WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CS 564: Database Management Systems

Lecture 2: SQL Basics I

Xiangyao Yu
1/26/2024

Module A1: SQL

SQL: Basics I

- Relational models
- Single-table operations

SQL: Basics II

Advanced SQL I

Advanced SQL II

Outline of this Lecture

Recap of Relational Model

SQL: Basics

- Creating a table

SQL: Single-table queries

- SELECT-FROM-WHERE structure
- DISTINCT/ORDER BY/LIMIT
- Aggregation

Recap of Relational Model

Relation: a table with rows and columns

Product

name	**category**	**price**	**manufacturer**
iPad	tablet	$399.00	Apple
Surface	tablet	$299.00	Microsoft
…	…	…	…

The schema of a relation:

relation name + attribute names

Product (name, price, category, manufacturer)

Primary Key

A primary key is a selected subset of attributes that is a unique identifier of tuples in a relation

Product

name	category	price	manufacturer
iPad	tablet	$399.00	Apple
Surface	tablet	$299.00	Microsoft
…	…	…	…

For example, **Product.name** can be the primary key

There can be only one primary key, but many unique keys

Accessing tuples using primary keys is preferable

Outline

- Recap of Relational Model
- SQL: Basics
  - Creating a table
- SQL: Single-table queries
  - SELECT-FROM-WHERE structure
  - DISTINCT/ORDER BY/LIMIT
  - Aggregation

Structured Query Language (SQL)

Pronounced “sequel” or “S.Q.L.”

The most widely used database language

- Many standards: SQL-92, SQL:1999, SQL:2011, SQL:2016, SQL:2023

Data Definition Language (DDL)

- Creation, deletion, modification of definitions of tables

Data Manipulation Language (DML)

- Query
- Insert, delete, and update rows

SQL: Create a Table

Product

name	category	price	manufacturer
iPad	tablet	$399.00	Apple
Surface	tablet	$299.00	Microsoft
…	…	…	…

table name

attributes

```
CREATE TABLE Product (
    name CHAR(30) PRIMARY KEY,
    category CHAR(20),
    price REAL,
    manufacturer CHAR(20)
);
```

“name” is the primary key

SQL: Create a Table — Unique Keys

```
CREATE TABLE Product (
    pid INTEGER PRIMARY KEY
    name CHAR(30) UNIQUE,
    category CHAR(20),
    price REAL,
    manufacturer CHAR(20)
);
```

“pid” is the primary key
“name” is a unique key

Insert to a Table

To insert a single tuple:

INSERT INTO *<relation>*

VALUES *(<list of values>);*

Product

name	**category**	**price**	**manufacturer**
iPad	tablet	$399.00	Apple
Surface	tablet	$299.00	Microsoft
…	…	…	…

Insert to a Table

To insert a single tuple:

```
    INSERT INTO <relation>
    VALUES ( <list of values>);
```

For example

```
    INSERT INTO Product
    VALUES
          (‘iphone’, ‘phone’, 999.00, ‘Apple’),
          (‘chromebook’, ‘laptop’, xxx, ‘Google’);
```

Product

name	category	price	manufacturer
iPad	tablet	$399.00	Apple
Surface	tablet	$299.00	Microsoft
…	…	…	…

Example Database

sailors(**sid**: integer, **sname**: string, **rating**: integer, **age**: real)

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

$\nwarrow$ Primary Key

Example Database

sailors(**sid**: integer, **sname**: string, **rating**: integer, **age**: real)

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

```
CREATE TABLE sailors(
        sid INTEGER PRIMARY KEY,
        sname CHAR(20),
        rating INTEGER,
        age REAL
);


INSERT INTO sailors VALUES
    (22, 'Dustin', 7, 45.0)
```

Outline

Recap of Relational Model

SQL: Basics

- Creating a table

SQL: Single-table queries

- SELECT-FROM-WHERE structure
- DISTINCT/ORDER BY/LIMIT
- Aggregation

Basic SQL Query

```
SELECT   attributes
FROM     table
WHERE    conditions
```

Basic SQL Query

```
SELECT   attributes
FROM     table
WHERE    conditions
```

Pseudo code for the SELECT-FROM-WHERE structure

```
Foreach row in table
    if conditions are satisfied
        send attributes in table to output
```

Example

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

```
SELECT sname
FROM Sailors
WHERE rating > 7;
```

Output

sname
Lubber
Andy
Rusty
Zorba
Horato

* in Select Clauses

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

When there is one relation in the **FROM** clause, * in the **SELECT** clause stands for *“all attributes of this relation”*

```
SELECT *
FROM Sailors
WHERE rating > 7;
```

Output

sid	sname	rating	age
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
71	Zorba	10	16
74	Horato	9	35

Renaming Attributes

If we want the output schema to have different attribute names, we can use **AS** <*new name*> to rename an attribute

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

```
SELECT sname AS SailorName
FROM Sailors
WHERE rating > 7;
```

Output

SailorName
Lubber
Andy
Rusty
Zorba
Horato

Arithmetic Expressions

We can use arithmetic expression in the **SELECT** clause

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

```
SELECT sname,
          (60 - age) AS YearToRetire
FROM Sailors
WHERE rating > 7;
```

Output

same	YearToRetire
Lubber	5
Andy	35
Rusty	25
Zorba	44
Horato	25

What can we use in WHERE Clauses?

Attribute names of the relations that appear in the **FROM** clause

Comparison operators: =, <>, <, >, <=, >=

Arithmetic operations: (+, -, /, *)

AND, **OR**, **NOT** to combine conditions

Operations on strings (e.g. concatenation)

Pattern matching: *s* **LIKE** *p*

Special functions for comparing dates and times

Pattern Matching

s **`LIKE`** p: pattern matching on strings

- % = any sequence of characters
- _ = any single character

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

```
SELECT *
FROM Sailors
WHERE sname LIKE ‘%st%’;
```

Output

sid	sname	rating	age
22	Dustin	7	45
58	Rusty	10	35

AND, OR, NOT

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

```
SELECT *
FROM Sailors
WHERE sname LIKE ‘%st%’
    OR age > 50;
```

Output

sid	sname	rating	age
22	Dustin	7	45
58	Rusty	10	35
31	Lubber	8	55

Using Distinct

The default semantics of SQL allow duplicate tuples in the output

Use **`DISTINCT`** in the **`SELECT`** clause to removes duplicates

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

```
SELECT DISTINCT rating
FROM Sailors;
```

Output

rating
7
1
8
10
9

Order By

ORDER BY orders the tuples by the attribute we specify in decreasing (**DESC**) or increasing (**ASC**) order

```
SELECT sname, age
FROM Sailors
WHERE age < 35
ORDER BY age DESC;
```

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

Output

sname	age
Brutus	33
Andy	25
Zorba	16

Limit

`LIMIT <`*`number`*`>` limits the output to be a specified number of tuples

Usually used with **`ORDER BY`** to get the **top K** records

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

```
SELECT sname, age
FROM Sailors
ORDER BY age DESC
LIMIT 2;
```

Output

sname	age
Lubber	55
Dustin	45

Aggregation

SUM,AVG,COUNT,MIN,MAX can be applied to a column in a **SELECT** clause to produce that aggregation on the column

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

```
SELECT AVG(age)
FROM Sailors
WHERE rating > 4;
```

Aggregation: Example

Count the number of sailors

```
SELECT COUNT(*)
FROM Sailors S;
```

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

Aggregation: No Duplicates

We can use

`COUNT(DISTINCT <attribute>)`

to remove duplicate tuples before counting!

```
SELECT COUNT (DISTINCT rating)
FROM Sailors;
```

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

Aggregation: Example

Find the age of the oldest sailor

```
SELECT MAX(S.age)
FROM Sailors S;
```

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

Aggregation: Example

Find the name and age of the oldest sailor

sid	sname	rating	age
22	Dustin	7	45
29	Brutus	1	33
31	Lubber	8	55
32	Andy	8	25
58	Rusty	10	35
64	Horatio	7	35
71	Zorba	10	16
74	Horato	9	35

Sailors

HINT: Use the output of one query (i.e., SELECT-FROM-WHERE) within another query

More Jupyter Notebook Examples

Activity-1a.ipynb

Activity-1b.ipynb

Summary

The Relational Model

- Tables, records/tuples/rows, attributes/fields/columns, schema, primary key

SQL: Basics

- DDL, DML, Creating a table, insert

SQL: Single-table queries

- SELECT-FROM-WHERE structure
- In SELECT: * in Select, rename attributes, arithmetic expressions, DISTINCT
- In WHERE: pattern matching, AND/OR/NOT
- ORDER BY, LIMIT
- Aggregation (e.g., SUM, COUNT, AVG, MAX, MIN)

Next lecture

SQL: Basics II