What we should consider when we deign a database:

Schemas: How should my data be logically organized?

Normalization: Should my data have minimal dependency and redundancy?

Views: What joins will be done most often?

Access control: Should all users of the data have the same level of access

DBMS: How do I pick between all the SQL and noSQL options?

and more!

Approaches to processing data

OLTP: Stand for the Online Transaction Processing. They are application oriented. We have a snapshot of data, with the gigabyte size. This is used by thousands of people like consumers. Examples: find a price of a book, customer transaction, employees' hours

OLAP: Stand for Online Analytical Processing. Oriented around a certain subject that's under analysis like average sale. OLAP is used in analysts and data science companies

OLAP contains the historical data, archive at the order of terabyte Calculate books with best profit, Find the average transaction, ..

Both of them working together. OLTP is stored in an operational database and provide data and clean them for the OLAP warehause. Before implementing anything we should figure out besinnes requirements

Sorting data: data can be saved in 3 different levels such as:

1. Structured data

It follows a schema and the data types and tables are defined and relationships between them should be defined e.g., SQL ,tables in a relational database

2. Unstructured data

Schemaless like most of data in the world e.g., photos, chat logs, MP3

3. Semi-structured data

Does not follow larger schema and it has self-describing structure e.g., NoSQL, XML, JSON

Beyond traditional databases

Traditional databases:

For storing real-time relational structured data such as OLTP

Data warehouses: For analyzing archived structured data such as OLAP

Organized for reading & aggregating data. They are optimized for read-only analytics. They contains data from multiple sources and used the massively Parallel Processing (MPP) for faster queries. In desining their data base they typically uses a denormalized schema and dimensional modeling. Amazon, Google and MicrosoftA provide data warehouse solution. A subset of data warehouses is called data mart and dedicated to a specie topic. They are Schema-on-write (the schema is predefined)

Data lakes: For storing data of all structures e.g., raw, operational databases, IoT device logs, real-time, relational and non-relational. They retains all data and can take up petabytes. Schema-on-read (the schema is created as the data is red) as opposed to schema-on-write. Data lakes need to catalog data otherwise becomes a data swamp. Run big data analytics using services such as Apache Spark and Hadoop.

Data lakes are useful for deep learning and data discovery because activities require so much data. When we think about where to store the data we should think about how the data get there and in what form:

ETL: Extract trasnform load: Traditional for smaller scale

ELT Extract load transform: Newer of bigger data

Database design is defined as how data is logically stored. And how is data going to be read and updated? Two important concepts for design: **Data base model, Schema.**

Data base model

Data base models defines the database structure such as relational model. This is the first step toward the database design.

Some other options: NoSQL models, object-oriented model, network model

Process of creating a data model for the data to be stored

1. Conceptual data model: describes entities, relationships, and attributes

Tools: data structure diagrams, e.g., entity-relational diagrams and UML diagrams

2. Logical data model: defines tables, columns, relationships

Tools: database models and schemas, e.g., relational model and star schema

3. Physical data model: describes physical storage

Tools: partitions, CPUs, indexes, backup systems and tablespaces

Schema

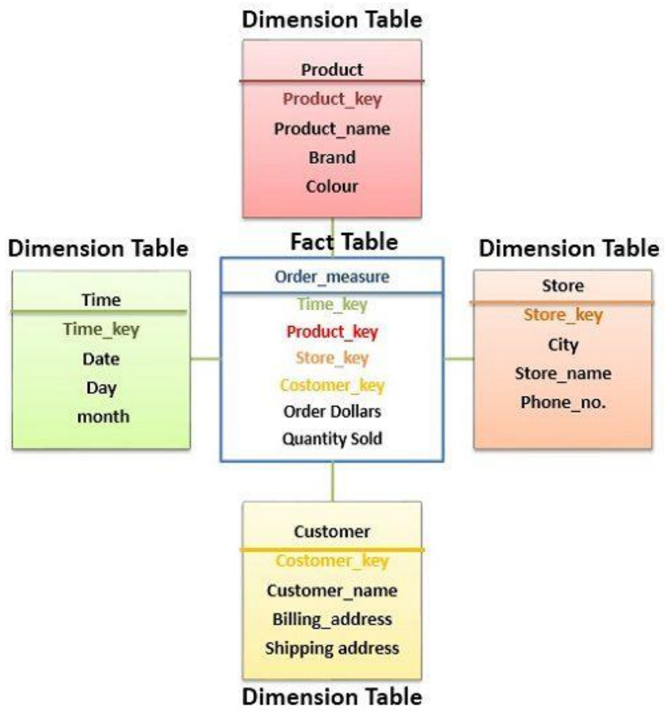
Schemas is the implementation of the database model. When we insert data in rational databases the schema should be followed. Defines tables, fields, relationships, indexes, and views

Dimensional modeling

This is the adaptation of the relational model for data warehouse design. It is optimized for OLAP queries: aggregate data, not updating (OLTP). This is Built using the star schema and Easy to interpret and extend schema.

Two types of tables dimensional modeling:

1. Fact **table**: Decided by business use-case. Holds records of a metric .Changes regularly Connects to dimensions via foreign keys
2. Dimensional table: Holds descriptions of attributes Does not change as often



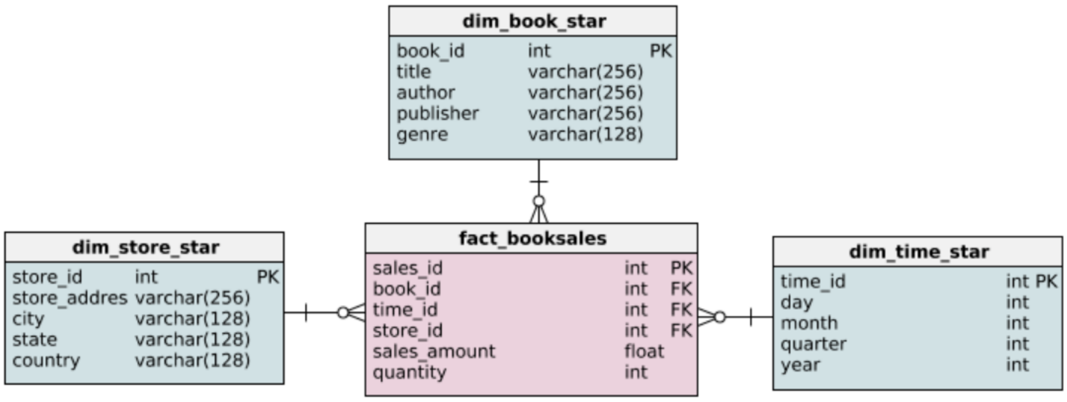
Star to snowflake schema: Normalization

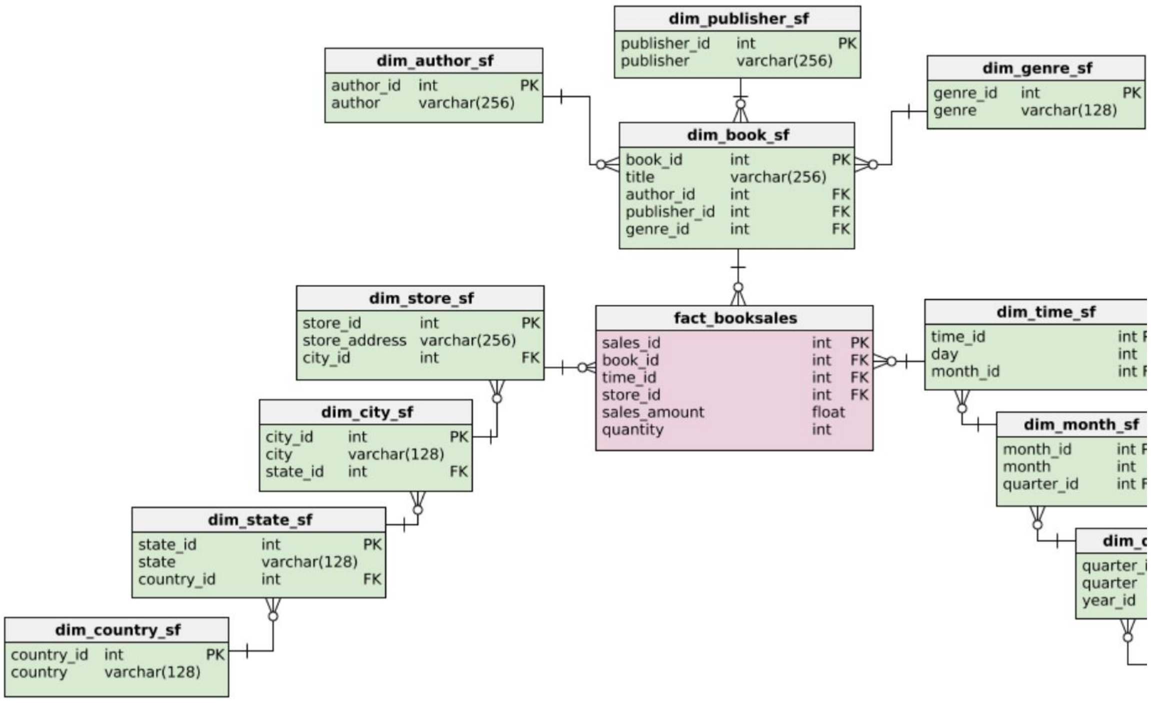
Snowflake is the extension of star schemas which dimension tables are normalized. Normalization divides them into smaller tables and connect them via relationships. The goal is to reduce redundancy. It is safer to update normalized data because you just new to update a record in one table. Disadvantage: More complex code and queries should be written. Normalization is useful for OLTP because we write there intensively and we want to write safer. OLAP is read intensive because we want to analyses it. With OLAP we try to avoid normalization/

Goal : Reduce redundancy and increase data integrity

Adding foreign keys

Foreign key references are essential to both the snowflake and star schema. When creating either of these schemas, correctly setting up the foreign keys is vital because they connect dimensions to the fact table. They also enforce a one-to-many relationship, because unless otherwise specified, a foreign key can appear more than once in a table and primary key can appear only once.





Normalization levels:

It shows the level of redundancy in a relational schema. It helps to updating and adding and deleting results.

1NF : each record must be unique (no duplicate rows) and each cell must hold one value

2NF : Must satisfy 1NF and If primary key is one column then automatically satisfies If there is a composite primary key then each non-key column must be dependent on all the keys

3NF: Satisfies 2NF. No transitive dependencies: non-key columns cant depend on the non-key columns,

Database views

View is the result of set of stored query on the data, which users can query just as they would in a persistent database. They are virtual table that is not part of the physical schema. View is not stored in the physical memory, instead the query that produce the view is stored.