```yaml
id: 1722872983-KDMB
aliases:
  - DBMS U1
tags:
Author: vortex
```

DBMS U1

Data Vs Information

Data: any set of raw facts, that serves as the foundation for knowledge. Needs to be processed to be of use.
Information: data that is organized, structured and presented in a given context to be of use.

Databases

- Collection of related data representing facts about a mini-world or a Universe of Discourse(UoD).
- Logically Coherent: not just a random collection of data
- Built for a specific purpose.
- Vary is size and complexity(small-scale to enterprise).

Database Management Systems

- Software system to manage large set of connected data.
  - Define Database
  - Construct Database
  - Manipulation of the database
  - Sharing of a database

> A DBMS is a collection of interrelated data and a set programs to manipulate that data

Purpose of DBMS

> Alias TFS -> Traditional File System

- Redundancy/Inconsistency of data
  - Redundancy: duplication of data
  - Inconsistency: changes are applied to redundant copied of data.

    > Solution: Store in centralized DB which avoids redundancy and applied changes to one copy of the data.

- Difficulty in Accessing data
  - Searching through TFS is manually taxing and required proprietary programs.

    > Solution: DB uses powerful queries and fast data structures to sift through the data.

- Data isolation
- Integrity
  - To apply a constraint in a TFS we must edit the application that parses it.

    > Solution: DBMS can specify constraints at the DB level.

- Atomicity
  - "All or None" philosophy.
  - System must either fully succeed or fail.

    > Solution: Auto rollback in case of failures.

- Concurrent Access anomalies
  - Anomalies exist when transactions to the DB happen concurrently.

    > Solution : Use of Locking to prevent conflicts.

- Security Issues

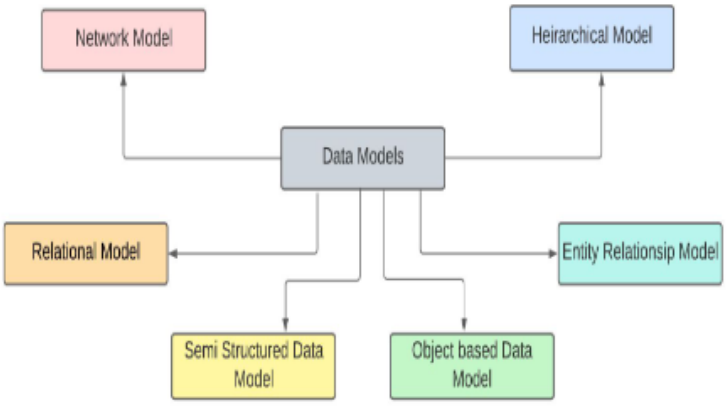
  > Admin can specify access controls per user and group.

Characteristics of DBMS

- Self Describing Nature
  - Contains a full description(Metadata) of the structure of the database.
- Program-Data Independance
  - The data is stored in different place than the access programs
- Multiple Views
  - Supports multiple views of the data. Each user and group can have their own views of the database
- Multi-user Transaction processing
  - Allows multiple users to access database at the same time.

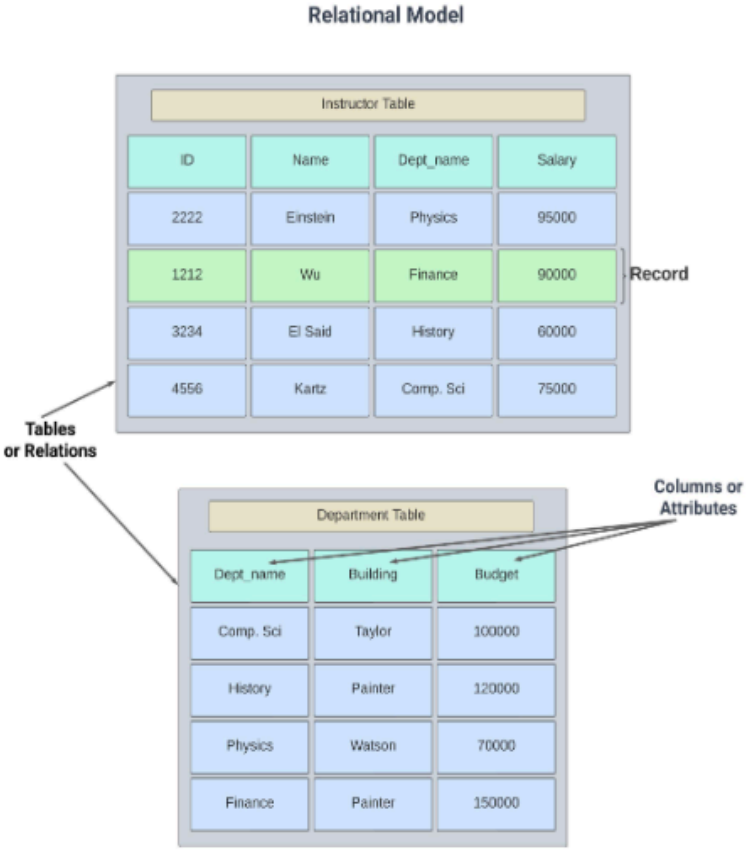
Data View

Data Models



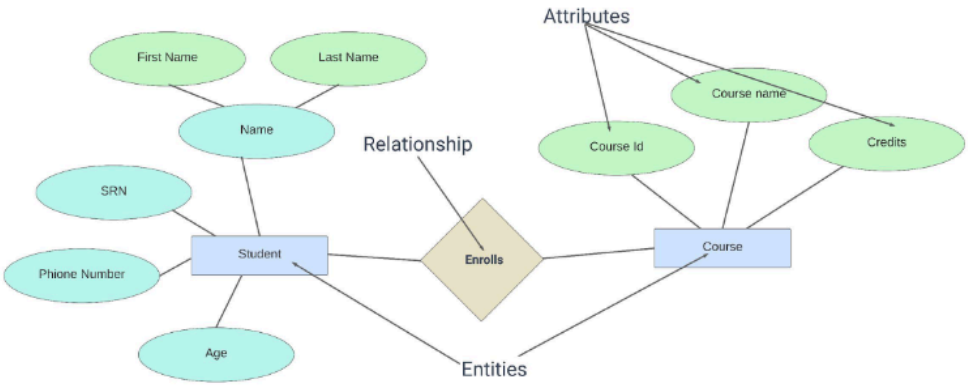
- Data Models: Set of tools to describe relationships, constraints and consistency.

Relational



- Collections of Tables
- Each column has a unique name
- Nomenclature:
  - Table = Relation
  - Row = Record
  - Column = Attribute

Entity Relation

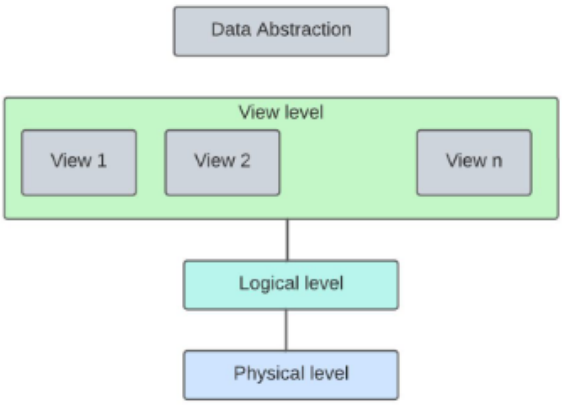


- ER data model: Collection of basic entities and relation between these entities.
- Entity: Thing or object distinguishable from other objects

Semi Structured

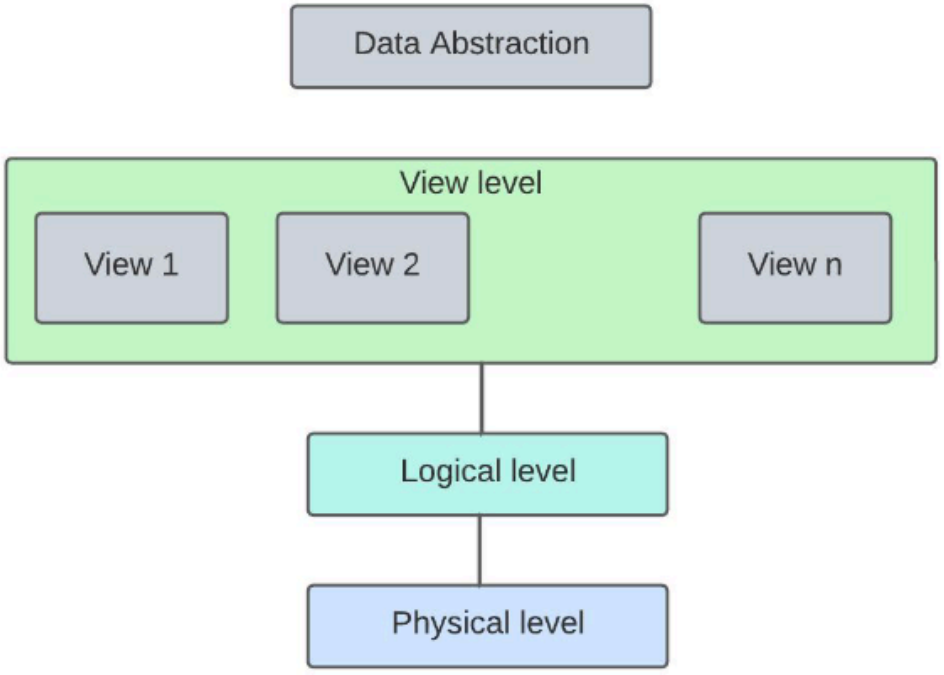
- Can't be constrained by a schema.
- Each entry may have a different set of attributes.
- Like JSON or XML.

Object Oriented



- Data Abstraction: Hide the complex data structures and represent data to users through high data abstraction

Data Abstraction

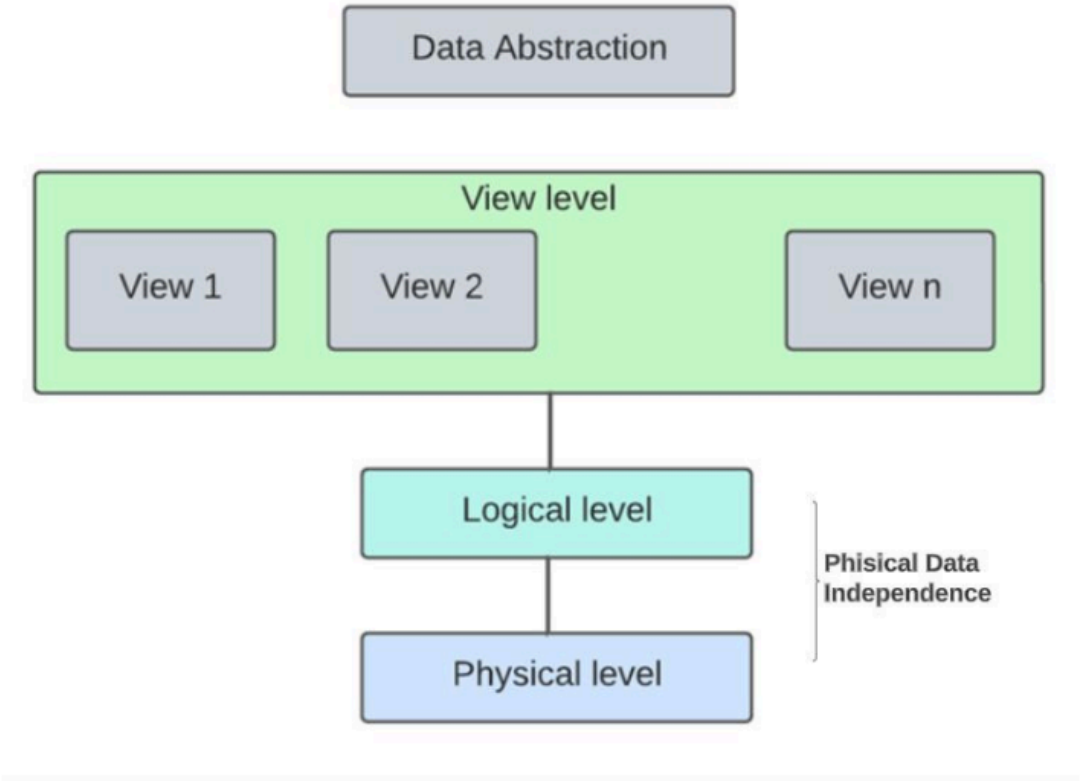


- Designers of database systems use many complex data structures to facilitate speedy access to data that is stored.
- End users of the DB don't really need to know all of these details.
- The designers therefore hide these details away from the users which is called database abstraction.
- The abstraction goes as follows:
  - Physical(lowest)
  - Logical
  - View(highest)
- The **Physical Level** describes how the data is stored physically on the computer.
- The **Logical Level** describes how the data is related to each other and what kind of information is stored in the DB.
- Not everyone who uses the DB needs to access the entire database and therefore the **View Level** lets different people see different parts of the DB relavent to their role.

> The simple structure of the logical view may entail really complex datastructures at the Physical Level. But the Logical Level user is unaware of this.
> The user at the View Level does need to know the workings of the Logical Level.

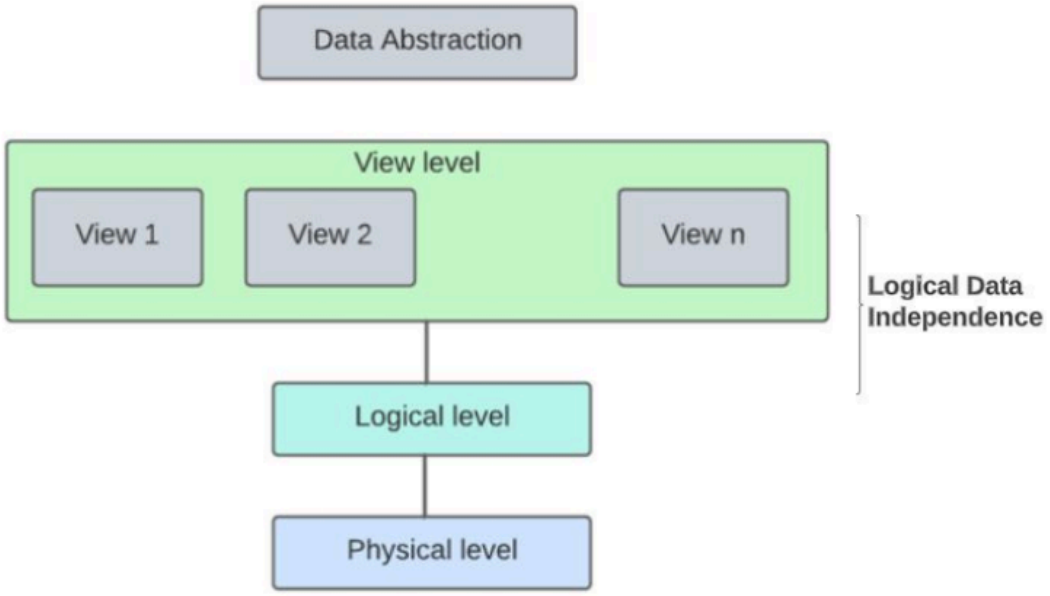
Physical Data Independence

- The Logical Level isn't affected by changes to the Physical Level. This includes changes to the structures storing data on the hard drive, altering storage, using a different file orgnaization.
- Without Physical Data Independence, any change to storage, could potentially break existing applications and queries requiring extensive rewrites.



Logical Data Independence

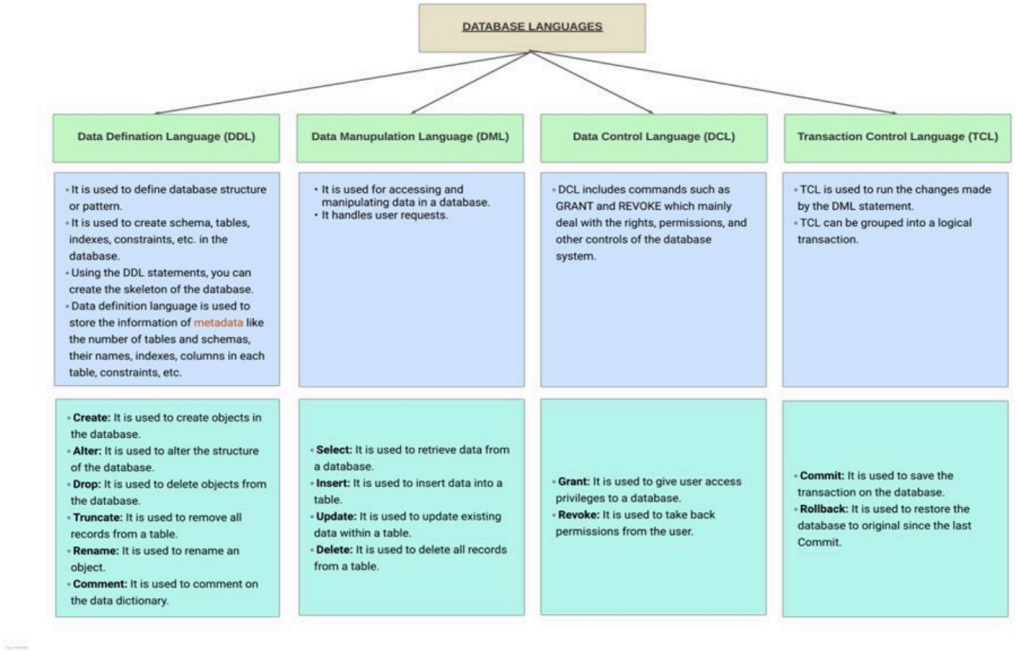
- Ability to modify logical schema without affecting application.



Some Terminologies

- Schema: variable declarations + type definitions
- Instance: snapshot of the DB at a particular point.
- Physical Schema:
  - Schema at Physical Level
  - Hidden below logical schema
  - Changing this doesn't really affect application programs
- Logical Schema:
  - Schema at logical level.
  - Most important to applications as programmers use this to develop apps.
- Subschema: Schema at view level

Database Languages



DDL

- Storage structure and access methods.
- The DDL compiler generates output that is stored in the **Data Dictionary**.
- Data Dictionary: contains metadata
  - schema
  - Integrity Constraints
  - Authorization
- Only the DBMS can alter the Data Dictionary. It is checked by the DBMS before modifying actual data.
- The data that we store must follow the following constraints:
  - Domain Constraints
  - Referential Integrity
  - Authorization

DML

- Languages used to manipulate already existing data in the DB.
- Functions:
  - Insertion
  - Update
  - Delete

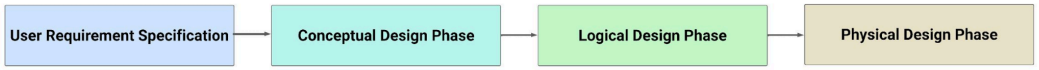
    - Retrieval
- Types:
    - Procedural: How and what to do
    - Declarative: only what to do
- **Query** : statement requesting retrieval of information
- The **query processor** translates the simple abstracted queries into complex actions at the physical level.

SQL

- Non-Procedural
- Not Turing-Complete
- Interaction with Applications made possible through Interfaces like ODBC(Open Data Base Connectivity) or JDBC(Java Data Base Connectivity).
- Has DDL(CREATE) and DML(SELECT) queries for easy operations.

Database Design

- Mainly refers to design of the schema.



1. User Requirement Spec:
    - Interaction with domain experts and survey the needs of the DB.
    - Documentation phase
2. Conceptual Design:
    - Schema preperation
    - Describe the relationship
        - Entity Relation(ER)
        - Normalization
3. Logical Design:
    - Map the high level schema into implementation data model.
4. Physical Design Phase
    - Designer uses the schema to design a system specific physical form of the database.
    - This includes describing internal structures and file formats

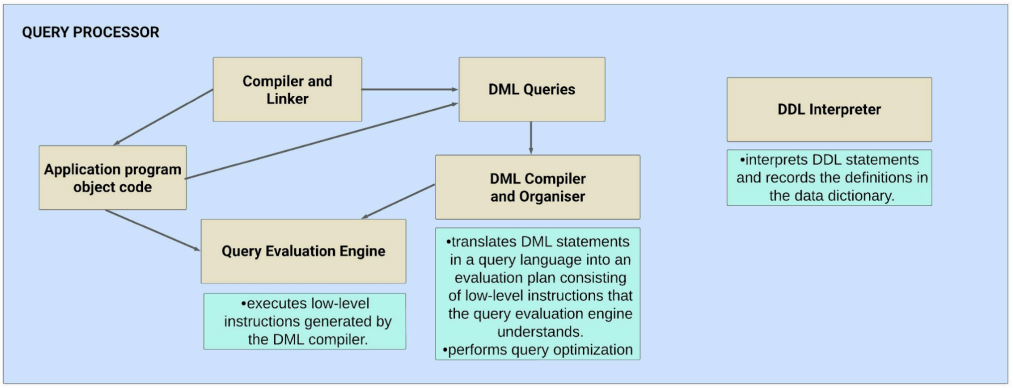
Database Engine

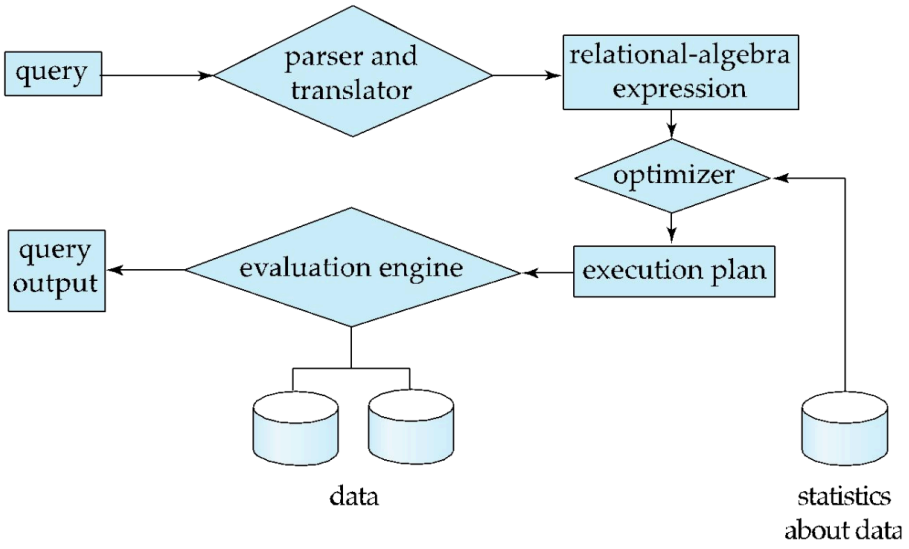
1. Storage Manager
    - Interface between high level queries and low level file manager.

- Interacts with the system file manager
- Efficiently stores and updates data.
- Data structures:
  - Data Files: for the DB itself.
  - Data Dictionary: for metadata
  - Indices: pointers to data items holding a value.
- Components:
  - Buffer Manager: checks what files to cache
  - File Manager: space allocations and data structures,
  - Transaction Manager: for consistency, concurrent transactions
  - Authentication: access permissions, integrity constraints

2. Query Processor

- Simplifies users who work at the view level by abstracting away complex data structures they need not know about.
- Components:
- DDL interpreter
- interprets the queries and stores definitions in the data dictionary.
- DML Compiler
- Compiler: translates queries into evaluation plan
- Optimizer: query optimization
- Query evaluation Engine: low level instructions are executed from output of compiler





3. Transaction Manager
    - Few important definitions:
        - Consistency: state of database being "correct".
        - Transaction: Any set of operations that ends with the performance of *one* logical operation in the application.
        - Durable: persistent changes in case of system failure
    - A transaction in a DB must always assure:
        1. Consistency
        2. Atomicity
        3. Durability
    - A temporary inconsistency is permitted within the runtime of the transaction if it is resolved by the time the transaction is completed.
    - Components:
        - Recovery Manager
        - Concurrency Control Manager

Database Architectures

1. Centralized
    - Shared memory architectures
    - Limited scalability
    - Few number of cores
2. Client-Server
    - One server does all the work
3. Parallel
    - Many Cores
    - Shared Disk
    - Better scalability

    - Runs on clusters
4. Distributed
    - Across geo-distanced databases
    - Schema/data heterogenity(?)

Database Applications

1. 2 Tier
    - Application in client
    - Invocation sent to server through queries
2. 3 Tier
    - client is just the frontend
    - the actual business logic resides in an application server.
    - The client connects to this application server and retrieves information.

Database workers

- People working with a DBMS are:
    1. Users
    2. Admins

Users

1. Naive Users:
    - Inexperienced users who interact via something like a web interface.
2. Application Programmers
    - Skilled Computer professionals
    - Well versed with toolchains
3. Sophisticated users
    - Raw interations with the DBMS
    - Query writers
    - No code writing
4. Database Admins(DBA)
    - Central Control
    - Functions:
        - Schema Definition
        - Storage structure and access method definition
        - Schema and physical organisation modification
        - Granting Authorization for accessing data
        - Maintenence

Constraints

Broad Classification:

1. Inherent/Implicit: As a consequence of the database model.
2. Schema-based/Explicit: Constraints enforced through the schema definitions
3. Application Based/Semantic: Enforced at the application level

Schema Based Constraints

1. Domain Constraints:
   - Restricting the value that an attribute can assume.
   - Every value in a tuple must be a value from the domain of it's attribute(or `NULL` if its permitted).
   - Datatypes like `int`, `float`, `char`, `bool` etc are examples of constraining the domain of an attribute.
2. Key Constraints:
   - Way of distinguishing between tuples.

     > No 2 tuples in a relation can have the exact same values for all attributes.

   SuperKey
   - Collection of one or more set of attributes when taken togeather help identify a tuple uniquely.
   - In a relation R , if there exist 2 tuples t_1 and t_2 such that $t_1 \neq t_2$, then $t_1[SK] \neq t_2[SK]$.
   - Any superset of a superkey is also a superkey.
   - In practical applications, it is preferred to have the *minimal superkey* which is essentially the superkey for which no subset is also a superkey. Such keys are also called *candidate keys*.

- The relation has multiple candidate keys out of which one is chosen randomly to be the *primary key*.
- This can uniquely identify a tuple in a relation.
- General rules:
  - Primary keys are usually the *smallest* in terms of size.
  - Primary key attributes are often listen before other attributes.

> $DB = \{r1, r2, \ldots, rm\}$ such that each ri is a state of Ri and such that the ri relation states satisfy the integrity constraints specified in IC.

- A DB that doesn't satisfy all Integrity Constraints is called *not valid*.
- Else it is called *Valid* state.

Entity Integrity Constraint

- Primary key attributes can't be `NULL`.
- If it has multiple participating attributes, none of them can be `NULL`.
- Other attributes can be enforced as not null.

Foreign Key Constraint

- Constraint applied from Attribute(A) of relation r_1 to Attribute(B) of relation r_2.
- B must be the *primary key* of r_2.
- A is called the *referencing relation* and B is called the *referenced relation*.

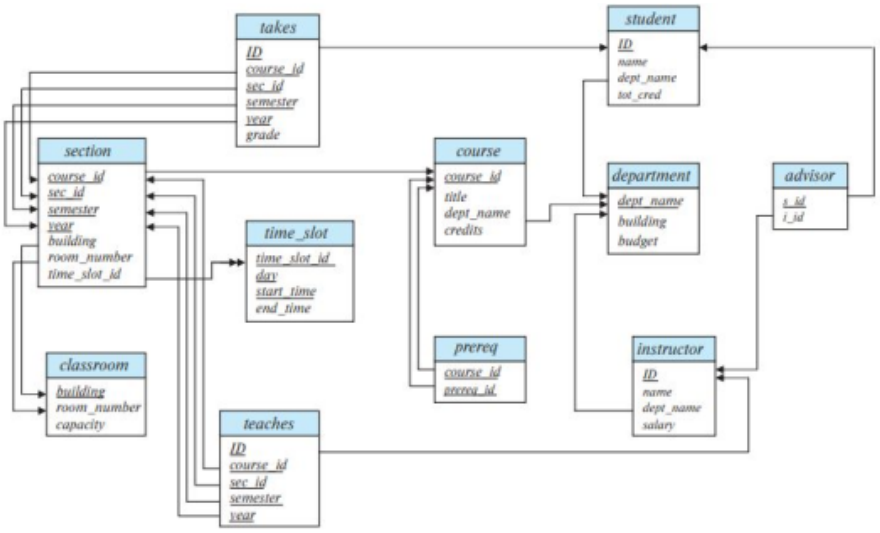
> Formal Definition: The referenced attribute must be the primary key of the referenced relation.

Referential Integrity Constrainsts

- Foreign Key constraints are a *special case* of this.
- Removes the restriction that the referenced attribute must be the primary key of the referenced relation.
- Statement:
  - Foreign key column in the referencing relation must either be
    1. Primary key of the referenced relation
    2. NULL
  - In the second case, it must not be the primary key of its own table.

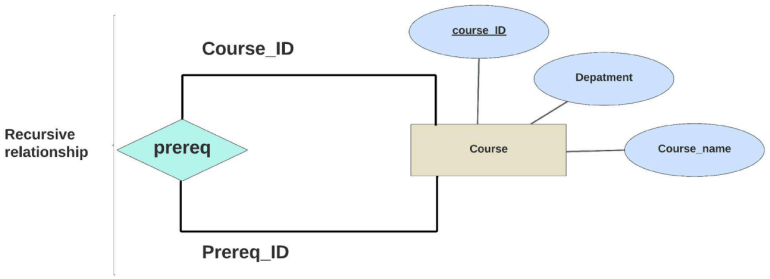
Schema Diagrams

- To depict constraints
- Relations: Boxes
- Primary Key: underline
- Foreign Key: Single headed arrow from referencing to referenced relation.
- Referential Integrity Constraint: Double headed arrow.

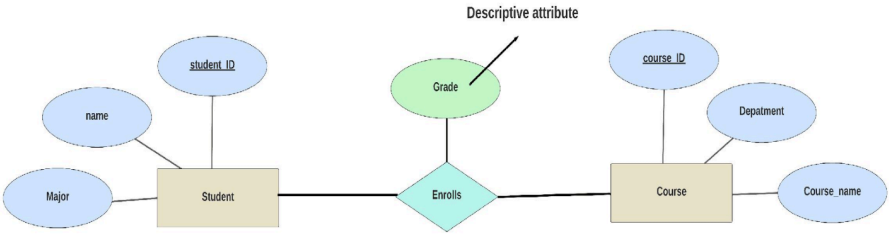


ER Model

- Data modeling method to assist in DB design.
  Concepts:
- Entity Sets: corresponds to a table in the DB.
- Relationship Sets: Associations between different entity sets.
- Attributes: Characteristics of entities or relations.
- Entity
  - Types:
    - Concrete - represents a physical object in the real world.
    - Abstract - represents a concept rather than a physical object
- Attributes
  - Type of information stored for each entity.
  - Entity's value provides distinct values for identification.
- Relationships
  - Connections between entities
  - A relationship set is a set of relations between similar entities.
  - Relationship instance is the association between 2 named entities.
  - Function that an entity plays in a relationship is its **role**.
    - Most times entities participating in a relationship are distinct.
    - Sometimes though when the meaning of a relation needs clarification, the same entity set participates in a relation. This is called a **recursive relation**.

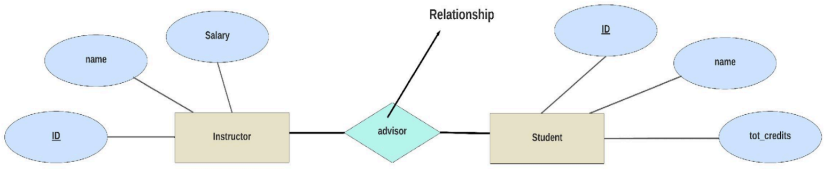


- Relationships can have attributes called the **descriptive attribute**.



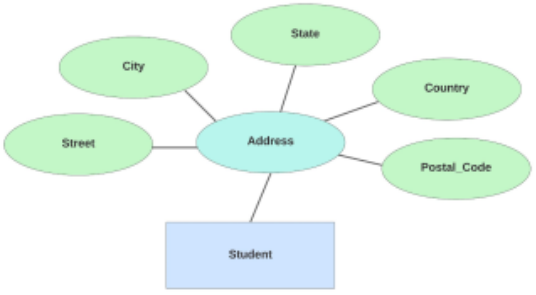
- Number of entities that take part in a relation is called the **degree** of the relation.

- Model Rules
  - Entity Sets - rectangle
  - Attribute - oval
  - Relationship - diamond
  - Underline primary keys

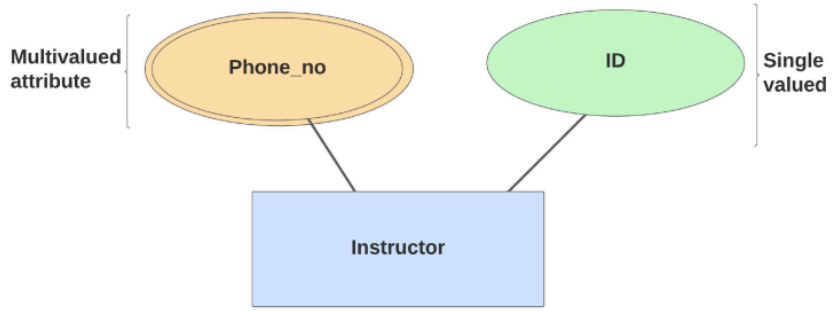


Complex Attributes

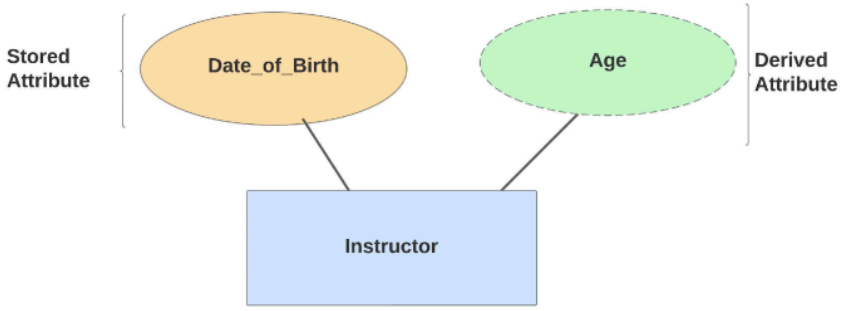
1. Composite Attributes:
   - Can be split into multiple smaller attributes.



2. Multivalued Attributes:
    - An attribute that may accomodate more than one values.



3. Derived Attribute:
    - *Stored* attributes are those that are actually stored on the database.
    - Some attributes can be derived through operations on these stored attributes and they are called *derived* attributes.

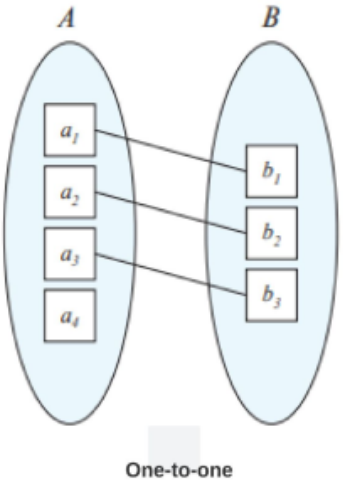


4. Complex Attributes:
    - Nested composite and multi-valued attributes.

Relationship Constraints

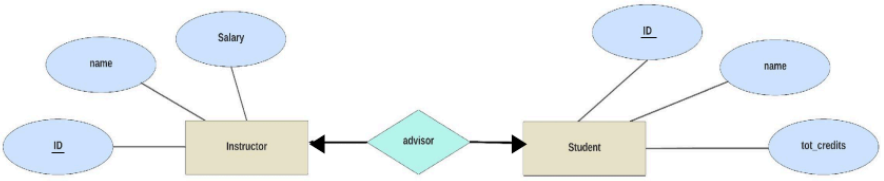
1. Cardinality Ratio

- Number of entities to which an entity can be associated to via a relationship set.
- Most useful for binary relationships:
    - One-to-One
        - An entity in A has one image in B and an entity in B has one pre-image in A.

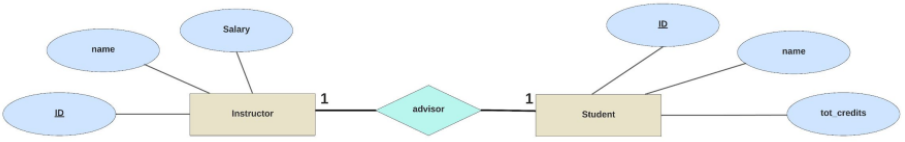


One-to-one

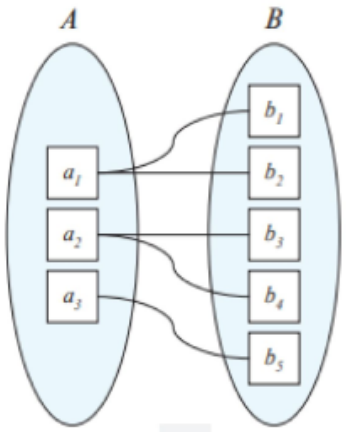
- Represented by drawing a directed line from the relationship to the entity sets.



Alternatively:

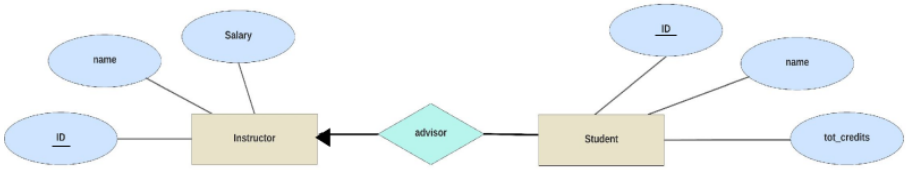


- One-to-Many
  - Entity in A can be associated to any number of entities in B but an entity in B is associated with max 1 entity in A.

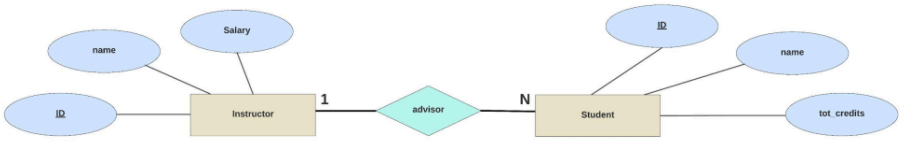


One-to-many

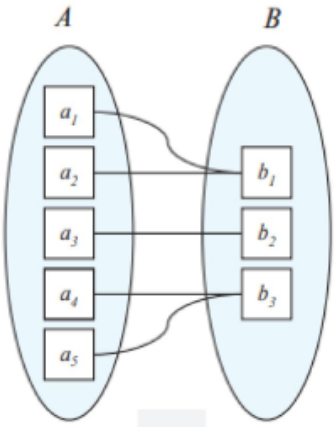
- Represented by a directed line from the relationship to the "ONE" side of the relationship.



alternatively:

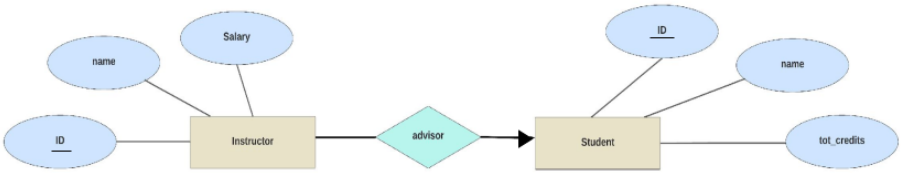


- Many-to-One
  - Entity in A associated with atmost 1 entity in B but entities in B associated with any number of entities in A.

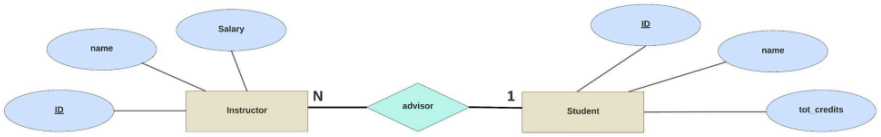


Many-to-one

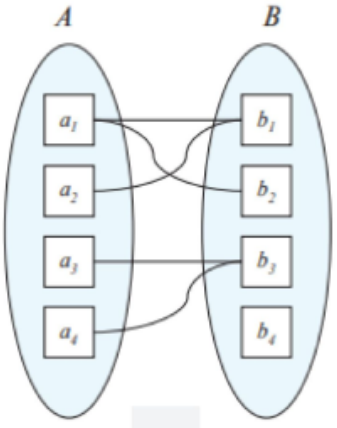
- Opposite of One-to-Many representation.



alternatively:

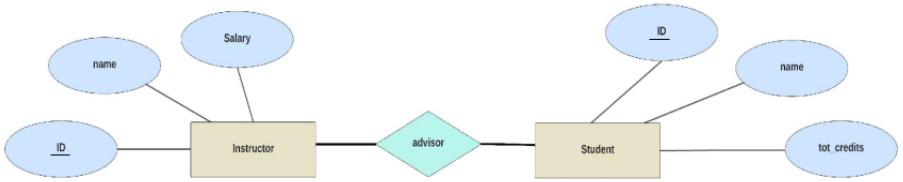


- Many-to-Many
  - Entity in A can be associated to any number of entities in B and vice versa.

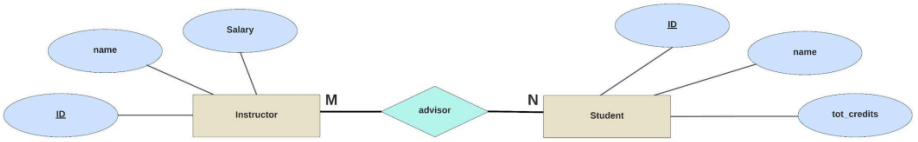


Many-to-many

- Represented by 2 undirected lines from the relationship.



alternatively:



2. Participation Constraints

- Total Participation
  - Every entity in entity set(E) participates in relationship R.
  - Represented using a *double line*.
- Partial Participation
  - Some entities in the entity set(E) don't participate in the relationship R.

> A line having a min-max cardinality can be represented as $l..h$ where:
> l is the $minimumcardinality$ and h is the $maximumcardinality$.
> Min Value = 1 (Total Paricipation)
> Max Value = 1 (Entity participates in atmost 1 relationship)
> Max Value = * (No limits)

Types of Attributes

- Composite Attributes
  - An attribute that can be further subdivided(like an address).
  - This is suitable when sometimes we would need to retreive only a part of the attribute.
- Simple Attributes
  - Cannot be further Subdivided.
- Multivalued
  - When an attribute can take multiple values(a person having many phone numbers)
  - Depicted by a double ellipse
- Single Valued
  - Can take only one value
- Derived Attribute

        - Whose value can be calculated from another attribute(like age from DOB).
        - Depicted by dotted ellipse.
    - Stored Attribute
        - Whose value is *stored* on db.
    - Complex Attribute
        - Combination of Composite+Multivalued

Reducing ERD to Relation Schema

- Steps to convert ER diagram into a relational schema
    - Mapping of strong entity sets
        - Convert into a schema with the attributes as the attributes of the entity in schema.
    - Mapping of weak entity sets
        - PK of the weak entity will be PK of strong entity+discriminator of weak entity(true for any weak entity derived from this weak entity.)
        - There is a foreign key relation from weak-strong entity
    - Mapping of strong entity sets with complex attributes
        - Ignore derived attributes
        - Split composite attributes into individual attributes.
        - Multivalued attributes are usually made into different relation referencing the main relation.
    - Mapping of relational sets
        - 1:1 or one-to-one
            - Foreign Key
                - Take S as relation with total participation
                - Take foreign key from S to primary key of T(other entity)
            - Merged relation
                - When both participations are total
                - Merge the 2 entity and relationship into one relation.
            - Relationship relation
                - Make new relation
                - This relation holds PK of S and T and some descriptive attributes of the relation.
        - 1:N or one-to-many or many-to-one
            - Take S as the relation with N side.
            - Foreign Key from S to T[PK]
        - M:N or many-to-many
            - Create New relation to depict R.
        - N-ary relations
            - Include as foreign key attributes, the primary keys of the n-ary relations