LUT University

LAND OF THE CURIOUS

CT60A4304 - BASICS OF DATABASE SYSTEMS

INTRO TO DATABASES

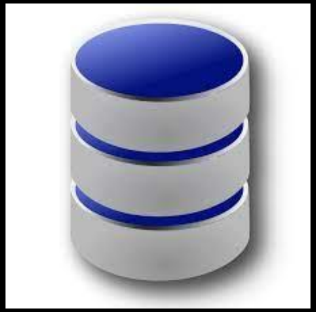
Lecture

Jiri Musto, D.Sc.

TABLE OF CONTENTS

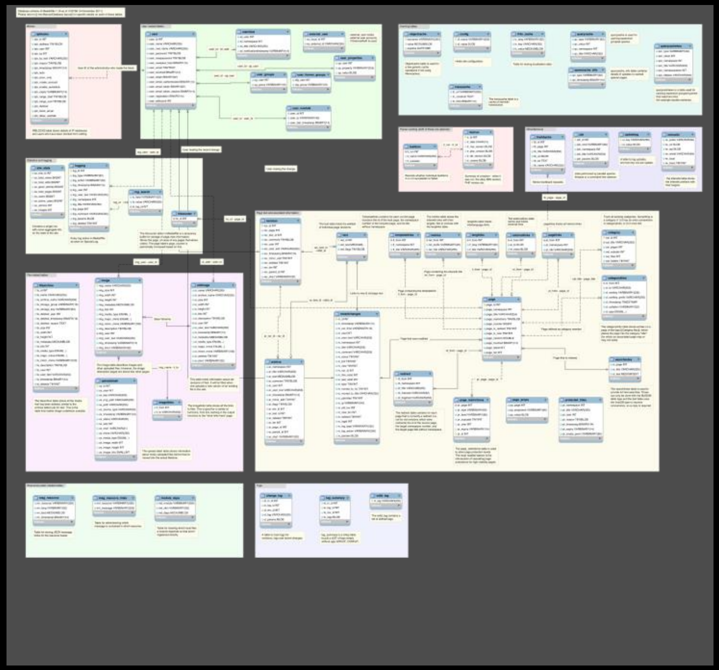
WHAT IS A DATABASE?

- In theory - Anything that stores interrelated data
  - Sheets of paper, files (excel sheets, text files), databases, etc.
- In practice - Has the following properties:
  1. Database illustrates the real world. If changes happen in the real world that the database illustrates, the changes are reflected in the database.
  2. Database is a logically coherent collection of data that has a natural significance. Random subgroup of data from a database is not a database.
  3. The structure and data are well designed and implemented for a specific purpose and the database has a target userbase and application.



MOTIVATION: WHY?

- Databases are used everywhere
- Data storage is a complex issue
  - Storing data in each software requires repeating the same functionality
  - Storing data in software limits the access to data
- Advantages of database systems:
  - Efficiency
  - Security
  - Portability
  - Reliability (ACID: Atomic, consistent, independent, durable)



MOTIVATION: WHEN?

- Multiple users
  - Simultaneous users, different users, different access rights
- Data changes over time
  - Add new data, delete old data, make changes to existing data
- Large amount of data
  - The amount is not feasible or reasonable to store in software code
- Multiple data sources
  - Homogenous sources, database
  - Heterogenous sources, data warehouse

MOTIVATION: WHERE?

- Three basic solutions:

1. Local device storage, clientside
    - Can be a database or flat file
    - Widely used in: mobile apps and temp files on computers
2. Physical database, serverside
    - Server can be in local network or accessed remotely
    - Used in businesses, games, universities, etc.
3. Cloud database, cloud platform
    - Data is accessed only through cloud platform
    - Many companies offer their services: Amazon, Google, Heroku, Oracle

ADDITIONAL INFORMATION: HISTORY

1960s

- Navigational database

1970s

- Relational DBMS

Late 1970s

- SQL

1990s

- Object-orientet

2000s

- NoSQL, NewSQL

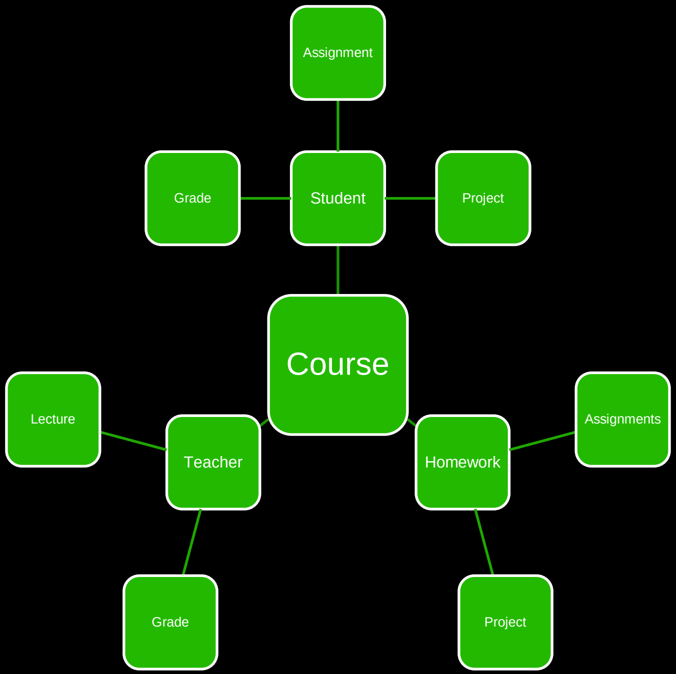
ADDITIONAL INFORMATION: DATABASE TYPES

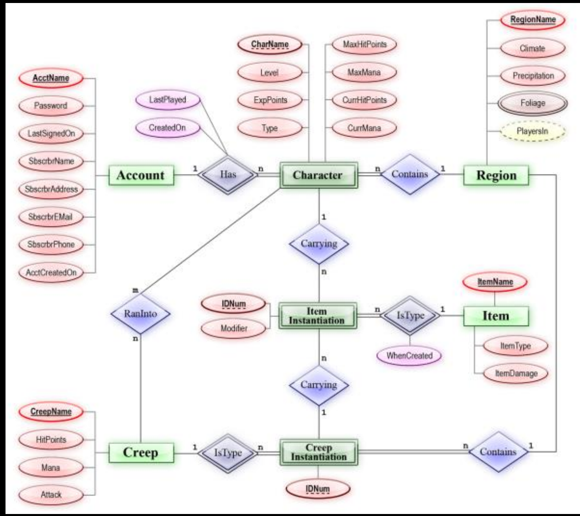
- Relational databases – Based on relational model and most use SQL
- Flat file – TXT, Excel, CSV, (JSON, XML)
- Object-oriented
- NoSQL
  - Key-value store
  - Document store – A specialized key-value store, most often JSON or XML documents
  - Graph
- NewSQL – Main usage is online transaction processing (OLTP)

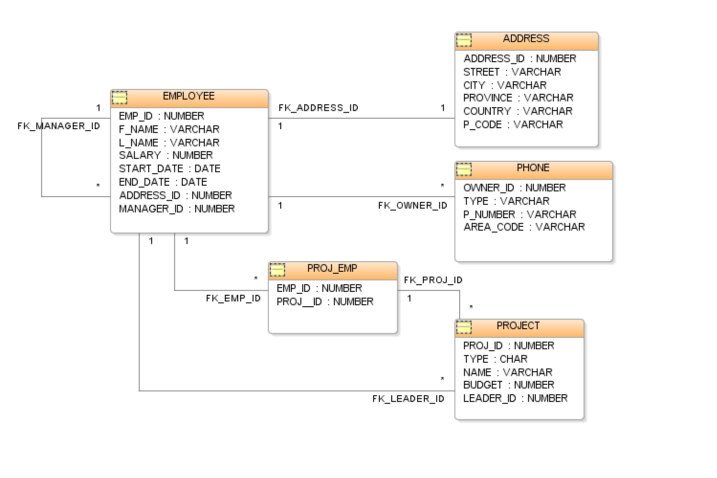
CONCEPTS: DATA MODELS

- Data model is an abstract definition of the data structure, its operations, and other related concepts. Data models can be divided into three categories
- Conceptual: The parts of data are expressed on a higher level.
  - ER models, mind map
  - Design based on an abstract idea
- Logical: Data is described without a connection to the physical implementation.
  - Database diagrams / schemas
  - Conceptual model transformed into a more concrete form.
- Physical: How data is composed / stored
  - Files, indices, hard drive segments
  - Design decisions based on security, technical requirements, recovery, scalability

CONCEPTS: DATA MODELS







CONCEPTS: DATABASE AND DBMS

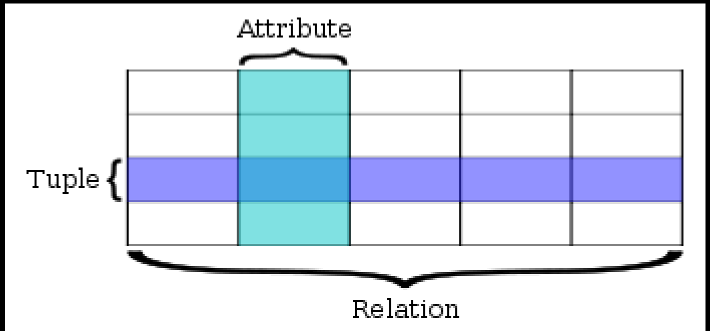
- Database is a collection of related data. Generally speaking, it has the following properties:
  - Reflects the real world. Is there are changes in the real world, the database is modified accordingly.
  - Logically uniform collection of data that has a purpose
  - Data and its structure are designed and implemented for a specific userbase and software
- Database management system (DBMS) is the software that oversees the database
  - Can define the structure of the database
  - Can search, insert, delete, alter data
  - Defines the access protocols, maintenance and sharing of the database
  - Oracle, DB2, SQL Server, Postgres, MySQL, SQLite, Hadoop, MongoDB, Cassandra, HBase, etc.

CONCEPTS: DATABASE SYSTEM

- Database system is the combination of:
  - Database
  - Database management system (DBMS)
  - Software(s) that use the aforementioned
- Database system can include one or multiple of the subparts

EXAMPLES: DATA TABLE

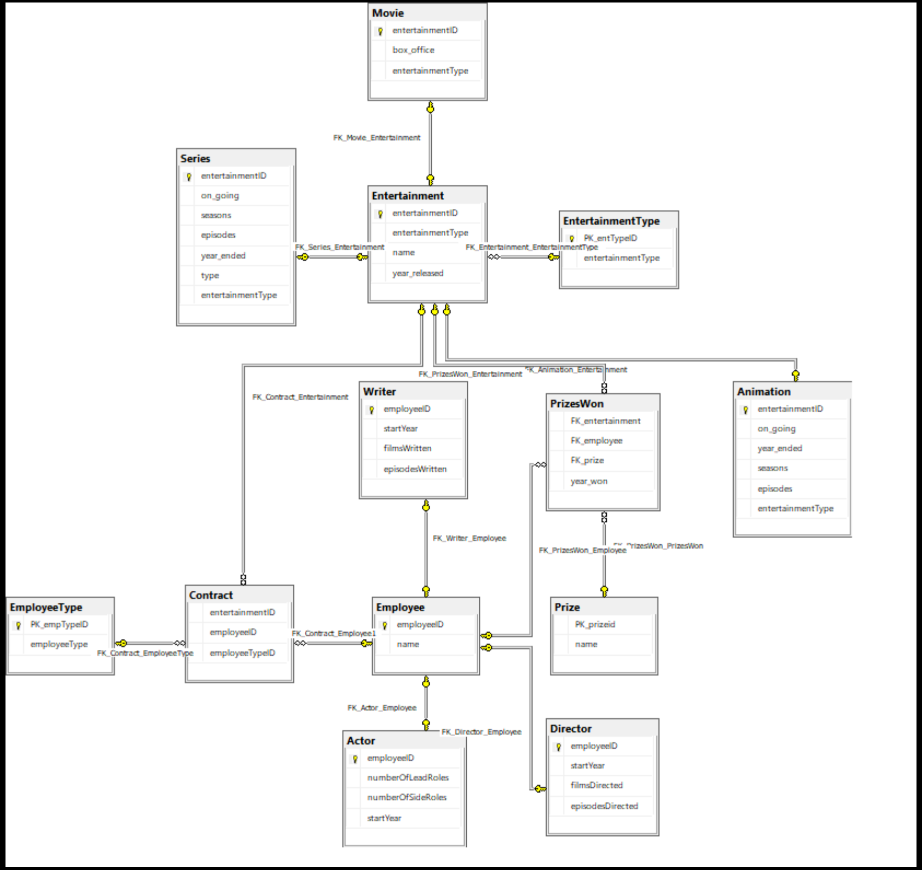
- Relational model vs SQL terminology
  - Tuple = Row
  - Attribute or field = Column
  - Relation = Table



	Movie	year_released	Employee	employeeType
1	Iron Man	2008	Robert Downey Jr.	Actor
2	The Avengers	2012	Joss Whedon	Director
3	Captain America, Civil War	2016	Robert Downey Jr.	Actor
4	Captain America, Civil War	2016	Tom Holland	Actor
5	Doctor Strange	2008	Benedict Cumberbatch	Actor
6	Avengers: Endgame	2008	Robert Downey Jr.	Actor
7	Avengers: Endgame	2008	Tom Holland	Actor
8	Avengers: Endgame	2008	Benedict Cumberbatch	Actor
9	The Witcher	2019	Henry Cavill	Actor
10	Sherlock	2010	Benedict Cumberbatch	Actor
11	The Office	2005	Paul Feig	Director
12	Friends	1994	Gary Halvorson	Director
13	Friends	1994	David Crane	Writer
14	Supernatural	2005	Robert Singer	Director
15	Supernatural	2005	Eric Kripke	Writer
16	Family Guy	1999	Seth MacFarlane	Actor
17	Family Guy	1999	Seth MacFarlane	Writer
18	One Piece	1999	Eiichiro Oda	Writer

EXAMPLE: ENTERTAINMENT DATABASE

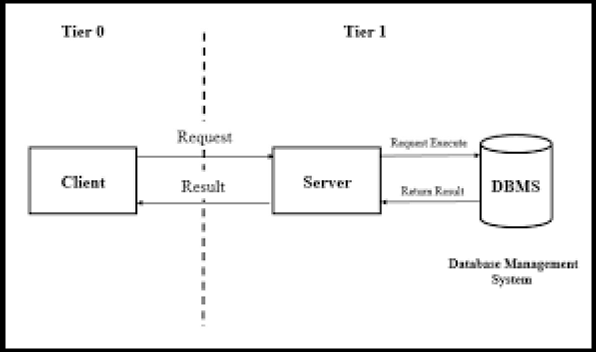
- Who worked in what movie/series?
- What prizes films have won?
- How many have worked as actors and writers?
  - In the same movie/series?
  - And won prizes from their work?
- With relational databases, these questions can be answered using SQL queries
  - Some are simple
  - Some are complex



OTHER FACTS

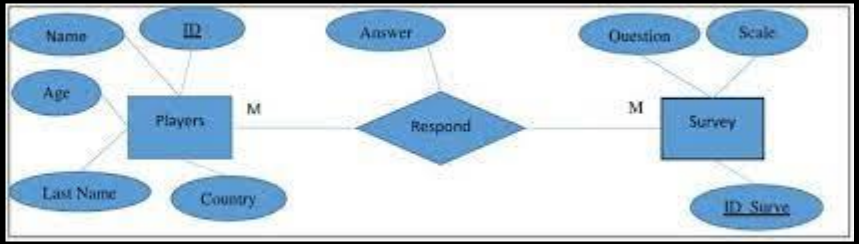
- Database vs. Cache
  - Short-term vs. Long-term memory
  - Database is used for a long-term storage
  - Cache (e.g. Browser cache, RAM) is for a short-term storage and quick access
- Data vs. Information
  - Data is meaningless raw input such as numbers and text.
  - Information is obtained when context is added to data.
  - Data and information are often used interrelated when it is not important to differentiate between them

WHAT IS TO COME: DATA MODELS, ARCHITECTURE

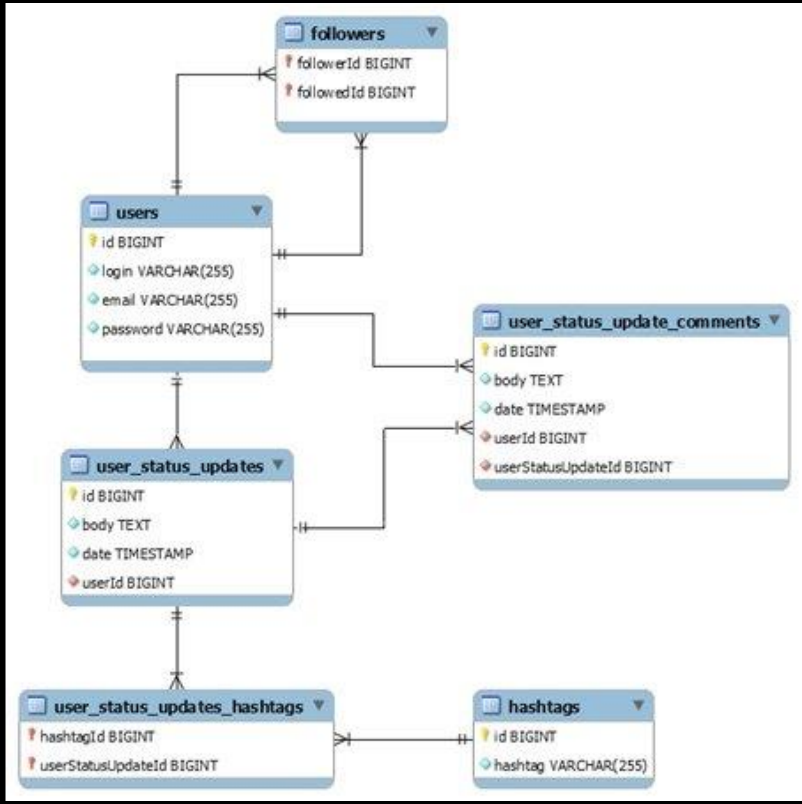


Architecture

ER-model



WHAT IS TO COME: RELATIONAL MODELS



WHAT IS TO COME: STRUCTURED QUERY LANGUAGE (SQL)

- SQL is used for managing relational databases
- Syntax resembles English
  - SELECT name FROM employee;
- SQL is easy
  - ...Right?

WHAT IS TO COME: STRUCTURED QUERY LANGUAGE (SQL)

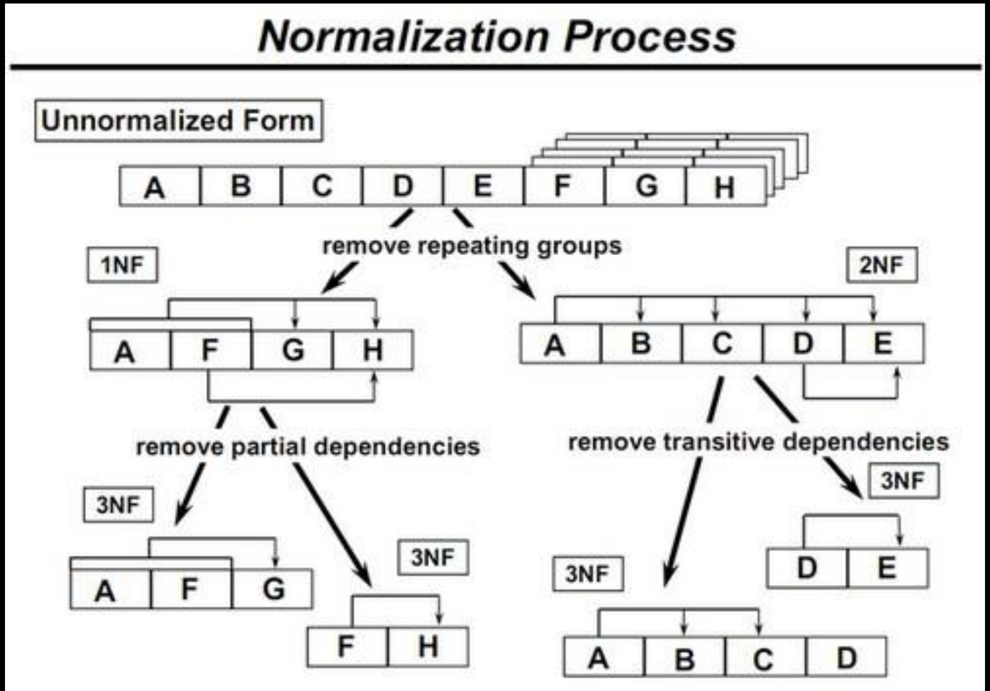
```sql
SELECT
    country.country_name_eng,
    SUM(CASE WHEN call.id IS NOT NULL THEN 1 ELSE 0 END) AS calls,
    AVG(ISNULL(DATEDIFF(SECOND, call.start_time, call.end_time),0)) AS avg_difference
FROM country
LEFT JOIN city ON city.country_id = country.id
LEFT JOIN customer ON city.id = customer.city_id
LEFT JOIN call ON call.customer_id = customer.id
GROUP BY
    country.id,
    country.country_name_eng
HAVING AVG(ISNULL(DATEDIFF(SECOND, call.start_time, call.end_time),0)) > (SELECT AVG(DATEDIFF(SECON
ORDER BY calls DESC, country.id ASC;
```

Source:https://www.sqlshack.com/learn-sql-how-to-write-a-complex-select-query/

WHAT IS TO COME: STRUCTURED QUERY LANGUAGE (SQL)

```sql
SELECT P1.Name, P2.Name
    FROM Person P1, Person P2, Student S1, Student S2, Enroll E1, Enroll E2
    WHERE P1.Name = S1.Name AND P1.DateOfBirth = S1.DateOfBirth
    AND S1.StudNo = E1.StudNo AND E1.Grade IS NOT NULL
    AND P2.Name = S2.Name AND P2.DateOfBirth = S2.DateOfBirth
    AND S2.StudNo = E2.StudNo AND E2.Grade IS NOT NULL
    AND S1.StudNo < S2.StudNo AND
    NOT EXISTS (
            SELECT * FROM Lecture AS L
                WHERE L.CourseNo IN
                (SELECT B.CourseNo FROM Enroll AS B
                    WHERE S1.StudNo = B.StudNo
                    OR S2.StudNo = B.StudNo)
                AND
                NOT EXISTS (
                    ( SELECT * FROM Enroll AS B1
                        WHERE S1.StudNo = B1.StudNo
                            AND B1.CourseNo = L.CourseNo)
                        UNION
                        (SELECT * FROM Enroll AS B2
                            WHERE S2.StudNo = B2.StudNo
                            AND B2.CourseNo = L.CourseNo)
                    )
                );
```

WHAT IS TO COME: NORMALIZATION



Source: https://learn.saylor.org/mod/page/view.php?id=23139

WHAT IS TO COME: SQL AND PYTHON

```python
import sqlite3
conn = sqlite3.connect('example.db')

c = conn.cursor()

# Create table
c.execute('''CREATE TABLE stocks
             (date text, trans text, symbol text, qty real, price real)''')

# Insert a row of data
c.execute("INSERT INTO stocks VALUES (?,?,?,?,?)", ('2006-01-05','BUY','RHAT',100,35.14))
c.execute("INSERT INTO stocks VALUES (?,?,?,?,?)", ('2006-01-05','BUY','NOKIA',106,24.12))

# Save (commit) the changes
conn.commit()

for row in c.execute('SELECT * FROM stocks ORDER BY price'):
    print(row)

# We can also close the connection if we are done with it.
# Just be sure any changes have been committed or they will be lost.
conn.close()
```

WHAT IS TO COME: INDEX AND OPTIMIZATION

- Indices are used for optimization
- In general: No need to go through all rows when indices point to the correct ones
- How they actually work (B+ tree algorithm) is outside the scope of this project
  - Better to consider how indices function in different scenarios


LUT
University