Lecture Notes

Data Management and Relational Modelling

Data Management

Data Management

Over the years, the amount of data that is received by an enterprise has increased in terms of volume, types and sources. The ways in which data can be manipulated have also increased. Data is used by companies to run their daily operations, which involves knowing their customers and suppliers, and managing their business activities. Data-driven decisions have given an edge in data analytics and, now, managing data in order to analyse it for various business queries has become more important. Data management is the ability of an enterprise to collect, store, integrate, secure and analyse data to run its daily operations effectively as well as use data analytics to handle various business queries. Data management includes planning, governing and managing data usage effectively.

On a logical level, data management includes policies for governing, securing, integrating and providing quality data flow within an organisation. On the implementation level, data management involves implementing various policies through various applications and software.

Components of Data Management

The main components of data management are data governance, data quality, data integrity and data security.
Data Governance involves making policies for flow and use of data within an organization.
Data governance answers the following questions:

- What is the need for setting certain rules for data usage?
- What are the rules regarding data usage?
- Where can one find the data of a specific company?
- Who has access to certain data?
- Which technologies can be used for data management and governance?
- How can rules be implemented for efficient usage of data?

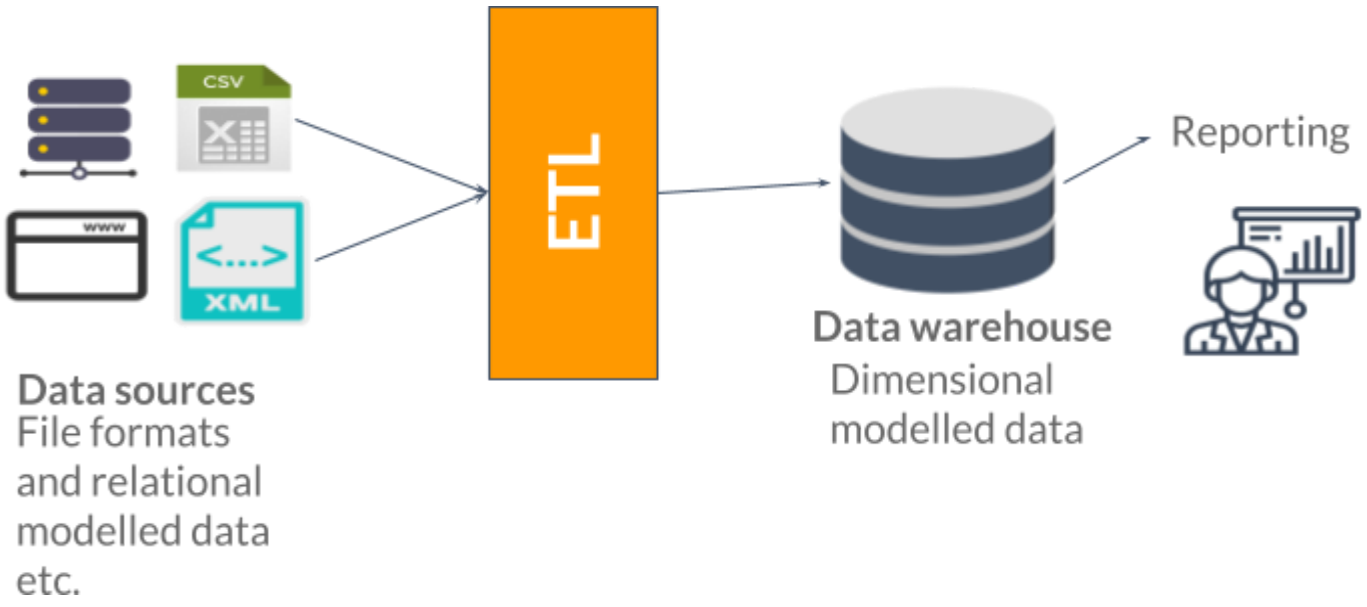
Data integrity refers to the maintenance and assurance of the accuracy and consistency of data over its life cycle. An example of this is providing one copy of consistent data to every department that uses the data.

Data quality refers to the state of qualitative or quantitative information. It is ensured through data cleaning, data structuring and data enrichment. A data quality check determines whether the data used for analysis is fit for usage, that there are no null values in non-null fields, that every data field permits only values and data types, and whether the data is in the required structured format.

Data security involves providing secure access to authorised users for data usage.

Uses of Managed Data

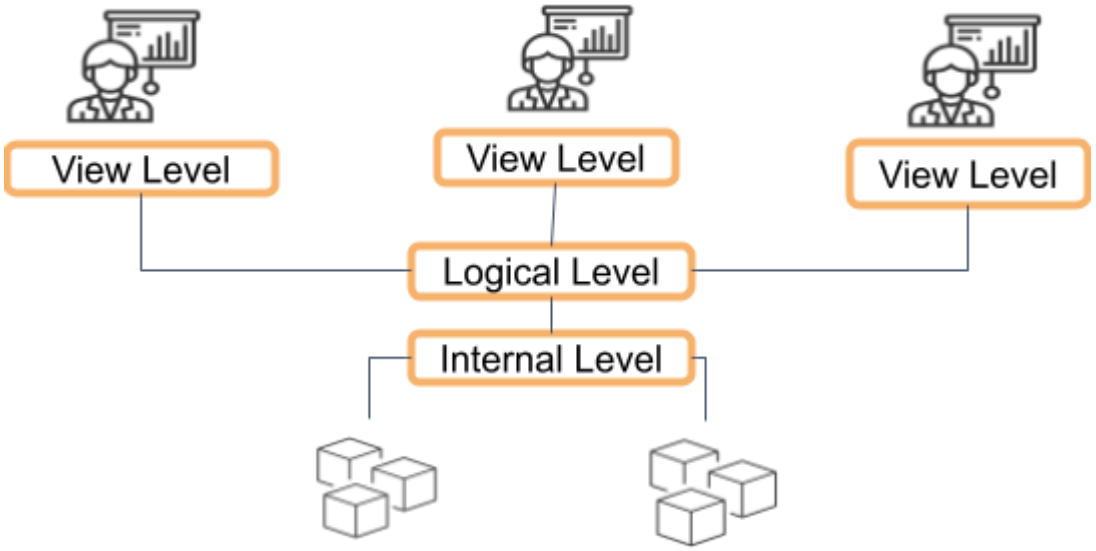
The flow of data in an organisation begins with data extraction from various data sources, which include XML files, CSV files, JSON files, Weblogs, relational modelled database and many more. To extract data effectively from each source, it is important to understand the format in which it is stored. Once the data is extracted, it goes through the processing stream where it is transformed, structured, cleaned, enriched and loaded into a central repository. The data is integrated into this central repository from various sources. Such a central repository can be either a data warehouse or a data lake. Data warehouses are structured storage systems that integrate data about a particular business subject. On the other hand, data lakes store both structured and unstructured data. The data stored in central repositories are used for reporting various business queries, including quarterly sales, sales in a particular region, promotion charges, product performance for the month, etc. Based on the results derived from these reports, businesses make decisions in order to plan their sales and marketing activities.



E-R Models

Three-Level Architecture

The three-level architecture of a DBMS describes the storage and management of data in a database.



The View Level is the application layer. Users use these applications to access data from databases. The Logical Level structures the data into a schema. This schema could be optimised for querying and updating data or for analysis of data. The schema is specific to the business requirement. The third level, internal level, designs and implements the physical implementation of data in physical storage devices.

These three levels are implemented independently. This means that if the schema implementation is changed at the logical level, then there is no change in the physical-level implementation. Each view level displays only specific parts of the logical schema, i.e., only the view level that displays the changed part of the logical schema changes; the implementation of the other view levels remains the same.

Data Models

Data models are logical designs that are first created on paper and then implemented physically. Business users use data models to understand a database. Hence, database designers build data models based on the business requirements defined by business users. Consider these two types of data models.

ER Model:

An entity-relationship model (ER model) describes real-world objects as entities and their properties as attributes. It identifies the relation between these entities along with the degree of the relation and the participation of each entity in such a relation.

Relational Model:

A relational model stores each business concept in a table and describes it as a relation. Each relation or table has keys that uniquely identify the data in every record. Foreign keys are used to describe how two different relations or tables are related. Foreign keys implements the relations described in an ER model.

Building an E-R Model

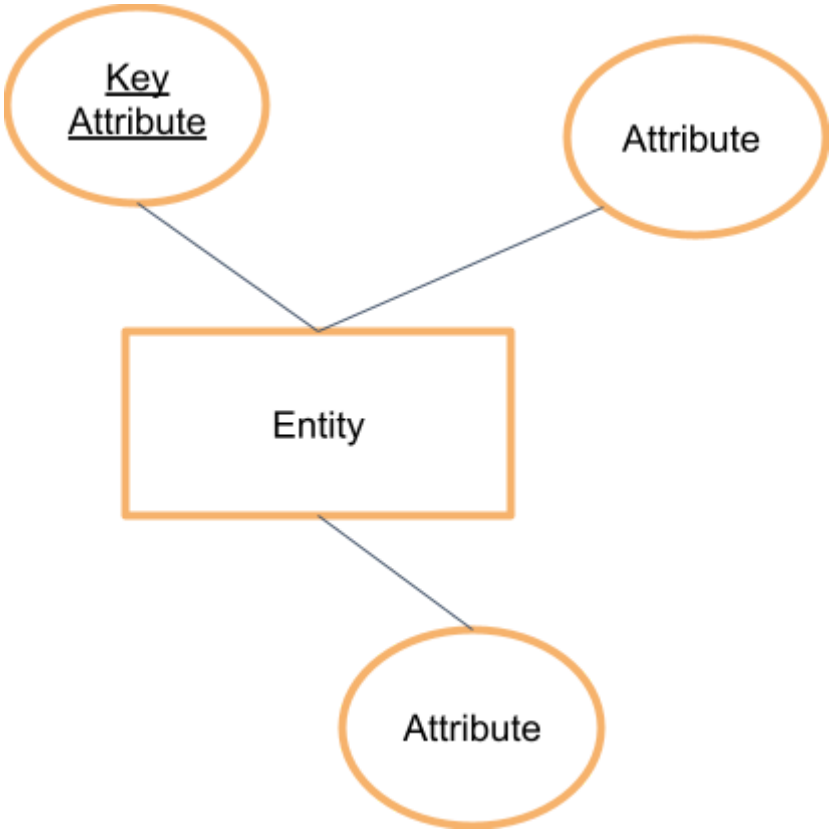
An ER model consists of entities, attributes and the relation between entities. Businesses store information about their main components, such as customers, suppliers, products, sales, orders, stores, employees and promotions. An ER model maps these real-world objects to entities, such as customer, student, employee, user, product and department.

The attributes in an ER model represent the properties of a real-world object. For example, a particular product's properties, such as name, weight, type, category, size and brand, are its properties. The attributes in an ER model store such properties of every entity. They define what information is required regarding a business entity.

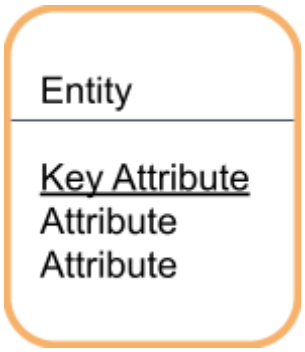
Entities and attributes are two of the main components of an ER model. Every entity is a table in a relational model, and every attribute of that entity is a column in that table.

Entities and attributes can be depicted in the following two ways:

1) Represent every entity in a rectangle and every attribute in an oval as shown in the diagram given below.



2) Represent all the attributes within an entity box as shown in the diagram given below.



Relation in an ER Model

A relation is generally a verb that connects entities. Some examples are as follows:

- An employee 'works' in a company.
- A department 'has' many employees.
- A team 'manages' a project.

A relation defines how the entities in an ER model are related to each other.
A unary relation or recursive relation relates one row of an entity to another. One employee may 'report' to another employee.
A binary relation connects two different entities. For example, a customer 'buys' products.

A ternary relation connects three different entities.

The degrees of relation between two entities specify how each entity participates in a relation. They are as follows:

- One-to-one
- One-to-many
- Many-to-many

In the case of a one-to-one relation, one row in one table is related to only one row in another table.
In the case of a one-to-many relation, one row in one particular table can be related to many rows in another table. Also, one row in the other table can be related to only one row in the first table.
In the case of a many-to-many relation, one row in each table can be related to many rows in another table.

Cardinality

Cardinality defines the minimum and maximum participation of an entity in a relation. Minimum cardinality refers to minimum participation and maximum cardinality refers to maximum participation. An entity may or may not participate in a relation; so, the minimum cardinality is 0. On the other hand, if an entity must participate in a relation, the minimum cardinality is 1.

Maximum cardinality can be defined as the degree of relation between entities. Suppose entity A and entity B are in a one-to-many relation. Entity A lies on the 'one' side of the relation, i.e., entity A has a maximum cardinality of 1, and entity B lies on the 'many' side of the relation, i.e., entity B has a maximum cardinality of N.

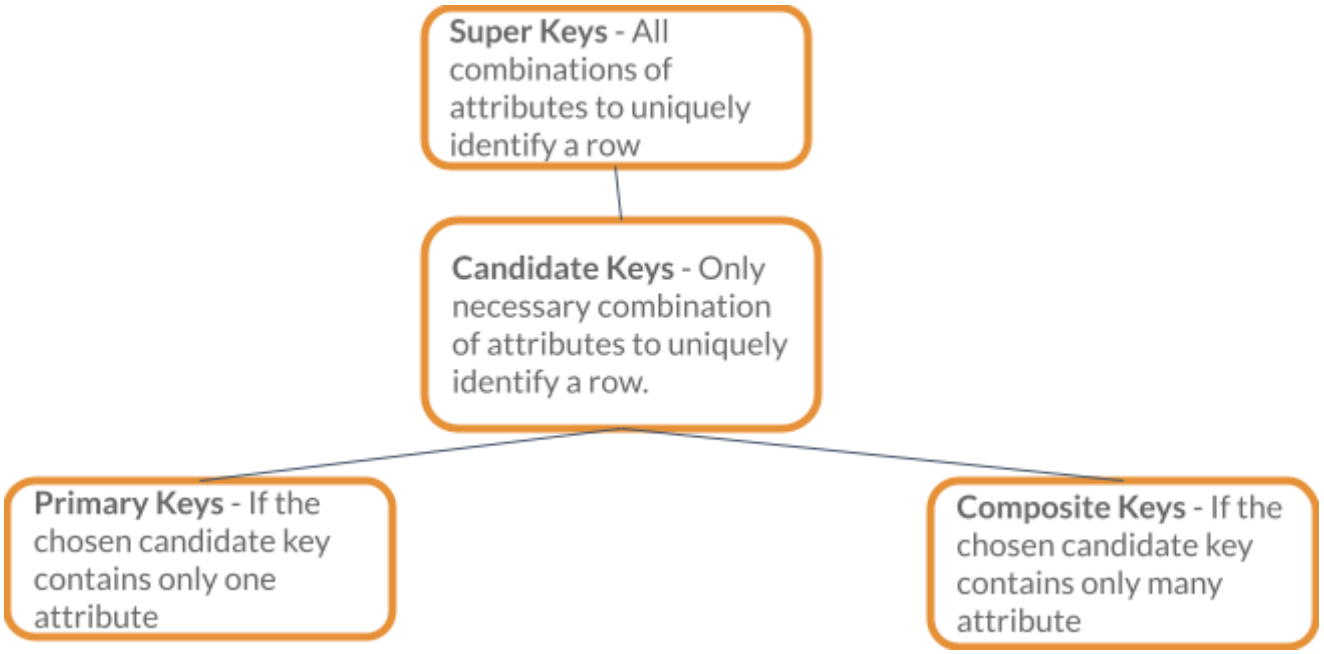
Relational Model

Relational Model

Every table in a relational model has the following characteristics.

1. The name of every table in a relational model database must be unique.
2. The name of every attribute of a table must be unique to that table.
3. Every table must have a key attribute. The value of this key attribute cannot be the same for any two rows in that table.
4. The data type for each attribute must be defined.
5. The order of the rows does not matter in a relation.
6. The order of the attributes does not matter in a relation.

Database Keys



Super keys are combinations of all possible attributes that can uniquely identify each row in a table. In a table with the attributes <student ID, name>, the student ID can uniquely identify each row. The super keys are as follows:

a. <student ID>
b. <student ID, name>

Only the 'name' attribute cannot be used to identify each row. Thus, <name> is not a super key.

Candidate keys are the super keys that contain only necessary combinations of attributes to identify each row of a table. In the table provided above, only <student ID> is necessary to identify each row. So, the candidate keys include only <student ID>.

If a table has many candidate keys, then one of them is chosen by the database designer to uniquely identify each row. This chosen key is the primary key of that table. Also, if the primary key contains more than one attribute, then it becomes a composite key.

Foreign keys are used to implement relations between tables. One of the tables thus related contains the foreign key that refers to the primary key of the other table. The values present in the foreign key column of one table must be present in the primary key column of the other table.

Building a Relational Model

If the relation between two entities is one-to-one, then the entity with mandatory participation will get the foreign key column. If the minimum cardinality is the same for both the entities, then either entity can get

the foreign key column.

The foreign key column is unique in a one-to-one relation, as one row in one table can correspond to only one row in another table.

If the relation between two entities is one-to-many, then the entity on the 'many' side will get the foreign key.

If the relation between two entities is many-to-many, then a new entity that contains the foreign keys to both the entities is created. This new entity will have a one-to-many relation with the other two entities.

ACID Property

A transaction in a relational database is an activity that cannot be divided into smaller activities. If a user transfers money from one bank account to another, then this transaction becomes atomic, which means that either the whole transaction occurs or it does not occur at all.

Every transaction must be isolated, which means that one transaction should take place only after the previous transaction is complete.

Every transaction must be durable. This means that if a transaction occurs, then the changes made by this transaction to the database remain inside it even in the event of a system failure.

The atomicity, isolation and durability of transactions make the data in a database consistent.

Data Normalization

Anomalies in a Database

information about more than one entity. Anomalies cause data to be deleted and inconsistent. The three types of anomalies are as follows:

1. Insertion anomaly: It occurs when you insert a data record as a row but the information is not available for all the columns, causing null values in a row.
2. Updation anomaly: It occurs when you update information in one row of a table, causing the previous information to be deleted, as there is no other copy of that data.
3. Deletion anomaly: It occurs when deletion of one row to remove data from some columns causes the data in other columns to be deleted as well. This occurs when data about multiple entities is stored in the same table. If information about one entity is deleted, then the information about another entity in the same row gets deleted.

First Normal Form

The first step in data normalisation is to convert a table to first normal form (1NF). This means that every field of a table will contain only single values, and every table will have a primary key.

If you store multiple values in one field, separated by commas, then the relational database (RDBMS) will consider the entire line to be one string and not a list. Hence, the application written to retrieve the data from the table will have to convert this string into a list in order to access each data element separately.

Second Normal Form

For a table to be in second normal form (2NF), all the non-prime attributes should be fully functionally dependent on the composite key of the table. There should not be any partial dependency in the table.

Suppose two columns, Column B and Column C, form a composite key for a table. The value of Column B can determine the value of Column D. The value of column C can determine the values of Columns E and F. The values of Columns B and C together can determine the values of Columns A, D, E and F. This can be notationally represented as follows:

a. (B, C) -> A
b. B -> D
c. C -> E, F

B and C are the prime attributes. The values of columns D, E and F depend on a prime attribute and not on the entire composite key. This is a case of partial dependency on the composite key. The value of column A is dependent on both B and C. This is a case of a full functional dependency on the composite key.

To remove partial dependencies, you need to separate them into new tables. In this case, the following three tables will be formed:

a. Table 1: <A, B, C>. B and C together will act as a composite key, and A will be fully functionally dependent on the composite key.
b. Table 2: <B, D>. B will act as a primary key, and D will be dependent on the primary key.
c. Table 3: <C, E, F>. C will act as a primary key and E and F will be dependent on the primary key.

Third Normal Form

Consider a table with the three attributes: <A, B, C>. A transitive dependency occurs when a non-prime attribute, C, is dependent on another non-prime attribute, B, which depends on the prime attribute, A. In this case, you create a new table with Columns B and C. Column B acts as a primary key for this new table. The two tables are as follows:
<A, B> and <B, C>

The conditions for a table to be in third normal form (3NF) are as follows:

a. Each field of the table must have single values. The table must be in 1NF first.
b. Every non-prime attribute must fully functionally depend on the composite key. The table must be in 2NF.
c. The table must not have any transitive dependency.

Disclaimer: *All content and material on the upGrad website is copyrighted material, belonging to either upGrad or its bona fide contributors, and is purely for the dissemination of education. You are permitted to access, print, and download extracts from this site purely for your own education only and on the following basis:*

- *You can download this document from the website for self-use only.*
- *Any copies of this document, in part or full, saved to disc or to any other storage medium, may be used for subsequent, self-viewing purposes or to print an individual extract or copy for non-commercial personal use only.*
- *Any further dissemination, distribution, reproduction, and copying of the content of the document herein, or the uploading thereof on other websites, or use of the content for any other commercial/unauthorized purposes in any way that could infringe the intellectual property rights of upGrad or its contributors is strictly prohibited.*
- *No graphics, images, or photographs from any accompanying text in this document will be used separately for unauthorized purposes.*
- *No material in this document will be modified, adapted, or altered in any way.*
- *No part of this document or upGrad content may be reproduced or stored on any other website or included in any public or private electronic retrieval system or service without upGrad's prior written permission.*
- *Any rights not expressly granted in these terms are reserved.*