Engineering Databases

Lecture 6 – Sorting, Limits, Triggers, Views, and Transactions

November 30, 2022

M. Saeed Mafipour & Mansour Mehranfar

Content of Lecture 5

- Explicit renaming of output columns $\rho_{new\ name/old\ name}(relation)$

- Aggregate by a column GROUP BY
- Aggregate group columns by functions e.g. COUNT(column)

- Use nested queries in WHERE and SELECT clause
- Use quantifier [NOT] EXISTS to check if a SELECT has content
- Special language elements are helpful e.g. BETWEEN

- Exercise students, professors, rooms and classes
- Exercise bar example

Pitfalls with SQL

- Table or column names containing spaces
  - Use ` (not ‘ or “)

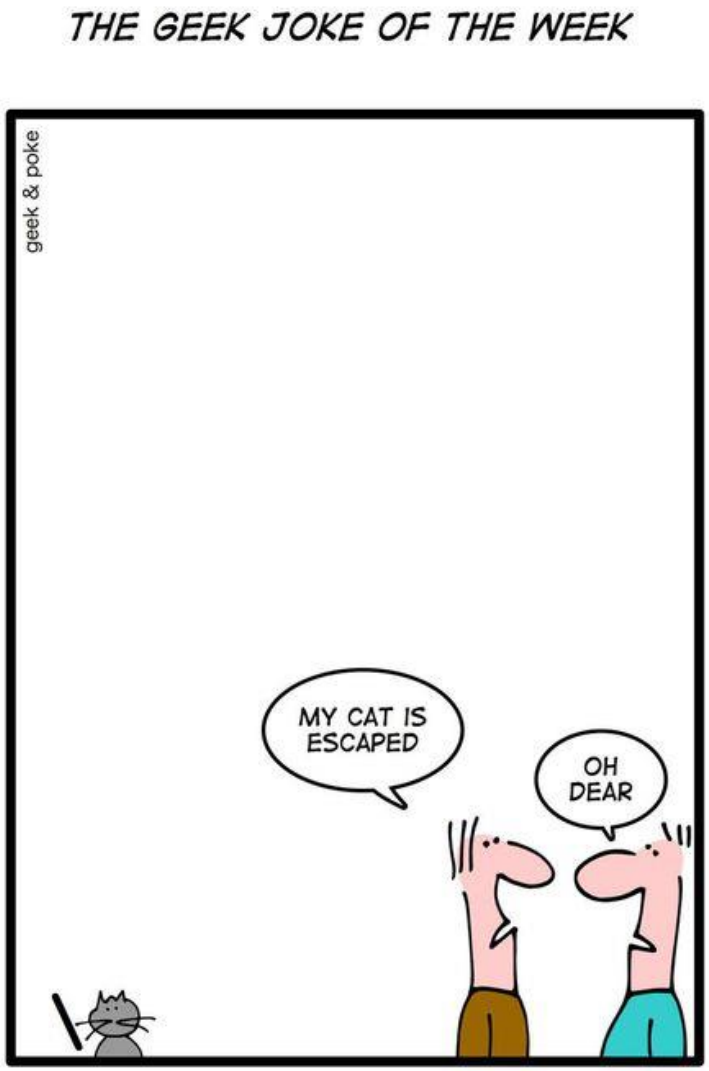
```
CREATE TABLE `Two Words`
(   `Word one` varchar(20),
    `Word two` varchar(20));
```

- What if your data contains “ or ‘ ?
  - Escape character \

```
INSERT INTO `Two Words`
VALUES ("can't",‘won\'t');
```

Word one	Word two
can't	won't

  - What if you want to use \ in your data?

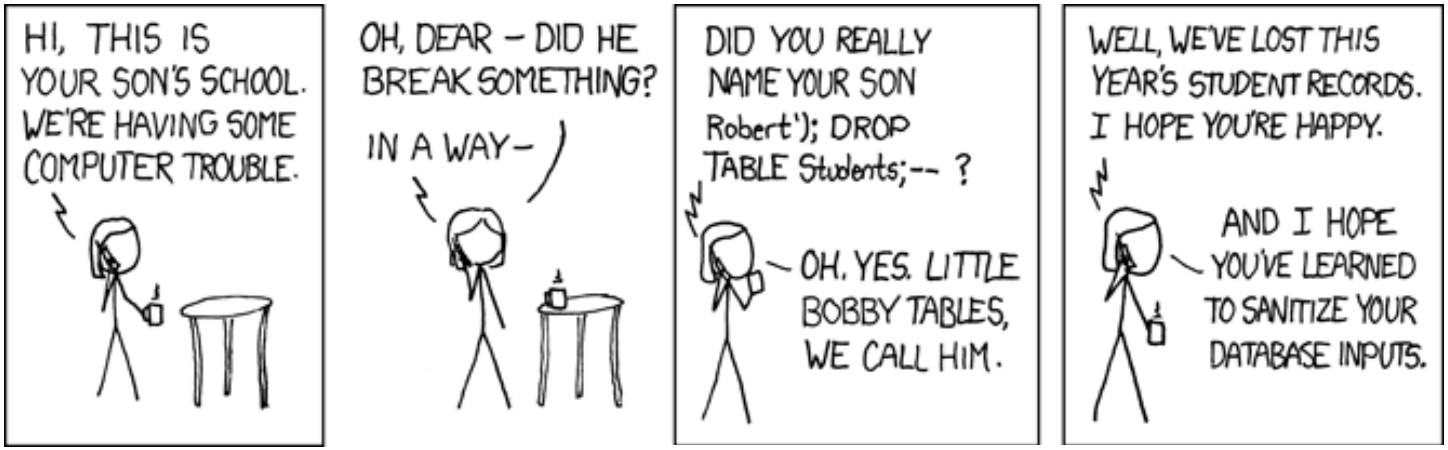


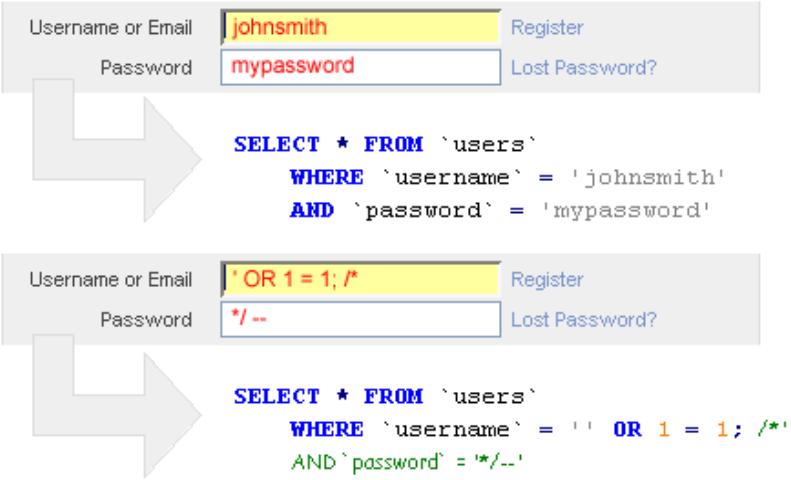
Pitfalls with SQL

- Why is this funny?

Pitfalls with SQL

xkcd





- `/* comment */`
- `-- end of statement`
- Pitfall is called

  "SQL INJECTION"

Sorting

- In many cases it is desired to sort the output of queries

- SQL syntax:
  ORDER BY a <direction>, b <direction>, …
  <direction> **ASC** | **DESC**

- Relational Algebra Operator: $\tau_{a,b}\,(R)$

- Example SQL:
  SELECT * **FROM** persons **ORDER BY** firstName **ASC;**

- Example Relational Algebra: $\tau_{firstName}\,(persons)$

firstName	lastName
Max	Bügler
Max	Mustermann
Max	Müller
Parker	James
Parker	Miller
Jennifer	Milan
Jennifer	Turner
John	Turner
Jennifer	Turner

firstName ▲ 1	lastName
Jennifer	Milan
Jennifer	Turner
Jennifer	Turner
John	Turner
Max	Bügler
Max	Mustermann
Max	Müller
Parker	James
Parker	Miller

https://dev.mysql.com/doc/refman/5.7/en/sorting-rows.html

Sorting

- Multiple sorting columns

- Example SQL:

```
SELECT * FROM persons ORDER BY firstName, age;
```

- Example Relational Algebra: $\tau_{firstName,\, age}(persons)$

firstName	lastName	age
Max	Bügler	33
Max	Mustermann	20
Max	Müller	25
Parker	James	28
Parker	Miller	24
Jennifer	Milan	22
Jennifer	Turner	28
John	Turner	25
Jennifer	Turner	21

No sort

firstName ▲ 1	lastName	age ▲ 2
Jennifer	Turner	21
Jennifer	Milan	22
Jennifer	Turner	28
John	Turner	25
Max	Mustermann	20
Max	Müller	25
Max	Bügler	33
Parker	Miller	24
Parker	James	28

Multiple sort

firstName ▲ 1	lastName	age
Jennifer	Milan	22
Jennifer	Turner	28
Jennifer	Turner	21
John	Turner	25
Max	Bügler	33
Max	Mustermann	20
Max	Müller	25
Parker	James	28
Parker	Miller	24

Single sort

https://dev.mysql.com/doc/refman/5.7/en/sorting-rows.html

Sorting

- Different sort orders
  ASC for Ascending and DESC for Descending

- Example SQL:

```
SELECT * FROM persons ORDER BY firstName DESC, age ASC;
```

firstName	lastName	age	male
Max	Bügler	33	1
Max	Mustermann	20	1
Max	Müller	25	1
Parker	James	28	0
Parker	Miller	24	1
Jennifer	Milan	22	0
Jennifer	Turner	28	0
John	Turner	25	1
Jennifer	Turner	21	0

firstName ▼ 1	lastName	age ▲ 2	male
Parker	Miller	24	1
Parker	James	28	0
Max	Mustermann	20	1
Max	Müller	25	1
Max	Bügler	33	1
John	Turner	25	1
Jennifer	Turner	21	0
Jennifer	Milan	22	0
Jennifer	Turner	28	0

https://dev.mysql.com/doc/refman/5.7/en/sorting-rows.html

Limits

- Obtain a certain number of results
- SQL: **LIMIT** <number>
- Example SQL:

```
SELECT * FROM persons WHERE firstName='Jennifer'
ORDER BY Age LIMIT 1
```

firstName	lastName	age	male
Max	Bügler	33	1
Max	Mustermann	20	1
Max	Müller	25	1
Parker	James	28	0
Parker	Miller	24	1
Jennifer	Milan	22	0
Jennifer	Turner	28	0
John	Turner	25	1
Jennifer	Turner	21	0

firstName	lastName	age	male
Jennifer	Turner	21	0

Return the youngest person named Jennifer

https://dev.mysql.com/doc/refman/5.7/en/limits.html

Triggers

- Triggers execute commands when tables are modified

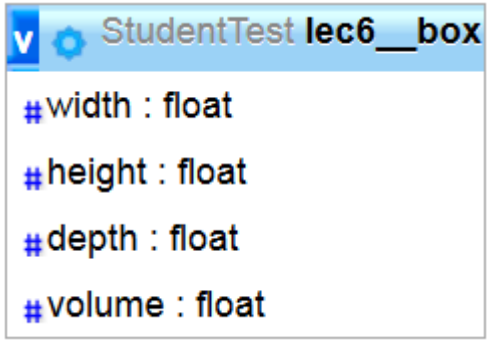
- SQL:

```
CREATE TRIGGER <name> <time> <event>
ON <table>
FOR EACH ROW [order]
        <body>
```

- `<time>`: **BEFORE** / **AFTER**
- `<event>`: **INSERT** / **UPDATE** / **DELETE**
- `[order]` : **FOLLOWS** / **PRECEDES** `<other trigger>` (only MySQL > 5.7)
- `<body>` : contains the code to be executed when the trigger is triggered

http://dev.mysql.com/doc/refman/5.7/en/create-trigger.html

Trigger



- Example SQL:

```
CREATE TABLE box(width FLOAT, height FLOAT,
     depth FLOAT, volume FLOAT);

CREATE TRIGGER calculateVolume BEFORE INSERT
ON box
FOR EACH ROW
     SET NEW.volume = NEW.width * NEW.height * NEW.depth;
```

- In <body>, refer to values prior to and after modification using the **OLD** and **NEW** keywords.
- In <body>, assign values using **SET** keyword

http://dev.mysql.com/doc/refman/5.7/en/create-trigger.html

Triggers

- Example SQL:

```
INSERT INTO box(width, height, depth)
        VALUES(1,2,3), (4,5,6), (7,8,9), (10,11,12);
```

width	height	depth	volume
1	2	3	6
4	5	6	120
7	8	9	504
10	11	12	1320

Before each insert
the volume is computed

-
```
CREATE TRIGGER calculateVolume BEFORE INSERT
ON Box
FOR EACH ROW
    SET NEW.volume = NEW.width * New.height * New.depth;
```

http://dev.mysql.com/doc/refman/5.7/en/create-trigger.html

orderNumber	numberOfPizzas	pizza	price	totalPrice
1	1	Salami	5	5
1	2	Funghi	6	12
2	10	Salami	5	50

orderNumber	price
1	17
2	50

```
CREATE TABLE orderedPizzas(orderNumber INT, numberOfPizzas INT,
        pizza VARCHAR(55), price FLOAT, totalPrice FLOAT);

CREATE TABLE pizzaOrders(orderNumber INT PRIMARY KEY,price FLOAT);
```

```
CREATE TRIGGER calculateTotalPrice BEFORE INSERT
ON orderedPizzas
FOR EACH ROW
        SET NEW.totalPrice = NEW.numberOfPizzas * NEW.price;
```

http://dev.mysql.com/doc/refman/5.7/en/create-trigger.html

orderNumber	numberOfPizzas	pizza	price	totalPrice
1	1	Salami	5	5
1	2	Funghi	6	12
2	10	Salami	5	50

orderNumber	price
1	17
2	50

- Pizza Trigger SQL Example:

Name Time Event

```
CREATE TRIGGER calculateOrder AFTER INSERT
ON orderedPizzas
FOR EACH ROW
REPLACE INTO pizzaOrders
VALUES(
    NEW.orderNumber,
    (SELECT SUM(orderedPizzas.totalPrice)
            FROM orderedPizzas
            WHERE orderedPizzas.orderNumber = NEW.orderNumber
            GROUP BY orderedPizzas.orderNumber));
```

Table

Body

http://dev.mysql.com/doc/refman/5.7/en/create-trigger.html

Triggers – Pizzeria Example

- Pizza Trigger SQL Example:

```
INSERT INTO orderedPizzas(orderNumber,numberOfPizzas,pizza,price)
        VALUES(1,1,'Salami',5),
              (1,2,'Funghi',6),
              (2,10,'Salami',5);
```

orderNumber	numberOfPizzas	pizza	price	totalPrice
1	1	Salami	5	5
1	2	Funghi	6	12
2	10	Salami	5	50

orderNumber	price
1	17
2	50

Intermediate Topic - LAST_INSERT_ID

- Remember: **AUTO_INCREMENT**
- SQL: **LAST_INSERT_ID()** Use this to access the last generated value

- Example SQL:

auto_inc	last_id
1	0
2	1
3	2

```
CREATE TABLE AUTOINC(
auto_inc int AUTO_INCREMENT PRIMARY KEY,
last_id int);

INSERT INTO AUTOINC(last_id) VALUES(0);
INSERT INTO AUTOINC(last_id) VALUES(LAST_INSERT_ID());
INSERT INTO AUTOINC(last_id) VALUES(LAST_INSERT_ID());
```

https://dev.mysql.com/doc/refman/5.7/en/information-functions.html#function_last-insert-id

Views

- A view is a virtual table
- Can be created using many select statements
- Can use
  - Joins
  - Unions
  - Sub-queries
  - Projections
  - Selections

- SQL:

```
CREATE VIEW <name> AS <query>
```

http://dev.mysql.com/doc/refman/5.7/en/views.html

Views

- Pizza View SQL Example:

```
CREATE VIEW OrderSummary AS
SELECT orderNumber as 'Order',
       GROUP_CONCAT(CONCAT(numberOfPizzas,"x",pizza)) as 'Pizzas',
       SUM(orderedPizzas.totalPrice) as 'Price'
       FROM orderedPizzas
       GROUP BY orderNumber
```

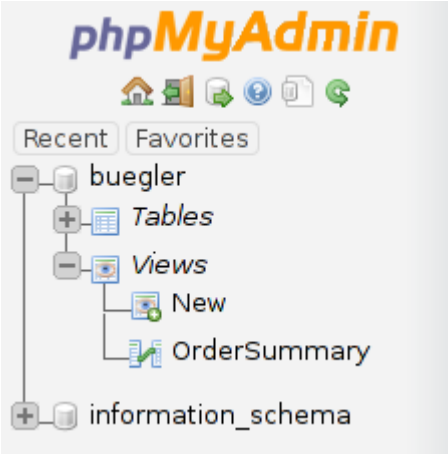
orderNumber	numberOfPizzas	pizza	price	totalPrice
1	1	Salami	5	5
1	2	Funghi	6	12
2	10	Salami	5	50

1 1xSalami, 2xFunghi 34

http://dev.mysql.com/doc/refman/5.7/en/views.html

Views

- Pizza View SQL Example:



orderNumber	numberOfPizzas	pizza	price	totalPrice
1	1	Salami	5	5
1	2	Funghi	6	12
2	10	Salami	5	50

```
SELECT * FROM OrderSummary;
```

orderNumber	Pizzas	price
1	1xSalami,2xFunghi	17
2	10xSalami	50

http://dev.mysql.com/doc/refman/5.7/en/views.html

Transactions

50 € are transferred from account A to account B

1. Read balance of account A
   a := read(A);
2. Reduce balance by 50 €
   a := a - 50;
3. Write new balance to database:
   write(A; a);
4. Read balance of B into b
   b := read(B);
5. Increase balance by 50 €
   b := b + 50;
6. Write balance to database
   write(B; b);

Consideration of system crash between 3th and 4th operation

Requirement:
Either all or none operations are performed

http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-transactions.html

Transactions

- Transactions are a mechanism to
  - Ensure database consistency
  - Run a bundle of operations as one unit
  - Handle disturbance by other progresses/crashes
- Encapsulation of a logically uninterruptible process
- A data base without a transaction system is useless in practice

http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-transactions.html

Transactions

- Properties of Transaction

- ACID = Atomicity, Consistency, Isolation, and Durability

- Atomicity: A transaction runs either entirely or not at all
- Consistency: A transaction takes the database from one consistent state to another
- Isolation: A transaction runs in isolation from other transactions
- Durability: Changes by a committed transaction must persist in the database

http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-transactions.html

Transactions

- There is no transaction command in the SQL standard

- MySQL supports
- SQL:
  - **`START TRANSACTION`**
    - <SQL Statements>
  - **`COMMIT`**
    - or
  - **`ROLLBACK`**

- In PhPMyAdmin, single SQL statements are automatically committed in MySQL
- Full functionality not usable in PhpMyAdmin

http://dev.mysql.com/doc/refman/5.7/en/sql-syntax-transactions.html

Homework – Freight Company

- You are running a freight company transporting containers around the world.
- Each container you transport has an id, a weight and a value, as well as a origin and destination location.
- Containers are transported on ships.
- Each ship can carry a certain number of containers and has a weight limit.
- Ships are waiting in some location and can have a destination assigned to them.
- When you get a mission to transport a container, all you want to do is, to enter the container weight, value, origin and destination.
- The database should then find a ship to put the container on, so it is transported to the destination.
- If a ship is already assigned the destination and can fit the container the container is assigned to the ship. If no such ship is available, a free ship is assigned with the new destination.

Homework – Freight Company – Tables

- Note: When creating the tables, carefully think about the use of data types, and modifiers such as primary keys, foreign keys, unique,…

- Create a table locations with columns locationId, locationName
- Create a table containers with columns containerId, weight, value, origin, destination
- Create a table ships with columns shipId, weightCapacity, containerCapacity
- Create a table shipAtLocation with columns shipId, locationId
- Create a table shipGoingToDestination width columns shipId, locationId
- Create a table containerOnShip with columns containerId, shipId

Homework – Freight Company – Data

- Initial data explanations:
- Insert the locations Rotterdam, Miami, and Hamburg
- Insert Ships with following data:
  - Weight capacity: 1000, container capacity 100
  - Weight capacity: 10, container capacity 2
  - Weight capacity: 200, container capacity 2
  - Weight capacity: 100, container capacity 10
  - Weight capacity: 100, container capacity 10
  - Weight capacity: 300, container capacity 10
- Assign the first two ships to be in Rotterdam, the second two to be in Miami, and the last two to be in Hamburg

Homework – Freight Company – Data

- Additional data:
- Add containers with the following data:
  - Weight: 1, value: 10, from Rotterdam to Miami
  - Weight: 10, value: 20, from Rotterdam to Hamburg
  - Weight: 10, value: 10, from Miami to Rotterdam
  - Weight: 50, value: 40, from Miami to Hamburg
  - Weight: 30, value: 10, from Hamburg to Rotterdam
  - Weight: 20, value: 1000, from Hamburg to Miami

- Assign each container to a ship (1 -> 1, 2 -> 2,…6 -> 6)

Homework – Freight Company – Queries

- Basic queries:
- List the locationId for each ship
- List the locationName for each ship
- List the shipId in Hamburg
- List all containers on ship 1
- List all containers on ships in Hamburg
- List all ships which have reached their weight capacity
  (sum of container weights on ship are equal to the weight capacity)
- List all ships which have reached their capacity by either weight or container count
  - Extension 1: Also show the location of the ships (locationId, then locationName)
  - Extension 2: Also show the total weight of each ship as a column
- List the lightest ship in Hamburg
- List the lightest ship in Hamburg which has a free weight capacity of 100
- List the lightest ship in Hamburg which has a free weight capacity of 100 and a free container capacity of 2

Homework – Freight Company – Extensions

- Add more data using nesting SQL
- Each Ship has one container loaded.
  Assign each ship's destination location based on the container it has stacked.
  The destination location should go into the shipGoingToLocation table

  Special case of subquery. Insert all results of one query into another table.

-
```
INSERT INTO shipGoingToLocation(shipId,destinationId)
     SELECT shipId, destination FROM Ship
     NATURAL JOIN containerOnShip
     NATURAL JOIN container
     GROUP BY shipId;
```

Homework – Freight Company – Extensions

- Retrieve all ships and their respective origin and destination
  - Remember that you have to rename a relation that you want to use twice in one query.
- Add a new ship with weight capacity 100, container capacity 10, that is currently in Hamburg.
- Retrieve all ships that do not have a destination assigned.
- Create a trigger that assigns a newly added container to the lightest ship matching origin, destination and not exceeding the ship's capacity.
  - **This one is REALLY HARD!**

End of Lecture

Thank you for your attention