CS 199 ACC
Databases

Prof. Robert J. Brunner

Ben Congdon
Tyler Kim
Bhuvan Venkatesh

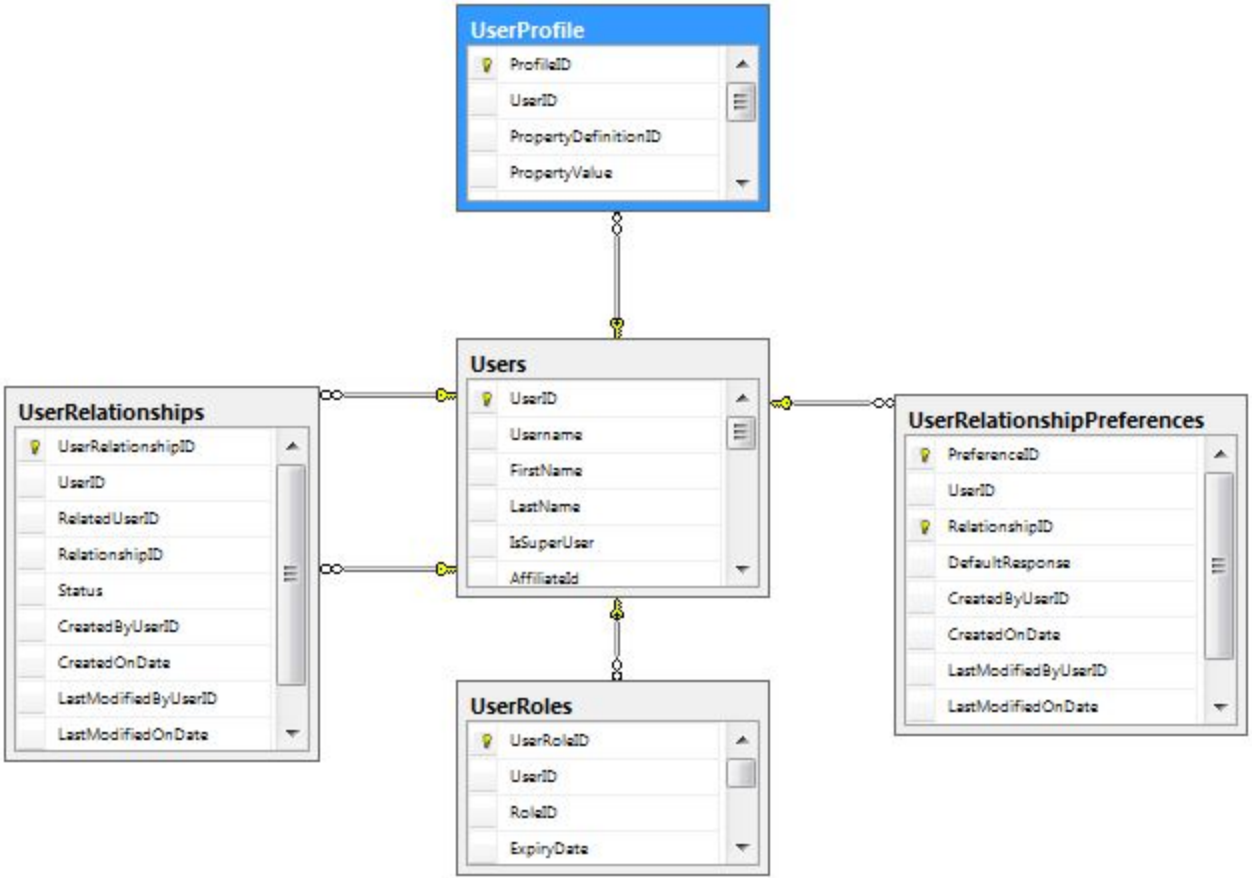
This Week

- Traditional Databases
- SQL
- Optimizations

Traditional Databases

RDBMS

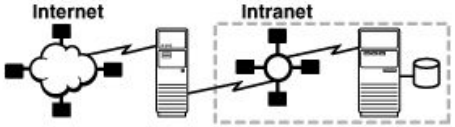
- Relational Database Management Systems or RDBMS is a database management system that deals with relational data (data that points to other data)
- A database management system manages how the data is stored and retrieved. Usually the data is modified with SQL
- Popular RDBMS include MySQL, PostgreSQL, OracleDB, etc

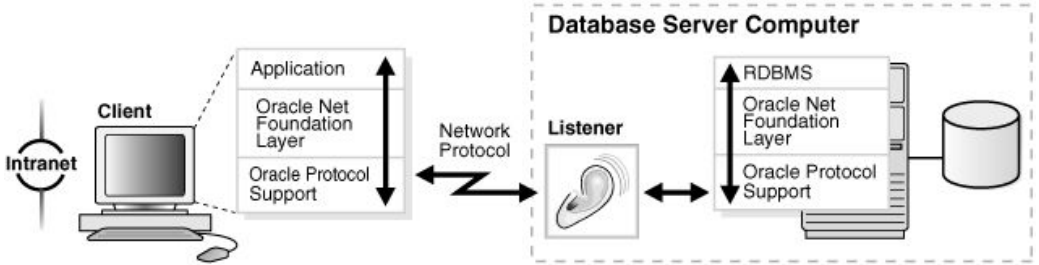


http://www.mytecbits.com/wp-content/uploads/RDBMS.png

Other Features

- RDBMS’s handle data backups, logically storing data, distributing data to leader followers, permissions, data integrity, handling and load balancing queries, and optimization.
- RDBMSs do all of this under the hood, so you can handle your day to day operations





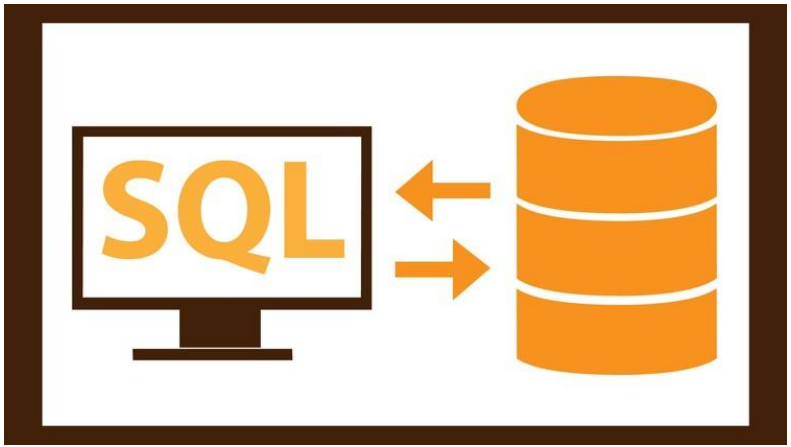
RDBMS Types of Data

- RDBMSs like simple data: INTEGERS, STRINGS, etc
- They don’t like handling JSON, HASHMAP, LISTS
- It makes the SQL databases job harder to optimize so it leaves it out.
- If you think you need this functionality, seriously rethink your application design. If you are absolutely sure that you need it, then you should probably use another application server.

SQL

What is it?

- Most of you have had some interaction with SQL
- SQL stands for the Structured Query Language
- SQL was made for both programmers and for accountants who were used to spreadsheets
- We can imagine taking data from spreadsheets, join from different sheets etc



https://udemy-images.udemy.com/course/750x422/743174_8046_5.jpg

Basic Commands - Data Definition Language (DDL)

- There are two types of SQL commands DDL and DML
- DDL lets you create, destroy, alter, and modify constraints on data
- You can think of them as operations that set up where data will go

- `CREATE TABLE (id INTEGER, name VARCHAR(255), location VARCHAR(255))`
- `ALTER TABLE ADD status INTEGER`
- `ALTER TABLE ADD blah INTEGER NOT NULL`
- `DROP TABLE`

Data Modification Language

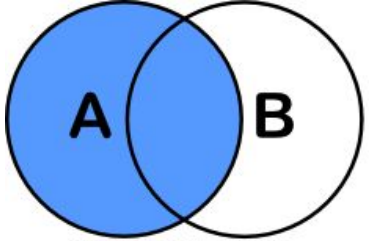
- This adds, deletes, selects, and updates data (basic CRUD operations)
- This lets you put data into the database tables

- `INSERT INTO table (col1, col2, ..) VALUES (v1, v2, ..), ..`
- `DELETE FROM table where col1 = ...`
- `UPDATE table SET col1='asdf' WHERE col2='asd'`
- `SELECT * FROM table`

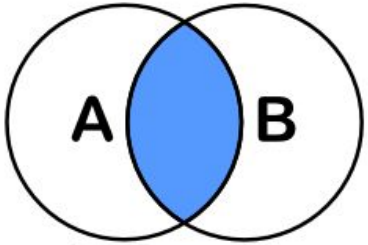
Data modification language extensions

- The data modification language also lets you do more powerful things when retrieving data.
  - We can have data `GROUP BY` a certain column(s)
  - Have data `ORDER BY` some column(s)
  - We can `JOIN` multiple spreadsheets based on a column
- We can have SQL calculate functions or aggregations on the fly
- Usually RDBMSs are optimized for read heavy workloads, especially because SQL provides so many ways of reading data into the database

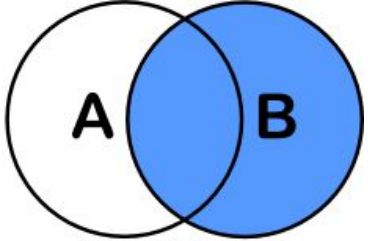
CHEATSHEET SQL JOINS



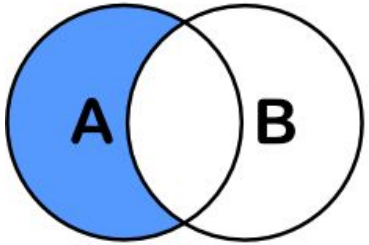

```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
```



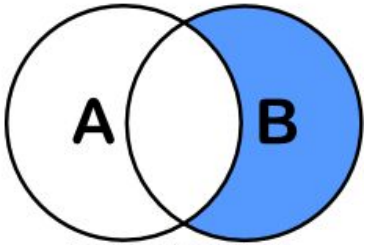

```
SELECT <auswahl>
FROM tabelleA A
INNER JOIN tabelleB B
ON A.key = B.key
```



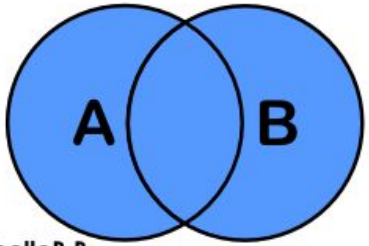

```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
```



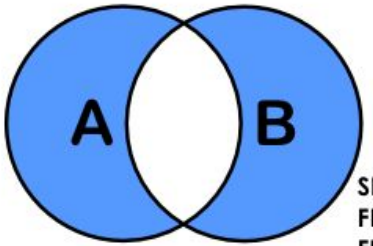

```
SELECT <auswahl>
FROM tabelleA A
LEFT JOIN tabelleB B
ON A.key = B.key
WHERE B.key IS NULL
```




```
SELECT <auswahl>
FROM tabelleA A
RIGHT JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
```




```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
```




```
SELECT <auswahl>
FROM tabelleA A
FULL OUTER JOIN tabelleB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

https://dsin.files.wordpress.com/2013/03/sqljoins_cheatsheet.png

SQL prepared statements

- One last things about sql is prepared statements. They aren’t an inherent feature of the original SQL but they deal with the actual interactions with the database. Consider the following query
- `INSERT INTO table VALUES (`+userid+`);`
- What if userid = “1; SELECT * FROM table WHERE col1 NOT IN (“
- `INSERT INTO table VALUES (1); SELECT * FROM table WHERE col1 NOT IN ();`
- This will give us back all the results from the database!

SQL Prepared Statements

- To avoid this, we have prepared statements
- `INSERT INTO table VALUES (?)` and you send userid separately
- This avoids the injection problem but doesn’t let sql server optimize database queries
- More specifically, you cannot optimize queries because you don’t know the queries to optimize (usually these are select statements and whatnot not usually INSERT statements)

Optimization

SQL Turing Completeness

- The 30-second overview of this is that every SQL statement (in ANSI SQL) will terminate
- This is important because in programming language theory, this may not always be the case. Since this may not always be the case, we cannot optimize general programs super effectively.
- The Non-Turing Completeness of SQL let’s us optimize it away

User tips for optimizing SQL queries

- Don’t use `SELECT *` statements, you usually are selecting more rows than need be
- If you have multiple levels of joins then you may want to consider staging your data into an intermediate table in order to reduce communication overhead
- Add indexes! Indexes slow down updates but drastically speed up complex queries if the indexes are on the appropriate columns

SQL Optimizer: Prediction

- Consider a query like `select col1 from table where col1=1 AND col2=2;`
- Your server has the choice of filtering by col2 and then col1 or by col1 then col2.
- If the server knows that there are a lot of NULL values in col2 which would reduce the number of rows in consideration a lot, it will filter based on col2 first and then filter on col1 because the complexity will be NUM_ROWS * SMALL_NUMBER

SQL Optimizer: Lazy Joins

- A join is when you combine two tables on a column

c1	c2
1	2
2	4

c3	c4
1	1
2	1
3	2

Example Join

SELECT * FROM t1 JOIN t2 USING (c1, c4);

c1	c2	c3	c4
1	2	1	1
1	2	2	1
2	4	3	2

Lazy Join

- SQL may filter the data before joining, may group by before joining if you know that one of the columns is in one of the table
- This is very ad-hoc prediction because SQL usually doesn’t keep track of super in depth statistics
- As a SQL server runs longer, then it gets better at this prediction
    - The main reason that it can’t keep track of all of this information is due to concurrency bottlenecks so it makes static analyses instead

Spark SQL

- Basically use SQL in a spark database. Just don’t UPDATE, INSERT, or DELETE. (Spark SQL is really good for retrieving data)
- Does all of the optimization that you love
- Doesn’t do JOINs well so don’t do those
- Pretty much a compatibility layer for people who are using RDBMS systems to consider switching over
- Useful for the lab!