Week 10: Relational Databases and SQL

LSE MY472: Data for Data Scientists
https://lse-my472.github.io/

Autumn Term 2024

Ryan Hübert

Slides last updated: 30 September 2024

Outline

➜ **Relational** vs non-relational databases

➜ **S**tructured **Q**uery **L**anguage

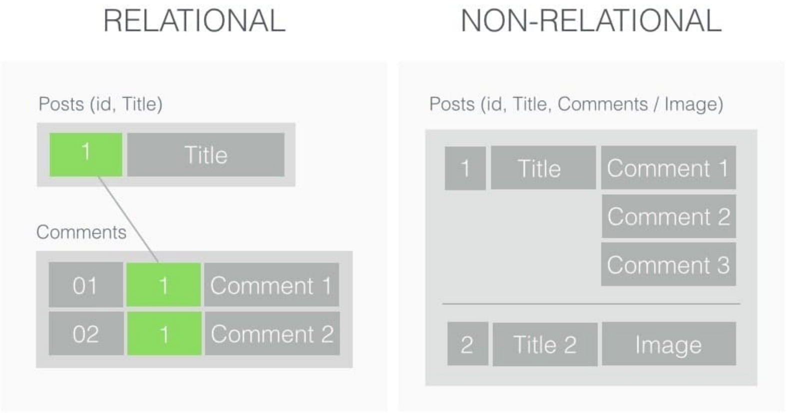
➜ Coding session

Relational vs non-relational databases

Databases

- **Database system**: An organized collection of data that is stored and accessed via a computer
- **Relational databases**: Data stored in multiple tables to avoid redundancy. Tables are linked based on common keys
- **Non-relational databases:** Data stored in a way that is not based on tabular relations (e.g. MongoDB uses JSON like documents)

Relational vs non-relational databases



RELATIONAL

Posts (id, Title)

1 Title

Comments

01 1 Comment 1

02 1 Comment 2

NON-RELATIONAL

Posts (id, Title, Comments / Image)

1 Title Comment 1

Comment 2

Comment 3

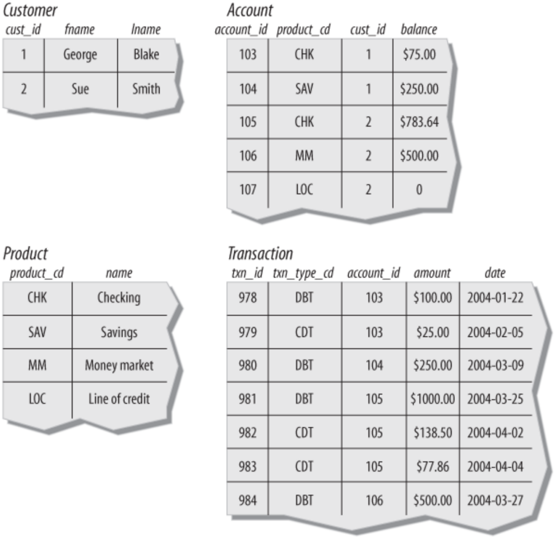
2 Title 2 Image

From: Codewave Insights

Relational databases

- **Relational Database Management Systems (RDBMS)**:
  - The underlying software system used to maintain relational databases
  - Examples: MySQL, PostgreSQL, SQLite, MariaDB, etc.
- **Online Transaction Processing (OLTP) Services**:
  - High frequency (many transactions per minute), fast response, many write operations
  - Examples: Amazon RDS, Google Cloud SQL, Azure SQL Database
- **Online Analytical Processing (OLAP) Services**:
  - Large volume (petabytes of data), lower frequency (few transactions), slower response, mostly read operations
  - Examples: Amazon RedShift, Google BigQuery, Microsoft Azure SQL Server, Snowflake

Relational databases in action



Customer

cust_id	fname	lname
1	George	Blake
2	Sue	Smith

Account

account_id	product_cd	cust_id	balance
103	CHK	1	$75.00
104	SAV	1	$250.00
105	CHK	2	$783.64
106	MM	2	$500.00
107	LOC	2	0

Product

product_cd	name
CHK	Checking
SAV	Savings
MM	Money market
LOC	Line of credit

Transaction

txn_id	txn_type_cd	account_id	amount	date
978	DBT	103	$100.00	2004-01-22
979	CDT	103	$25.00	2004-02-05
980	DBT	104	$250.00	2004-03-09
981	DBT	105	$1000.00	2004-03-25
982	CDT	105	$138.50	2004-04-02
983	CDT	105	$77.86	2004-04-04
984	DBT	106	$500.00	2004-03-27

Some vocabulary

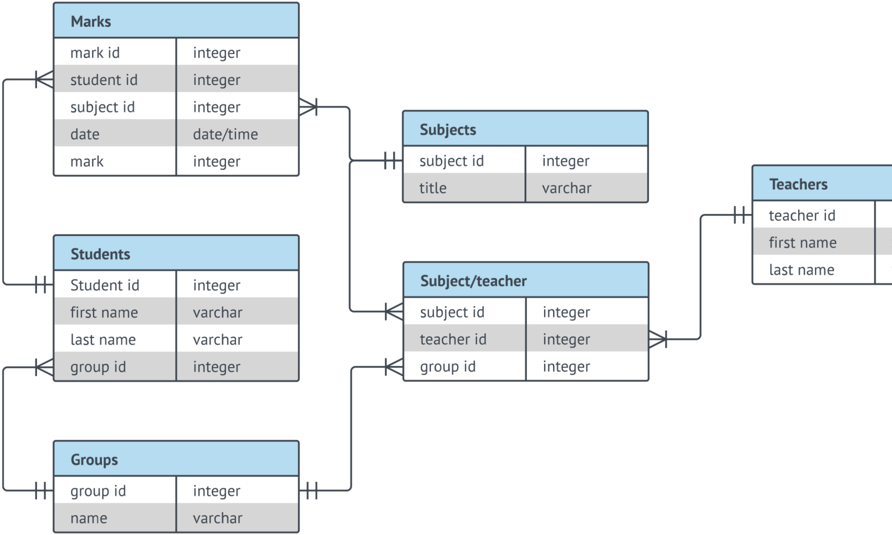
Relational database term	SQL term
Relation	Table
Tuple, record	Row
Attribute, field	Column

(Excerpt from: https://en.wikipedia.org/wiki/Relational_database)

Keys

- Keys are **critical**, allowing the rows of different tables to be connected
- Primary key: A column or set of columns (composite key) which uniquely identifies each row/record in the table
- Foreign key: A primary key of another table

Entity relationship diagrams (ERDs)



Marks	
mark id	integer
student id	integer
subject id	integer
date	date/time
mark	integer

Subjects	
subject id	integer
title	varchar

Teachers
teacher id
first name
last name

Students	
Student id	integer
first name	varchar
last name	varchar
group id	integer

Subject/teacher	
subject id	integer
teacher id	integer
group id	integer

Groups	
group id	integer
name	varchar

From: Lucidchart

Structured Query Language

SQL: Structured Query Language

- **Language** designed to define, control access to, manipulate, and query **relational databases**
- Initially written SEQUEL (Structured English Query Language), but later changed to SQL because of trademark issues
- Pronounced both S-Q-L and SEQUEL today
- It is a **nonprocedural/declarative language**: User defines what to do, inputs, and outputs, but not the control flow; how the statement is executed, is left to the *optimizer*
- How long SQL queries depends on optimization that is opaque to user
- Performance will vary, but generally faster than standard data frame manipulation in R (and much more scalable)

Some common components of SQL queries

- The result of a SQL query is a table
- **SELECT** columns
- **FROM** a table in a database
- **WHERE** rows meet a condition
- **GROUP BY** values of a column
- **ORDER BY** values of a column when displaying results
- **LIMIT** to only X number of rows in resulting table
- Always required: **SELECT** and **FROM**; rest are optional
- **SELECT** can be combined with operators such as **SUM**, **COUNT**, **AVG**. . .

Some more components of SQL queries

- To merge multiple tables, use **JOIN**
  - Variety of ______ **JOIN** types: **INNER**, **RIGHT**, **LEFT FULL OUTER**
  - For anti-joins, use **RIGHT** or **LEFT** and a **WHERE** clause
  - When handling multiple tables, use aliases (e.g. **FROM table AS t**)
- More complex ways of combining tables include (non-exhaustive):
  - **CROSS JOIN**: Produce all combinations of the two ids
  - **UNION**: De-duped vertical combination of both tables (add **ALL** for dupes)
- SQL also supports common table expressions (CTEs):
  - Lets you build multiple sub-tables within a single query
  - Connect these together with a subsequent **SELECT** statement

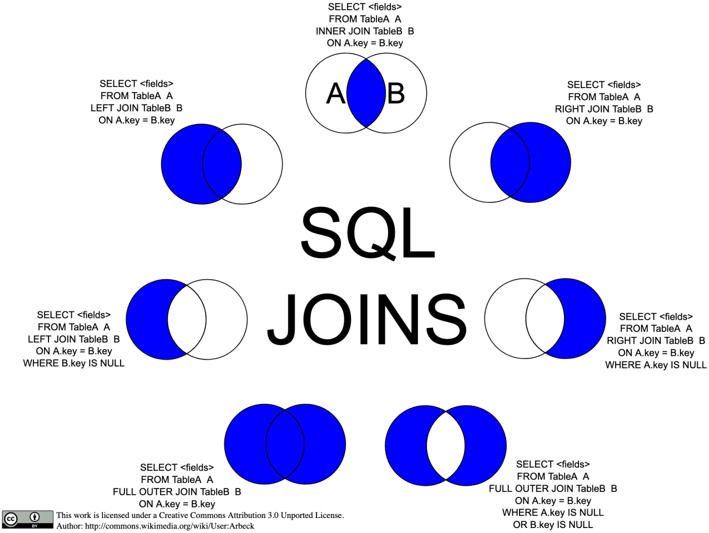
SQL query examples

```sql
SELECT name, account_id FROM client;
```

```sql
SELECT * FROM client WHERE gender = 'F';
```

```sql
SELECT SUM(billed) AS total_billed,
       AVG(billed) AS avg_billed
FROM client
WHERE gender = 'F';
```

SQL JOINs



```
SELECT <fields>
FROM TableA  A
INNER JOIN TableB  B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA  A
LEFT JOIN TableB  B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA  A
RIGHT JOIN TableB  B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA  A
LEFT JOIN TableB  B
ON A.key = B.key
WHERE B.key IS NULL
```

```
SELECT <fields>
FROM TableA  A
RIGHT JOIN TableB  B
ON A.key = B.key
WHERE A.key IS NULL
```

```
SELECT <fields>
FROM TableA  A
FULL OUTER JOIN TableB  B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA  A
FULL OUTER JOIN TableB  B
ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

From: https://upload.wikimedia.org/wikipedia/commons/9/9d/SQL_Joins.svg

SQL JOIN examples

```sql
SELECT client.name, account.balance
FROM client JOIN account
ON client.account_id = account.id;
```

```sql
WITH

 cte_one AS (
  SELECT * FROM client WHERE gender = 'F'
 ),

 cte_two AS (
  SELECT * FROM sales
 )

SELECT co.account_id, ct.sales_count, ct.sales_revenue
FROM cte_one AS co
INNER JOIN cte_two AS ct
ON co.account_id = ct.acc_id;
```

Coding session

Coding session

Download from moodle:

➜ `public Facebook data (individual csv files)`

Code:

➜ `01-sql-intro.Rmd`

➜ `02-sql-join-and-aggregation.Rmd`

General information on how to connect to SQL databases with R:
https://solutions.rstudio.com/db/