Principles of Database Systems

Introduction to SQL(2)

Basic Structure of SQL Queries

First Sight on SELECT

select *name*
from *instructor*
where *dept_name* = 'Comp. Sci.'

- The basic structure of an SQL query consists of three clauses (子句): **select**, **from**, and **where**.

Basic Query Structure

- A typical SQL query has the form:

  select $A_1, A_2, \ldots, A_n$
  from $r_1, r_2, \ldots, r_m$
  where P

  - A_i represents an attribute (属性)
  - R_i represents a relation (关系)
  - P is a predicate (谓词).

- The query takes as its input the relations listed in the **from clause**, operates on them as specified in the **where** and **select clauses**, and then produces a relation as the result

Informal Vocabulary useful in the very beginning

- select *XX* → I want get *XX* attribute.
- from *YY* → I need data from *YY*.
- where *ZZ* → The data needed should satisfy the condition of *ZZ*.

Try…

- Explain the meaning of following statement

  select *name* **from** *instructor*;

- "Find the names of all instructors."

Try…

- Explain the meaning of following statement

select *name*
from *instructor*
where *salary* > 70000;

- "Find the names of all instructors who have salary greater than $70,000."

Try…

- Construct statement for the following query
- “Find the department names of all instructors.”

select *dept_name*
from *instructor*;

Queries on a Single Relation

- The **select clause** list the attributes desired in the result of a query

  Example:

  find the names of all instructors:

  select *name*
  from *instructor*

- An asterisk(星号) in the select clause denotes "all attributes"

  select * **from** instructor

Queries on a Single Relation

- In the formal, mathematical definition of the relational model, a **relation is a set**. Thus, duplicate tuples would never appear in relations. In practice, duplicate elimination is time-consuming. Therefore, **SQL allows duplicates in relations as well as in the results of SQL expressions.**

- Run the following statement and try to understand the result.

select *dept_name* **from** *instructor*;

Queries on a Single Relation

- Insert the keyword **distinct** after **select** to force the elimination of duplicates.

  select distinct *dept_name*
  from *instructor*;

- Try it …

Queries on a Single Relation

- The **select clause** may also contain arithmetic expressions (算数表达式) involving the operators +, −, ∗, and / operating on constants or attributes of tuples. The literal (文字量) can also be contained in **select clause**.

  select *ID*, *name*, *dept_name*, *salary* *1.1
  from *instructor*;

- Note: it does not result in any change to the instructor relation.

Queries on a Single Relation

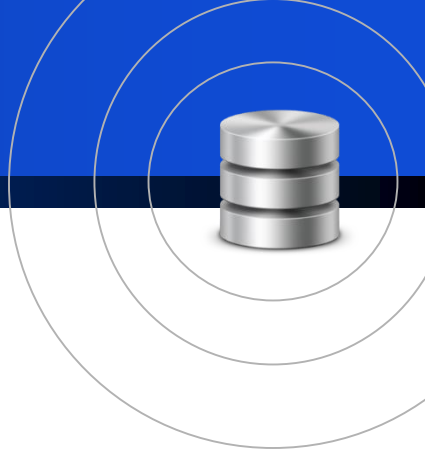
- The **where clause** allows us to select only those rows in the result relation of the from clause that satisfy a specified predicate.

- To find all instructors in Comp. Sci. dept with salary > 80000

```
select name
from instructor
where dept_name = 'Comp. Sci.'
    and salary > 80000
```

- Comparison results (比较结果) can be combined using the logical connectives **and**, **or**, and **not**.

- Comparisons can be applied to results of arithmetic expressions.



Additional Basic Operations

The Rename Operation(更名运算)

- The SQL allows renaming **relations** and **attributes** using the **as clause**(as子句):

  old-name **as** new-name

- E.g.

  select ID, name, salary/12 as monthly_salary
  from instructor

String Operations(字符串运算)

- SQL specifies strings by enclosing them in **single quotes**(单引号), for example, 'Computer'.

- A single quote character that is part of a string can be specified by using two single quote characters; for example, the string "**It's right**" can be specified by "**It''s right**"

String Operations

- **The SQL standard** specifies that the equality operation on strings is **case sensitive**

- SQL also permits a variety of functions on character strings, such as concatenating (using "**||**"), converting strings to uppercase (using **upper(s)**) and lowercase (using the function **lower(s)**) and so on.

String Operations

- SQL includes a string-matching operator for comparisons on character strings. The operator "**like**" uses patterns(模式) that are described using two special characters:
  - percent (**%**). The % character matches any **substring**(子字符串).
  - underscore (_). The _ character matches any **character**(字符).

String Operations

- Try to explain the following patterns:
  - 'Intro%'
  - '%Comp%'
  - '_ _ _ '
  - '_ _ _ %'

String Operations

- **Try:** Find the names of all instructors whose name includes the substring “dar”.

String Operations

- **Try:** Find the names of all instructors whose name includes the substring “dar”.

```
select name
from instructor
where name like '%dar%'
```

String Operations

- Backslash (\) is used as the escape character.

- E.g.
  - **like** 'ab\%cd%' **escape** '\' matches all strings beginning with "ab%cd".

  - **like** 'ab\\cd%' **escape** '\' matches all strings beginning with "ab\cd".

Where Clause Predicates

- SQL includes a between comparison operator
  - Example: Find the names of all instructors with salary between $90,000 and $100,000 (that is, >=$90,000 and <=$100,000)
  - **select** name
    from instructor
    where salary **between** 90000 **and** 100000

- Tuple comparison
  - **select** name, course_id
    from instructor, teaches
    where (instructor.ID, dept_name) = (teaches.ID, 'Biology');
  - SQL server doesn't support this feature

Ordering the Display of Tuples

- The order by clause causes the tuples in the result of a query to appear in sorted order

```
select name
from instructor
where dept_name = 'Physics'
order by name;
```

Ordering the Display of Tuples

- We may specify **desc** for descending order or **asc** for ascending order, for each attribute; ascending order is the default. 默认升序
  - Example: **order by** name **desc**

- Can sort on multiple attributes
  - Example: **order by** dept_name, name