INTRODUCTION TO DATABASES

DATABASE

- A database is a collection of data stored electronically, usually organized and structured in some form.
- Databases can be very simple – a text file or a CSV file, for example.
- When they are complex, we use formal methods of design and engineering principles, to build, update and maintain these systems.
- A DataBase Management System (DBMS) is the software used to interact between the database and an application that uses the database.
- Data in databases is usually stired as tables.

DIFFERENCES BETWEEN TABLES AND DATABASES

- When we think of databases, we often think of tables of information.
- However, we encounter several types of tables in software, and they focus on different aspects.
- HTML tables
  - Focus on display – tables have no basis of "identity" for data
- Spreadsheets
  - Focus on positional data – fields are arranged and calculated based on their position in the spreadsheet
- Databases
  - Focus on the data itself – metadata is used to identify data fields.
- Data stored in transience during program execution (classes, structs, etc) do not count, since they do not offer permanence.

DATA IN 3 LEVELS

- Level 1 – The actual database
  - A file that stores all the information. The format depends on the type of the Database)text, relational, graph, etc.)
- Level 2 – The DataBase Management System (DBMS) – interaction between data and an application
  - A software package that helps to update, maintain and secure the data.
- Level 3 – Application
  - Any software package that is built on top of the other 2 layers that provides interaction with the user. A web app, a mobile app, BI/ERP, etc.

TYPES OF DATABASES

- Flat Model
- Navigational databases
- ▫ Hierarchical (tree) database model
- ▫ Network/Graph model
- Relational Model
- Object model
- Document model
- Entity–attribute–value model
- Star schema

DISADVANTAGES OF FILE STORAGE

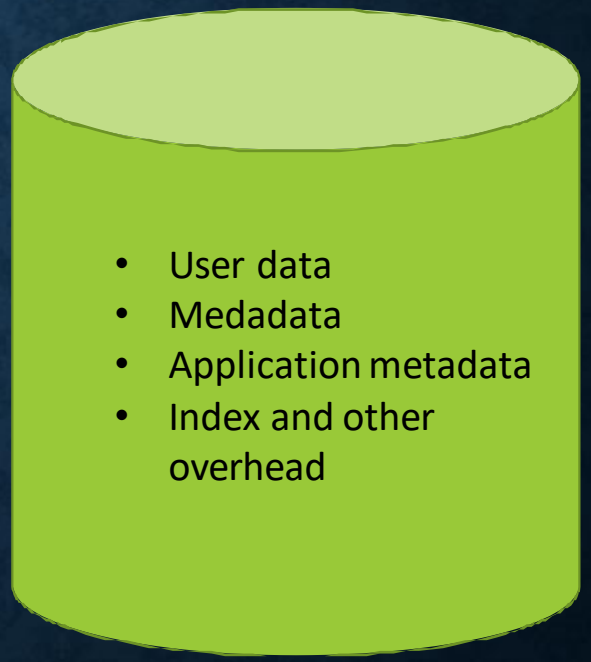
- Redundancy – We might have multiple copies of the same data.
- Inconsistency – Data on the same business entity may appear in different forms, for example, state name, phone number, etc. This makes it hard to modify data and track changes.
- Mixture of data – No clear logical relationships between the data columns, making it hard to understand and manage complex structures.
- Hard to maintain and manage
- No concurrency support – limiting scalability

HOW DO DATABASES HELP?

- Self-describing data collection of related records (meta data, data about data)
- Databases contains not just data, but also include the structure of data
- At the application level, databases can also store other application related meta data.
- This facilitates the personalization and customization of the application according to a user's profile.
- Two of the most important roles in defining metadata
  - Identify the type of data with a unique tag
  - Define the affinity of the data (tags enclose all data that is logically related)

CONTENTS OF A DATABASE

- User Data: The actual data used by the application
- Meta data: The structure (schema) of the data, including table names, column names and types, and constraints over the column(s)
- Application meta data: Meta data on user settings or functions of the application
- Index and other overhead data: used for performance improvement and maintenance. Logs, tracks, security, etc.



3 ACRONYMS

- ACID
  - 4 essential properties that a DBMS has to implement to effectively interact with a database
- SQL – Structured Query Language
  - A query is a result of a request to the database for information or for modification.
  - SQL is the language most relational databases (the most common for of databases) use for interaction with the data.
- CRUD – Create, Read, Update, Delete
  - Self-explanatory
  - The 4 major operations performed on the database.

ACID

- Atomicity – transactions are either all or none (commit/rollback)
- Consistency – only valid data is saved
- Isolation – transactions should not affect each other
- Durability – written data will not be lost

- Good example : Bank transactions

- Most of the challenges related to ACID compliance come from multiple users/concurrent use of the database

RELATIONAL DATABASES - RELATIONS

- A relational database describes the relationships among different kinds of data
  - Captures ideas like those defined in the Affinity and Collection rules
  - Allows software to answer queries about them
- Any relational DB can be described in XML
  - But it is not the case that every XML description defines a relational DB

RELATIONS (TABLES)

- Tables (formally called ‘relations’) – are the building blocks of relational databases.
- Tables store data in 2D, where each row reflects one instance of a record, and each column reflects one aspect of the attributes for all instances.
- Rows may also be called ‘tuples’.
- Columns may also be called ‘fields’.

- For example, a ‘student’ table may contain (student id, first name, last name, grade, school name, home address, …), and each row may represent one student’s information, and each column of the table represents one piece of information for all students. And this is called a ‘relation’.

ENTITIES

- Entities are theoretical constructs that help us in conceptualization of relations.
- Anything that can de identified by a fixed number of its characteristics (attributes)
  - Attributes have names and values
  - Attributes have a data type
- A relational database table can be empty (NULL instance)
- The values are the data that's stored in the table
- An entity could be defined by a relation (table) or be spread across several relations.
- Instances are Unordered
  - Order of the rows and columns does not matter in databases
  - Freedom to move the data is limited to exchanging entire rows or exchanging entire columns

PROPERTIES OF ENTITIES

- Uniqueness
  - No two rows can be the same
  - Two rows can have the same value for some attributes, just not all attributes
- Atomic Data
  - Values stored for attributes
  - Not decomposable into any smaller parts. Separate fields for street, city, state, postal code
  - "Only atomic data" rule relaxed for certain types of data. Dates, times, currency

PROPERTIES OF ENTITIES

- Keys
  - Any set of attributes for which all attribute values are different is called a candidate key
  - Pick one and call it the primary key to decide uniqueness
  - Key must distinguish all potential and actual entities, not just those that happen to be in the table at a given time
  - If no combination of attributes qualify as a candidate key, assign a unique ID to each entity

KEYS - PRIMARY KEY AND FOREIGN KEY

- Primary key: Unique Identifier made of one or more columns to uniquely identify rows in a table.
  - If the primary key contains more than one column, it can be called 'composite key' as well.
- Foreign Key: is the primary key of another table, which is referenced in the current table.
  - It's the key to establish the relationship between the two tables, and through DBMS, to impose referential integrity.

KEYS – SURROGATE KEY

- Surrogate key is a unique column
- Added to a relation to use as the primary key when there is a lack of a natural column that serves as the primary key, or when the composite key needs to be replaced for various reasons.
- Surrogate key is usually in the form of an auto increment numeric value, and of no meaning to the user, and thus could often be hidden in the table, form. or other entity for internal use.
- Surrogate keys are often used in the place of composite keys to add more flexibility to the table.

DATABASE SCHEMAS

- Database schema – way to define a table
- Collection of table definitions that gives the name of the table, lists the attributes and their data types, and identifies the primary key

OPERATIONS ON TABLES

- A database is a collection of database tables
- Main use of database is to look up information
- Users specify what they want to know and the database software finds it
- We can perform operations on tables to produce new tables (that are not explicitly part of the schema)
- The questions we ask of a database are answered with a whole table
- Five fundamental operations that can be performed on tables: Select, Project, Union, Difference, Product

SELECT OPERATION

- Takes rows from one table to create a new table
- Specify the table from which rows are to be taken, and the test for selection
- For SQL Syntax, look at SQL Reference

PROJECT OPERATION

- Builds a new table from the columns of an existing table
- Specify name of exiting table and the columns (field names) to be included in the new table
- The new table will have the number of columns given in the operation, and the same number of rows as the original table, unless
  - The new table eliminates a key field; if the new table has duplicate rows, the duplicates will be eliminated
- SQL does not implement projects. Instead it is handled as a variation of select, where certain columns are selected.

OTHER OPERATIONS

- Union - Combines two tables (that have the same set of attributes)
- Difference - Remove from one table the rows also listed in a second table (remove from Table1 any rows also in Table2)
- Product – Gives the cartesian product of the two tables – one instance of a row in Table2 added to each instance of a row in Table1
- The five basic operations (select, project,+, -, x) are all we need to make any relational table
- Other operations we can imagine can be done with combinations of the basic 5
- One combination is so commonly wanted and useful that we name it and provide it as a direct operation: Join

JOINS

- Combines two tables (like the Product operation) but doesn't necessarily produce all pairings
- If the two tables each have fields with a common data type, the new table combines only the rows from the given tables that match on the fields
- When match is true for a row from each table it produces a result row that is their concatenation
- Lookup operation on tables
  - For each row in one table, locate a row (or rows) in the other table with the same value in the common field; if found, combine the two; if not, look up the next row.
  - This match on equality is called a natural join
  - Possible to join using any relational operator, not just = (equality) to compare fields

STRUCTURE OF A DATABASE

- We want to arrange the information in a database in a way that users see a relevant-to-their-needs view of the data that they will use continually
- Physical database (stored on disk)
- Logical view of the database (made on the fly and customized for a user)
- The point of the two-level system is to separate the management of the data (physical database) from the presentation of the data (logical view of the database)
- All users work with the same physical database
- Different users will work with different views, one for each

NORMALIZATION

- Database normalization, or simply normalization, is the process of restructuring a relational database in accordance with a series of so- called normal forms in order to reduce data redundancy and improve data integrity.
- It was first proposed by Edgar F. Codd as an integral part of his relational model.
- Normalization entails organizing the columns (attributes)
- and tables (relations) of a database to ensure that their dependencies are properly enforced by database integrity constraints. It is accomplished by applying some formal rules either by a process of synthesis (creating a new database design) or decomposition (improving an existing database design).

ADVANTAGES OF NORMALIZATION

- No Redundancy
- No Inconsistency – changes can only be made at one place and are consistent (because of the key constraints), in DB
- Normalization is the process of decomposition, so all business concepts can be modeled with clear logical relationships
- The entire database system remains consistent over time as the database grows with least redundancy and much durability.
- Strong support to be ACID compliant