3).

Comparison Operators

- A range condition is specified using the <,<=,> and >= operators as

```
SELECT …
FROM …
WHERE attribute<value1 AND attribute>value2;
```

- Example: Query the first and last names of all students with GPA between 3.0 and 4.0:

```
SELECT fname, lname
FROM Students
WHERE gpa >= 3.0 AND gpa <= 4.0;
```

BETWEEN operator

- We may render the same select condition in a form that is closer to English using the BETWEEN operator.

- The query on the previous slide can be rewritten as

```
SELECT fname, lname
FROM Students
WHERE gpa BETWEEN 3.0 AND 4.0;
```

Comparison Operators

- The BETWEEN operator has a negation: NOT BETWEEN.
- The BETWEEN operator is defined for most data types including numeric and temporal data.

BETWEEN and NOT BETWEEN

BETWEEN treats the endpoint values as included in the range. NOT BETWEEN does the opposite comparison.

a BETWEEN x AND y
is equivalent to

a >= x AND a <= y

a NOT BETWEEN x AND y
is equivalent to

a < x OR a > y

Pattern Matching

SQL provides the

- % and _ characters to match strings
- LIKE operator to support comparisons of partial strings.

The LIKE operator is used in conjunction with % and _ characters.

Pattern Matching

- The % character matches an arbitrary number of characters, including spaces.
- So, vinc% would match each of the following:
  vince, vincent, vincenzo, vinc

- The _ character matches a single arbitrary character.
- So, v_nce will match each of the following:
  vince, vance, vbnce, vnnce, v1nce, and so on.

Pattern Matching

- Example with *%*: Query the phone number if it starts with 412.

```
SELECT phone
FROM Contacts
WHERE phone LIKE ‘412%’;
```

Pattern Matching

- Example with _: Query the phone number if it starts with ‘20’ and ends with ‘-555-4335’.

```
SELECT phone
FROM Contacts
WHERE phone LIKE ‘20_-555-4335’;
```

Converting Data Types

- PostgreSQL CAST is used to convert from one data type into another.
- First, you specify an expression that can be a constant, a table column, etc., that you want to convert. Then, you specify the target type which you want to convert to.
- Syntax:

```
CAST (expression AS type)
```

- Example:

```
SELECT CAST ('100' AS INTEGER);
```

Converting Data Types

- Besides the type CAST syntax, following syntax can be used to convert a type into another:

  expression::type

- Notice that the cast syntax with
  :: is PostgreSQL specific and does not conform to SQL.

- Example:

  SELECT '100'::INTEGER;

Books

Connolly, Thomas M. Database Systems: A Practical Approach to Design, Implementation, and Management / Thomas M. Connolly, Carolyn E. Begg.- United States of America: Pearson Education

Garcia-Molina, H. Database system: The Complete Book / Hector Garcia-Molina.- United States of America: Pearson Prentice Hall

Sharma, N. Database Fundamentals: A book for the community by the community / Neeraj Sharma, Liviu Perniu.- Canada

www.postgresql.org/docs/manuals/

www.postgresql.org/docs/books/

Online SQL Training

- sqlzoo.net
- sql-ex.ru

Question

When specifying a selection criterion in SQL, attributes can be renamed with which of the following operators?

a) RENAME
b) AS
c) ALIAS
d) @

Question

In SQL, which of the following operators can be used to express searches that test for a range in a selection condition?

a) RANGE
b) FROM and TO
c) BETWEEN
d) START and END

Question

With SQL, how do you select all the columns from a table named "Persons"?

a) SELECT Persons;
b) SELECT [all] FROM Persons;
c) SELECT *.Persons;
d) SELECT * FROM Persons;

Question

With SQL, how do you select all the records from a table named "Persons" where the value of the column "FirstName" starts with an "a"?

a) SELECT * FROM Persons WHERE FirstName='%a%';
b) SELECT * FROM Persons WHERE FirstName LIKE '%a';
c) SELECT * FROM Persons WHERE FirstName LIKE 'a%';
d) SELECT * FROM Persons WHERE FirstName='a';

Question

Which SQL keyword is used to return only different (unique) values?

a) UNIQUE
b) DIFFERENT
c) *
d) DISTINCT

Question

What is the meaning of LIKE '%0%0%'

a) Feature begins with two 0's
b) Feature ends with two 0's
c) Feature has more than two 0's
d) Feature has two 0's in it, at any position

Question

What is meant by the following relational algebra statement: STUDENT X COURSE

a) Compute the right outer join between the STUDENT and COURSE relations
b) Compute the left outer join between the STUDENT and COURSE relations
c) Compute the cartesian product between the STUDENT and COURSE relations
d) Compute the full outer join between the STUDENT and COURSE relations