Tomas Sandnes
tomas.sandnes@kristiania.no

Week 11

DB1102 / PGR 111 – DATABASES

Today's topics

(Today's chapters: See the chart on Canvas, several small ones, won't list all here.)

- RDBMS operations:
  - SQL transactions
  - ACID properties

- DB user administration

- NoSQL databases
  - Document DB
  - Graph DB

- *Bonus topic:* Hacking SQL databases
  - SQL injection

Regarding feedback on the hand-ins

- For feedback on the coursework requirement: (and exam)
  - Talk to **Per Lauvås!**

PER

PER !!!

Per Lauvås

(NOT me!)

SQL transactions

Commit and rollback

When we want to change data and/or tables in SQL, this can be handled in two different ways.

- The changes can be applied continuously.
  - This is called auto-commit and is the default setting in MySQL Workbench.

- The changes can be staged and then applied on our request. Meaning we can chain several changes together, then finally carry them all out or undo them all.
  - We control this with the SQL commands START TRANSACTION, COMMIT and ROLLBACK.
  - *Note:* "Transaction" in this setting means "database operation".
  - Not necessarily related money transfers.

ACID properties

ACID properties

- The ACID properties are a collection of properties that all database commands for relational DBs should fulfil.

- They have been named ACID after the first letter of each property:
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability

ACID properties – cont.

- Atomicity:
  - The "all or nothing" principle: Either an entire transaction is completed, or everything is reset.

- Consistency:
  - A transaction must move the database from one full state to another.
  - This responsibility rests with both the developer and DBMS.

- Insulation:
  - What happens internally in a transaction must be invisible to the outside world (invisible to other transactions) until the transaction is completed.

- Durability:
  - The result of a completed transaction must be stored in the database, regardless of what happens in future transactions.

User admin

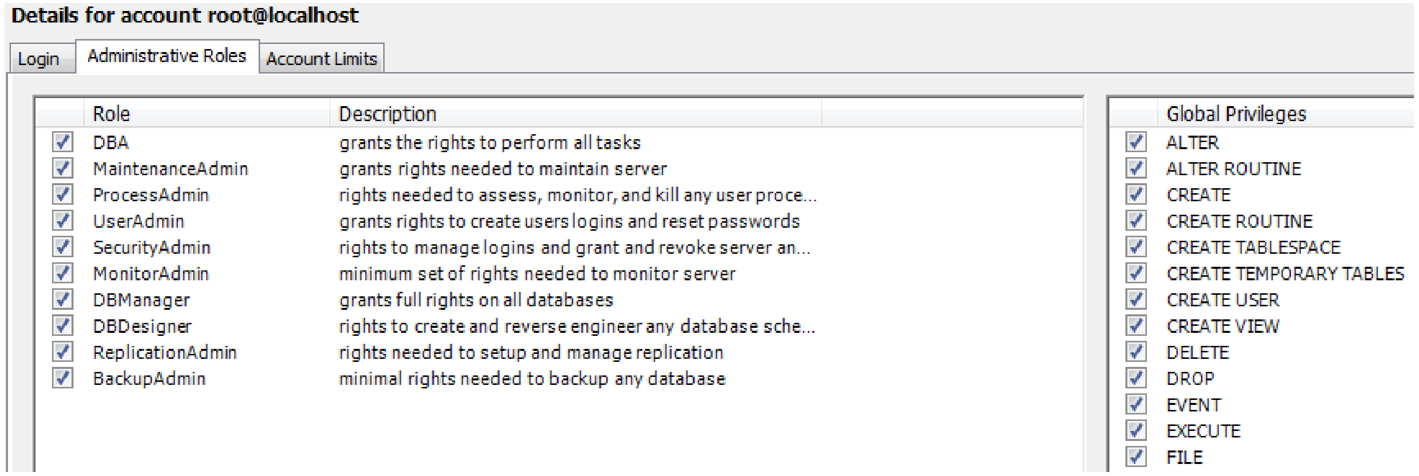
DBMS user administration

- A user can be given rights to specific parts of the database:
  - We can grant rights to various functions in SQL, and for specific schemas / tables.

- The purpose is for a person to be able to perform the tasks he / she is intended to perform, but not other (more) tasks than that.

DBMS user administration – cont.

- The root user can do everything, and can therefore give all possible rights to others:
  - In MySQL Workbench: Server -> Users and Privileges
  - Select a user.



Details for account root@localhost

Login | Administrative Roles | Account Limits

	Role	Description
✓	DBA	grants the rights to perform all tasks
✓	MaintenanceAdmin	grants rights needed to maintain server
✓	ProcessAdmin	rights needed to assess, monitor, and kill any user proce...
✓	UserAdmin	grants rights to create users logins and reset passwords
✓	SecurityAdmin	rights to manage logins and grant and revoke server an...
✓	MonitorAdmin	minimum set of rights needed to monitor server
✓	DBManager	grants full rights on all databases
✓	DBDesigner	rights to create and reverse engineer any database sche...
✓	ReplicationAdmin	rights needed to setup and manage replication
✓	BackupAdmin	minimal rights needed to backup any database

	Global Privileges
✓	ALTER
✓	ALTER ROUTINE
✓	CREATE
✓	CREATE ROUTINE
✓	CREATE TABLESPACE
✓	CREATE TEMPORARY TABLES
✓	CREATE USER
✓	CREATE VIEW
✓	DELETE
✓	DROP
✓	EVENT
✓	EXECUTE
✓	FILE

Creating a new user

- As an example, let's say we want to create a new user who can only perform SELECT statements against the world database.
  - So can not do DELETE FROM, not do DROP TABLE, etc.

- In this context, we must:
  1. Create a user.
  2. Grant (restrict) the user's rights.
  3. Create a new connection to the database and log in as the new user.
  4. Check that the user's rights are working as intended.

Example

- SQL-syntax to create a user:

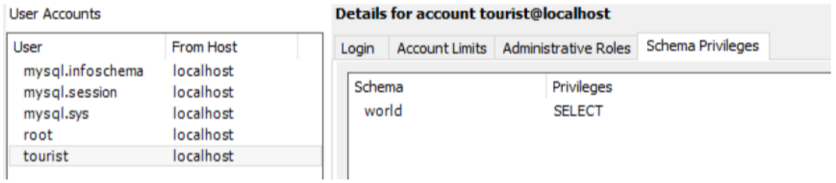
```
CREATE USER username@hostname
IDENTIFIED BY password;
```

- *Example:*
  - Creating user *tourist* on *localhost*, with password *seeTheWorld*:

```
CREATE USER 'tourist'@'localhost'
IDENTIFIED BY 'seeTheWorld';
```

  - Then we grant *SELECT* rights to user *tourist* (on *localhost*) for the world database: (change * for tablename to grant rights to a single table only)

```
GRANT SELECT
ON world.*
TO 'tourist'@'localhost';
```



User Accounts

User	From Host
mysql.infoschema	localhost
mysql.session	localhost
mysql.sys	localhost
root	localhost
tourist	localhost

Details for account tourist@localhost

Schema	Privileges
world	SELECT

More on user rights

- If we want to give access to a specific view:

```
GRANT SELECT
ON <database name>.<view name>
TO <user>@<host>
```

- To grant all rights: (but why would you?)
  - GRANT ALL PRIVILEGES

- Google for more options. :-)

Checking that the user works as expected

- In addition, we should test that our user works. We do this from the Home page of MySQL Workbench: (the small "house" in the upper left corner)
  - Click the plus sign next to MySQL Connections.
  - Give the new connection a name and enter the relevant username
  - Enter the user's password when prompted.

- Test that new user has the correct rights.
  - (Can do SELECT, but no other commands, in our tourist-example.)

- To delete a user: DROP USER <username>@<hostname>

- Example:

```
DROP USER 'tourist'@'localhost';
```

SQL vs. NoSQL

Different types of databases

- Relational databases – RDBMS (MySQL, …)
  - Object-relational databases – ORDBMS (PostgreSQL, …)

- Non-relational databases – NoSQL ("Not only SQL") DBMS variants:
  - Document (MongoDB, …)
  - Graph (Neo4j, …)
  - Key-value (Redis, …)
  - Column (HBase, …)

Sources: Databasesystemer, 5th edition, Bjørn Kristoffersen | What is Object-Relational Database Systems? | Which Modern Database Is Right for Your Use Case?

Relational databases – RDBMS

- The dominant database type since the 1970s.
  - This is what we have been working on for 8 days now! :-D
  - MySQL, Microsoft SQL Server and Oracle are all examples of RDBMS.

- Advantages:
  - ACID compliance.
  - Ideal for consistent data systems.
  - Great support options.

- A close relative to the RDBMS is object-relational databases (ORDBMS)
  - *Pro:* A version of relational databases that also support user-defined objects.
  - *Con:* Can result in higher complexity (and thus increased costs).

Non-relational databases – NoSQL DBMS

- Comes in several variants: Document, Graph, Key-value, Column.
  - MongoDB is probably the best known NoSQL database (of the Document variant).

- Advantages:
  - Excellent for handling "big data" analytics.
  - No limits on types of data you can store.
  - Easier to scale.
  - No data preparation required.

- Disadvantages:
  - Don't follow the full ACID principles.
  - More difficult to find support and possibly lack of tools.
  - Compatibility and standardization challenges.

Two types of non-relational databases

- Document
  - Semi-structured data in the form of "documents" (XML, JSON, …). Only each document has a unique ID.
  - Good as a backbone for web solutions, as well as for free text searches.

- Graph
  - *Note: Graph DBs are quite different than the other types!*
  - Uses graph structures. A graph is built with nodes and edges (and then properties on these).
  - Good for modelling maps and social media relations (and searches / pathfinding in these).

Hacking DBs

The Injection security threat

- Injection is a HUGE security threat!

- According to OWASP, as of 2021 "Injection" is the 3rd-largest application security risk. (And it was the number-1 risk in 2017.)

- Notable weakness types included in "Injection" are:
  - SQL injection
  - Cross-site Scripting
  - External Control of File Name or Path

Get into the correct developer mindset

- The injection problem starts with software developers expecting application users to be intelligent and nice guys.
  - That is an absolutely wrong expectation!
- Your software users are, at the very best, idiots!
  - Don't except a user to input a number, just because an input box says "Phoner number".
- And idiots are not the scary ones. Some of the users are EVIL!
  - They are trying to hurt you and your application.

Being the GOOD guys

- So why am I teaching you how to become a malicious hacker?
  - I am not!
  - Well, I kind of am. :-\ But that's just the (unwanted) bi-product.

- I'm teaching you what the SQL injection security threat is.
  - That way you can take measures that keeps your applications and your databases safe from SQL injection threats. :-)

- We are the GOOD guys! (Right?)
  - Our intention is Ethical hacking. (Being "white hat" hackers.)

SQL injection, possibilities

- SQL injection is the process of adding ("injecting") extra SQL code into an application's DB-statements.
  - This is done by "creative" use of the input-fields.
  - Resulting in the program's behaviour altered from what the developer intended.

- Some example uses:
  - Logging in without knowing the correct password. (Also without knowing the username, if you like.)
  - Seeing (stealing) hidden content. (Like personal info for other users, etc…)
  - Changing existing content. (For example, giving yourself better grades, extra shopping credits, …)
  - Deleting content. (All or just some of it. At any time you're asked to input a value, like name, item category or a search word, you could delete the whole DB.)

SQL injection, historical examples

- Here are some examples of real-world, malicious SQL injection. These attacks could have been prevented with better security knowledge on the software developers' end!
  - Hackers targeted 53 universities using SQL injection and stole and published 36,000 personal records belonging to students, faculty, and staff.
  - Hackers used SQL injection to breach the Turkish government website and erase debt to government agencies.
  - A team of attackers used SQL injection to penetrate corporate systems at several companies, primarily the 7-Eleven retail chain, stealing 130 million credit card numbers.

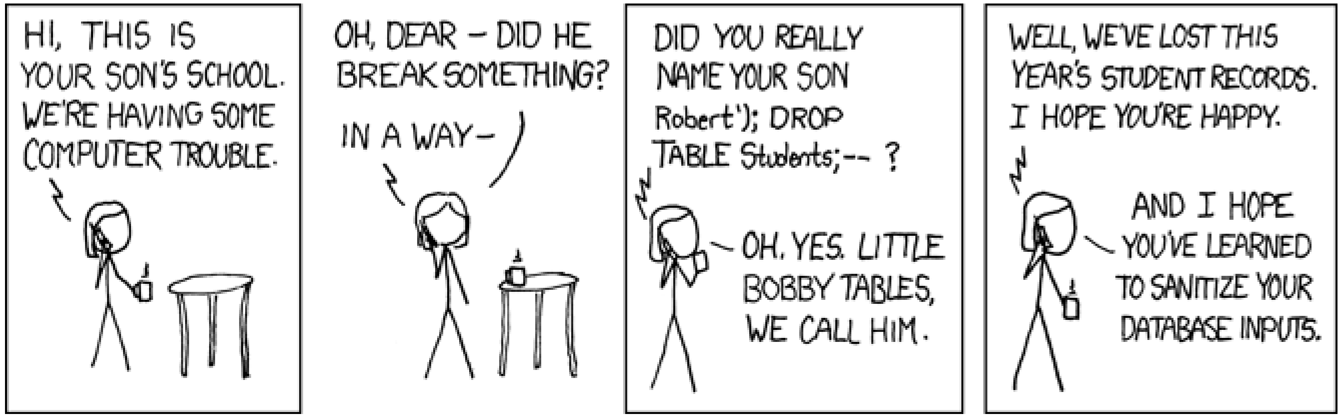
- Here are a couple of discovered SQL injection vulnerabilities:
  1. Fortnite is an online game with over 350 million users. In 2019, a SQL injection vulnerability was discovered which could let attackers access user accounts.
  2. In 2014, security researchers publicized that they were able to breach the website of Tesla using SQL injection, gain administrative privileges and steal user data.

Source: SQL Injection Attack: Real Life Attacks and Code Examples

SQL injection, the technical stuff

- For those attending the lesson: I'll show a live demo of SQL injection.
  - *Note:* I'm NOT hacking a real site!
  - I'm hacking my own "sandbox-site for showcasing SQL injections". :-)

- Short (and funny!? :-P) sum-up of how it works:


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

Today's exercises & looking ahead

- Now: 2 hours of exercises.

- Exercises are on Canvas, as usual. Short summary:
  - Exercises related todays topics.
  - Then start on repetition / remaining topics from this autumn.

- Main contents for the next lesson:
  - Repetition of whatever *you want!* (Tell me by Canvas message.)

The