Introduction: An overview of database management system, database system Vs file system, Database system concept and architecture, data model schema and instances, data independence and database language and interfaces, data definitions language, DML, Overall Database Structure.
Data Modeling using the Entity Relationship Model: ER model concepts, notation for ER diagram, mapping constraints, keys, Concepts of Super Key, candidate key, primary key, Generalization, aggregation, reduction of an ER diagrams to tables, extended ER model, relationship of higher degree.

An overview of database management system

A database-management system (DBMS) is a collection of interrelated data and a set of programs to access those data. This is a collection of related data with an implicit meaning and hence is a database. The collection of data, usually referred to as the database, contains information relevant to an enterprise. The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.

By data, we mean known facts that can be recorded and that have implicit meaning. Database systems are designed to manage large bodies of information. Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information. In addition, the database system must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access. If data are to be shared among several users, the system must avoid possible anomalous results

Database System Vs File System

- A database management system coordinates both the physical and the logical access to the data, whereas a file-processing system coordinates only the physical access.
- A database management system reduces the amount of data duplication by ensuring that a physical piece of data is available to all programs authorized to have access to it, whereas data written by one program in a file-processing system may not be readable by another program.
- A database management system is designed to allow flexible access to data (i.e., queries), whereas a file-processing system is designed to allow predetermined access to data (i.e., compiled programs).
- A database management system is designed to coordinate multiple users accessing the same data at the same time. A file-processing system is usually designed to allow one or more programs to access different data files at the same time. In a file-processing system, a file can be accessed by two programs concurrently only if both programs have read-only access to the file.
- Redundancy is control in DBMS, but not in file system
- Unauthorized access is restricted in DBMS but not in file system.
- DBMS provide backup and recovery. When data is lost in file system then it not recover.
- DBMS provide multiple user interfaces. Data is isolated in file system

Advantages and Disadvantages of a DBMS

Using a DBMS to manage data has many advantages:

- **Data independence**: Application programs should be as independent as possible from details of data representation and storage. The DBMS can provide an abstract view of the data to insulate application code from such details.
- **Efficient data access**: A DBMS utilizes a variety of sophisticated techniques to store and retrieve data efficiently. This feature is especially important if the data is stored on external storage devices.
- **Data integrity and security**: If data is always accessed through the DBMS, the DBMS can enforce integrity constraints on the data. For example, before inserting salary information for an employee,

the DBMS can check that the department budget is not exceeded. Also, the DBMS can enforce access controls that govern what data is visible to different classes of users.

- **Data administration**: When several users share the data, centralizing the administration of data can offer significant improvements. Experienced professionals, who understand the nature of the data being managed, and how different groups of users use it, can be responsible for organizing the data representation to minimize redundancy and fine-tuning the storage of the data to make retrieval efficient.
- **Concurrent access and crash recovery**: A DBMS schedules concurrent accesses to the data in such a manner that users can think of the data as being accessed by only one user at a time. Further, the DBMS protects users from the effects of system failures.
- **Reduced application development time**: Clearly, the DBMS supports many important functions that are common to many applications accessing data stored in the DBMS. This, in conjunction with the high-level interface to the data, facilitates quick development of applications. Such applications are also likely to be more robust than applications developed from scratch because many important tasks are handled by the DBMS instead of being implemented by the application.

Disadvantages of a DBMS

- **Danger of Overkill**: For small and simple applications for single users a database system is often not advisable.
- **Complexity**: A database system creates additional complexity and requirements. The supply and operation of a database management system with several users and databases is quite costly and demanding.
- **Qualified Personnel**: The professional operation of a database system requires appropriately trained staff. Without a qualified database administrator nothing will work for long.
- **Costs:** Through the use of a database system new costs are generated for the system itself but also for additional hardware and the more complex handling of the system.
- **Lower Efficiency**: A database system is a multi-use software which is often less efficient than specialized software which is produced and optimized exactly for one problem.

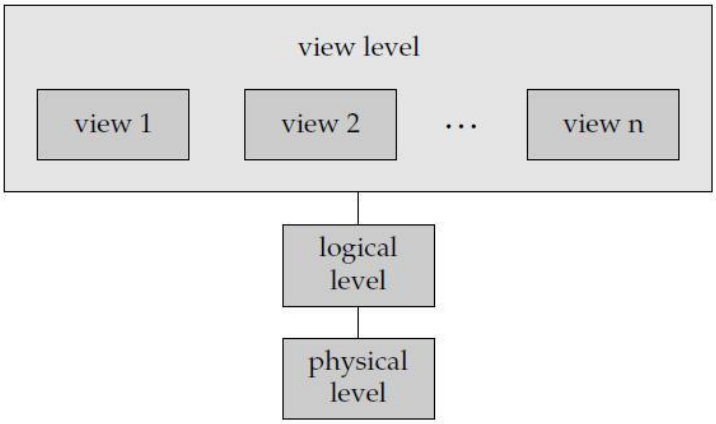
A database system is a collection of interrelated files and a set of programs that allow users to access and modify these files. A major purpose of a database system is to provide users with an abstract view of the data. That is, the system hides certain details of how the data are stored and maintained

Data Abstraction

Since many database-systems users are not computer trained, developers hide the complexity from users through several levels of abstraction, to simplify users‘ interactions with the system:

- **Physical level**. The lowest level of abstraction describes how the data are actually stored. The physical level describes complex low-level data structures in detail.
- **Logical level**. The next-higher level of abstraction describes what data are stored in the database, and what relationships exist among those data. The logical level thus describes the entire database in terms of a small number of relatively simple structures. Although implementation of the simple structures at the logical level may involve complex physical-level structures, the user of the logical level does not need to be aware of this complexity. Database administrators, who must decide what information to keep in the database, use the logical level of abstraction.
- **View level**. The highest level of abstraction describes only part of the entire database. Even though the logical level uses simpler structures, complexity remains because of the variety of information

stored in a large database. Many users of the database system do not need all this information; instead, they need to access only a part of the database. The view level of abstraction exists to simplify their interaction with the system. The system may provide many views for the same database



Database Instance

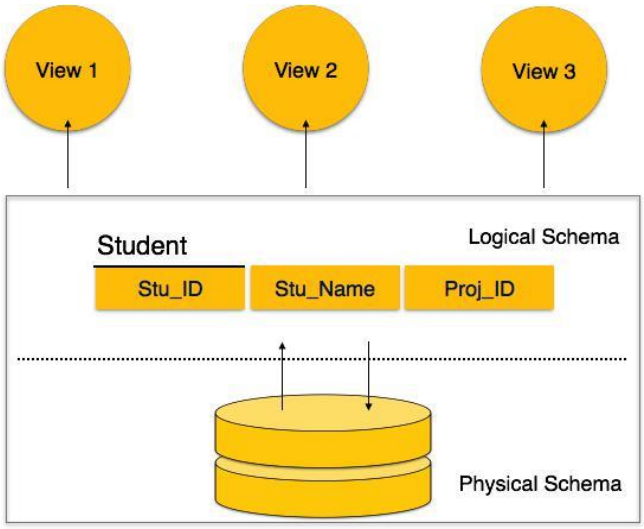
The collection of information stored in the database at a particular moment is called an **instance** of the database. A database instance is a state of operational database with data at any given time. It contains a snapshot of the database. Database instances tend to change with time. A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.

Database Schema

The overall design of the database is called the database schema. Database systems have several schemas, partitioned according to the levels of abstraction.

- The physical schema describes the database design at the physical level,
- The logical schema describes the database design at the logical level.
- A database may also have several schemas at the view level, sometimes called sub-schemas that describe different views of the database.

A database schema corresponds to the variable declarations (along with associated type definitions) in a program. Each variable has a particular value at a given instant. The values of the variables in a program at a point in time correspond to an instance of a database schema. Therefore Database schema skeleton structure of and it represents the logical view of entire database. It tells about how the data is organized and how relation among them is associated. A database schema defines its entities and the relationship among them. Database schema is a descriptive detail of the database, which can be depicted by means of schema diagrams.



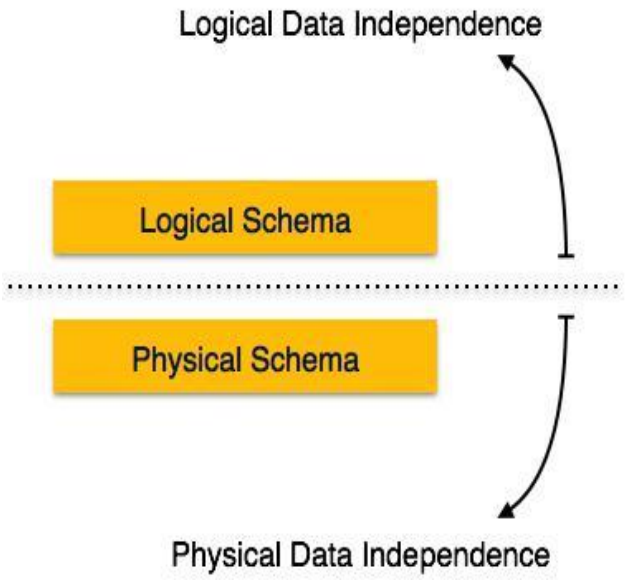
Data Independence

A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily. It is rather difficult to modify or update a set of metadata once it is stored in the database. But as a DBMS expands, it needs to change over time to satisfy the requirements of the users. If the entire data is dependent, it would become a tedious and highly complex job.

Logical Data Independence

Logical data is data about database, that is, it stores information about how data is managed inside. For example, a table (relation) stored in the database and all its constraints, applied on that relation.

Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk. If we do some changes on table format, it should not change the data residing on the disk.



Physical Data Independence

All the schemas are logical, and the actual data is stored in bit format on the disk. Physical data independence is the power to change the physical data without impacting the schema or logical data.

For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

DBMS - Data Models

Data model: A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

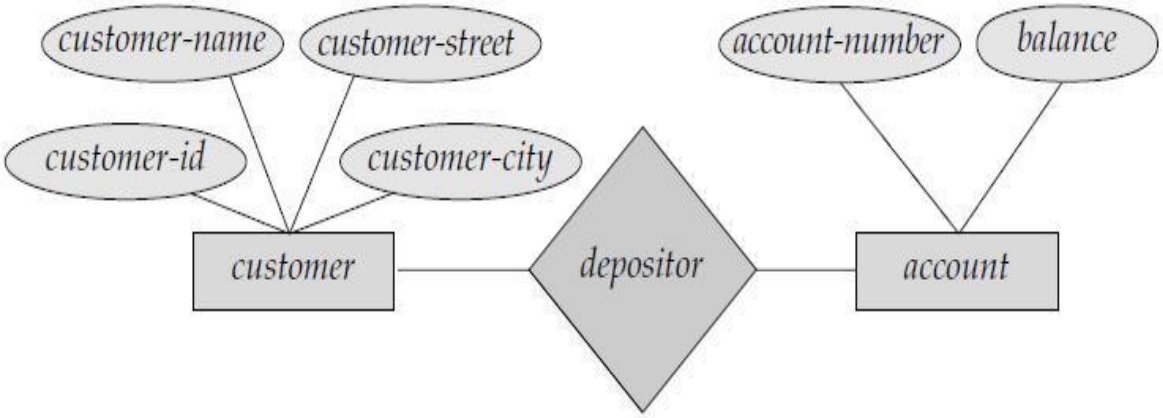
Entity-Relationship Model: Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database. ER Model is based on the following

- Entities and their attributes.
- Relationships among entities.

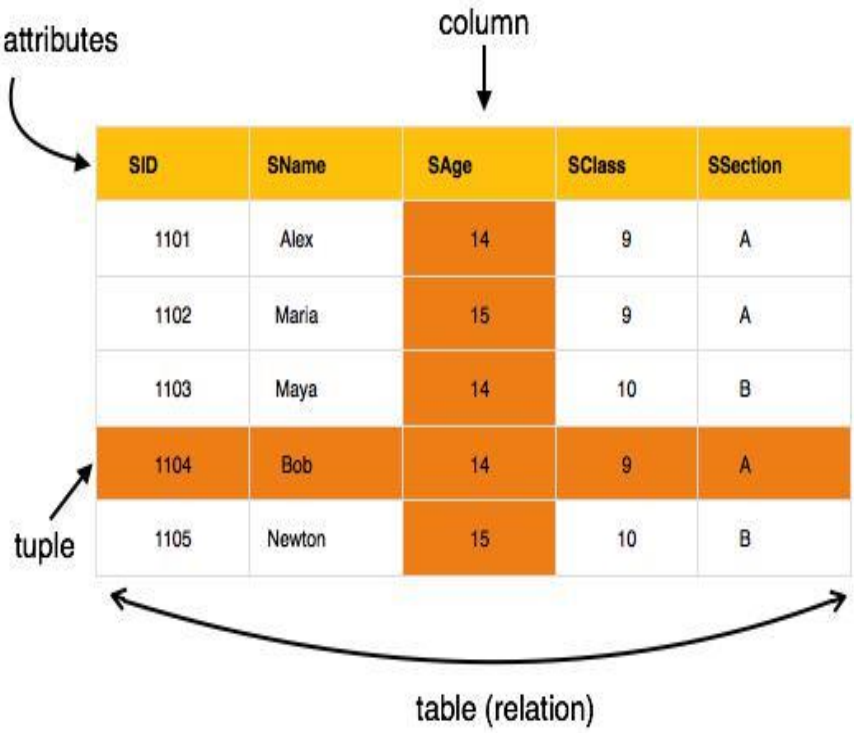
Entity: An entity is a ―thing‖ or ―object‖ in the real world that is distinguishable from other objects. For example, each person is an entity, and bank accounts can be considered as entities. Entities are described in a database by a set of attributes. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.

Relationship: The logical association among entities is called relationship. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.



Relational Model

The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on first-order predicate logic and defines a table as an n-ary relation.



The relational model uses a collection of tables to represent both data and the relationships among those data. Each table has multiple columns, and each column has a unique name.

The relational model is an example of a record-based model. Record-based models are so named because the database is structured in fixed-format records of several types. Each table contains records of a particular type. Each record type defines a fixed number of fields, or attributes. The columns of the table correspond to the attributes of the record type.

The relational model is at a lower level of abstraction than the E-R model. Database designs are often carried out in the E-R model, and then translated to the relational model.

Other Data Models

Object-oriented data model: Object-oriented data model is another data model that has seen increasing attention. The object-oriented model can be seen as extending the E-R model with notions of encapsulation, methods (functions), and object identity.

Object-Relational data model: The object-relational data model combines features of the object-oriented data model and relational data model.

Semi Structured data model: Semi structured data models permit the specification of data where individual data items of the same type may have different sets of attributes. This is in contrast with the data models mentioned earlier, where every data item of a particular type must have the same set of attributes. The extensible markup language (XML) is widely used to represent semi-structured data.

Database Languages

A database system provides a data definition language to specify the database schema and a data manipulation language to express database queries and updates. The data definition and data manipulation languages are not two separate languages; instead they simply form parts of a single database language, such as the widely used SQL language.

Data-Definition Language (DDL)

DDL is used for specifying the database schema. Following statements comes under DDL.

- To create the database instance – CREATE
- To alter the structure of database – ALTER
- To drop database instances – DROP
- To delete tables in a database instance – TRUNCATE
- To rename database instances – RENAME

All these commands specify or update the database schema that‘s why they come under Data Definition language.

Data Manipulation Language (DML): DML is used for accessing and manipulating data in a database.

- To read records from table(s) – SELECT
- To insert record(s) into the table(s) – INSERT
- Update the data in table(s) – UPDATE
- Delete all the records from the table – DELETE

A data-manipulation language (DML) is a language that enables users to access or manipulate data as organized by the appropriate data model.

There are basically two types:

- Procedural DMLs require a user to specify what data are needed and how to get those data.
- Declarative DMLs (also referred to as nonprocedural DMLs) require a user to specify what data are needed without specifying how to get those data.

Data Control language (DCL): DCL is used for granting and revoking user access on a database

- To grant access to user – GRANT
- To revoke access from user – REVOKE

Transaction Control (TCL) : Statements are used to manage the changes made by DML statements. It allows statements to be grouped together into logical transactions.

- COMMIT - save work done
- SAVEPOINT - identify a point in a transaction to which you can later roll back
- ROLLBACK - restore database to original since the last COMMIT
- SET TRANSACTION - Change transaction options like isolation level and what rollback segment to use.

Database Users

Database users are the one who really use and take the benefits of database. There will be different types of users depending on their need and way of accessing the database.

- **Naive Users** - these are the users who use the existing application to interact with the database. For example, online library system, ticket booking systems, ATMs etc which has existing application and users use them to interact with the database to fulfill their requests.
- **Application Programmers** - They are the developers who interact with the database by means of DML queries. These DML queries are written in the application programs like C, C++, JAVA, Pascal etc. These queries are converted into object code to communicate with the database. For example, writing a C program to generate the report of employees who are working in particular department will involve a query to fetch the data from database. It will include a embedded SQL query in the C Program.
- **Sophisticated Users** - They are database developers, who write SQL queries to select/insert/delete/update data. They do not use any application or programs to request the database. They directly interact with the database by means of query language like SQL. These users will be scientists, engineers, analysts who thoroughly study SQL and DBMS to apply the concepts in their requirement. In short, we can say this category includes designers and developers of DBMS and SQL.
- **Specialized Users** - These are also sophisticated users, but they write special database application programs. They are the developers who develop the complex programs to the requirement.
- **Stand-alone Users** - These users will have stand –alone database for their personal use. These kinds of database will have readymade database packages which will have menus and graphical interfaces.

Database Administrator

One of the main reasons for using DBMSs is to have central control of both the data and the programs that access those data. A person who has such central control over the system is called a database administrator (DBA). The functions of a DBA include:

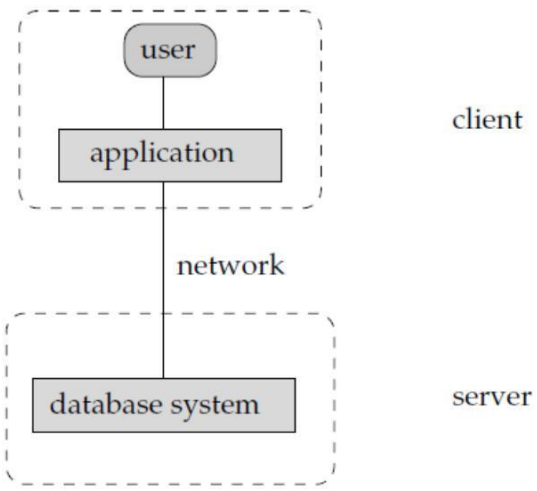
- **Schema definition**. The DBA creates the original database schema by executing a set of data definition statements in the DDL.
- **Storage structure and access-method definition.**
- **Schema and physical-organization modification**. The DBA carries out changes to the schema and physical organization to reflect the changing needs of the organization, or to alter the physical organization to improve performance.
- **Granting of authorization for data access.** By granting different types of authorization, the database administrator can regulate which parts of the database various users can access. The authorization information is kept in a special system structure that the database system consults whenever someone attempts to access the data in the system.
- **Routine maintenance**. Examples of the database administrator‘s routine maintenance activities are:
  - Periodically backing up the database, either onto tapes or onto remote servers, to prevent loss of data in case of disasters such as flooding.
  - Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required.
  - Monitoring jobs running on the database and ensuring that performance is not degraded by very expensive tasks submitted by some users.

Database System Architectures

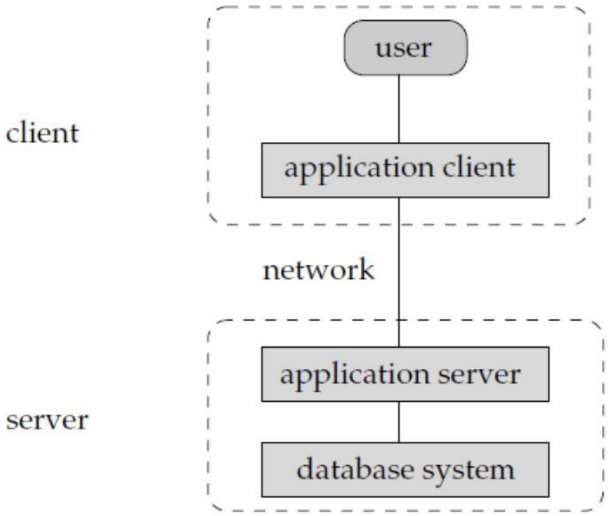
Database architecture is logically divided into two types.

- Logical two-tier Client / Server architecture
- Logical three-tier Client / Server architecture

1. **Two-tier Architecture**: In two-tier architecture, the application is partitioned into a component that resides at the client machine, which invokes database system functionality at the server machine through query language statements. Application program interface standards like ODBC and JDBC are used for interaction between the client and the server.

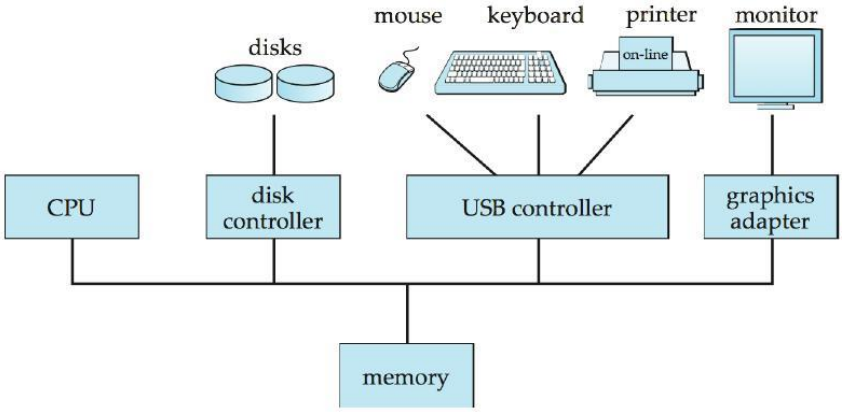


2. **Three-tier Architecture:** In three-tier architecture, the client machine acts as merely a front end and does not contain any direct database calls. Instead, the client end communicates with an application server, usually through a forms interface. The application server in turn communicates with a database system to access data. The business logic of the application, which says what actions to carry out under what conditions, is embedded in the application server, instead of being distributed across multiple clients. Three-tier applications are more appropriate for large applications, and for applications that run on the World Wide Web.

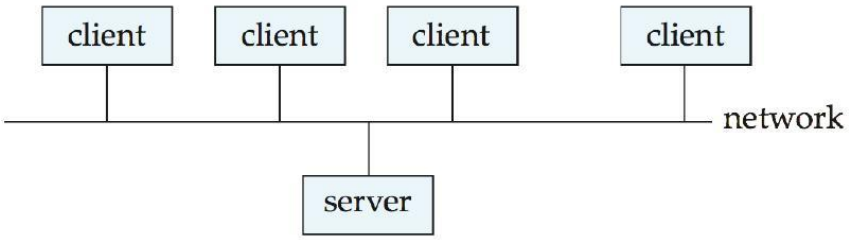


Other Architecture:
- Centralized Systems
- Client-Server Systems

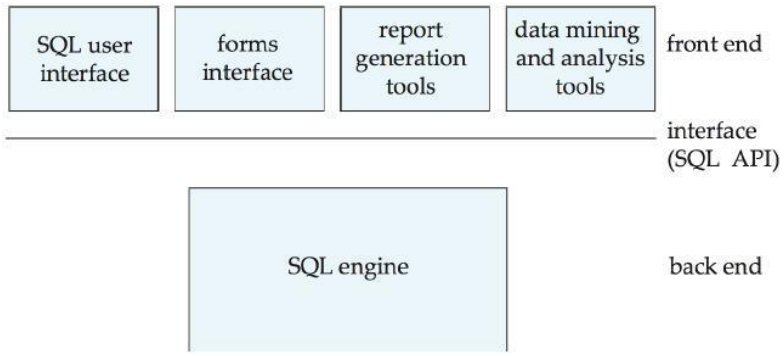
Centralized Systems
- Run on a single computer system and do not interact with other computer systems.
- General-purpose computer system: one to a few CPUs and a number of device controllers that are connected through a common bus that provide access to shared memory.
- Single-user system (e.g., personal computer or workstation): desk-top unit, single user, usually has only one CPU and one or two hard disks; the OS may support only one user.
- Multi-user system: more disks, more memory, multiple CPUs, and a multi-user OS. Serve a large number of users who are connected to the system vie terminals. Often called server systems



Client-Server Systems: Server systems satisfy requests generated at _m' client systems, whose general structure is shown below:



Database functionality can be divided into:



1. Back-end: manages access structures, query evaluation and optimization, concurrency control and recovery.
2. Front-end: consists of tools such as forms, report-writers, and graphical user interface facilities.

The interface between the front-end and the back-end is through SQL or through an application program interface.

Advantages of replacing mainframes with networks of workstations or personal computers connected to back-end server machines:

- better functionality for the cost
- flexibility in locating resources and expanding facilities
- better user interfaces
- easier maintenance

Overall Structure of DBMS

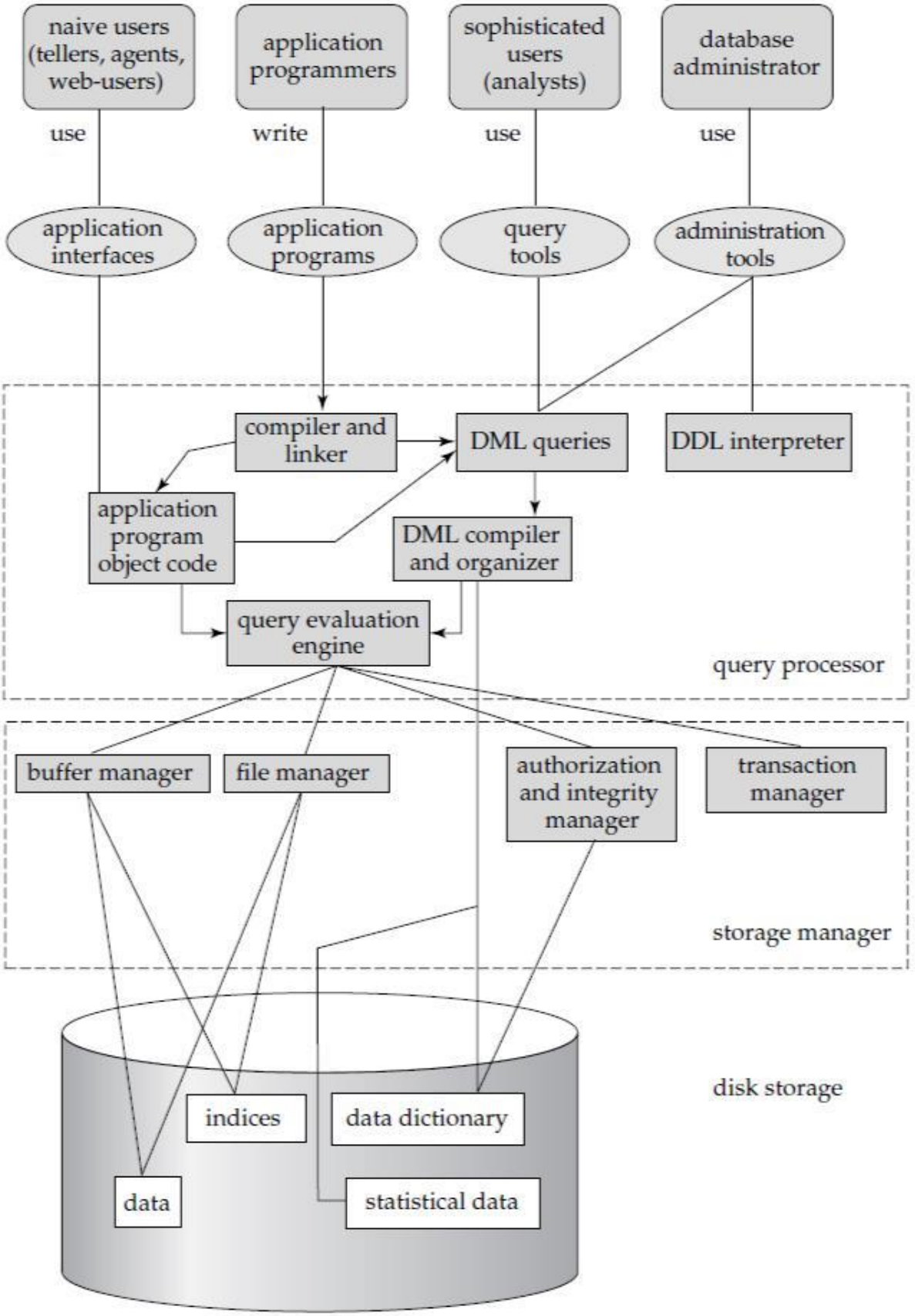
Database Users and User Interfaces

- **Naïve Users:** Naıve users are unsophisticated users who interact with the system by invoking one of the application programs that have been written previously. The typical user interface for naıve users is a forms interface, where the user can fill in appropriate fields of the form. Naıve users may also simply read reports generated from the database.

- **Application programmers** are computer professionals who write application programs. Application programmers can choose from many tools to develop user interfaces.

- **Sophisticated users** interact with the system without writing programs. Instead, they form their requests either using a database query language or by using tools such as data analysis software.

- **Specialized users** are sophisticated users who write specialized database applications that do not fit into the traditional data-processing framework. Among these applications are computer-aided design systems, knowledgebase and expert systems, systems that store data with complex data types (for example, graphics data and audio data), and environment-modeling systems.


naive users (tellers, agents, web-users)
application programmers
sophisticated users (analysts)
database administrator
use
write
use
use
application interfaces
application programs
query tools
administration tools
compiler and linker
DML queries
DDL interpreter
application program object code
DML compiler and organizer
query evaluation engine
query processor
buffer manager
file manager
authorization and integrity manager
transaction manager
storage manager
disk storage
indices
data dictionary
data
statistical data

Storage Manager:
The storage manager is the component of a database system that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system. The storage manager is responsible for storing, retrieving, and updating data in the database.

The storage manager components include:

- **Authorization and integrity manager**, which tests for the satisfaction of integrity constraints and checks the authority of users to access data.
- **Transaction manager**, which ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting.
- **File manager**, which manages the allocation of space on disk storage and the data structures used to represent information stored on disk.
- **Buffer manager**, which is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

The storage manager implements several data structures as part of the physical system implementation:

- **Data files**, which store the database itself.
- **Data dictionary**, which stores metadata about the structure of the database, in particular the schema of the database.
- **Indices,** which can provide fast access to data items. A database index provides pointers to those data items that hold a particular value. For example, we could use an index to find the instructor record with a particular ID, or all instructor records with a particular name. Hashing is an alternative to indexing that is faster in some but not all cases.

The Query Processor
The query processor components include:

- **DDL interpreter**, which interprets DDL statements and records the definitions in the data dictionary.
- **DML compiler**, which translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands. The DML compiler also performs query optimization
- **Query evaluation engine**, which executes low-level instructions generated by the DML compiler.

Entity-Relationship Model
The entity-relationship (E-R) data model perceives the real world as consisting of basic objects, called entities, and relationships among these objects. It was developed to facilitate database design by allowing specification of an enterprise schema, which represents the overall logical structure of a database.
The E-R model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema. Because of this usefulness, many database-design tools draw on concepts from the E-R model.

Entity

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities.

All these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Attributes

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Types of Attributes

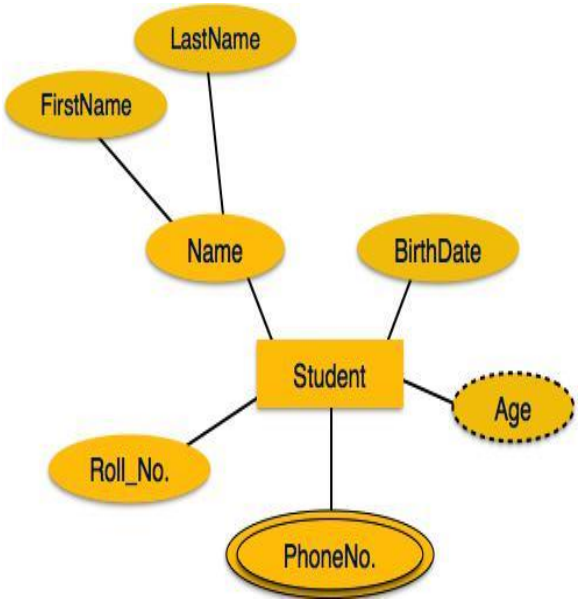
- **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's Births Date is an atomic.

- **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.

- **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For another example, age can be derived from BirthDate.

- **Single-value attribute** − Single-value attributes contain single value. For example – Roll_No.

- **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.



Keys: Key is an attribute or collection of attributes that uniquely identifies an entity among entity set. For example, the roll_number of a student makes him/her identifiable among students.

- **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.

- **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Relationship

The association among entities is called a relationship. For example, an employee works_at a department, a student enrolls in a course. Here, Works_at and Enrolls are called relationships.

Relationship Set: A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

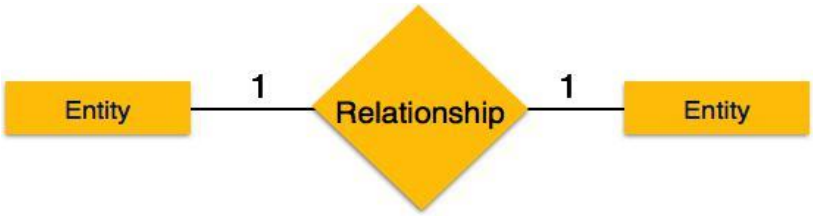
Degree of Relationship: The number of participating entities in a relationship defines the degree of the relationship.

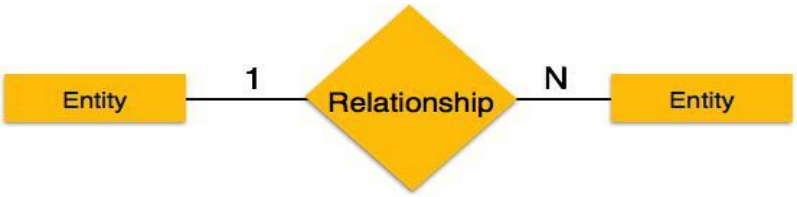
- Binary = degree 2
- Ternary = degree 3
- n-ary = degree

Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

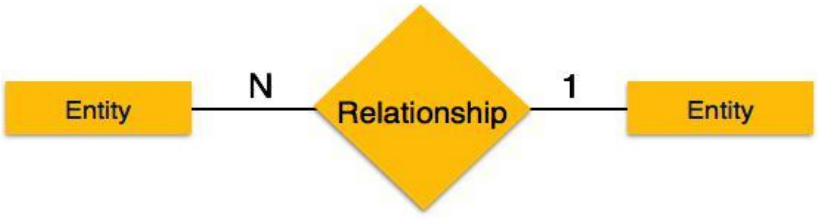
- **One-to-one** − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.

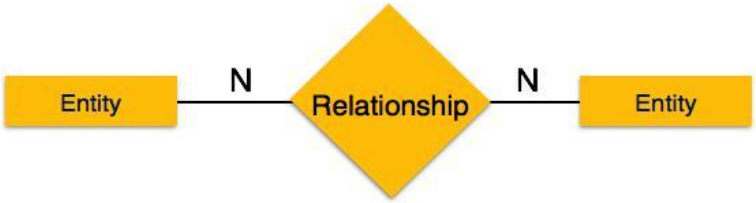




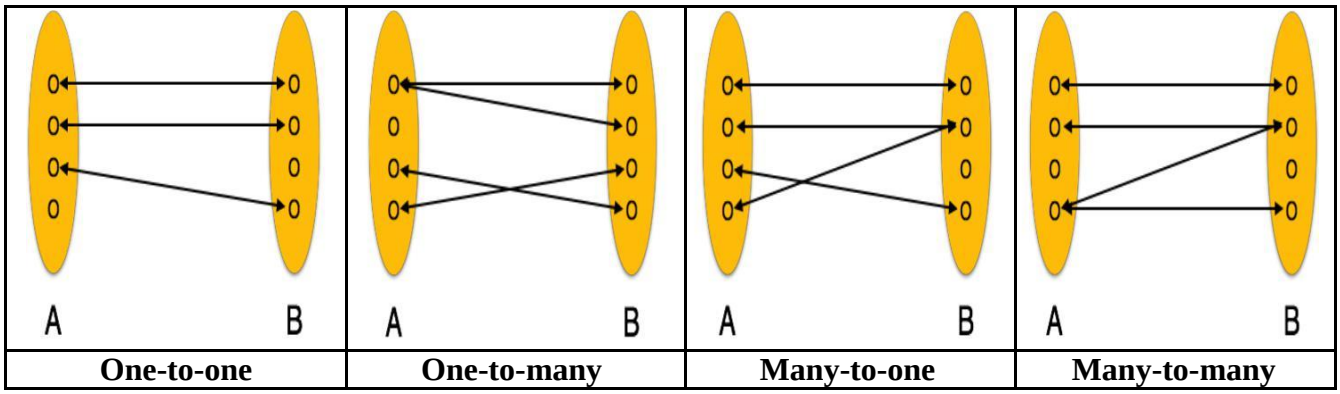
- **One-to-many** − One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.

- **Many-to-one** − More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



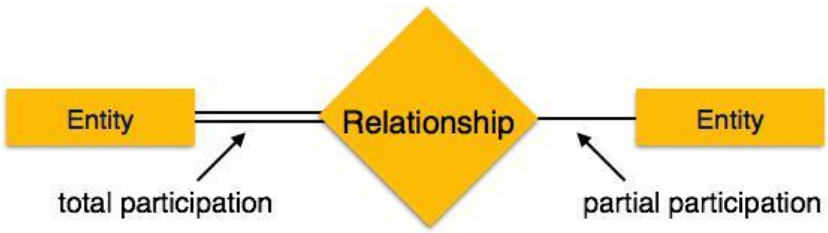


- **Many-to-many** − One entity from A can be associated with more than one entity from B and vice versa.

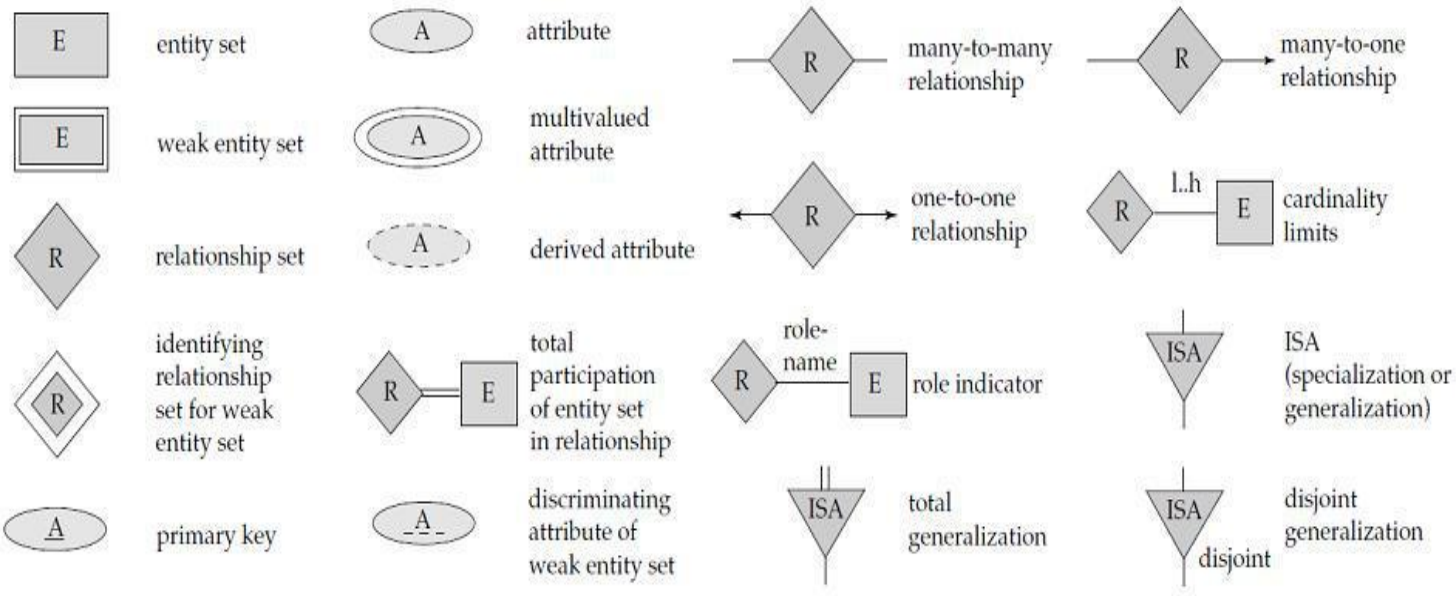


Participation Constraints

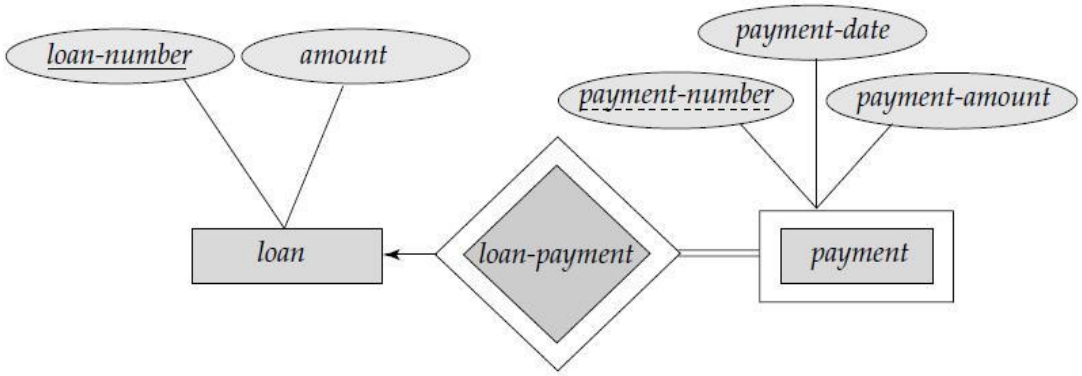
1. Total Participation − each entity is involved in the relationship. Total participation is represented by double lines.
2. Partial participation − Not all entities are involved in the relationship. Partial participation is represented by single lines.



Entity-Relationship Symbols:



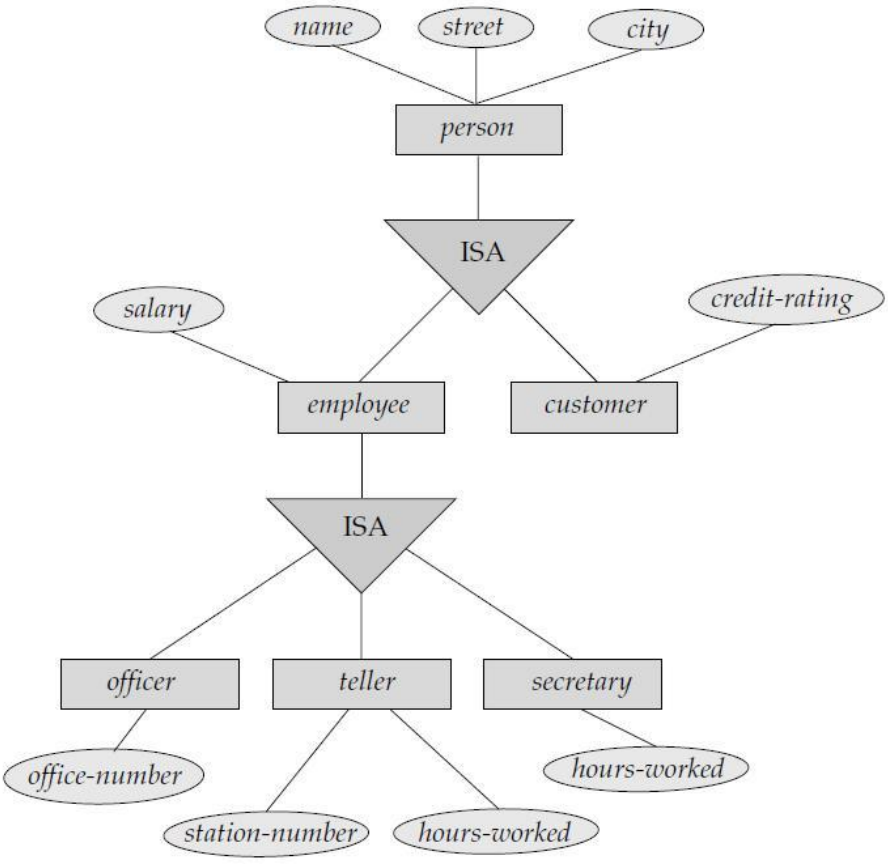
Weak Entity Sets: An entity set may not have sufficient attributes to form a primary key. Such an entity set is termed a weak entity set. An entity set that has a primary key is termed a strong entity set.



For a weak entity set to be meaningful, it must be associated with another entity set, called the identifying or owner entity set. Every weak entity must be associated with an identifying entity; that is, the weak entity set is said to be existence dependent on the identifying entity set. The identifying entity set is said to own the weak entity set that it identifies. The relationship associating the weak entity set with the identifying entity set is called the identifying relationship. The identifying relationship is many to one from the weak entity set to the identifying entity set, and the participation of the weak entity set in the relationship is total.

Specialization

An entity set may include sub groupings of entities that are distinct in some way from other entities in the set. For instance, a subset of entities within an entity set may have attributes that are not shared by all the entities in the entity set. The E-R model provides a means for representing these distinctive entity groupings. The process of designating sub groupings within an entity set is called specialization. In an E-R diagram, specialization is depicted by a triangle component labeled ISA.



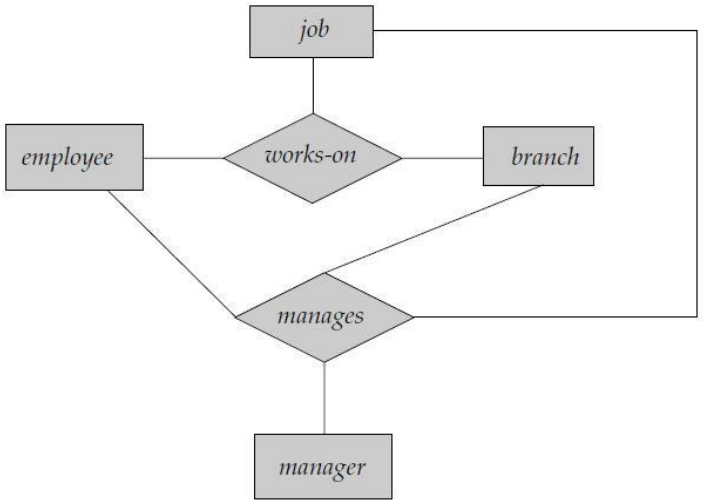
The label ISA stands for ―is a‖ and represents, for example, that a customer ―is a‖ person. The ISA relationship may also be referred to as a super class-subclass relationship. Higher- and lower-level entity sets are depicted as regular entity sets—that are, as rectangles containing the name of the entity set.

Generalization:

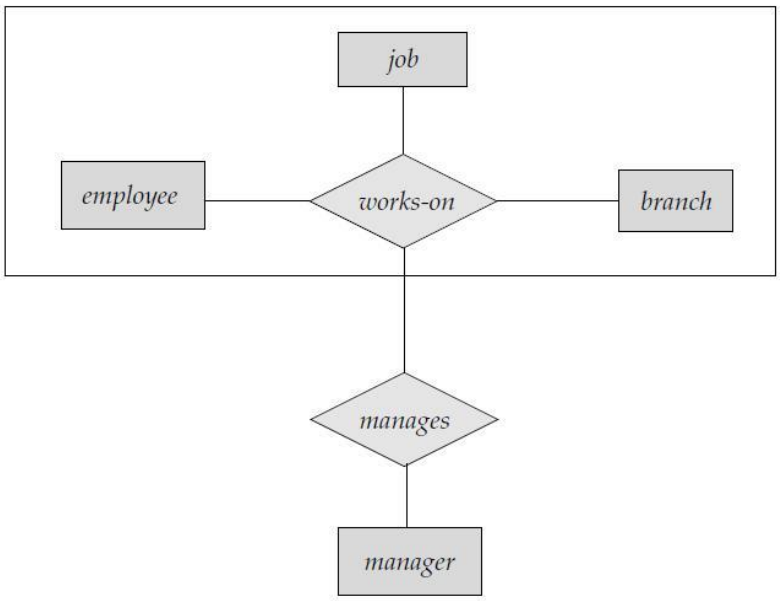
The refinement from an initial entity set into successive levels of entity sub groupings represents a top-down design process in which distinctions are made explicit. The design process may also proceed in a bottom-up manner, in which multiple entity sets are synthesized into a higher-level entity set on the basis of common features. This commonality can be expressed by generalization, which is a containment relationship that exists between a higher-level entity set and one or more lower-level entity sets. In our example, person is the higher-level entity set and customer and employee are lower-level entity sets.

Aggregation: One limitation of the E-R model is that it cannot express relationships among relationships. The best way to model a situation such as the one just described is to use aggregation. Aggregation is an abstraction through which relationships are treated as higher level entities.

For example the following figure shows the ternary relationship works-on, which we saw earlier, between an employee, branch, and job.



There is redundant information in the figure, however, since every employee, branch, job combination in manages is also in works-on.



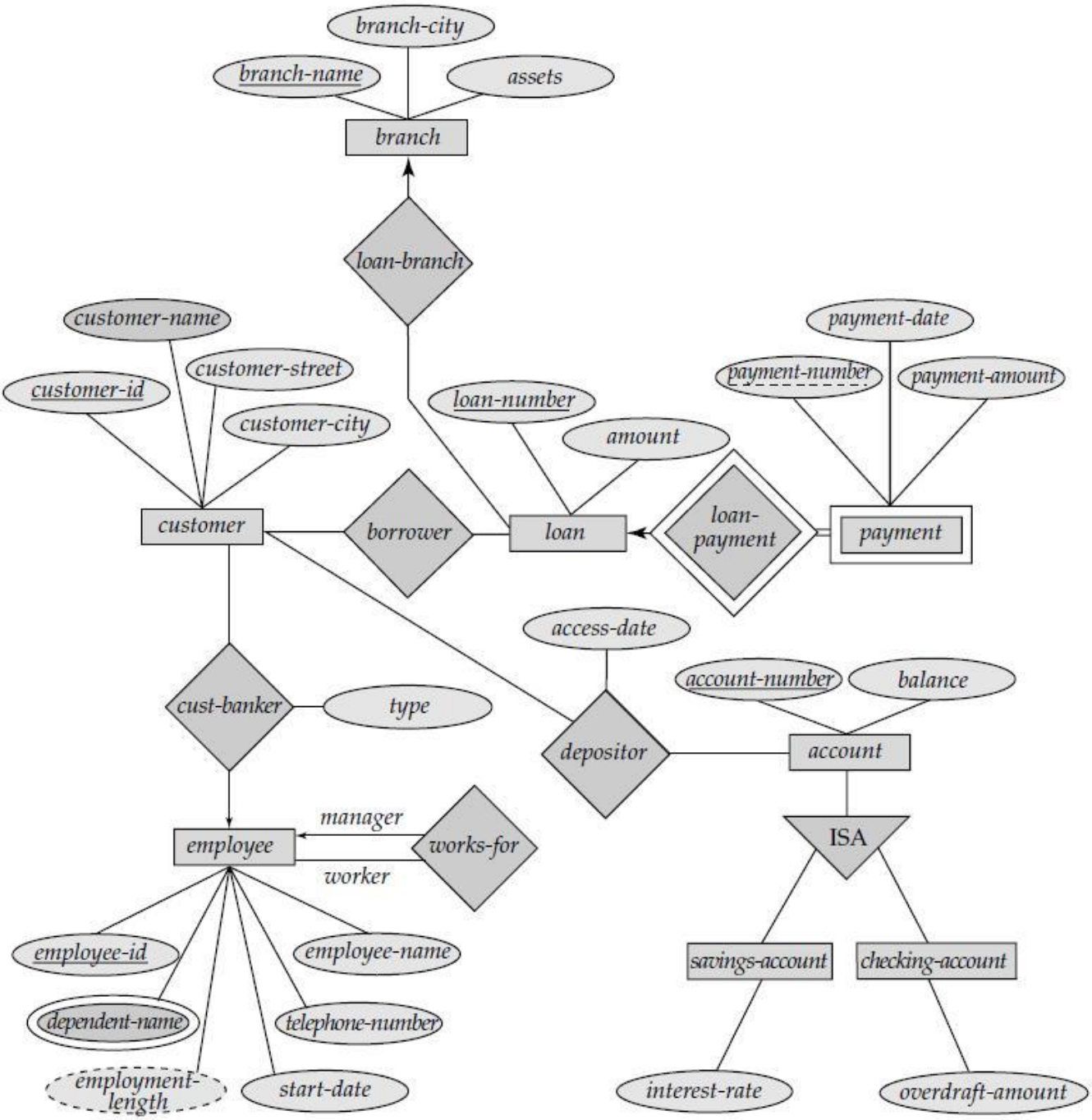
That is, from the above figure following results may be depicted:

- Employee manages job
- Job manages Employee
- Employee manages branch and vice-versa

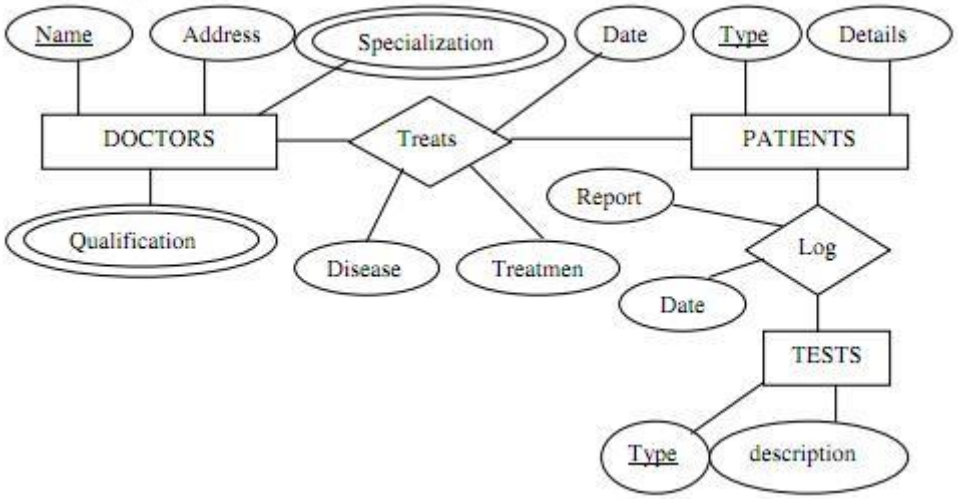
To remove these anomalies we use aggregation.

Example of E-R Diagrams:

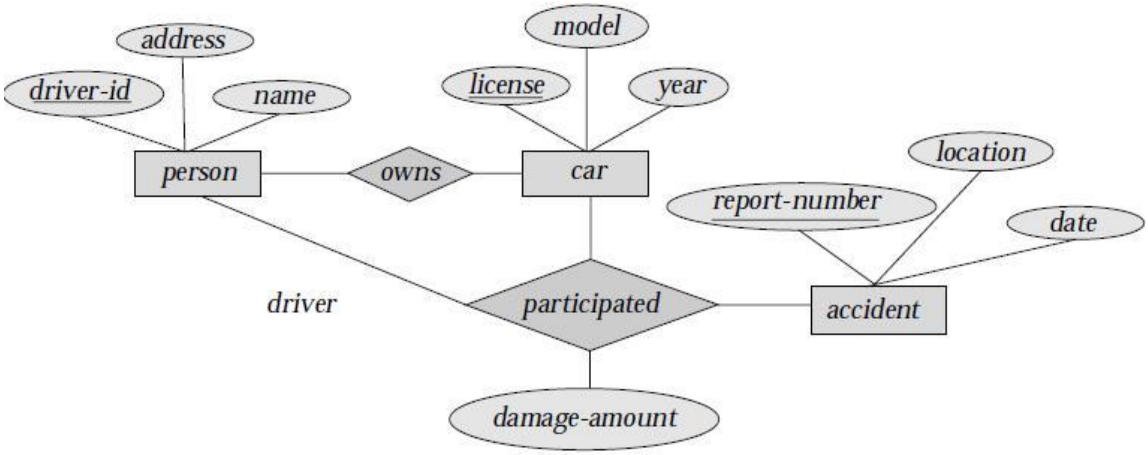
1. E-R diagram for a banking enterprise.



2. An E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.



3. An E-R diagram for a car-insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents.

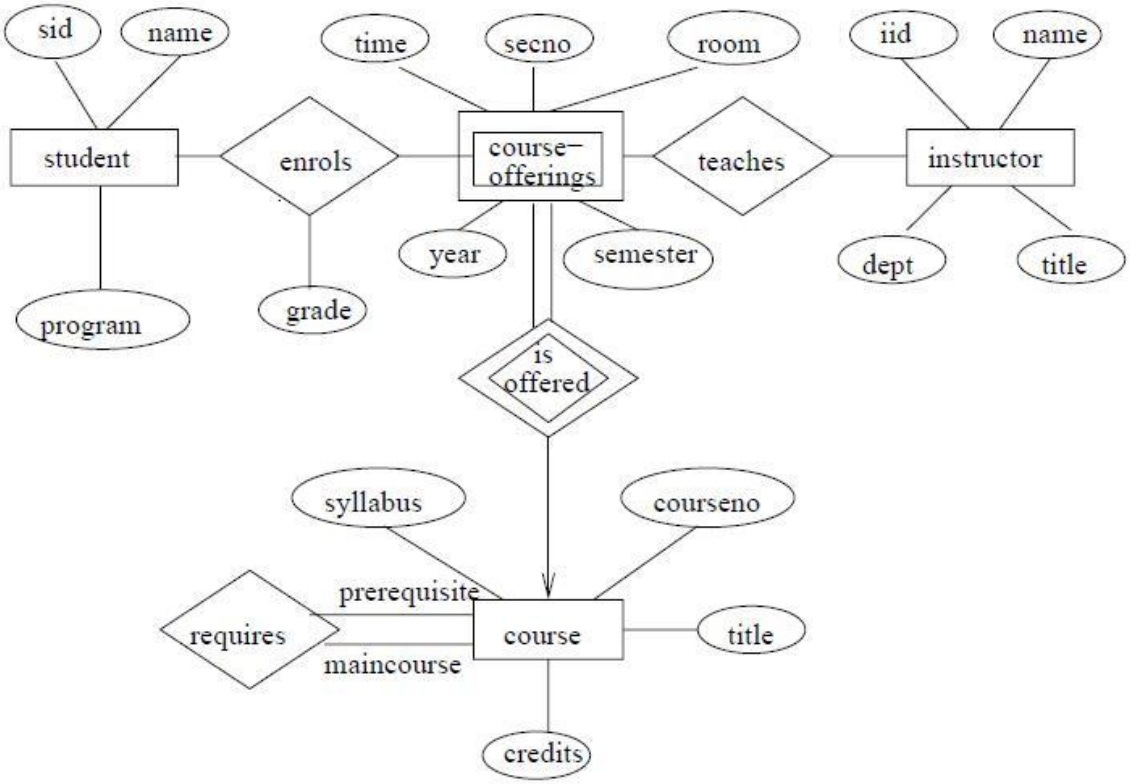


4. A university registrar‘s office maintains data about the following entities:

- **a.** courses, including number, title, credits, syllabus, and prerequisites;
- **b.** course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;
- **c.** students, including student-id, name, and program;
- **d.** instructors, including identification number, name, department, and title.

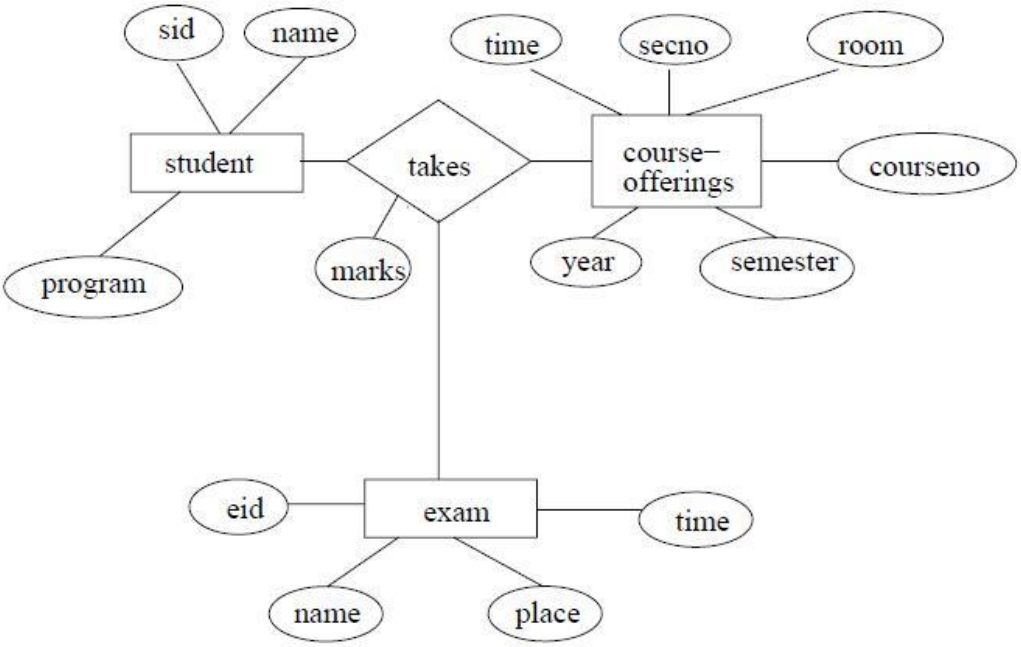
Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.

Construct an E-R diagram for the registrar‘s office. Document all assumptions that you make about the mapping constraints.

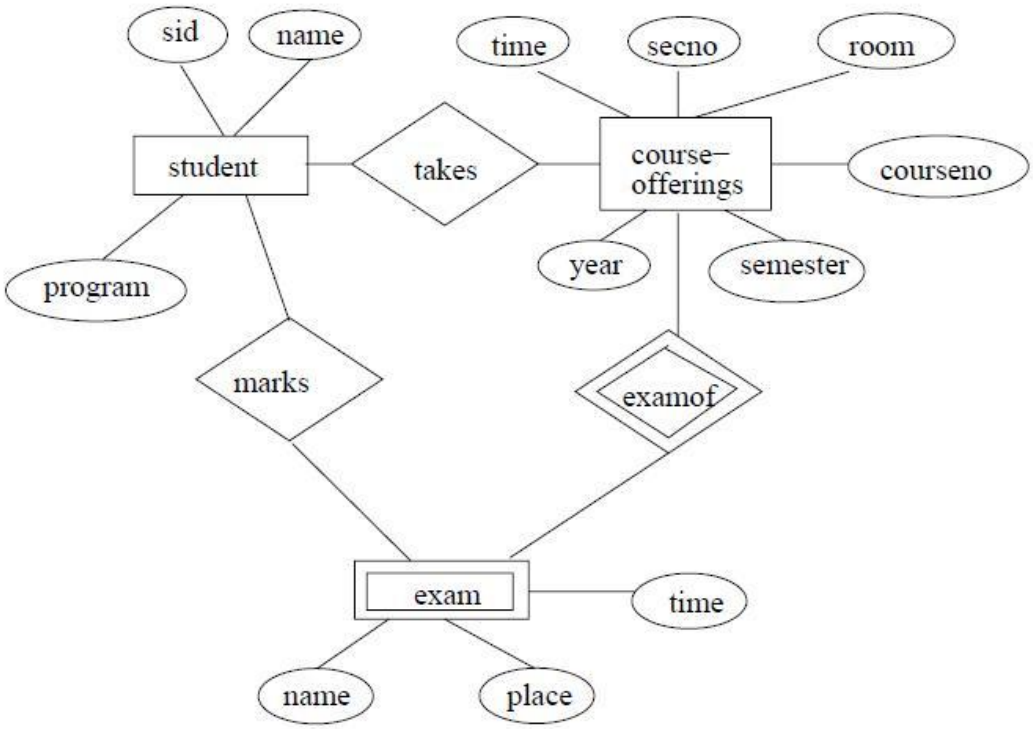


5. Consider a database used to record the marks that students get in different exams of different course offerings.

a. Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database



b. Construct an alternative E-R diagram that uses only a binary relationship between students and course-offerings. Make sure that only one relationship exists between a particular student and course-offering pair, yet you can represent the marks that a student gets in different exams of a course offering.



6. Design an E-R diagram for keeping track of the exploits of your favourite sports team. You should store the matches played, the scores in each match, the players in each match and individual player statistics for each match. Summary statistics should be modeled as derived attributes