Chapter 4

DESIGN

CHAPTER OUTLINE

After studying this chapter, students will be able to understand the:

- Designing Databases
- Designing Forms and Reports
- Designing Interfaces and Dialogues

INTRODUCTION

Systems development is systematic process which includes phases such as planning, analysis, design, deployment, and maintenance. Here we discuss about the designing phase only. It is a process of planning a new business system or replacing an existing system by defining its components or modules to satisfy the specific requirements. Before planning, we need to understand the old system thoroughly and determine how computers can best be used in order to operate efficiently.

System design is the phase that bridges the gap between problem domain and the existing system in a manageable way. This phase focuses on the solution domain, i.e. "how to implement?". It is the phase where the SRS document is converted into a format that can be implemented and decides how the system will operate.

DATABASE DESIGN

Database design is the organization of data according to a database model. The designer determines what data must be stored and how the data elements interrelate. With this information, they can begin to fit the data to the database model. Database Design is a collection of processes that facilitate the designing, development, implementation and maintenance of enterprise data management systems. It helps produce database systems:

- That meets the requirements of the users
- Have high performance.

The main objectives of database designing are to produce logical and physical designs models of the proposed database system.

- The logical model concentrates on the data requirements and the data to be stored independent of physical considerations. It does not concern itself with how the data will be stored or where it will be stored physically.
- The physical data design model involves translating the logical design of the database onto physical media using hardware resources and software systems such as database management systems (DBMS).

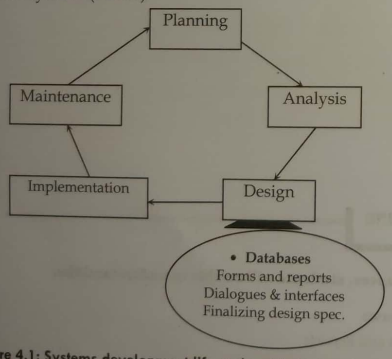


Figure 4.1: Systems development life cycle with design phase highlighted

THE PROCESS OF DATABASE DESIGN

Figure below shows that database modeling and design activities occur in all phases of the systems development process. Here, we discuss methods that help us finalize logical and physical database designs during the design phase. In logical database design, we use a process called normalization, which is a way to build a data model that has the properties of simplicity, non-redundancy, and minimal maintenance.

- Enterprise wide data model (ER with only attributes
- Conceptual data model (ER with only attributes for specific project)

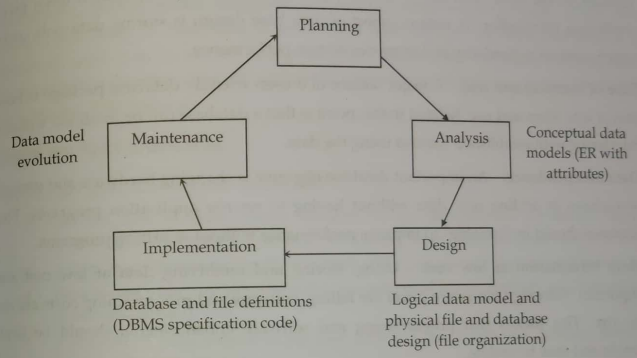


Figure 4.2: Relationship between data modeling and the SDLC

In most situations, many physical database design decisions are implicit or eliminated when we choose the data management technologies to use with the application. We concentrate on those decisions that will make most frequently and use Oracle to illustrate the range of physical database design parameters you must manage. There are four key steps in logical database modeling and design:

1. Develop a logical data model for each known user interface (form and report) for the application using normalization principles.
2. Combine normalized data requirements from all user interfaces into one con-solidated logical database model; this step is called view integration.
3. Translate the conceptual E-R data model for the application or enterprise, developed without explicit consideration of specific user interfaces, into normalized data requirements.
4. Compare the consolidated logical database design with the translated E-R model and produce, through view integration, one final logical database model for the application.

OBJECTIVES OF DATA BASE

The general theme behind a database is to handle information as an integrated whole. There is none of the artificiality that is normally embedded in separate file or applications. A database is a collection of interrelated data stored with minimum redundancy to serve many users quickly and efficiently. The general objective is to make information access easy, quick, inexpensive and flexible for the user. In data base design, several specific objectives are considered:

1. **Controlled redundancy: -** Redundant data occupies space and, therefore, is wasteful. If versions of the same data are in different phases of updating, the system often gives conflicting information. A unique aspect of data base design is storing data only once, which controls redundancy and improves system performance.
2. **Ease of learning and use: -** A major feature of a user- friendly database package is how easy it is to learn and use. Related to this point is that a database can be modified without interfering with established ways of using the data.
3. **Data independence: -** An important database objective is changing hardware and storage procedures or adding new data without having to rewrite application programs. The database should be "tunable" to improve performance without rewriting programs.
4. **More information at low cost: -** Using, storing and modifying data at low cost are important. Although hardware prices are falling, software and programming costs are on the rise. This means that programming and software enhancements should be kept simple and easy to update.
5. **Accuracy and integrity: -** The accuracy of a database ensures that data quality and content remain constant. Integrity controls detect data inaccuracies where they occur.
6. **Recovery from failure: -** With multi-user access to a database, the system must recover quickly after it is down with no loss of transactions. This objective also helps maintain data accuracy and integrity.
7. **Privacy and security: -** For data to remain private, security measures must be taken to prevent unauthorized access. Database security means that data are protected from various forms of destruction; users must be positively identified and their actions monitored.
8. **Performance: -** This objective emphasizes response time to inquiries suitable to the use of the data. How satisfactory the response time is depends on the nature of the user-data base dialogue. For example, inquiries regarding airline seat availability should be handled in a few seconds. On the other extreme, inquiries regarding the total sale of a product over the past two weeks may be handled satisfactorily in 50 seconds.

DATA MODELING

Data modeling is the process of creating a data model for the data to be stored in a database. This data model is a conceptual representation of

- Data objects
- The associations between different data objects
- The rules.

Data modeling helps in the visual representation of data and enforces business rules, regulatory compliances, and government policies on the data. Data Models ensure consistency in naming conventions, default values, semantics, and security while ensuring quality of the data. Data model emphasizes on what data is needed and how it should be organized instead of what operations need to be performed on the data. Data Model is like architect's building plan which helps to build a conceptual model and set the relationship between data items. The two types of Data Models techniques are

USES OF DATA MODEL

The primary goals of using data model are:

- Ensures that all data objects required by the database are accurately represented. Omission of data will lead to creation of faulty reports and produce incorrect results.
- A data model helps design the database at the conceptual, physical and logical levels.
- Data Model structure helps to define the relational tables, primary and foreign keys and stored procedures.
- It provides a clear picture of the base data and can be used by database developers to create a physical database.
- It is also helpful to identify missing and redundant data.
- Though the initial creation of data model is labor and time consuming, in the long run, it makes your IT infrastructure upgrade and maintenance cheaper and faster.

TYPES OF DATA MODELS

There are mainly three different types of data models:

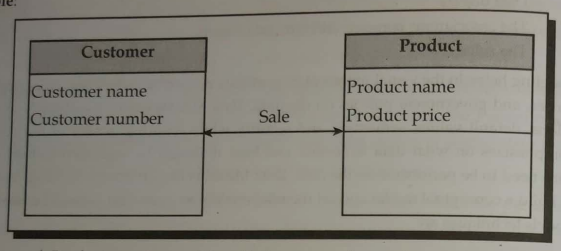
- Conceptual model
- Logical data model
- Physical data model

Conceptual data model

This data model defines **what** the system contains. This model is typically created by business stakeholders and Data Architects. The purpose is to organize scope and define business concepts and rules. The main aim of this model is to establish the entities, their attributes, and their relationships. In this Data modeling level, there is hardly any detail available of the actual database structure. The 3 basic tenants of data model are

- **Entity**: It is real-world thing.
- **Attribute**: It is characteristics or properties of an entity.
- **Relationship**: It is dependency or association between two entities.

Example:



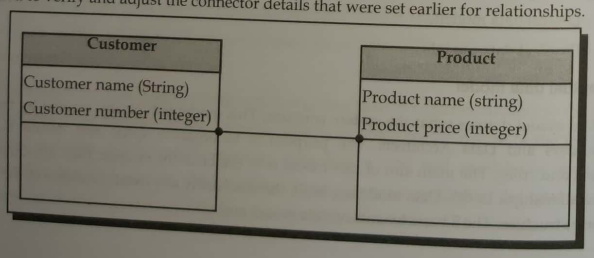
Customer and Product are two entities. Customer number and name are attributes of the Customer entity. Product name and price are attributes of product entity. Sale is the relationship between the customer and product.

Characteristics of a conceptual data model

- Offers organization-wide coverage of the business concepts.
- This type of Data Models are designed and developed for a business audience.
- The conceptual model is developed independently of hardware specifications like data storage capacity, location or software specifications like DBMS vendor and technology. The focus is to represent data as a user will see it in the "real world."

Logical data model

It defines **how** the system should be implemented regardless of the DBMS. This model is typically created by Data Architects and Business Analysts. The purpose is to developed technical map of rules and data structures. Logical data models add further information to the conceptual model elements. It defines the structure of the data elements and set the relationships between them. The advantage of the Logical data model is to provide a foundation to form the base for the Physical model. However, the modeling structure remains generic. At this Data Modeling level, no primary or secondary key is defined. At this Data modeling level, you need to verify and adjust the connector details that were set earlier for relationships.



Characteristics of a Logical data model

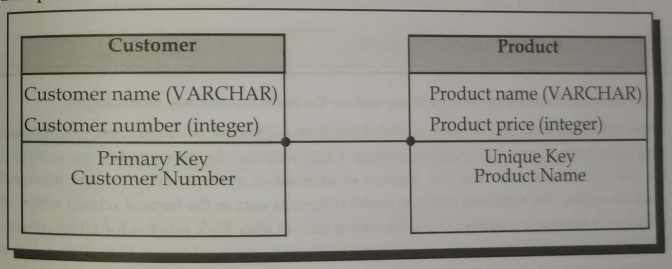
- Describes data needs for a single project but could integrate with other logical data models based on the scope of the project.
- Designed and developed independently from the DBMS.
- Data attributes will have data types with exact precisions and length.
- Normalization processes to the model is applied typically till 3NF.

Physical data model

This data model describes **how** the system will be implemented using a specific DBMS system. This model is typically created by DBA and developers. The purpose is actual implementation of the database. A Physical Data Model describes the database specific implementation of the data model. It offers an abstraction of the database and helps generate schema. This is because of the richness of meta-data offered by a Physical Data Model.

This type of Data model also helps to visualize database structure. It helps to model database columns keys, constraints, indexes, triggers, and other RDBMS features.

Example:



Characteristics of a physical data model

- The physical data model describes data need for a single project or application though it may be integrated with other physical data models based on project scope.
- Data Model contains relationships between tables that which addresses cardinality and null ability of the relationships.
- Developed for a specific version of a DBMS, location, data storage or technology to be used in the project.
- Columns should have exact data types, lengths assigned and default values.
- Primary and Foreign keys, views, indexes, access profiles, and authorizations, etc. are defined.

Advantages and Disadvantages of Data Model

Advantages of Data model

- The main goal of a designing data model is to make certain that data objects offered by the functional team are represented accurately.
- The data model should be detailed enough to be used for building the physical database.
- The information in the data model can be used for defining the relationship between tables, primary and foreign keys, and stored procedures.
- Data Model helps business to communicate the within and across organizations.
- Data model helps to documents data mappings in ETL process
- Help to recognize correct sources of data to populate the model

Disadvantages of Data model

- To develop Data model one should know physical data stored characteristics.
- This is a navigational system produces complex application development, management. Thus, it requires knowledge of the biographical truth.
- Even smaller changes made in structure require modification in the entire application.
- There is no set data manipulation language in DBMS.

RELATIONAL DATABASE MODEL

Many different database models are in use and are the bases for database technologies.

Although hierarchical and network models have been popular in the past, these are not used very often today for new information systems. Object-oriented database models are emerging but are still not common. The vast majority of information systems today use the relational database model. The relational database model represents data in the form of related tables, or relations. A relation is a named, two-dimensional table of data. Each relation (or table) consists of a set of named columns and an arbitrary number of unnamed rows. Each column in a relation corresponds to an attribute of that relation. Each row of a relation corresponds to a record that contains data values for an entity.

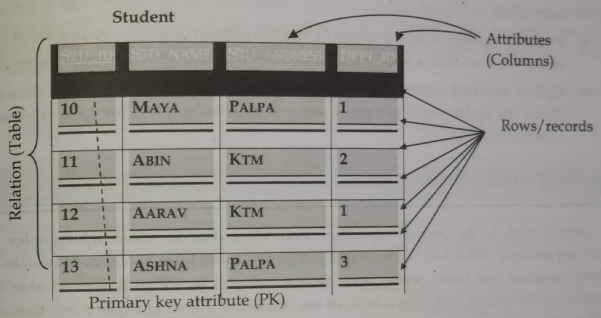
In relational model, the data and relationships are represented by collection of inter-related tables. Each table is a group of column and rows, where column represents attribute of an entity and rows represents records. Relational data model represents the logical view of how data is stored in the relational databases. There exist some concepts related to this, which includes the following terms.

- **Table:** In relational data model, data is stored in the tables. The table consists of a number of rows and columns. Thus, table is used because it can represent the data in the simplest form possible making data retrieval very easy.
- **Attribute:** Any relation have definite properties that are called as attributes.

- **Tuple:** Rows of table represents the tuple which contains the data records.
- **Domain:** Domain is a set of values which is indivisible i.e. value for each attribute present in the table contains some specific domain in which the value needs to lie. For

Example: The value of date of birth must be greater than zero. As, it cannot be negative. This is called domain of an attribute.

- **Relation:** A relation in relational data model represents the respective attributes and the correlation among them.



Relations have several properties that distinguish them from non-relational tables:

1. Entries in cells are simple. An entry at the intersection of each row and column has a single value.
2. Entries in a given column are from the same set of values.
3. Each row is unique. Uniqueness is guaranteed because the relation has a non-empty primary key value.
4. The sequence of columns can be interchanged without changing the meaning or use of the relation.
5. The rows may be interchanged or stored in any sequences.

NORMALIZATION

Normalization is the process of organizing data into a related table; it also eliminates redundancy and increases the integrity which improves performance of the query. To normalize a database, we divide the database into tables and establish relationships between the tables.

Database normalization can essentially be defined as the practice of optimizing table structures. Optimization is accomplished as a result of a thorough investigation of the various pieces of data that will be stored within the database, in particular concentrating upon how this data is interrelated.

Normalization Avoids

- **Duplication of Data -** The same data is listed in multiple lines of the database
- **Insert Anomaly -** A record about an entity cannot be inserted into the table without first inserting information about another entity - Cannot enter a customer without a sales order
- **Delete Anomaly -** A record cannot be deleted without deleting a record about a related entity. Cannot delete a sales order without deleting all of the customer's information.
- **Update Anomaly -** Cannot update information without changing information in many places. To update customer information, it must be updated for each sales order the customer has placed

DE-NORMALIZATION

De-normalization is the process of adding redundant data to speed up complex queries involving multiple tables JOINS. One might just go to a lower form of Normalization to achieve De-normalization and better performance. Data is included in one table from another in order to eliminate the second table which reduces the number of JOINS in a query and thus achieves performance.

Problems without Normalization

If a table is not properly normalized and has data redundancy then it will not only eat up extra memory space but will also make it difficult to handle and update the database, without facing data loss. Insertion, updating and Deletion Anomalies are very frequent if database is not normalized. To understand these anomalies let us take an example of a Student table.

Student

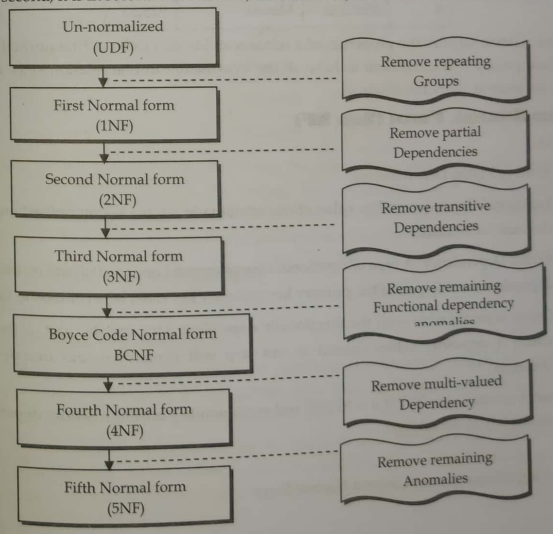
Sid	Sname	Branch	HOD	Office Phone
100	Abin	Pokhara branch	Mr. Nabin	4434564
101	Aarav	Pokhara branch	Mr. Nabin	4434564
102	Ashana	Pokhara branch	Mr. Nabin	4434564
103	Geeta	Pokhara branch	Mr. Nabin	4434564
104	Umesh	Banepa branch	Mr. Prabin	5212343
105	Ramesh	Banepa branch	Mr. Prabin	5212343

In the table above, we have data of six Computer Sci. students. As we can see, data for the fields branch, HOD and Office_Phone is repeated for the students who are in the same branch in the college, this is Data Redundancy.

- **Insertion Anomaly**
  Suppose for a new admission, until and unless a student opts for a branch, data of the student cannot be inserted, or else we will have to set the branch information as NULL. Also, if we have to insert data of 100 students of same branch, then the branch information will be repeated for all those 100 students. These scenarios are nothing but Insertion anomalies.
- **Update Anomaly**
  What if Mr. Prabin leaves the college? Or is no longer the HOD of computer science department? In that case all the student records will have to be updated, and if by mistake we miss any record, it will lead to data inconsistency. This is Updating anomaly.
- **Deletion Anomaly**
  In our Student table, two different information are kept together, Student information and Branch information. Hence, at the end of the academic year, if student records are deleted, we will also lose the branch information. This is Deletion anomaly.

TYPES OF NORMALIZATION

Normalization is a Six stage process - After the first stage, the data is said to be in first normal form, after the second, it is in second normal form, after the third, it is in third normal form and so on.



First Normal Form (1st NF)

In 1st NF:

- The table cells must be of single value.
- Eliminate repeating groups in individual tables.
- Create a separate table for each set of related data.
- Identify each set of related data with a primary key.

Definition: An entity is in the first normal form if it contains no repeating groups. In relational terms, a table is in the first normal form if it contains no repeating columns. Repeating columns make your data less flexible, waste disk space, and make it more difficult to search for data.

Example

Order	Customer	Contact Person	Total
1	Rishabh	Manish	134.23
2	Preeti	Rohan	521.24
3	Rishabh	Manish	1042.42
4	Rishabh	Manish	928.53

The above relation satisfies the properties of a relation and is said to be in first normal form (or 1NF). Conceptually it is convenient to have all the information in one relation since it is then likely to be easier to query the database.

Second Normal Form (2nd NF)

In 2nd NF:

- Remove Partial Dependencies.
- Functional Dependency: The value of one attribute in a table is determined entirely by the value of another.
- Partial Dependency: A type of functional dependency where an attribute is functionally dependent on only part of the primary key (primary key must be a composite key).
- Create separate table with the functionally dependent data and the part of the key on which it depends. Tables created at this step will usually contain descriptions of resources.

Definition: A relation is in 2NF if it is in 1NF and every non-key attribute is fully dependent on each candidate key of the relation.

Example

The following relation is not in Second Normal Form:

Order	Customer	Contact Person	Total
1	Rishabh	Manish	134.23
2	Preeti	Rohan	521.24
3	Rishabh	Manish	1042.42
4	Rishabh	Manish	928.53

In the table above, the order number serves as the primary key. Notice that the customer and total amount are dependent upon the order number -- this data is specific to each order. However, the contact person is dependent upon the customer. An alternative way to accomplish this would be to create two tables:

Customer	Contact Person
Rishabh	Manish
Preeti	Rohan

Order	Customer	Total
1	Rishabh	134.23
2	Preeti	521.24
3	Rishabh	1042.42
4	Rishabh	928.53

The creation of two separate tables eliminates the dependency problem. In the second table, contact person is dependent upon the primary key -- customer name. The first table only includes the information unique to each order. Someone interested in the contact person for each order could obtain this information by performing a Join Operation.

Third Normal Form (3rd NF)

In 3rd NF:

- Remove transitive dependencies.
- Transitive Dependency A type of functional dependency where an attribute is functionally dependent on an attribute other than the primary key. Thus its value is only indirectly determined by the primary key.
- Create a separate table containing the attribute and the fields that are functionally dependent on it. Tables created at this step will usually contain descriptions of either resources or agents. Keep a copy of the key attribute in the original file.

Definition: A relation is in third normal form, if it is in 2NF and every non-key attribute of the relation is non-transitively dependent on each candidate key of the relation.

Example

Company	City	State	ZIP
Sanima Ltd	Kathmandu	State 3	10169
Shivm Ltd.	Biratnagar	State 2	33196
API Company Ltd.	Darchula	State 7	21046

The above table is not in the 3NF.

In this example, the city and state are dependent upon the ZIP code. To place this table in 3NF, two separate tables would be created -- one containing the company name and ZIP code and the other containing city, state, ZIP code pairings.

Company	ZIP
Sanima Ltd	10169
Shivm Ltd.	33196
API Company Ltd.	21046

City	State	ZIP
Kathmandu	State 3	10169
Biratnagar	State 2	33196
Darchula	State 7	21046

This may seem overly complex for daily applications and indeed it may be. Database designers should always keep in mind the tradeoffs between higher level normal forms and the resource issues that complexity creates.

Boyce-Codd Normal Form (BCNF)

In BCNF:

- When a relation has more than one candidate key, anomalies may result even though the relation is in 3NF.
- 3NF does not deal satisfactorily with the case of a relation with overlapping candidate keys I.e. composite candidate keys with at least one attribute in common.
- BCNF is based on the concept of a determinant.
- A determinant is any attribute (simple or composite) on which some other attribute is fully functionally dependent.
- A relation is in BCNF is, and only if, every determinant is a candidate key.

Definition: A relation is in Boyce-Codd Normal Form (BCNF) if every determinant is a candidate key.

Example

Client Interview

ClientNo	InterviewDate	InterviewTime	StaffNo	RoomNo
CR76	13-may-11	10:30	SG5	G101
CR76	13-may-11	12:00	SG5	G101
CR74	13-may-11	12:00	SG37	G102
CR56	02-july-11	10:30	SG5	G102

- Functional dependency 1 (FD1): Client No, Interview Date→Interview Time, Staff No, Room No (Primary Key)
- FD2: Staff No, Interview Date, Interview Time → Client No (Candidate key)
- FD3: Room No, Interview Date, Interview Time → Client No, Staff No (Candidate key)
- FD4: Staff No, Interview Date → Room No (not a candidate key)

As a consequence, the Client Interview relation may suffer from update anomalies. To transform the Client Interview relation to BCNF, we must remove the violating functional dependency by creating two new relations called Interview and Staff Room as shown below,

Interview (Client No, Interview Date, Interview Time, Staff No)

Staff Room (Staff No, Interview Date, Room No)

Interview

Client No	Interview Date	Interview Time	Staff No
CR76	13-may-11	10:30	SG5
CR76	13-may-11	12:00	SG5
CR74	13-may-11	12:00	SG37
CR56	02-july-11	10:30	SG5

Staffroom

Staff No	Interview Date	Room No
SG5	13-may-11	G101
SG37	13-may-11	G102
SG5	02-july-11	G102

Fourth Normal Form (4th NF)

In 4th NF:

An entity is in Fourth Normal Form (4NF) when it meets the requirement of being in Third Normal Form (3NF) and additionally:

- Has no multiple sets of multi-valued dependencies. In other words, 4NF states that no entity can have more than a single one-to-many relationship within an entity if the one-to-many attributes are independent of each other.
- Fourth Normal Form applies to situations involving many-to-many relationships.

In relational databases, many-to-many relationships are expressed through cross-reference tables.

Definition: A table is in fourth normal form (4NF) if and only if it is in BCNF and contains no more than one multi-valued dependency.

Example

Take an example of Employee Table

Employee	Skills	Hobbies
1	Programming	Golf
1	Programming	Bowling
1	Analysis	Golf
1	Analysis	Bowling
2	Analysis	Golf
2	Analysis	Gardening
2	Management	Golf
2	Management	Gardening

This table is difficult to maintain since adding a new hobby requires multiple new rows corresponding to each skill. This problem is created by the pair of multi-valued dependencies Employee → Skills and Employee → Hobbies. A much better alternative would be to decompose INFO into two relations:

Employee	Skills
1	Programming
1	Analysis
2	Analysis
2	Management

Hobbies

Employee	Hobbies
1	Golf
1	Bowling
2	Golf
2	Gardening

Fifth Normal Form (5th NF)

In 5th NF:

- A relation that has a join dependency cannot be decomposed by a projection into other relations without spurious results
- A relation is in 5NF when its information content cannot be reconstructed from several smaller relations i.e. from relations having fewer attributes than the original relation

Definition: A table is in fifth normal form (5NF) or Project-Join Normal Form (PJNF) if it is in 4NF and it cannot have a lossless decomposition into any number of smaller tables.

Fifth normal form, also known as join-projection normal form (JPNF), states that no non-trivial join dependencies exist. 5NF states that any fact should be able to be reconstructed without any anomalous results in any case, regardless of the number of tables being joined. A 5NF table should have only candidate keys and its primary key should consist of only a single column.

Example

Take an example of a buying table. This is used to track buyers, what they buy, and from whom they buy. Take the following sample data:

Buyer	Vendor	Item
Kamala	Alphabeta House	Jeans
Abin	Alphabeta House	Jeans
Kamala	Radhika Sarees	Saree
Abin	Radhika Sarees	Saree
Kamala	Radhika Sarees	Suit

The problem with the above table structure is that if Claiborne starts to sell Jeans then how many records must you create to record this fact? The problem is there are pair wise cyclical dependencies in the primary key. That is, in order to determine the item you must know the buyer and the item, and buyer and vendor, and to determine the vendor you must know the buyer and the item. And the solution is to break finally to know the buyer you must know the vendor and the item. So following this one table into three tables; Buyer-Vendor, Buyer-Item, and Vendor-Item. So following tables are in the 5NF.

Buyer-Vendor

Buyer	Vendor
Kamala	Alphabeta House
Abin	Alphabeta House
Kamala	Radhika Sarees
Abin	Radhika Sarees

Buyer-Item

Buyer	Item
Kamala	Jeans
Abin	Jeans
Kamala	Saree
Abin	Saree
Kamala	Suit

Vendor- Item

Vendor	Item
Alphabeta House	Jeans
Radhika Sarees	Saree
Radhika Sarees	Suit

TRANSFORMING E-R DIAGRAMS INTO RELATIONS

Transforming an E-R diagram into normalized relations and then merging all the relations into one final, consolidated set of relations can be accomplished in four steps. These steps are summarized briefly here, and then steps 1, 2, and 4 are discussed in detail in the remainder of this chapter.

1. **Represent entities.** Each entity type in the E-R diagram becomes a relation. The identifier of the entity type becomes the primary key of the relation, and other attributes of the entity type become non-primary key attributes of the relation.
2. **Represent relationships.** Each relationship in an E-R diagram must be represented in the relational database design. How we represent a relationship depends on its nature. For example, in some cases we represent a relationship by making the primary key of one relation a foreign key of another relation. In other cases, we create a separate relation to represent a relationship.

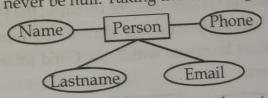
3. **Normalize the relations.** The relations created in steps 1 and 2 may have unnecessary redundancy. So we need to normalize these relations to make them well structured.
4. **Merge the relations.** So far in database design we have created various relations from both a bottom-up normalization of user views and from transforming one or more E-R diagrams into sets of relations. Across these different sets of relations, there may be redundant relations (two or more relations that describe the same entity type) that must be merged and renormalized to remove the redundancy.

TRANSFORMING ENTITIES AND RELATIONSHIPS E-R DIAGRAMS INTO RELATIONS

The ER Model is intended as a description of real-world entities. Although it is constructed in such a way as to allow easy translation to the relational schema model, this is not an entirely trivial process. The ER diagram represents the conceptual level of database design meanwhile the relational schema is the logical level for the database design. We will be following the simple rules:

Entities and Simple Attributes

An entity type within ER diagram is turned into a table. You may preferably keep the same name for the entity or give it a sensible name but avoid DBMS reserved words as well as avoid the use of special characters. Each attribute turns into a column (attribute) in the table. The key attribute of the entity is the primary key of the table which is usually underlined. It can be composite if required but can never be null. Taking the following simple ER diagram:

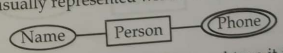


The initial relational schema is expressed in the following format writing the table names with the attributes list inside a parenthesis as shown below ;

Persons (personid , name, lastname, email)

Multi-Valued Attributes

A multi-valued attribute is usually represented with a double-line oval.



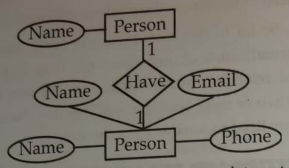
If you have a multi-valued attribute, take the attribute and turn it into a new entity or table of its own. Then make a 1: N relationship between the new entity and the existing one. In simple words;

- Create a table for the attribute.
- Add the primary (id) column of the parent entity as a foreign key within the new table as shown below:

Persons(personid , name, lastname, email)

Phones (phoneid , personid, phone)

1:1 Relationships



To keep it simple and even for better performances at data retrieval, I would personally recommend using attributes to represent such relationship. For instance, let us consider the case where the Person has or optionally has one wife. You can place the primary key of the wife within the table of the Persons which we call in this case foreign key as shown below.

Persons(personid , name, lastname, email , wifeid)

Wife (wifeid , name)

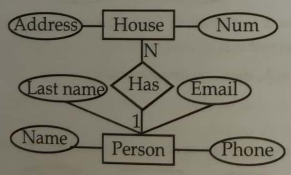
Or vice versa to put the personid as a foreign key within the Wife table as shown below:

Persons(personid , name, lastname, email)

Wife (wifeid , name , personid)

1: N Relationships

This is the tricky part! For simplicity, use attributes in the same way as 1:1 relationship but we have only one choice as opposed to two choices. For instance, the Person can have a House from zero to many, but a House can have only one Person. To represent such relationship the personid as the Parent node must be placed within the Child table as a foreign key but not the other way around as shown next:



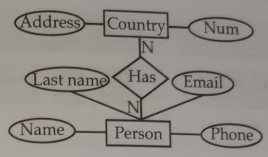
It should be converted to,

Persons(personid , name, lastname, email)

House (houseid , num , address, personid)

N: N Relationships

We normally use tables to express such type of relationship. This is the same for N – ary relationship of ER diagrams. For instance, The Person can live or work in many countries. Also, a country can have many people. To express this relationship within a relational schema we use a separate table as shown below:



It should convert into;

Persons (personid , name, lastname, email)

Countries (countryid , name, code)

HasRelat (hasrelatid , personid , countryid)

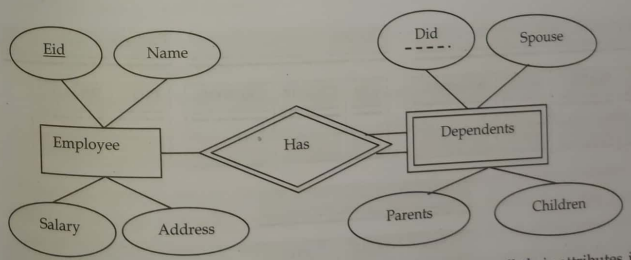
Mapping Weak entity sets to ER

A weak entity set does not have its own primary key and always participates in one-to-many relationship with owner entity set and has total participation. For a weak entity set create a relation that contains all simple attributes (or simple components of composite attributes). In addition, relation for weak entity set contains primary key of the owner entity set as foreign key and its primary key is formed by combining partial key (discriminator) and primary key of the owner entity set.

Mapping Process

- Create table for weak entity set.
- Add all its attributes to table as field.
- Add the primary key of identifying entity set.
- Declare all foreign key constraints

Example: Let's take a weak entity set Dependents as shown in ER diagram below;



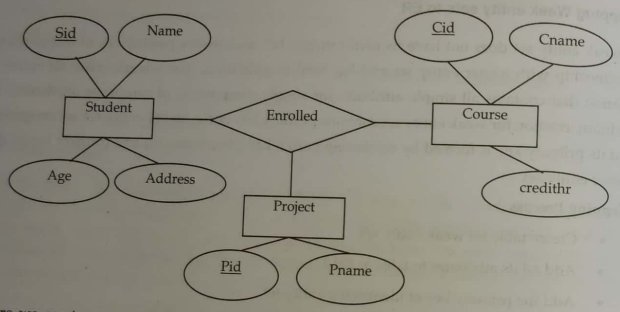
Here to draw table of weak entity set 'Dependents' we simply set all their attributes i.e. Did, spouse, Parents, children and also set primary key of Employee table to the Dependents table as below,

EMPLOYEE			
EID	NAME	SALARY	ADDRESS

DEPENDENTS				
DID	DID	PARENTS	SPOUSE	CHILDREN

Mapping of N-ary relationship types to ER

For each n-ary relationship set for n>2, a new relation is created. Primary keys of all participating entity sets are included in the relation as foreign key attributes. Besides this all simple attributes of the n-ary relationship set (or simple components of composite attributes) are included as attributes of the relation.



Here we create separate table for each entities and also create a single table for relationship Enrolled that contains the primary keys of each of the entities associated with this relationship.

STUDENT			
SID	NAME	AGE	ADDRESS

COURSE		
CID	CNAME	CRIDITHR

PROECT	
PID	PNAME

ENROLLED		
SID	CID	PID

MERGING RELATIONS

As part of the logical database design, normalized relations likely have been created from a number of separate E-R diagrams and various user interfaces. Some of the relations may be redundant—they may refer to the same entities. If so, you should merge those relations to remove the redundancy. This section describes merging relations, or view integration, which is the last step in logical database design and prior to physical file and database design.

An example of Merging Relations

Suppose that modeling a user interface or transforming an E-R diagram results in the following 3NF relation:

Employee1 (Emp_ID, Name, Address, Phone)

Modeling a second user interface might result in the following relation:

Employee2 (Emp_ID, Name, Address, Jobcode, Number_of_Years)

Because these two relations have the same primary key (Emp_ID) and describe the same entity, they should be merged into one relation. The result of merging the relations is the following relation:

Employee (Emp_ID, Name, Address, Phone, Jobcode, Number_of_Years)

Notice that an attribute that appears in both relations (such as Name in this example) appears only once in the merged relation.

PHYSICAL FILE AND DATABASE DESIGN

Designing physical files and databases requires certain information that should have been collected and produced during prior SDLC phases. This information includes the following:

- Normalized relations, including volume estimates
- Definitions of each attribute
- Descriptions of where and when data are used: entered, retrieved, deleted, and updated (including frequencies)
- Expectations or requirements for response time and data integrity
- Descriptions of the technologies used for implementing the files and database so that the range of required decisions and choices for each is known.

Normalized relations are, of course, the result of logical database design. Statistics on the number of rows in each table as well as the other information listed above may have been collected during requirements determination in systems analysis. If not, these items need to be discovered to proceed with database design. We take a bottom-up approach to reviewing physical file and database design. Thus, we begin the physical design phase by addressing the design of physical fields for each attribute in a logical data model.

Designing Fields

A field is the smallest unit of application data recognized by system software, such as a programming language or database management system. An attribute from a logical database model may be represented by several fields. For example, a student name attribute in a normalized student relation might be represented as three fields: last name, first name, and middle name. In general, we will represent each attribute from each normalized relation as one or more fields. The basic decisions we must make in specifying each field concern the type of data (or storage type) used to represent the field and data integrity controls for the field.

Calculated Fields: It is common for an attribute to be mathematically related to other data. For example, an invoice may include a total due field, which represents the sum of the amount due on each item on the invoice. A field that can be derived from other database fields is called a calculated field (or a computed field or a derived field). Recall that a functional dependency between attributes does not imply a calculated field. Some database technologies allow you to explicitly define calculated fields along with other raw data fields. If you specify a field as calculated, you would then usually be prompted to enter the formula for the calculation; the formula can involve other fields from the same record and possibly fields from records in related files. The database technology will either store the calculated value or compute it when requested.

Designing Physical Tables

A relational database is a set of related tables (tables are related by foreign keys referencing primary keys). In logical database design, you grouped into a relation those attributes that concern some unifying, normalized business concept, such as a customer, product, or employee. In contrast, a physical table is a named set of rows and columns that specifies the fields in each row of the table. A physical table may or may not correspond to one relation. Whereas normalized relations possess properties of well-structured relations, the design of a physical table has two goals different from those of normalization: efficient use of secondary storage and data processing speed.

- The efficient use of secondary storage (disk space) relates to how data are loaded on disks. Disks are physically divided into units (called pages) that can be read or written in one machine operation. Space is used efficiently when the physical length of a table row divides close to evenly into the length of the storage unit. For many information systems, this even division is very difficult to achieve because it depends on factors, such as operating system parameters, outside the control of each database.
- A second and often more important consideration when selecting a physical table design is efficient data processing. Data are most efficiently processed when they are stored close to one another in secondary memory, thus minimizing the number of input/output (I/O) operations that must be performed. Typically, the data in one physical table are stored close together on disk. De-normalization is the process of splitting or combining normalized relations into physical tables based on affinity of use of rows and fields.

Designing Forms and Reports: Introduction

The forms are used to present or collect information on a single item, such as a customer, product, or event. Forms can be used for both input and output. Reports, on the other hand, are used to convey information on a collection of items. Form and report design is a key ingredient for successful systems. Because users often equate the quality of a system with the quality of its input and output methods, you can see that the design process for forms and reports is an especially important activity. And because information can be collected and formatted in many ways, gaining an understanding of design dos and don'ts and the tradeoffs between various formatting options is useful for all systems analysts.

Both forms and reports are the product of input and output design and are business document consisting of specified data. The main difference is that forms provide fields for data input but reports are purely used for reading.

Designing Forms and Reports

Forms are used to present or collect information on a single item such as a customer, product, or event. Forms can be used for both input and output. Reports, on the other hand, are used to convey information on a collection of items. Forms and report design is a key ingredient for successful systems. As users often equate the quality of a system to the quality of its input and out methods, the design process of for forms and reports is an especially important activity.

Forms and reports are identified during requirements structuring. The kinds of forms and reports the system will handle are established as a part of the design strategy formed at the end of the analysis phase of the system development process. Forms and reports are integrally related to various diagrams developed during requirements structuring. For example, every input form will be associated with a data flow entering a process on a DFD, and every output form or report will be a data flow produced by a process on a DFD. Further, the data on all forms and reports must consists of data elements on data stores and on the E-R data model for the application, or must be computed from these data elements.

The Process of Designing Forms and Reports

Designing forms and reports is a user-focused activity that typically follows a prototyping approach. First, you must gain an understanding of the intended user and task objectives by collecting initial requirements during requirements determination. During this process, several questions must be answered. These questions attempt to answer the "who, what, when, where, and how" related to the creation of all forms and reports as given below.

1. Who will use the form or report?
2. What is the purpose of the form or report?
3. When is the form or report needed or used?
4. Where does the form or report need to be delivered and used?
5. How many people need to use or view the form or report?

Gaining an understanding of these questions is a required first step in the creation of any form or report. After collecting the initial requirements, you structure and refine this information into an initial prototype. Structuring and refining the requirements are completely independent of the users, although you may need to occasionally contact users in order to clarify some issue overlooked during analysis. Finally, you ask users to review and evaluate the prototype. After reviewing the prototype, users may accept the design or request that changes be made. If changes are needed, you will repeat the construction-evaluate-refinement cycle until the design is accepted. Usually, several iterations of this cycle occur during the design of a single form or report. As with any prototyping process, you should make sure that these iterations occur rapidly in order to gain the greatest benefits from this design approach.

The initial prototype may be constructed in numerous environments. The obvious choice is to use CASE tool or the standard development tools used within your organization. Often, initial prototypes are simply mock screens that are not working modules or systems. Mock screens can be produced from a word processor; computer graphics design package, or electronic spreadsheet.

Deliverables and Outcomes

Design specifications are the major deliverables and are inputs to the system implementation phase. Design specifications have three sections:

1. **Narrative overview:** This section contains general overview of the characteristics of the target users, tasks, system, and environmental factors in which the form or report will be used. The purpose is to explain to those who will actually develop the final form, why this form exists, and how it will be used so that they can make the appropriate decisions.
2. **Sample design:** This section provides a sample design of the form. This design may be hand drawn using a coding sheet although, in most instances, it is developed using CASE or standard development tool. Using actual development tools allows the design to be more thoroughly tested and assessed.
3. **Testing and usability assessment:** This section provides all testing and usability assessment information. Assessing usability depends on speed, accuracy, and satisfaction.

Formatting Forms and Reports

A wide variety of information can be provided to users of information systems and, as technology continues to evolve, a greater variety of data types will be used. There are numerous guidelines for formatting information.

General Formatting Guidelines

Over the past several years, industry and academic researchers have spent considerable effort investigating how information formatting influences individual task, performance, and perceptions of usability. Through this work, several guidelines for formatting information have emerged as given below:

1. **Meaningful titles:** Titles should be clear and specific describing content and use. They should include version information and current date.
2. **Meaningful information:** Forms should include only necessary information with no need to modification.
3. **Balance of layout:** Adequate spacing and margins should be used. All data and entry fields should be clearly labeled.
4. **Easy navigation system**: Clearly show how to move forward and backward. Clearly show where you are currently. Notify the user when on the last page of a multi-paged sequence.

Assessing Usability

There are many factors to consider when you design forms and reports. The objective for designing forms, reports, and all human–computer interactions is usability. Usability typically refers to the following three characteristics:

1. **Speed.** Can you complete a task efficiently?
2. **Accuracy.** Does the system provide what you expect?
3. **Satisfaction.** Do you like using the system?

In other words, usability means that your designs should assist, not hinder, user performance. Thus, usability refers to an overall evaluation of how a system performs in supporting a particular user for a particular task. In the remainder of this section, we describe numerous factors that influence usability and several techniques for assessing the usability of a design.

General Design Guidelines for Usability of Forms and Reports

Usability Factor	Guidelines for Achievement of Usability
Consistency	Consistent use of terminology, abbreviations, formatting, titles, and navigation within and across outputs. Consistent response time each time a function is performed.
Organization	Formatting should be designed with an understanding of the task being performed and the intended user. Text and data should be aligned and sorted for efficient navigation and entry. Entry of data should be avoided where possible (e.g., computing rather than entering totals).
Clarity	Outputs should be self-explanatory and not require users to remember information from prior outputs in order to complete tasks. Labels should be extensively used, and all scales and units of measure should be clearly indicated.
Format	Information format should be consistent between entry and display. Format should distinguish each piece of data and highlight, not bury important data. Special symbols, such as decimal places, dollar signs, and ± signs, should be used as appropriate.
Flexibility	Information should be viewed and retrieved in a manner most convenient to the user. For example, users should be given options for the sequence in which to enter or view data and for use of shortcut key strokes. and the system should remember where the user stopped during the last use of the system.

Designing Interfaces and Dialogues: Introduction

Interface design focuses on how information is provided to and captured from users; dialogue design focuses on the sequencing of interface displays. Dialogues are analogous to a conversation between two people. The grammatical rules followed by each person during a conversation are analogous to the interface. Thus, the design of interfaces and dialogues is the process of defining the manner in which humans and computers exchange information. A good human computer interface provides a uniform structure for finding, viewing, and invoking the different components of a system.

Measures of Usability

User-friendliness is a term often used, and misused, to describe system usability. Although the term is widely used, it is too vague from a design standpoint to provide adequate information because it means different things to different people. Consequently, most development groups use several methods for assessing usability, including the following considerations:

- **Learnability:** How difficult is it for a user to perform a task for the first time?
- **Efficiency:** How quickly can users perform tasks once they know how to perform them?
- **Error rate:** How many errors might a user encounter, and how easy it is to recover from those errors?
- **Memorability:** How easy is it to remember how to accomplish a task when revisiting the system after some period of time?
- **Satisfaction and aesthetics:** How enjoyable is the system's visual appeal and how enjoyable is the system to use?

In assessing usability, you can collect information by observation, interviews, keystroke capturing, and questionnaires. Time to learn simply reflects how long it takes the average system user to become proficient using the system. Equally important is the extent to which users remember how to use inputs and outputs over time.

Characteristics for consideration when designing forms and reports

Characteristic	Consideration for Form and Report Design
User	Issues related to experience, skills, motivation, education, and personality should be considered.
Task	Tasks differ in amount of information that must be obtained from or provided to the user. Task demands such as time pressure, cost of errors, and work duration (fatigue) will influence usability.
System	The platform on which the system is constructed will influence interaction styles and devices.
Environment	Social issues such as the users' status and role should be considered in addition to environmental concerns such as lighting, sound, task interruptions, temperature, and humidity. The creation of usable forms and reports may necessitate changes in the users' physical work facilities.

Designing Interfaces and Dialogues

Similar to designing forms and reports, the process of designing interfaces and dialogues is a user-focused activity. This means that you follow a prototyping methodology of iteratively collecting information, constructing a prototype, assessing usability, and making refinements. To design usable interfaces and dialogues, you must answer the same who, what, when, where, and how questions used to guide the design of forms and reports. Thus, this process parallels that of designing forms and reports.

Deliverables and outcomes

The deliverable and outcome from system interface and dialogue design is the creation of a design specification. This specification is also similar to the specification produced for form and report designs—with one exception:

1. Narrative overview
2. Sample design
3. Testing and usability assessment

For interface and dialogue designs, one additional subsection is included: a section outlining the dialogue sequence—the ways a user can move from one display to another.

Interaction Methods and Devices

The human–computer interface defines the ways in which users interact with an information system. All human–computer interfaces must have an interaction style and use some hardware devices for supporting this interaction.

Methods of Interacting

When designing the user interface, the most fundamental decision you make relates to the methods used to interact with the system. Given that there are numerous approaches for designing the interaction, we briefly provide a review of those most commonly used. Our review will examine the basics of five widely used styles: command language, menu, form, object, and natural language. We will also describe several devices for interacting, focusing primarily on their usability for various interaction activities.

Command Language Interaction

In command language interaction, the user enters explicit statements to invoke operations within a system. This type of interaction requires users to remember command syntax and semantics. For example, to rename a copy of a file called "file.doc" in the current directory as "newfile.doc" at the command prompt within Linux, you would type:

```
$ cp file.doc newfile.doc
```

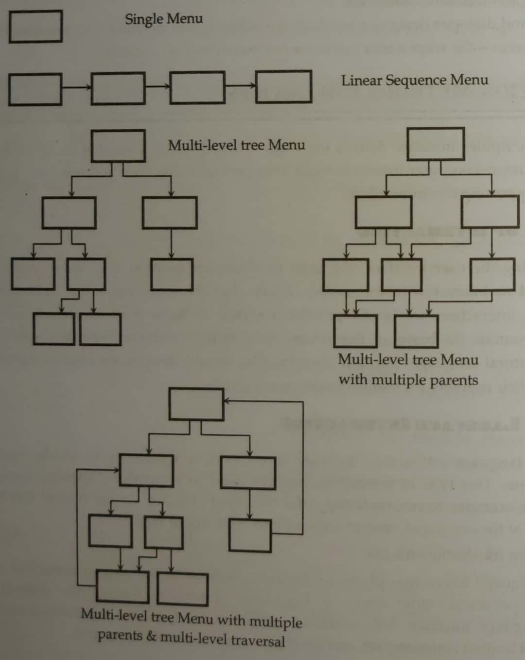
Command language interaction places a substantial burden on the user to remember names, syntax, and operations. Most newer or large-scale systems no longer rely entirely on a command language interface. Yet command languages are good for experienced users, for systems with a limited command set, and for rapid interaction with the system.

MENU INTERACTION

A menu is simply a list of options; when an option is selected by the user, a specific command is invoked or another menu is activated. Menus have become the most widely used interface method because the user only needs to understand simple signposts and route options to effectively navigate through a system.

Menus can differ significantly in their design and complexity. The variation of their design is most often related to the capabilities of the development environment, the skills of the developer, and the size and complexity of the system. For smaller and less complex systems with limited system options, you may use a single menu or a linear sequence of menus. A single menu has obvious advantages over a command language but may provide little guidance beyond invoking the command.

For large and more complex systems, you can use menu hierarchies to provide navigation between menus. These hierarchies can be simple tree structures or variations wherein children menus have multiple parent menus. Some of these hierarchies may allow multilevel traversal. Variations as to how menus are arranged can greatly influence the usability of a system.



Guidelines for Menu Design

Working	• Each menu should have a meaningful title • Command verbs should clearly and specifically describe operations • Menu items should be displayed in mixed uppercase and lowercase letters and have a clear, unambiguous interpretation.
Organization	• A consistent organizing principle should be used that relates to the tasks the intended users perform; for example, related options should be grouped together, and the same option should have the same wording and codes each time it appears.
Length	• The number of menu choices should not exceed the length of the screen • Submenus should be used to break up exceedingly long menus
Selection	• Selection and entry methods should be consistent and reflect the size of the application and sophistication of the users. • How the user is to select each option and the consequences of each option should be clear (e.g., whether another menu will appear)
Highlighting	• Highlighting should be minimized and used only to convey selected options (e.g., a check mark) or unavailable options (e.g., dimmed text)

FORM INTERACTION

The premise of form interaction is to allow users to fill in the blanks when working with a system. Form interaction is effective for both the input and presentation of information. An effectively designed form includes a self-explanatory title and field headings, has fields organized into logical groupings with distinctive boundaries, provides default values when practical, displays data in appropriate field lengths, and minimizes the need to scroll windows. Form interaction is the most commonly used method for data entry and retrieval in business-based systems. Using interactive forms, organizations can easily provide all types of information to web surfers.

OBJECT-BASED INTERACTION

The most common method for implementing object based interaction is through the use of icons. Icons are graphic symbols that look like the processing option they are meant to represent. Users select operations by pointing to the appropriate icon with some type of pointing device. The primary advantages to icons are that they take up little screen space and

can be quickly understood by most users. An icon may also look like a button that, when selected or depressed, causes the system to take an action relevant to that form, such as cancel, save, edit a record, or ask for help.

Natural Language Interaction

One branch of artificial intelligence research studies techniques for allowing systems to accept inputs and produce outputs in a conventional language such as English. This method of interaction is referred to as natural language interaction. Presently, natural language interaction is not as viable an interaction style as the other methods presented. Current implementations can be tedious, frustrating, and time consuming for the user and are often built to accept input in narrowly constrained domains (e.g., database queries). Natural language interaction is being applied within both keyboard and voice entry systems.

Designing Interfaces

User interface is the front-end application view to which user interacts in order to use the software. User can manipulate and control the software as well as hardware by means of user interface. Today, user interface is found at almost every place where digital technology exists, right from computers, mobile phones, cars, music players, airplanes, ships etc. User interface is part of software and is designed such a way that it is expected to provide the user insight of the software. UI provides fundamental platform for human-computer interaction. UI can be graphical, text-based, audio-video based, depending upon the underlying hardware and software combination. UI can be hardware or software or a combination of both. The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to understand
- Consistent on all interfacing screens

Designing Layouts

To ease user training and data recording, you should use standard formats for computer-based forms and reports similar to those used on paper-based forms and reports for recording or reporting information. This form has several general areas common to most forms:

- Header information
- Sequence and time-related information
- Instruction or formatting information
- Body or data details
- Totals or data summary
- Authorization or signatures
- Comments

In many organizations, data are often first recorded on paper-based forms and then later recorded within application systems. When designing layouts to record or display information on paper-based forms, you should try to make both as similar as possible. Additionally, data entry displays should be consistently formatted across applications to speed data entry and reduce errors. Another concern when designing the layout of computer-based forms is the design of between-field navigation. Because you can control the sequence for users to move between fields, standard screen navigation should flow from left to right and top to bottom just as when you work on paper-based forms.

When designing the navigation procedures within your system, flexibility and consistency are primary concerns. Users should be able to freely move forward and backward or to any desired data entry fields. Users should be able to navigate each form in the same way or in as similar a manner as possible. Additionally, data should not usually be permanently saved by the system until the user makes an explicit request to do so. This allows the user to abandon a data entry screen, back up, or move forward without adversely affecting the contents of the permanent data.

Designing Dialogues

The process of designing the overall sequences that users follow to interact with an information system is called dialogue design. A dialogue is the sequence in which information is displayed to and obtained from a user. As the designer, our role is to select the most appropriate interaction methods and devices (described earlier) and to define the conditions under which information is displayed to and obtained from users. The dialogue design process consists of three major steps:

- Designing the dialogue sequence
- Building a prototype
- Assessing usability

For a dialogue to have high usability, it must be consistent in form, function, and style. All other rules regarding dialogue design are mitigated by the consistency guideline. For example, the effectiveness of how well errors are handled or feedback is provided will be significantly influenced by consistency in design. If the system does not consistently handle errors, the user will often be at a loss as to why certain things happen.

Guidelines for the Design of Human–Computer Dialogues

Guideline	Explanation
Consistency	Dialogues should be consistent in sequence of actions, keystrokes, and terminology (e.g., the same labels should be used for the same operations on all screens, and the location of the same information should be the same on all displays).
Shortcuts and Sequence	Allow advanced users to take shortcuts using special keys (e.g., CTRL-C to copy highlighted text). A natural sequence of steps should be followed (e.g., enter first name before last name, if appropriate).

Feedback	Feedback should be provided for every user action (e.g., confirm that a record has been added, rather than simply putting another blank form on the screen).
Closure	Dialogues should be logically grouped and have a beginning, middle, and end (e.g., the last in the sequence of screens should indicate that there are no more screens).
Error Handling	All errors should be detected and reported; suggestions on how to proceed should be made (e.g., suggest why such errors occur and what user can do to correct the error). Synonyms for certain responses should be accepted (e.g., accept either "t," "T," or "TRUE").
Reversal	Dialogues should, when possible, allow the user to reverse actions (e.g., undo a deletion); data should not be deleted without confirmation (e.g., display all the data for a record the user has indicated is to be deleted).
Control	Dialogues should make the user (especially an experienced user) feel in control of the system (e.g., provide a consistent response time at a pace acceptable to the user).
Ease	It should be a simple process for users to enter information and navigate between screens (e.g., provide means to move forward, backward, and to specific screens, such as first and last).

DESIGNING THE DIALOGUE SEQUENCE

Your first step in dialogue design is to define the sequence. In other words, you must first gain an understanding of how users might interact with the system. This means that you must have a clear understanding of user, task, technological, and environmental characteristics when designing dialogues. Suppose that the marketing manager at Pine Valley Furniture (PVF) wants sales and marketing personnel to be able to review the year-to-date transaction activity for any PVF customer. After talking with the manager, you both agree that a typical dialogue between a user and the Customer Information System for obtaining this information might proceed as follows:

- Request to view individual customer information
- Specify the customer of interest
- Select the year-to-date transaction summary display
- Review customer information
- Leave system

As a designer, once you understand how a user wishes to use a system, you can then transform these activities into a formal dialogue specification.

DESIGNING INTERFACES AND DIALOGUES IN GRAPHICAL ENVIRONMENTS

Graphical user interface (GUI) environments have become the de facto standard for human-computer interaction. Although all of the interface and dialogue design guidelines presented previously apply to designing GUIs, additional issues that are unique to these environments must be considered. Here, we briefly discuss some of these issues.

Graphical interface Design issues

When designing GUIs for an operating environment such as Microsoft Windows or the Apple OSX, numerous factors must be considered. Some factors are common to all GUI environments, whereas others are specific to a single environment. We will not, however, discuss the subtleties and details of any single environment. Instead, our discussion will focus on a few general truths that experienced designers mention as critical to the design of usable GUIs. In most discussions of GUI programming, two rules repeatedly emerge as composing the first step to becoming an effective GUI designer:

- Become an expert user of the GUI environment.
- Understand the available resources and how they can be used.

The first step should be an obvious one. The greatest strength of designing within a standard operating environment is that standards for the behavior of most system operations have already been defined. For example, how you cut and paste, set up your default printer, design menus, or assign commands to functions have been standardized both within and across applications. This allows experienced users of one GUI-based application to easily learn a new application. Thus, in order to design effective interfaces in such environments, you must first understand how other applications have been designed so that you will adopt the established standards for "look and feel." Failure to adopt the standard conventions in a given environment will result in a system that will likely frustrate and confuse users.

MULTIPLE CHOICE QUESTIONS

1. Who are the people that actually use the system to perform or support the work to be completed?
   a. System analysts
   b. system designers
   c. System builders
   d. none of the above
2. Form of dependency in which set of attributes that is neither a subset of any of keys nor candidate key is classified as .....
   a. Transitive dependency
   b. full functional dependency
   c. Partial dependency
   d. prime functional dependency

EXERCISE

1. What column, row, and text formatting issues are important when designing tables and lists?
2. Describe how numeric, textual, and alphanumeric data should be formatted in a table or list.
3. What is meant by usability and what characteristics of an interface are used to assess a system's usability?
4. Discuss the benefits, problems, and general design process for the use of color when designing system output.
5. What type of labeling can you use in a table or list to improve its usability?
6. How can differences in user, task, system, or the environment influence the design of a form or report? Provide an example that contrasts characteristics for each difference.
7. Imagine the worst possible reports from a system. What is wrong with them? List as many problems as you can. What are the consequences of such reports? What could go wrong as a result? How does the prototyping process help guard against each problem?
8. Describe five methods of interacting with a system. Is one method better than all others? Why or why not?
9. List and describe the common interface and dialogue design errors found on websites.
10. Describe the properties of windows and forms in a GUI environment. Which property do you feel is most important? Why?
11. List four contributing factors that have acted to impede the design of high-quality interfaces and dialogues on Internet-based applications.
12. What is the purpose of normalization?
13. What is the purpose of de-normalization? Why might you not want to create one physical table or file for each relation in a logical data model?
14. What are the goals of designing physical tables?

15. What problems can arise when merging relations (view integration)?
16. What do you mean by relation? List out any five properties of relation.
17. Explain the purpose of data compression techniques.
18. Describe how numeric, textual, and alphanumeric data should be formatted in a table or list.
19. Provide some examples where variations in users, tasks, systems, and environmental characteristics might affect the design of system forms and reports.
20. Imagine the worst possible reports from a system. What is wrong with them? List as many problems as you can. What are the consequences of such reports? What could go wrong as a result? How does the prototyping process help guard against each problem?

❑❑❑