Zhongwei Wang

2021374489

COURSE # 87521, Sec 005

1.1

Data are facts and statistics collected together for the reference of analysis.

Database is a collection of related data extracted from any firm or organization.

DBMS (Database Management System) is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the process of defining, constructing, and manipulating database.

A **Database Systems** comprises a database of operational data, together with the processing functionality required to access and manage that data. The combination of the DBMS and the database is called database systems.

A **database catalog** contains complete description of the databases, database objects, databases structure, details of users, and constrains etc. that are stored.

Program-data independence refers to storing the data and programs separately. In traditional file-implemented database, the structure of data files is closely related to the programs. That means if we change the structure of the data file, those programs involving these data should be changed as well. But now DBMS allows data structure to be stored separately with the DBMS programs and the data definition. Overall, the new idea is the program-data independence.

A **user view** is the appearance that a particular user sees from the database.

DBA (Database Administrator) is a person who is responsible for authorizing access to the database, coordinating and monitoring its use, and acquiring software and hardware resources as needed.

End users are persons who want to access the database for various purposes.

Canned Transactions are actions such as standardized queries and updates to the database. Usually they are performed by using well-programed and well-tested programs.

A Deductive Database System is a database system provides capability of performing deductions based on rules and facts pre-stored in the system.

A **persistent object** refers to an object in the object-oriented database system. It is usually related to a programing language such as java or C++. Additionally, it shall survive the termination of the DBMS program.

Meta-data refers to those information that describes data. For example, information stored in the catalog is called meta data.

A **Transaction-processing application** is the application dealing with transaction processing. A Transaction is an indivisible operation, such as insert, delete, etc. Processing includes one or more transactions are transaction-processing.

1.3

First, the database approach has the nature of describing itself. Unlike a traditional file processing in which data definition is typically part of the application programs, the new approach contains not only the database itself, but also a complete definition describing the database. For example, the meta data are such descriptions and describes the structure, access time, and other information in database. Meanwhile, in the traditional file processing, data structures are declared in the application programs which will lead to huge work if one has to change the structure.

Second, the approach typically has an insulation between programs, data, and data abstraction. As mentioned before, in traditional file processing, the structure of data file is implemented in the application programs leading to huge changes. Now, the DBMS in the new approach does not require changes in lots of cases due to its separation of application program level and stored database level. In the database approach, the structure of data file is stored in DBMS catalog separately from the access programs.

Third, the new approach supports multiple view of data. A typical traditional approach does not support various views of the same data, but for a multi-user DBMS, user view varies from users to users.

Fourth, the new approach allows multiple users concurrency access and update the database at the same time. DBMS has a mechanism dealing with the concurrency central software to ensure that the concurrency access of data is handled correctly. In other words, the same data could be visited by multiple users at the same time. And they all have the capability of updating the data under the control of the DBMS. Meanwhile, in traditional file system, no such data sharing is possible, hence there is no concurrency access.

1.6

Controlling Redundancy is one of the main feature of DBMS. The redundancy leads to duplication of space and effort for updating for the same data.

Restricting Unauthorized Access is another big feature of DBMS. In real world, lots of users may access the database with different limits of authority. Some may only be permitted to read, while others could read and write. As a result, the DBMS must have a way to control the authority of accessing.

Providing Persistent Storage for Program Objects is also a great feature specially for object-oriented database system. Usually, the values of variables are discarded once a program halts. But in OO database system, if a programmer explicitly stored them in permanent files, they can be persistent stored.

Implementing good structures for efficient query processing. Queries and updates are main instructions used in daily life. Therefore, the DBMS must provide good data structure to be efficient on responding.

Backup and Recovery become more and more important nowadays. A good DBMS must provide capability to recover from hardware or software failures.

Providing Multiple User Interfaces is another requirement for DBMS. Customers, employees, programmers, and supervisors usually want to see the same data in their own perspective way. Providing multiple user interfaces is more user-friendly.

Representing Complex Relationships is also important for a database. Even though the mini world we try to represent in a database is relatively smaller than the real world, people always want the database has the capability in representing various relationships among the data as well as adding, deleting and changing relationships easily.

Enforcing Integrity Constraints. A DBMS should provide capabilities for defining and enforcing certain constraints that must hold for the data.

1.7

Database Approach

+ Like a file contains information about more than one entity and information as well as their relationships among those entities.

+ Each entity is stored to a table in the database

+ Supple for multiple users and can be used in various applications

- Complex to design, implement and maintain.

- Relatively slow due to its complicated mechanism.

+ Data independence makes it more reliable and less changing effort.

+ Less redundancy improves data consistence, enforcement of standards improved data quality.

+ Data are flexible and scalable.

Information Retrieval System

- Basically a file system developed on the route.

- Different OS has its own unique set of files to represent data.

- Users always see the same interface unless the system developer provides them a new interface.

+ Easy to design and implement.

+ Relatively fast because it is simple.

- Data are dependent, leading to redundancy or more effort to maintain and change.

- Data Redundancy!

- Not flexible or scalable

1.8

Get the transcript of Brown

List the prerequisites of the Data Structure course.

List the students who took the class CS1310-Intro to Computer Science in 2007.

Give a letter grade ‘A’ to Smith in the Database section 135 CS3380 database in 2008.

Create a new section of Data Structure course for 2008 Spring.

Change Brown’s major to Biotech.

1.14

a.

Table	Columns Name that Needs to be Changed
Student	**Major**
Course	**Course_number, Department**
Section	**Course_number**
Prerequisite	**Course_number, Prerequisite_number**

b.

Delete the department column in the Course table, we can still find the correct department info from its Course_number.

No needs to change the Section table.

I don’t know what to change in the Prerequisite table so that only one column modification is enough. But if we assume that all prerequisite course must be taking from the same department, ie. CS courses only has CS courses as its’ prerequisite, then we can change the prerequisite_number into a format that without the department initials. The prerequisite table would look like this:

Prerequisite table	Dept_Name	Course_Number (Only the digits)	Prerequisite_Number (Only the digits)

2.1

Data Model

A collection of concepts that can be used to described the structure of a database, providing the database abstractions. Data models is also collections of conceptual tools for describing data, data relationships, data semantics and consistency constrains.

Database schema

It is the design of the database. A database schema covers how data is stored, what data is stored, and how high-level app could make use of these data. Variables are declared in a program and can have one particular value at a given instant for each variable.

Database state

The data in the database at a certain moment is called database state.

Internal schema

It is the physical stage structure of the database.

Conceptual schema

The descriptions of the database structure for a community of users are called conceptual schema. It hides the details of physical storage.

External schema

External schema describes the part of the database that a user group can see and how it sees it. It is called a database view as well.

Data independence

The capability of changing the schema at one level without having to change the schema at the next higher level is called data independence.

DDL

Data Definition Language. A language used by DBA and developers to create a database with specific schemas.

DML

Data Manipulation Language. A language used to perform data manipulations/operands such as addition, insertion, deletion, etc.

SDL

Storage Definition Language. A language used to implement the internal schema of the database.

VDL

View Definition Language. A language used to implement the user view and mapping them back to the conceptual schema.

Query language

It is a high-level data manipulation language.

Host language

It is a computer language both used in high-level and low-level data manipulation. Usually data manipulation commands will trigger the embedded host language to perform. A programmer could code in this language to perform his order directly as well.

Data sublanguage

The data manipulation language embedded in the last language is called data sublanguage.

Data utility

Software modules that help the DBA to manage a database system is called database utility.

Catalog

A complete description of the database structure and constraints is called catalog.

Client/Server architecture

It is a database architecture in which the system functionality is distributed between two types of models. A client module consists of application programs to provide a user interface to access the database.

Three-tier architecture

It is a designed architecture for three separate levels. It separates the application logic process living in the middle-tiers from the data and the user interface (presentation) tiers. It benefits the system into a more scalable, less redundancy, and more flexible manner.

N-tier architecture

It uses several tiers of computers to interpret requests and transfer data between one place to another. The other tiers are at the source of the data. Each tier is completely independent of all the other tiers, except for those immediately above and below it. For each level, it only needs to know how to handle a request from n+1 tier and how to forward the request to the n-1 tier.

2.3

Database schema is a description of the database and the database state is the database itself. The description of a database, A.K.A the database schema, is defined during the design of a database. It should not be changed frequently. For data models, there is a certain way of displaying schemas as diagram. This is also called a schema diagram. It displays the structure of each record type but not the actual instances of records. A schema diagram only displays some aspects of a schema, such as the name of record types and data items, etc.

On the other hand, the data in the database at one moment is called a database state. It is also called the current set of occurrences and instances in the database. For each database state, each schema construct has its own current set of instances. Every time we insert or delete a record, we have change one state into another.

Database schema is defined and embed into DBMS during the time programmers define a new database. But at that time, there is no actual data stored in the database. So, the database state is an empty state with no data. Last but not least, the schema is also called the intension, while the state is called extension of the schema.

2.4

A three-schema architecture could be divided into 3 levels: internal level, conceptual level, and external level. The purpose for dividing is to separate the user applications and physical database.

The internal level has an internal schema, describing the physical storage structure of the database.

The conceptual level has a conceptual schema, describing the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations and constrains.

The external level includes a number of external schema such as user views. Each describes the part of the database that the user would be focus on, and hides the rest of information from the database.

Mapping is the process that transferring requests and results between levels. The conceptual internal mapping defines the correspondence between the conceptual view and the shared database. It also defines how conceptual records and fields are represented at the internal level.

In the same way, an external conceptual mapping defines the linkage between a particular external view and the conceptual view.

DDL is the language used to specify conceptual and internal schemas for the database as well as the mappings between them.

SDL is the language used in the internal schema to store data. It may define the type and how data are stored in the internal level, as well as dealing with those information in the mapping.

VDL is the language used to specify user view and their mapping to the conceptual schema.

2.5

Logical data independence is the idea that to change the conceptual schema without having to change the external schemas such as the application programs.

Physical data independence is the idea that to change the internal schema without having to change the conceptual schema, hence no need to change the external neither.

In my opinion, the logical data independence is more difficult to achieve. Since application programs are heavily dependent on the logical structure of the data, any change of the structures and constrains may affect the application programs and how they can be used.

2.6

Nonprocedural DML is the high-level language. It describes what data is needed without specifying how to get it. Commands can be embedded in a general-purpose programming language, such as SQL. On the other hand, procedure DML is a low-level language which describes what data and how data are stored. Stand-alone DML commands can be applied directly in procedural DML. It is more relevant to the algorithm.

2.14

Personally, I prefer the 3-tier client/server architecture.

Since our task is a web based application, numbers of users would try to access the database concurrently. Meanwhile, most of our users are not trained. So, it is better to keep processing away from the user end. User can interact with user interface and submit the transactions and a Web server can handle those transactions. Another important reason is that concurrency control is quite often in our task. 3-tier could handle it better than other architecture since the server is away from the end user.

As the application is web based, use of main frame computers, where user terminals give user only the capability of viewing and also doing all processing at the same server, will make system slow and inefficient for use. Additionally, users' location would vary widely. But providing application server at every location cost too much or even unrealistic. Thus, 2-tier architecture cannot be used for this task.

Overall, the 3-tier architecture is the best choice.

7.1

High-level data model does not have overwhelmed details, hence they are easy to understand and useful in communicating with non-professional users.

Meanwhile, they can still be the representative of mini-world data that all users' requirement should be met.

Third, it helps the designers to focus on the properties of data, without being distracted by other details such as storage details.

Overall, it makes the creation of good conceptual design easier.

7.3

Entity is an object with independent physical or conceptual existence in the real world.

Attribute is a feature or properties that represent its significance in real world or describe it. Attribute is important parts to entities.

Attribute value is the value of attributes. For example, car has the attribute of colors, and red is one possible attribute value for a certain car entity.

Relationship instance is an association of entities, where the association includes exactly one entity from each participating entity type. For example, in a factory's database, the employee entity has relationship with the department entity. An employee must work for one department. So, this relationship instance set the association between one employee and one department.

Composite Attribute is an attribute that can be divided into smaller sub-attributes. For example, the name can be divided into first name, middle name, last name, and suffix. So, the name attribute is a composite attribute.

Multivalued attribute is a kind of attribute that for a single entity, it can hold multiple values. For example, an employee could have one, two or more phone numbers. So, phone number is a multivalued attribute to the entity table of employee.

Derived attribute is another attribute that can be derived (calculate or figure out) by other attribute. For example, if we have the birthday of employee, then the age can be calculated. Thus, the age attribute is a derived attribute.

Complex attribute is an attribute that composite and multivalued attributes can be nested. For example, an employee could have more than one address, for each address it can associated with multiple phone numbers. Then the addr_phone attribute is a composite attribute.

Key attribute is the attribute that is unique. For each entity value, its key attribute must be different than the others. For example, the employee entity table could use SSN as the key attribute.

Value set is a range for an attribute of a real world entity. For example, the age of an employee must lay in the range of 18-70.

7.4

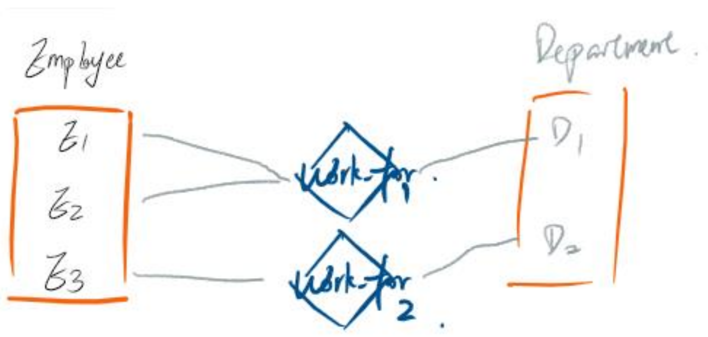
Entity type defines a collection of entities that have the same attributes. A database usually contains a group of entities that are similar. These entities have same attributes but different attribute values. For example, the airline ticket database. The fight is an entity type. It may have lots of entity values which have the same destination and depart location; but their airline company, price, and time would be different.

Entity set is an arbitrary set of the whole set of an entity table. For example, after a query of search tickets, user could see a special list of fights that meet his needs. The list is an entity set. It could contain 1,2,3, more or even 0 entity values.

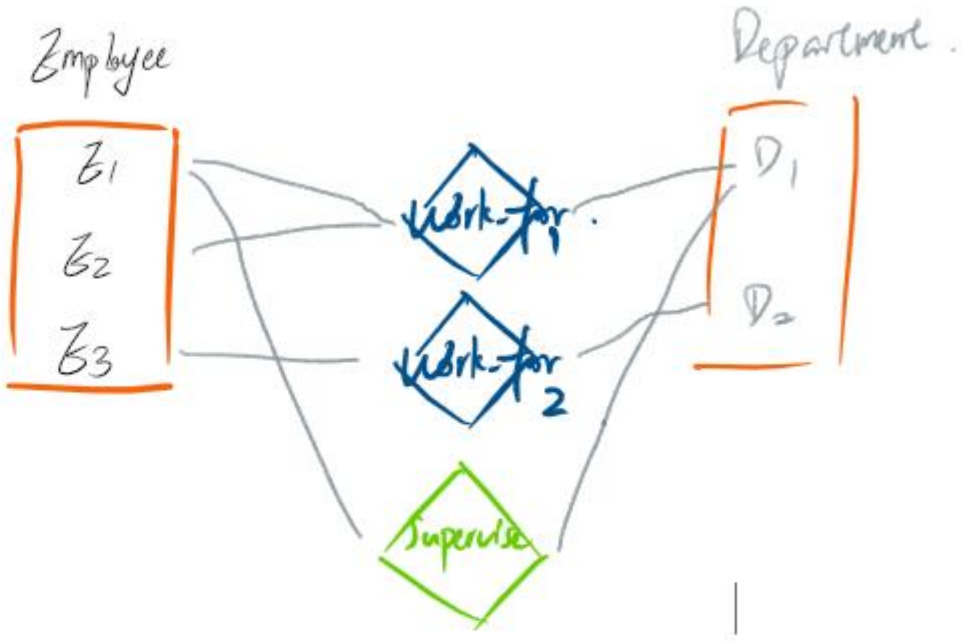
An entity is a real world object that has independent physical or conceptual existence. A collection of entities is called entity type. Usually, the entity type names after the common attribute of the collection of entities. At a particular time, some subset of entities stored in the database is an entity set.

7.7

Participation Role signifies role that a participating entity from the entity type plays in each relationship instance, and helps to explain what relationship means. The figure below is an illustration of PR.

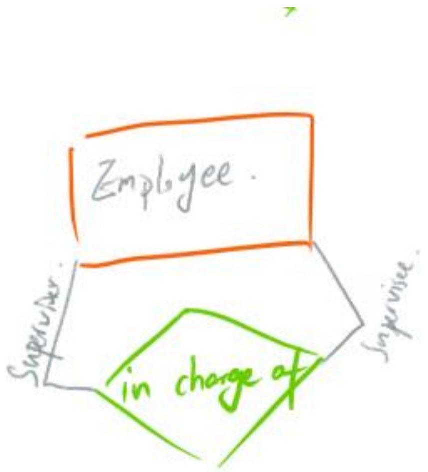


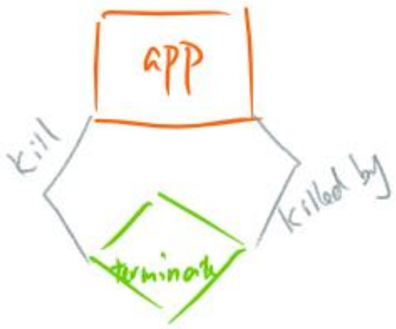
When one entity type participates in a relation in more than one role, it becomes necessary to use role names in the description of relationship types. (See the figure below)



7.11

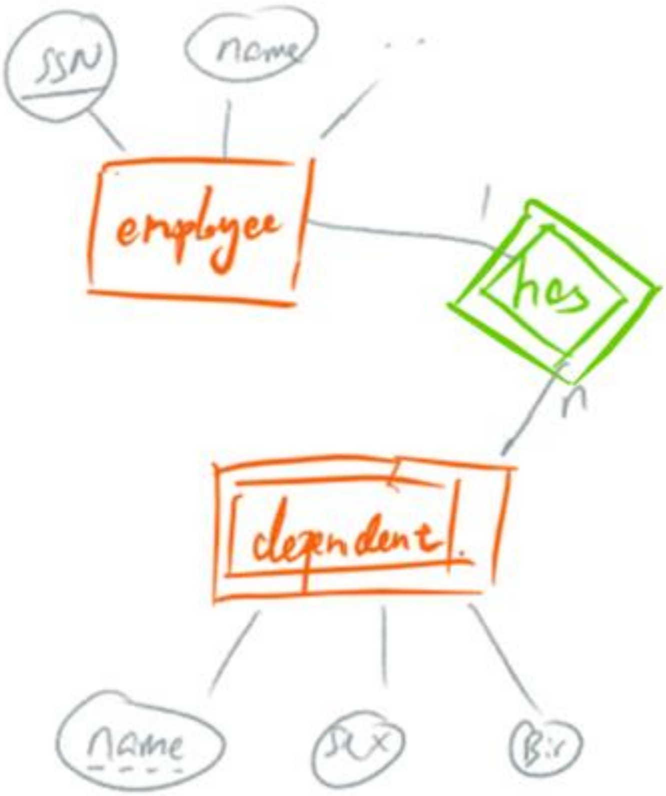
If the same entity type participates more than once in a relationship type in different roles then the relationship types are called recursive relationship. See the figures below.





7.12

The concept is used in conceptual phase of data modeling while modeling entity types who do not have key attributes of their own. See the figure below.



Owner entity type is a strong entity type (those entities having key attributes) who “owns” the weak entity type.

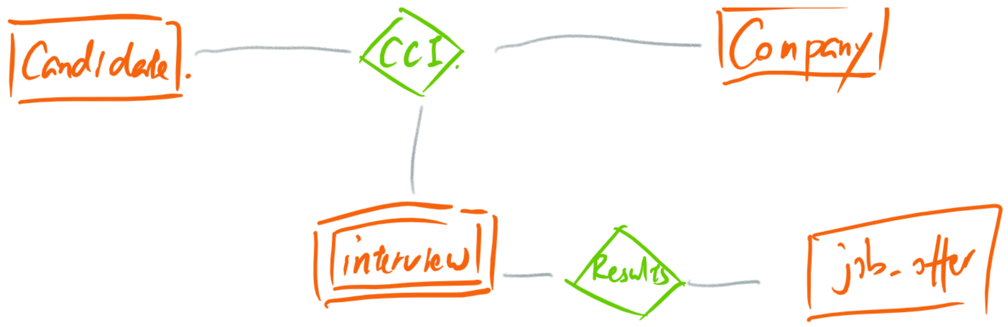
Weak entity type is on the contract of the strong entity type. It does not have any key attributes.

Identifying relationship type is a relationship type that relates weak entity to its owner entity type.

Partial key is a set of attributes in weak entity types that can uniquely identify weak entities related to the same owner entity. In other word, it can take duplicate values as long as the same duplicates does not belong to the same owner entity.

7.13

Yes, if we allow that. See the figure below.

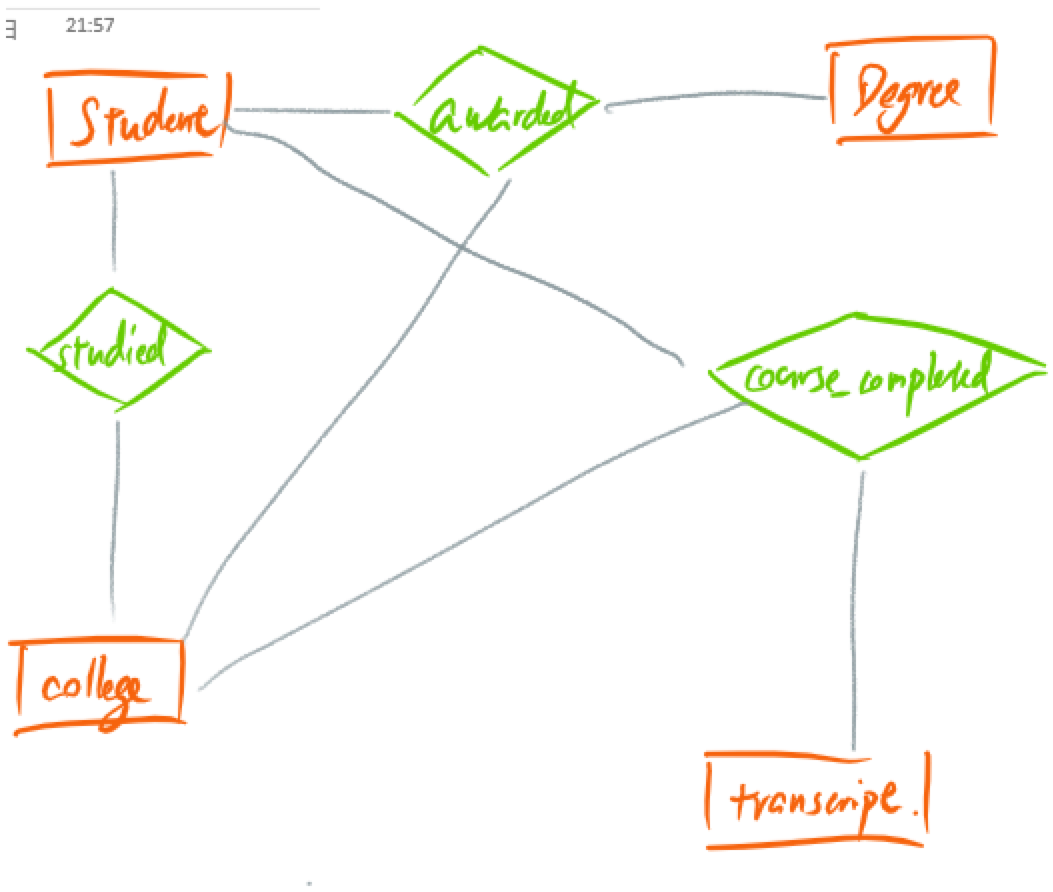


7.17

Prev_college{College_name, Start_date, End_date, {Degrees(Degree_name, month, year)}, {Transcript(Transcript_name, Semester, Year, Gade)}}

7.18

See the figure below:



7.19

Each flight is identified by Number, and consists of one or more Fight_Legs with Leg_no. And flies on certain weekdays.

Each Fight_Leg has scheduled arrival and departure time and arrival and departure airport and one or more Leg_Instances-one for each Date on which flight travels.

For each Flight_Leg instance, Seat_Reservations are kept, as are Airplane used on each leg and the actual arrival and departure times and airports.

Airplane is identified by an airplane id, and is of a particular Airplane_Type. It has a fixed No. of seats.

Can_Lan relates Airplane_Type to the Airports at which they can land.

Airport is identified by airport code.

7.20

Users (User_name, User_id, Kind_of_user)

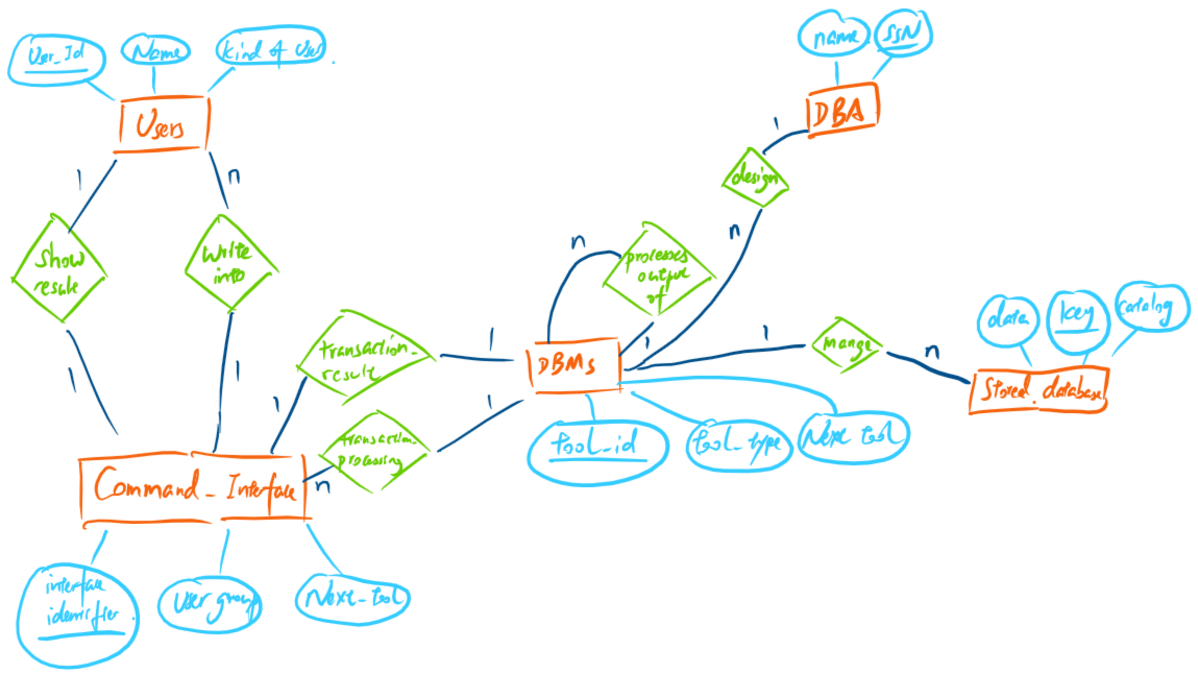
Command_interface_type (Interface_identifier, User_group, Next_tool)

DBMS (Tool_id, Tool_type, Next_tool)

DBA (Name, SSN)

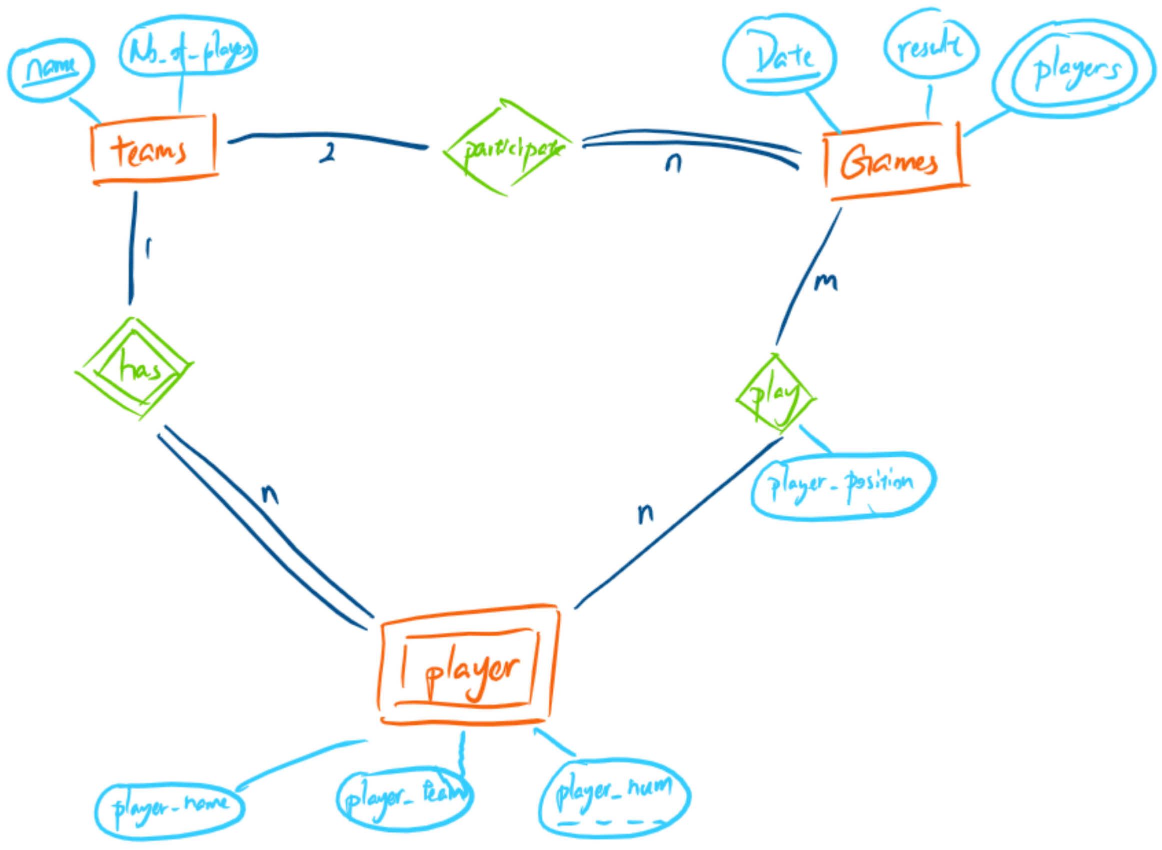
Stored_database (data, key, catalog)

All the constrains and relationships see the figure below.



7.22

ER diagram for soccer:



1. All players have to be part of a team.
2. Only two teams can participate in each game.
3. Each player in a team has unique number.
4. Each team must have a unique name.
5. On a date only one match takes place.
6. A player can play many games.

7.23

a. Loan, Customer, Account, Bank

b. Yes. Bank_Branch; partial key is Branch_no. and id relationship is Branches.

c. No two branches have the same number;

A bank can have any number of branches but a branch is of only one bank.

d.

Branches: Bank (min, max) = (1, 1) and Bank_Branch (min, max) = (1, n). A bank can have any number of branches but a branch can be owned by a single bank.

Accts: Account (min, max) = (1,n) and Bank_Branch (min, max) = (1, 1). An account can be with one branch but a branch can have many accounts.

Loans: Loan (min, max) = (1, n) and Bank_Branch (min, max) = (1, 1). A customer can have any number of loans but a loan is given from one branch only.

A_C: Account (min, max) = (1, n) and Customer (min, max) = (1, 1). A customer can have any number of accounts but an account is owned by only one customer.

L_C: Customer (min, max) = (1, 1) and Loan (min, max) = (1, n). A customer can take any number of loans but a loan is given to only one customer.

e. In a banking system;

Each bank has a unique code, name, and address.

A bank can have any number of Bank_Branch. Each Bank_Branch has number that is unique in branches of that bank.

Each Bank_Branch opens account and gives loans to customers.

Each account and loans is id by account number and has balance.

Each customer is id by SSN, name, address, phone number of customer are stored.

f.

Branches: Bank (min, max) = (1, 1) and Bank_Branch (min, max) = (1, n).

Accts: Account (min, max) = (1, n) and Bank_Branch (min, max) = (1, 1).

Loans: Loan (min, max) = (1, 1000) and Bank_Branch (min, max) = (1, 1).

A_C: Account (min, max) = (1, n) and Customer (min, max) = (1, 1).

L_C: Customer (min, max) = (1, 1) and Loan (min, max) = (1, 2).

7.27

Entity1	Cardinality Ratio	Entity2	Assumption
Student	1-1	Social_Security_Number	
Student	N-M	Teacher	
Classroom	2-5	Wall	
Country	1-1	President	A country can and must have only one president
Course	1-N	Textbook	There can be many text books for a course
Item (that can be found in an order)	N-M	Order	An item can be in several order
Student	N-1	Class	
Class	M-N	Instructor	

Instructor	N-1	Office	Many instructors can be in single office but an instructor will have only one office
Ebay_auction_item	1-N	Ebay-bid	Any number of people can bid for an item

7.28

a. True. Actor has full participation in Performs_In.

b. Maybe. The max cardinality on Actor Performs_In Movie is N, so this is neither required nor ruled out.

c. True. Relation has 2:N which says more than one actor done a lead role in many movies.

d. True. The max cardinality on Movie has Lead_Role is 2.

e. Maybe. This can't be false because the relationship Also_A_Director exists, but it can't be true because Director doesn't have total participation in it.

f. Maybe. We do not know whether there are common entities in the two entity type tables.

g. False. A producer can be an actor in some other movie.

h. Maybe. The max cardinality on Movie is Performed_In by actor is M. So, this is possible but we cannot tell from the ER diagram.

i. False. There is no relation between the producer and director.

j. Maybe. Most movies may have one director and one producer.

k. True. Some movies can have one director but several producers.

l. True. An actor can do a lead role in a moive, can direct a movie and can produce the movie.

m. Maybe. This is not specified in the ER diagram. But it is possible.