Databases for data analytics

https://github.com/evidencebp/databases-course/

Data integrity

Guidelines

גרוע מהטיפש, ההולך אחריו

- In data context - if you did not validate the data, at least assume it is wrong
- Data can be wrong and misleading in many ways
- It is better to prevent opportunities to error than to fix mistakes

Problems with IMDB

- No recent movies?
- No movies with long names?
- Did two actors the played “Himself” play the same role?
- Is ‘Maria’ the same person in all movies?
- Are roles case sensitive?
- What is the difference between a ‘Receptionist’ and a ‘Secretary’?
- Is ‘Himself - host’ role like ‘Himself’, ‘Host’, both or none?
- Who should come first, if any, ‘Superman/Clark Kent’ or ‘Clark Kent/Superman’?
- Under which name we should search for Madonna?
- Do we really need directors_generes? Is it harmless?

Database can help us with some of the problems

- With semantics we are generally alone
- We can and should
    - Use nulls properly
    - Avoid duplicates
    - Choose appropriate data type
    - Set proper constraints
    - Avoid redundancies
    - Use mechanisms to protect the data from future changes

Null - a very special value

- Semantically mean either “unknown” or “irrelevant”
- Null != Null since we don’t know their value
- Therefore, should be checked with “X is null” and not as “X = Null”
- When a column must have a value, add a constraint that prevents nulls

Avoid duplicates

- Unique keys prevent duplicates
- However, some entities are not in our control
  - There are many movies of the same name
  - Some are from the same year too
  - In this case our “movie_id” just allow to represent both, with their related relations
- Primary and unique keys are also indices
  - making search faster
  - Making updates slower
  - Require more storage

Choose appropriate data type

- Choose the minimal data type the can represent your needed value
  - Using a string for a number can lead to values like ‘NINE’
  - A too long string waste storage and hurts performance
- If there is a default value that is suitable (e.g., 0, current date) use it. Otherwise, enforce providing a value.
- You can add additional constraints upon a type (e.g., numbers up to million).

Set proper constraints

- We want only actors to play in roles
- We can use foreign keys to enforce the actor_id column in role to include only ids from the table of actors
- The possible values change as actors are added or deleted
- Foreign keys are usually “hinting” for join in SQL
- Foreign keys also slow down performance
- An actor should be added before adding the actor’s roles
- Foreign keys cannot check that all players played in a certain role

Avoid redundancies

- The data in directors_generes can be based on movies' directors and genres
- Its existence trades off computation time and storage
- If the basic tables will be updated, directors_generes will have to be recomputed
- In On Line Transaction Processing it is avoided.
- In On Line Analysis Processing it is very common

Use mechanisms to protect the data from future changes

- MySql allows to run triggers on events (e.g., inserting a row to a table).
- One can write stored procedures, functions, that will handle the trigger event.
- That allows to write code that protects the database when data is changed, validating it is kept correct.
- The validation is not dependent on external applications using the database.
- This validation might hurt performance
- The writing of triggers and stored procedure is out of the course scope.

Information schema

- Contains metadata on database (e.g., tables, columns)
- Very helpful to generate queries on all tables
- Such queries can be validation queries

In class exercises

- Generate queries to check each column for being unique
- Check if all directors in movies_directors are directors. If so, how come?
- Find the directors that did not directed. How can you prevent such cases?

Exercises

- Data integrity