

Lecture 13

- Triggers
- Chatty vs. Chunky – The Spaghetti Cook’s Tale

Triggers

- Many SQL-based RDBMS allow you to define *triggers* on database tables
- Think of a trigger as a procedure without parameters that is called whenever a particular event occurs on a table (usually a modification of the data in the table)
- Triggers can typically be defined for insert, update, or delete events, i.e., are called when you issue an INSERT, UPDATE, or DELETE statement
- You can define them to be called before or after the an INSERT, UPDATE, or DELETE statement takes effect

What INSERT or UPDATE triggers can be used for (examples)

- format data before it is inserted (e.g., trim extra whitespace)
- check data for validity before it is inserted
- additional inserts, updates, or deletes in other tables using new data from the INSERT

What DELETE triggers can be used for (examples)

- Guard against erroneous deletes (to an extent)
- Ensure that any child records are deleted (records in other tables that use a value from the record to be deleted as a foreign key). E.g., delete all records in the User table for the City record to be deleted
- Record the deletion in a log table

Disclaimer

- You cannot use triggers on the MySQL server at university – creating triggers in MYSQL 5 requires you to be a superuser
- You can try this at home if you are logged in as root

Creating a simple INSERT trigger

- Post a welcome message to a new user:

```
delimiter //

CREATE TRIGGER welcome AFTER INSERT ON User
FOR EACH ROW BEGIN
    INSERT INTO Message
        (`owner_id`,`subject`,`body`,
        `posted`,...)
    VALUES
        (NEW.user_id, ' Welcome!',
        'Hope you like it here!',...);
END;
//

delimiter ;
```

Creating a simple DELETE trigger

- Make sure all users are deleted for a particular city (to avoid FK violation):

```
delimiter //

CREATE TRIGGER zap_users BEFORE DELETE
ON City
FOR EACH ROW BEGIN
    DELETE FROM User
        WHERE User.city_id = OLD.city_id;
END;
//

delimiter ;
```

Trigger happy?

- Having triggers can have unforeseen consequences if you are not careful
- Don't use triggers as a replacement for transactions,
- E.g., don't use them to do inserts with the new data in a BEFORE INSERT trigger – if the main INSERT then fails (e.g., because of a duplicate key or FK violation), you have a problem
- Also, if you use lots of triggers, beware of inadvertent trigger loops (e.g., INSERT on table T1 triggering insert on TABLE T2 triggering INSERT on T1 etc.)

Having a sensible chat to your DB

- A typical query by a client application to a remote DB server involves the following:
 - a) Client sends query – this incurs a network delay
 - b) RDBMS processes query – this incurs a delay depending on the query's complexity (mostly number of joins involved) and size of result set to be compiled
 - c) RDBMS sends result set to client – this incurs a network delay linear in the size of the result set
- If the application does not maintain connection to the DB, additional communication delays may be incurred in the case of multiple queries
- Need to be careful to balance client's need for data with query complexity and result set size
- How often do we query, and how much do we retrieve each time?
- This conundrum is also known as "chatty vs. chunky"

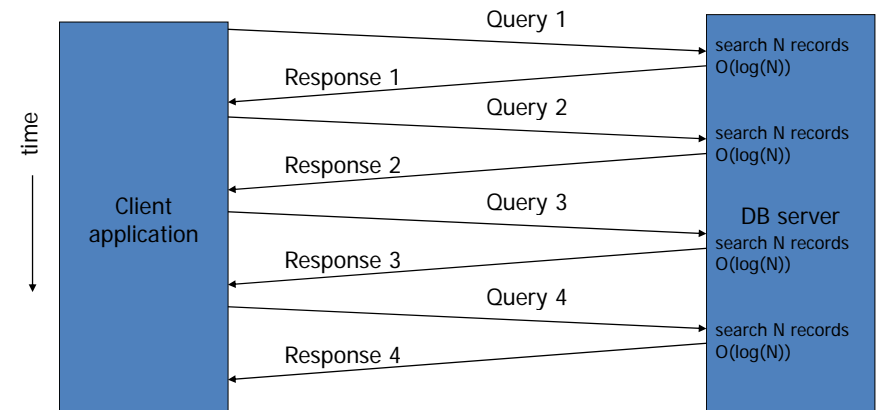
The Spaghetti Cook's Tale

- There are two ways to cook Spaghetti
- Algorithm 1: Go to the supermarket. Buy a single spaghetti noodle. Take it home. Cook it. Repeat the procedure until you have a bowl of spaghetti.
- Algorithm 2: Go to the supermarket. Buy a pack of spaghetti. Take it home. Cook the whole pack at once. You now have a bowl of spaghetti
- Which way is more efficient?
- But do you ever buy a year's worth of spaghetti at once?

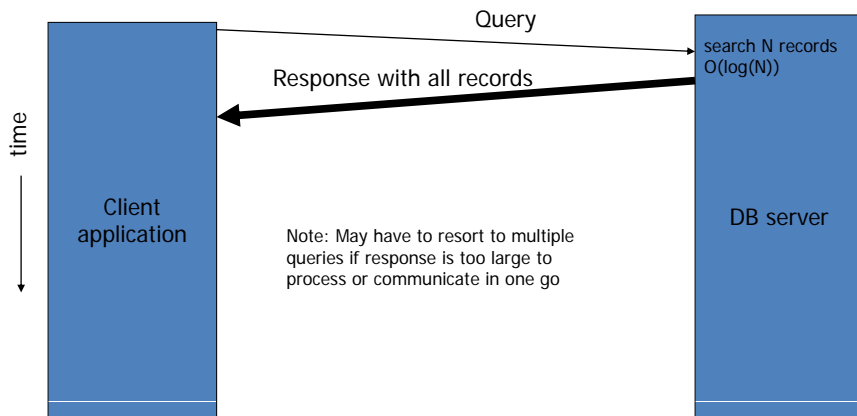
The Database User's Tale

- There are two ways of getting records out of a database
- Version 1: Connect to the database server. Send a query to retrieve the record you want. Process the record. Repeat these three steps until you have all the records you need.
- Version 2: Connect to the database server. Send a single query to retrieve all the records you need. Process the records locally.
- Which version is more efficient?
- Morale of the story: Don't open repetitive database connections inside a loop. Don't send database `select` queries on tables inside a loop if you can avoid it.

Database User's Tale (bad, chatty)



Database User's Tale (good, chunky)



Today's lab sheet

- For today, there is no lab sheet
- Please e-mail me with suggestions/questions for Friday's review session