Advanced SQL Topics in Software Development

CSc3350

Dr. William Greg Johnson
Department of Computer Science
Georgia State University

Introduction

Every organization has data that needs to be collected, managed, and analyzed. Most people are familiar with some kind of spreadsheet, such as Microsoft Excel. Spreadsheets are easy and convenient to use, and they may be employed by an individual. Spreadsheets are commonly used to store information in a tabular format. A spreadsheet can store data in rows and columns, it can link cells on one sheet to those on another sheet, and it can force data to be entered in a specific cell in a specific format. It's easy to calculate formulas from groups of cells on the spreadsheet, create charts, and work with data in other ways.

A database fulfills these needs. Along with the powerful features of a relational database come requirements for developing and maintaining the database. Data analysts, database programmers, and database administrators (DBAs) need to be able to translate the data in a database into useful information for both day-to-day operations and long-term planning.

Database

- Originally the database was flat
  - Information was stored in text file called delimited file
  - Each record (row) in file separated by special character, such as a vertical bar (|)
  - Difficult to search for specific information

```
Lname, FName, Age, Salary|Smith, John, 35, $280|Doe, Jane, 28, $325|Brown, Scott, 41, $265|Howard, Shemp, 48, $359|Taylor, Tom, 22, $250
```

Types of Databases

By Function

- Analytical Database – Also referred as On-Line Analytical processing (OLAP), are those used to keep track of statistics
  - Read only access to analyze the data
- Operational Database – Also referred as On-Line Transactional Processing (OLTP), are those, let you actually change and manipulate the data in database

Types of Databases

By Data Model

- Flat File database model
  - Data is stored in numerous files
  - No linkage between files so repetition of information in different files
- Relational Database Model
  - Data can be stored in different table/databases
  - The tables/databases can be connected using keys
- Object oriented Database Model
  - Stores **not only text, but also sounds, images, and all sorts of media clips**
- Not only SQL (NoSQL) is an umbrella term for any alternative system to traditional SQL databases. In NoSQL management systems, it indicates any database that doesn't use a relational model. NoSQL databases use a data model that has a different structure from the rows and columns table structures used with RDBMS.
  - document databases
  - key-value stores
  - column-oriented databases
  - graph databases

What is a DBMS?

Need for information management

- A very large, integrated collection of data
- Models the real-world enterprise
  - Entities (e.g., students, courses)
  - Relationships
- A Database Management System (DBMS) is a software package designed to store and manage databases.

Why Use a DBMS?

- Data independence and efficient access
- Data integrity and security
- Uniform data administration
- Concurrent access, recovery from crashes
- Replication control
- Reduced application development time

Why Study Databases?

?

- Shift from *computation* to *information*
  - at the “low end”: access to physical world
  - at the “high end”: scientific applications
- Datasets increasing in diversity and volume.
  - Digital libraries, interactive video, Human Genome project, e-commerce, sensor networks
  - Generative adversarial network (synthetic data)
- DBMS encompasses several areas of CS
  - OS, languages, theory, AI, multimedia, digital design

Data Models

- A *data model* is a collection of concepts for describing data.
- A *schema* is a description of a particular collection of data, using a given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.
  - Every relation has a *schema*, which describes the columns, or fields.

SQL Data Manipulation Language (DML)

- SQL includes a syntax to update, insert, and delete records:
  - SELECT - extracts data
  - UPDATE - updates data
  - INSERT INTO - inserts new data
  - DELETE - deletes data

SQL Data Definition Language (DDL)

- The Data Definition Language (DDL) part of SQL permits:
  - Database tables to be created or deleted
  - Define indexes (keys)
  - Specify links between tables
  - Impose constraints between database tables
- Some of the most commonly used DDL statements in SQL are:
  - CREATE TABLE - creates a new database table
  - ALTER TABLE - alters (changes) a database table
  - DROP TABLE - deletes a database table

Metadata

- SQL databases are based on the RDBM (Relational Database Model)
- One important fact for SQL Injection
  - Amongst Codd's 12 rules for a Truly Relational Database System:
    - Metadata (data about the database) must be stored in the database just as regular data is stored
  - Therefore, the database structure can also be created, read, and altered with SQL queries

The INSERT INTO Statement

The INSERT INTO statement is used to insert new rows into a table.

Syntax

INSERT INTO table_name
VALUES (value1, value2,....)

You can also specify the columns for which you want to insert data:

INSERT INTO table_name (column1, column2,...)
VALUES (value1, value2,....)

Insert a New Row

LastName	FirstName	Address	City
Pettersen	Kari	Stork 20	Nashville

And this SQL statement:

```
INSERT INTO Persons
VALUES ('Heartland', 'Camilla', 'Peachtree 24', 'Atlanta')
```

LastName	FirstName	Address	City
Pettersen	Kari	Stork 20	Baltimore
Heartland	Camilla	Peachtree 24	Atlanta

Insert Data in Specified Columns

LastName	FirstName	Address	City
Pettersen	Kari	Stork 20	Baltimore
Heartland	Camilla	Peachtree 24	Atlanta

And This SQL statement:

INSERT INTO Persons (LastName, Address)
VALUES (‘Rasmussen’, ‘Storgt 67’)

LastName	FirstName	Address	City
Pettersen	Kari	Stork 20	Baltimore
Heartland	Camilla	Peachtree 24	Atlanta
Rasmussen		Storgt 67	

The Update Statement

The UPDATE statement is used to modify the data in a table.

Syntax

```
UPDATE table_name
SET column_name = new_value
WHERE column_name = some_value
```

Update one Column in a Row

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	St. Louis
Rasmussen		Storgt 67	

We want to add a first name to the person with a last name of "Rasmussen":

```
UPDATE Person SET FirstName = 'Nina'
WHERE LastName = 'Rasmussen'
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	St. Louis
Rasmussen	Nina	Storgt 67	

Update several Columns in a Row

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	St. Louis
Rasmussen		Storgt 67	

We want to change the address and add the name of the city:

```
UPDATE Person
SET Address = 'Stien 12', City = 'St. Louis'
WHERE LastName = 'Rasmussen'
```

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	St. Louis
Rasmussen	Nina	Stien 12	St. Louis

The Delete Statement

The DELETE statement is used to delete rows in a table.

Syntax

DELETE FROM table_name

WHERE column_name = some_value

Delete a Row

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	St. Louis
Rasmussen	Nina	Stien 12	St. Louis

"Nina Rasmussen" is going to be deleted:

DELETE FROM Person WHERE LastName = ‘Rasmussen’

LastName	FirstName	Address	City
Nilsen	Fred	Kirkegt 56	St. Louis

Delete All Rows

It is possible to delete all rows in a table without deleting the table. This means that the table structure, attributes, and indexes will be intact:

DELETE FROM table_name

Or

DELETE * FROM table_name

Drop Table

- Sometimes we may decide that we need to get rid of a table in the database for some reason. In fact, it would be problematic if we cannot do so because this could create a maintenance nightmare for the DBA's.
- Use the **DROP TABLE** command.
  - **DROP TABLE "table_name"**
- So, if we wanted to drop a table called *customer* use
  - **DROP TABLE customer**.

Truncate Table

- Sometimes we wish to get rid of all the data in a table. If we wish to simply get rid of the data but not the table itself?
- Use the **TRUNCATE TABLE** command.
  - **TRUNCATE TABLE "table_name"**
- So, if we wanted to truncate the table called *customer* use
  - **TRUNCATE TABLE customer**

What is SQL Injection?

The ability to inject SQL commands into the database engine through an existing application.

It's an attack on web-based applications that connect to database back-ends in which the attacker executes unauthorized (and unexpected) SQL commands, taking advantage of insecure code and bad input validation.

How common is it?

- It is probably the most common Website vulnerability today!
- It is a flaw in "web application" development, it is not a DB or web server problem
  - Most programmers are still not aware of this problem
  - A lot of the tutorials & demo “templates” are vulnerable
  - Even worse, a lot of solutions posted on the Internet are not good enough
- ‘In our penetration tests, over 60% of our clients turn out to be vulnerable to SQL Injection.’ [1]

Vulnerable Applications

- Almost all SQL databases and programming languages are potentially vulnerable
  - MS SQL Server, Oracle, MySQL, Postgres, DB2, MS Access, Sybase, Informix, etc
- Accessed through applications developed using:
  - Perl and CGI scripts that access databases
  - ASP, JSP, PHP
  - XML, XSL and XSQL
  - Javascript
  - VB, MFC, and other ODBC-based tools and APIs
  - DB specific Web-based applications and API's
  - Reports and DB Applications
  - 3 and 4GL-based languages (C, OCI, Pro*C, and COBOL)
  - many more

How does SQL Injection work?

Common vulnerable login query

```
SELECT * FROM users
WHERE login = 'victor'
AND password = '123'
```

(If it returns something then login!)

ASP/MS SQL Server login syntax

```
var sql = "SELECT * FROM users
WHERE login = '" + formusr +
"' AND password = '" + formpwd + "'";
```

Preventing SQL Injection

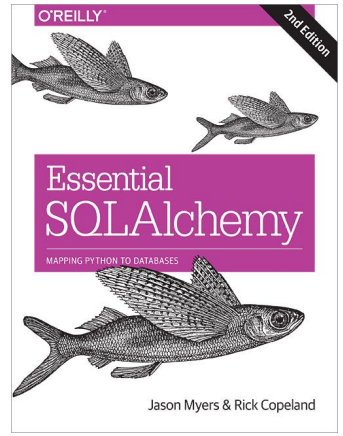
- To prevent SQL injection, *never* concatenate a SQL string with parameters
- Instead, use the right tools and libraries to safely inject parameters for you
- *For Python, use SQLAlchemy*

```
from sqlalchemy import create_engine, text

engine = create_engine('sqlite:///C:\\Users\\thoma\\Dropbox\\rexon_metals.db')
conn = engine.connect()


def customer_for_id(customer_id):
    stmt = text("SELECT * FROM CUSTOMER WHERE CUSTOMER_ID = :id")
    return conn.execute(stmt, id=customer_id).fetchone()


print(customer_for_id(2))
```



More info at:
http://www.sqlalchemy.org/

Georgia State University | COLLEGE OF ARTS & SCIENCES

Preventing SQL Injection

For Java, Scala, Kotlin, and other JVM languages use JDBC's PreparedStatement

```
int customerId = 2;

Connection connection =
        DriverManager.getConnection("jdbc:sqlite:C:\\Users\\thoma\\Dropbox\\rexon_metals.db");

String sql = "SELECT * FROM CUSTOMER WHERE CUSTOMER_ID = ?";

PreparedStatement ps = connection.prepareStatement(sql);
ps.setInt(1,customerId);

ResultSet rs = ps.executeQuery();
rs.next();

System.out.println(rs.getInt("CUSTOMER_ID") + " " +  rs.getString("NAME"));

connection.close();
```

More info at:
http://tutorials.jenkov.com/jdbc/index.html

Attributions

1. Shiva Kumar, https://www.scribd.com/presentation/175876322/DBMS-SQL
2. Dikesh Shah, https://www.scribd.com/presentation/416479214/Data-Analysis

Resources

1. Complete SQL Tutorial: https://www.1keydata.com/sql/sql.html
2. A Gentle Introduction to SQL: https://sqlzoo.net/wiki/SQL_Tutorial
3. A favorite: https://www.w3schools.com/sql/
4. JDBC: *http://tutorials.jenkov.com/jdbc/index.html*

END

Advanced SQL Topics in Software Development

CSc3350

Dr. William Greg Johnson

Department of Computer Science

Georgia State University