Data Base Management System

INTRODUCTION TO DBMS

WHAT IS DATABASE?

Database can be defined as an organized collection of interrelated data.

Data in the database:

(i) Is integrated (ii) Can be shared (iii) Can be concurrently accessed

WHAT IS DATABASE MANAGEMENT SYSTEM?

A Database Management System (DBMS) is a collection of interrelated files and a set of programs that allow users to access and modify these files. The primary goal of a DBMS is to provide a convenient and efficient way to store, retrieve and modify information.

The database systems are designed to:

(i) Define structures for the storage of data

(ii) Provide mechanisms for the manipulation of data

(iii) Ensure the security of the stored data even in case of system crash or attempts of unauthorized access

(iv) Share data among the different users.

File System Interface versus DBMS Interface

In the traditional file approach, data is stored in flat files which are maintained by the file system, beneath the operating system's control.

(i) **Flat files:** A flat file is a file containing records that has no structured interrelationship. Files used is structured programming (SP) Projects where essentially flat files.

(ii) In the **DBMS** approach, all requests to use the data stored in the database are managed by the DBMS. The end user can make use of either the application programs or the standard SQL to access the data.

SQL: SQL stands for structured query language. It is a language used by the relational database of fetch, update and manage data.

MASTER AND TRADITIONAL FILES

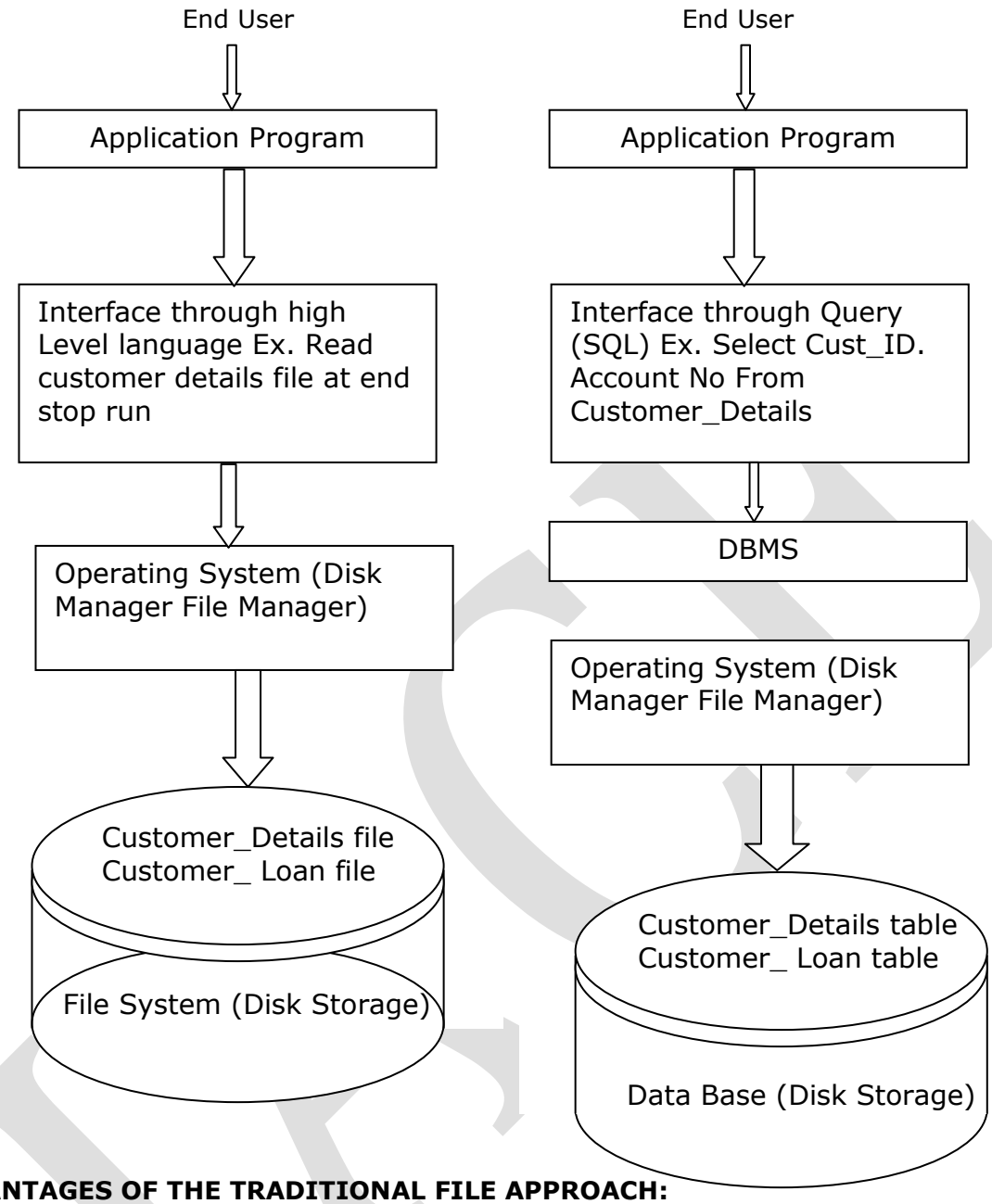
A Master file

(i) Stores relatively static data about an entity (ii) Changes rarely

A transaction file

(i) Stores relatively temporary data about a particular data processing task

(ii) Changes frequently as transaction happen periodically and in large numbers



DISADVANTAGES OF THE TRADITIONAL FILE APPROACH:

(i) **Data Security:** Data easily accessible by all and therefore not secure

(ii) **Data Redundancy:** Same data is duplicated in two or more files which may lead to update anomalies

(iii) **Data Isolation:** All the related data is not available in one file. Thus writing a new application program is difficult

(iv) **Program / Data Dependence:** Application programs are data-dependent. It is impossible to change the physical representation (how the data is physically represented in storage) or the access technique (how it is physically accessed) without affecting the application

(v) **Lack of Flexibility:** Only pre-determined requests for information can be met. It is not flexible enough to satisfy unanticipated queries.

(vi) **Concurrent Access Anomalies:** Same piece of data is allowed to be updated simultaneously which leads to inconsistencies

WHY DBMS

DBMS ensures the following:

(i) Data independence (ii) Allows for sharing of data among the different users

(iii) Allows concurrent access to the database (iv) Controls redundancy and inconsistency

(v) Provides a secure access to the database

(vi) Enforces integrity constraints by preventing the entry of invalid information into the database

(v) Enables backup and recovery from system crashes

Types of Database:

(i) **Centralized Database:** All data is located at a single site.

(ii) **Distributed Database:** The database is stored on several computers.

THREE LEVEL ARCHITECTURE FOR A DBMS:

(i) **External / View Level:** Enables users to view / access only a part of the database.

(ii) **Conceptual / Logical Level:** Describes what data is stored in the database and what relationships exist among those data.

(iii) **Internal / Physical Level:** Describes the data storage and access methods.

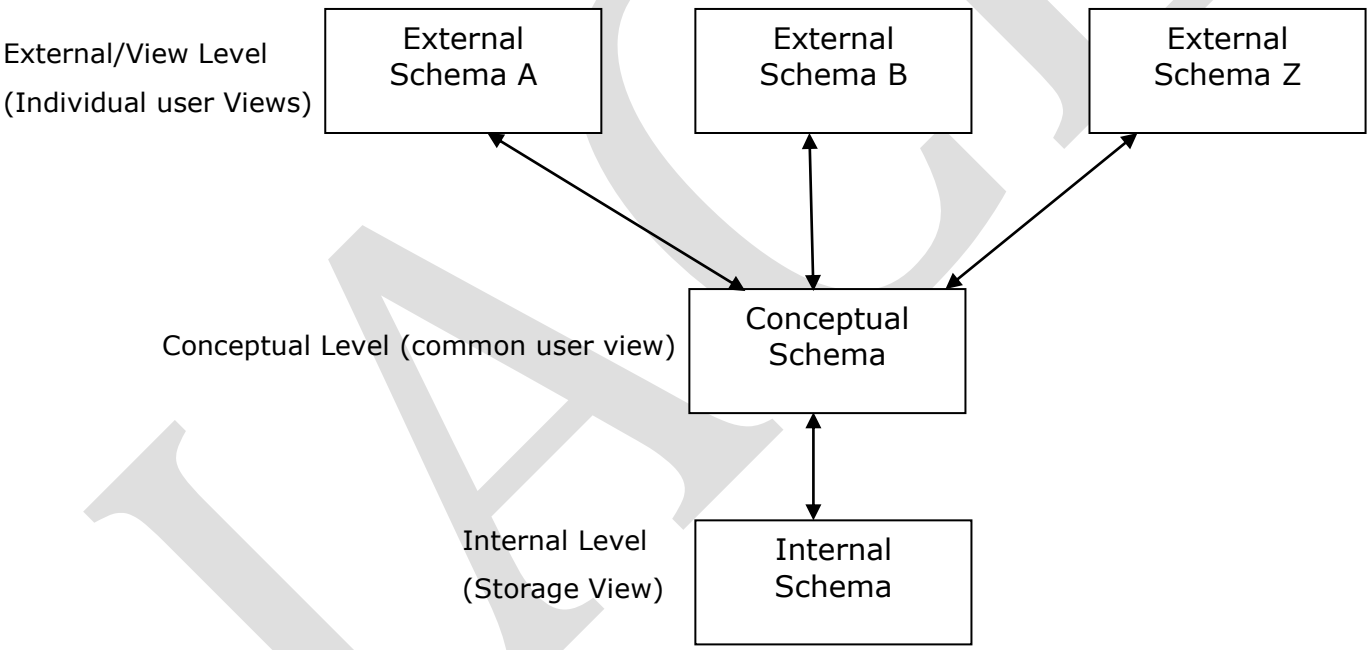


Fig: The three levels of the Architecture

DBMS USERS:

(i) **End User:** Works at the external level and generally makes updates to the database or executes queries on the database.

(ii) **Application Programmer:** Application programmer is responsible for writing database application programs in some programming languages such as COBOL, PL/1, C++, or some higher-level fourth generation language. The application programs access the database by issuing the appropriate request to the DBMS

(iii) **Database Administrator:** A database administrator (DBA) is responsible for the maintenance, performance, integrity and security of a database. Additional role requirements are likely to include planning, development and troubleshooting.

Some organizations have a hierarchical level of database administrators.

- Data Analysts/Query designers
- Junior DBAs
- Midlevel DBAs
- DBA consultants

Database administrator's activities can be listed as below

1. Transferring Data 2. Replicating Data
3. Maintaining database and ensuring its availability to users
4. Controlling privileges and permissions to database users 5. Monitoring database performance
6. Database backup and recovery 7. Database security

ENTITY:

Entity is a thing in the real world with an independent existence, and entity set is a collection of all entities of particular entity type in the database.

Example

A company have many employees, and these employees are defined as entities (e1, e2, e3……..) and all these entities having same attributes are defined under ENTITY TYPE employee, and set(e1, e2……) is called entity set.

Entity Type: Entity Type is a collection (set) of entities that have same attributes. There are two types of Entities.

1. **Strong Entity:** Objects are represented by their attributes and, as objects are inter distinguishable, a subset of these attributes forms a primary key for uniquely identified an instance of an entity. Entity types that have primary keys are called strong entities.

2. **Weak Entity:** An entity set may not have sufficient attributes to form a primary key, and its primary key compromises of its partial key and primary key of its parent entity, then it is said to be Weak Entity set.

Note: A weak entity set can always be making into strong entity set by adding to its attributes of its identifying entity set. For a weak entity set, we add columns to the table corresponding to the primary key of the strong entity set on which the weak set is dependent.

DATA MODELS:

It is a conceptual tools which can be used to describe data, data relationships, data semantics and consistency constraints.

Two widely used data models are:

(i) **Object based logical model:** Entity-Relationship Model (E-R Model) is a widely known object based logical model. These models are used to portray data at the conceptual and the view level.

The E-R Model is based on the inspection of the real world that consists of a collection of various basic objects also called entities, and of relationships among these objects or entities.

(ii) **Record based logical model:** They are used to depict data at the conceptual and the view level. They are used both to specify the general logical structure of the database and to supply a higher-level description of the implementation. There are three widely accepted record based logical models.

(1) **Hierarchical Data Model:** The hierarchical data model organizes data in a tree like structure. This hierarchy is also called parent-child hierarchy. This structure implies that a record can have repeating information (generally in the child data segments). **IBM's Information Management System** is a best example.

(2) **Network Data Model:** It permit's many to many relationship in data. Data in the network model is represented by a collection of records and the relationships among data are represented by links (pointers). The records in the database are organized as collections of graphs. Example: **IDMS.**

(3) **Relational Data Model:** Relational data model uses a collection of tables (relations) to represent data and the relationships among those data.

Example: Oracle, Sybase

A **table** has a specified number of columns but can have any number of rows. Rows stored in a table resemble records from flat files. A row in a table represents a relationship among a set of values.

Since a table is a collection of such relationships there is a close correspondence between the concept of table and the mathematical concept of relation, from which the relational data model takes its name.

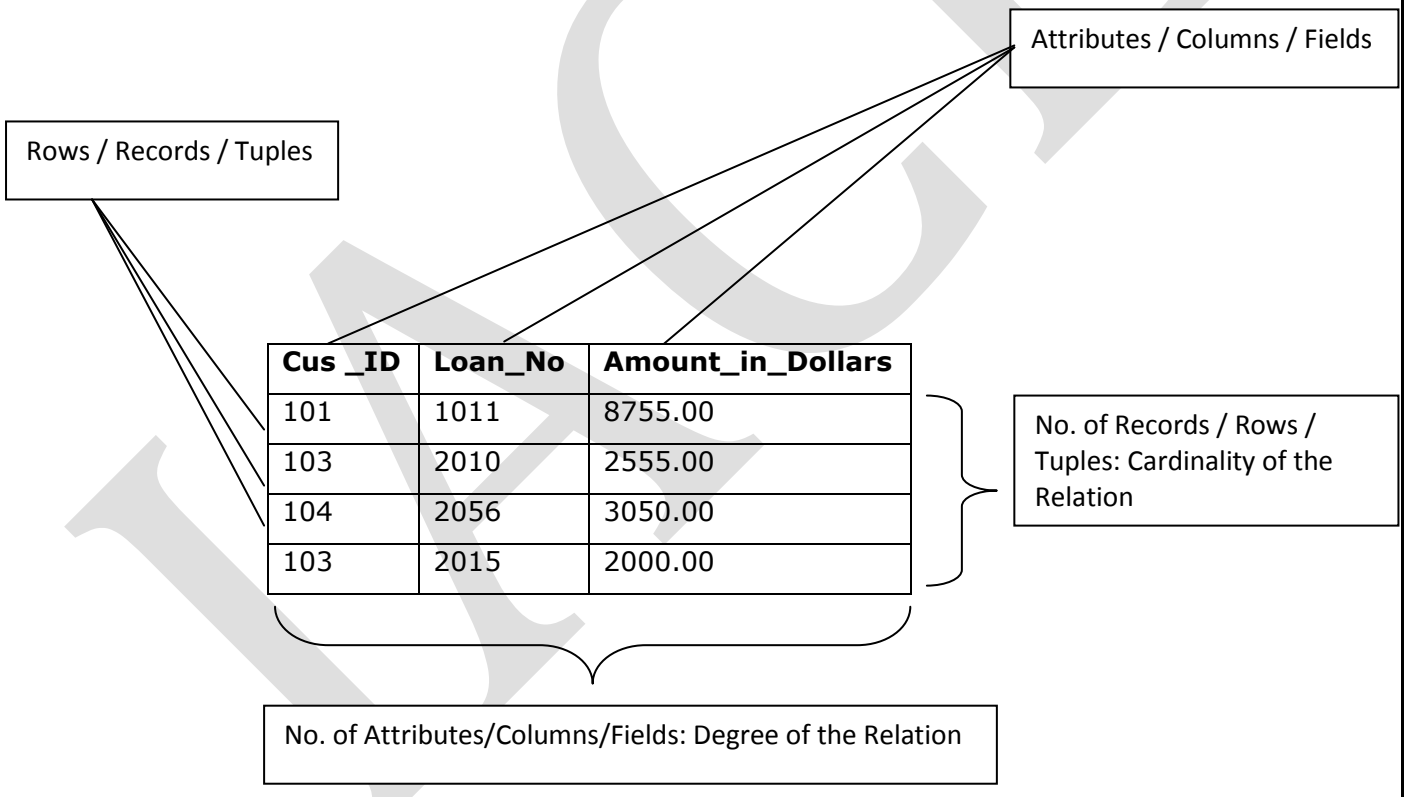


Figure: Relation Data Model

Attributes:

Attributes of a relation serve as names for the columns of the relation. Usually, an attribute describes the meaning of entries in the column below. Attributes is a characteristic of data. A real-world data feature modeled in the database, will be represented by an attribute. An attribute has to have a name has to be as relevant as possible for that feature. For example, for a person the attributes can be: Name, Sex, and Date of Birth, Informal terms used to define an attribute are column in a table or field in a data file.

Domains: A domain is a set of atomic values that are all of the same type. A value is the smallest unit of data in the relational model. For example, BMW, Mercedes, Audi, and VW are values for the attribute Producer. Those values are considered to be atomic, that is they are non-decomposable as far as the model is concerned. The domain for the Producer is the set of all possible car producer names. An attribute always has a domain associated with it. This domain gives the possible values for that attribute. Two or more attributes can be defined on the same domain.

Tuples: The rows of a relation, other than the header row containing the attribute names, are called tuples. A tuple has one component for each attribute of the relation. Tuple is an ordered set of values that describe data characteristics at one moment in time. Another formal term used to define a tuple is n-tuple. Informal terms used for tuples are: row in a table or record in a data file. A tuple usually represents an object and information about that object.

Structural Terminology	
Form Relational Term	Informal Equivalence
Relation	Table
Tuple	Row or Record
Cardinality of a Relation	Number of rows
Attribute	Column or Field
Degree of a Relation	Number of Columns
Primary Key	Unique Identifier
Domain	A pool of values from which the values of specific attributes of specific relations are taken

RDBMS

Relational Database: Any database in which the data is logically organized based on relational model.

RDBMS: It is a DBMS which manages the relational database. An RDBMS is a category of DBMS that stores data in related tables.

Some Popular RDBMS packages

RDBMS Package	**Company / Corporation**
Oracle	Oracle Corp.
Sybase	Sybase Inc.
Informix	Informix Software Inc.
MySQL	It is an Open source
DB2	IBM
Ingres II	Computer Associates International Inc.
SQL Server	Microsoft

RDBMS APPLICATION AREAS:

Database are widely used in real life applications such as:

1. **Sales:** For customer, product and purchase information in any industry.
2. **Airlines:** For reservations and Schedule information.
3. **Banking:** For customer information, accounts, loans
4. **Telecommunication:** For keeping track of calls made by customers, generating monthly bills of the customer and storing information about the communication networks.
5. **Universities:** For student information, course registrations and grades.

KEYS

1. **Candidate Key:** A candidate key of a table in defined as a set or more attributes of the table that can uniquely identify a row in a given table.

- A table can have more than one candidate keys.
- Candidate keys are identified during the design phase.
- One of the candidate key is chosen as primary key by the database designer while creating the table
- It is preferred to select a candidate key which is having a minimal number of attributes to function as a primary key.

2. **Simple Candidate Key:** A candidate key comprising of one attribute only
3. **Composite candidate Key:** A candidate key comprising of two or more attributes.
4. **Invalid Candidate Key:** A candidate key should be comprised of a set of attributes that can uniquely identify a row. A subset of the attributes should not possess the unique identification property.

PRIMARY KEY: At the time of the creation of table, the database designer choose one of the candidate key to uniquely identify rows in the table the candidate key so chosen in called the primary key.

- The primary key of a table is always not null and unique.
- The attributes which constitute the primary key cannot have duplicate values in the row of the table.
- It is mandatory to provide input for the primary key attributes.
- A null value is used to represent an unknown value. It is not a blank or a zero value.

How to select a primary Key:

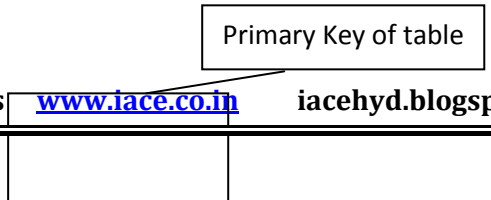
(i) Give preference to numeric column(s). The search algorithm performs better when the primary key is numeric.

(ii) Give preference to a single attribute. The search algorithm gives better output with a single attribute primary key than with a composite attribute primary key.

(iii) Give preference to the minimal composite key. A composite key is a collection of two or more attributes. Example: If the candidate keys are (x1, x2, x3) and (y1, y2), the composite key (y1, y2) is the minimal composite key and will therefore be chosen as the primary key

(iv) Primary keys are chosen according to business convenience.

Example: In below table account No. is chosen as Primary Key.

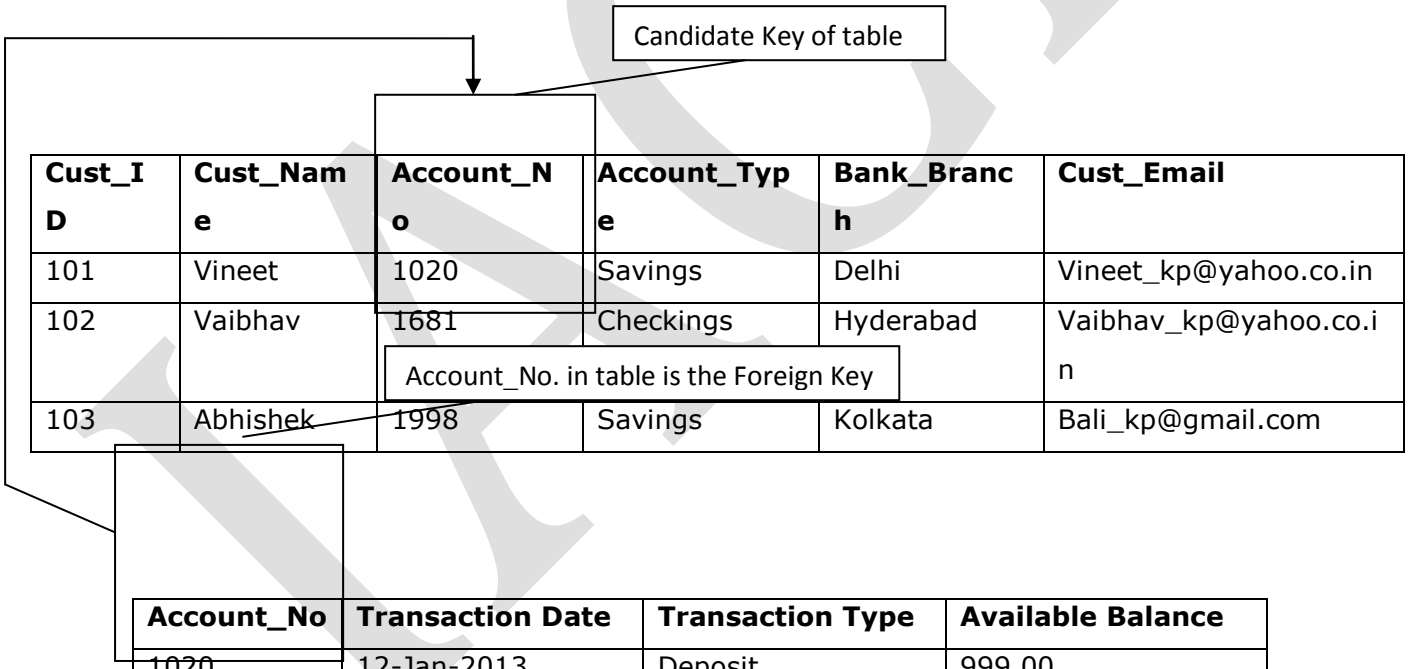


Cust_ID	Cust_Name	Account_No	Account_Type	Bank_Branch	Cust_Email
101	Vineet	1020	Savings	Delhi	Vineet_kp@yahoo.co.in
102	Vaibhav	1681	Checkings	Hyderabad	Vaibhav_kp@yahoo.co.in
103	Abhishek	1998	Savings	Kolkata	Bali_kp@gmail.com

Fig: Table with Account No. as Primary Key.

Foreign Key: A foreign key is defined as a set of attribute(s) in a table with a restriction that its value should be matched with the values of a candidate key in the same or another table. The foreign key attribute(s) can have duplicate or null values. As per the **referential integrity constraint,** the values of a foreign key attribute must match the values of the value of the corresponding candidate key. The relation that contains the foreign key is the referring relation (child table) and the relation that contains the corresponding candidate key is referenced relation (parent table).

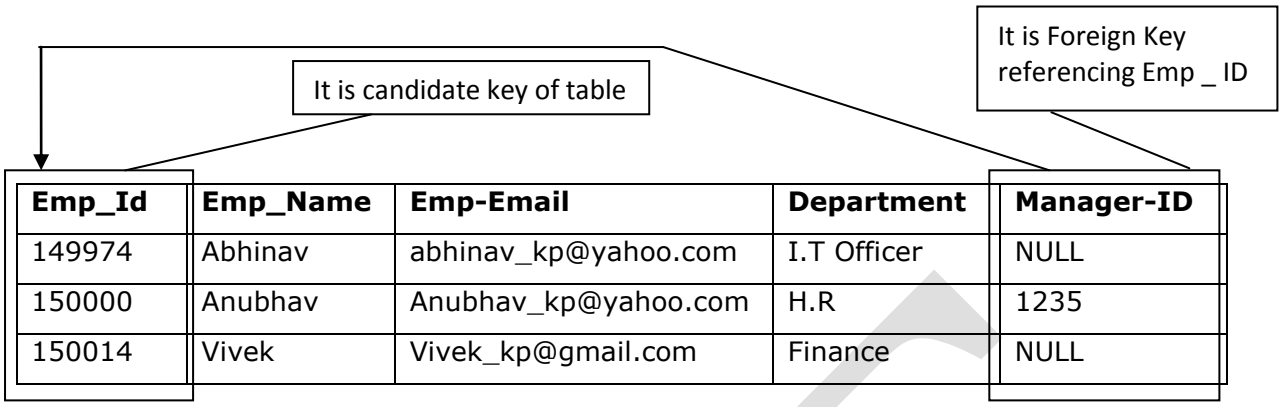
Self referencing: A table might include a foreign key, the values of which are required to match the value of a candidate key in the same table. This is known as self referencing.



Cust_ID	Cust_Name	Account_No	Account_Type	Bank_Branch	Cust_Email
101	Vineet	1020	Savings	Delhi	Vineet_kp@yahoo.co.in
102	Vaibhav	1681	Checkings	Hyderabad	Vaibhav_kp@yahoo.co.in
103	Abhishek	1998	Savings	Kolkata	Bali_kp@gmail.com

Account_No	Transaction Date	Transaction Type	Available Balance
1020	12-Jan-2013	Deposit	999.00
1681	14 Jan-2013	Deposit	1000.00
1998	19-Jan-2013	Withdrawal	No data

Fig: Demonstration of Referential Integrity



Emp_Id	Emp_Name	Emp-Email	Department	Manager-ID
149974	Abhinav	abhinav_kp@yahoo.com	I.T Officer	NULL
150000	Anubhav	Anubhav_kp@yahoo.com	H.R	1235
150014	Vivek	Vivek_kp@gmail.com	Finance	NULL

Fig. An example of self Referencing

- **Super Key:** Any superset of a candidate key is a super key. A super key may have unnecessary attributes.
- **Non-Key Attributes:** The attributes other than the candidate key attributes in a table / relation are called non-key attributes.

ENTITY RELATIONSHIP-E-R MODELLING

An E-R diagram is one of the many ways in which the business findings are pictorially represented.

Four types of cardinality of relationships are:

1. **One to one:** In this one instance of entity is related to another instance of the entity. Both participating entities have a one to one relationship.

Example: One country can have only one citizen as prime minister and one citizen can become prime minister of only one country.

2. **One to Many Relationship:** One instance of entity is related to multiple instance of another entity.

Example: One company can have many workers but one employee can work only for one company.

3. **Many to One Relationship:** This is the reverse of the one to many relationship.

Example: Many workers can work for only one section of department can have many workers. The relationship between workers and department is many to one.

4. **Many to Many Relationship**: In this many to many relationship multiple instance of one Entity are related to multiple instance of another entity.

Example: One student is enrolled for many subjects and one subject is enrolled by many students.

NORMALISATION:

- Normalization is a refinement process wherein it helps in removing anomalies in insert, up-date and delete operations.
- Normalization is also called "Bottom-up approach", because this technique requires full knowledge of every participating attribute and its dependencies on the key attributes, if you try to add new attributes after normalization is done, it may change the normal form of the database design itself.

First Normal Form (INF): A relation is considered to be in first normal form if all of its attributes have domains that are indivisible or atomic.

The idea of atomic values for attribute ensures that there are no 'repeating groups'. This is because a relational database management system is capable of storing a single value only at the intersection of a row and a column. Repeating Groups are when we attempt to store multiple values at the intersection of a row and a column and a table that will contain such a value is not strictly relational.

A table is in 1NF if and only if it satisfies the following five conditions:

- There is no top-to-bottom ordering to the rows.
- There is no left-to-right ordering to the columns.
- There are no duplicate rows.
- Every row-and-column intersection contains exactly one value from the applicable domain (and nothing else).
- All columns are regular [i.e. rows have no hidden components such as row IDs, object IDs, or hidden timestamps].

Second Normal Form (2NF): A relation is in second formal form when it is in 1NF and there is no such non-key attribute that depends on part of the candidate key, but on the entire candidate key.

It follows from the above definition that a relation that has a single attribute as its candidate key is always in 2NF.

Third Normal Form (3NF): A relation is in third normal form if it is in 2NF and there is no such non-key attribute that depends transitively on the candidate key. That is every attribute depends directly on the primary key and not through a transitive relation where an attribute Z may depend on a non-key attribute Y and Y in turn depends on the primary key X.

Transitivity, as seen earlier, means that when X'!Y and Y'!Z, then X'!Z.

It follows from 3NF relation that the non-key attributes are mutually independent.

The Fourth Normal Form (4NF): The fourth normal form can be understood in terms of multi-valued dependencies. The Fourth Normal Form (4NF) for a relational schema is said to exist when the non-related multi-valued dependencies that exist are not more than one in the given relation.

Normal Form	Test	Remedy (Normalization)
1NF	Attributes of every relation should be atomic. An attribute is atomic if domain of the attribute includes only atomic (simple, indivisible) values.	Form new relations for each non-atomic attribute.
2NF	For relations where candidate key contains multiple attributes (composite candidate key), non-key attribute should not be functionally dependent on apart of the candidate key.	Decompose to form a new relation for each partial key with its dependent attribute(s). Also retain the relation with the original candidate key and any attributes that are fully functionally dependent on it.

3NF	Relation should not have any non-key attribute functionally determined by any other non-key attribute. In other words there should be no transitive dependency of a non-key attribute on the candidate key through another non key attribute.	Decompose to form a relation that includes the non-key attribute(s) that functionally determine(s) other non-key attribute(s).

Few building blocks which are used to define normal forms.

(i) **Determinant:** Attribute X can be defined as determinant if it uniquely defines the attribute value Y in a given relationship or entity. To qualify as determinant attribute need NOT be a key attribute. Usually dependency of an attribute is represented as X → Y, which means attribute X decides attribute Y.

Example: In college selection relation, cut off attribute may decide the student attribute. This is represented as cut off → student and read as cut off decides student.

(ii) **Function Dependency:** In a given relation R, X and Y are attribute sets. Attribute set Y is functionally dependent on attribute set X if each value of X determines EXACTLY ONE value of Y. It is represented as:

X → Y.

Example: College report (student, course, course name, Room, Marks) where:

Student: A unique number associated with each student number.

Course: A unique number associated with each course called course number.

Course name: Name of the course

Room: Room number assigned to respective instructor.

Marks: Marks obtained

Student course together (called composite attribute) determines Exactly one value of Marks. Marks is functionally dependent on student course.

(iii) **Full Functional Dependency:** In a given relation R, X and Y are attributes. Y is fully functionally dependent on attribute X only if it is not functionally dependent on sub-set of a X. However X may be composite in nature.

In above example marks is fully functionally dependent on **Student Course** and not on subset of student course. Also course name is not fully functionally dependent on **Student course** because one of the subset is course.

(iii) **Partial Dependency:** In a given relation R, X and Y are attributes sets. Attribute set Y is partially dependent on the attribute set X only if it is dependent on subset of attribute set X.

In the above relationship Course Name, Room, are partially dependent on attributes Student Course because Course alone is enough to determine the Course Name, Room.

(iv) **Transitive Dependency:**

Example: Grade depends on marks, in turn marks depends on **Student Course** hence grade is fully tranisitive depends on **Student Course.**

(v) **Key attributes:** In a given relationship R, if the attribute X uniquely defines all other attributes, then the attribute X is a Key attribute which is nothing but the candidate key which is defined in earlier.

Example: Student Course together is a composite key attribute which attributes in relationship REPORT (Student, Course, Course Name, Room, Marks, Grade) uniquely. Hence Student and Course are key attributes.

(vi) **Non Key attributes:** The attributes other than the candidate key attributes in a table/relation are called Non-Key attributes. The attributes which do not participate in the candidate key.

Advantages of Normalization:

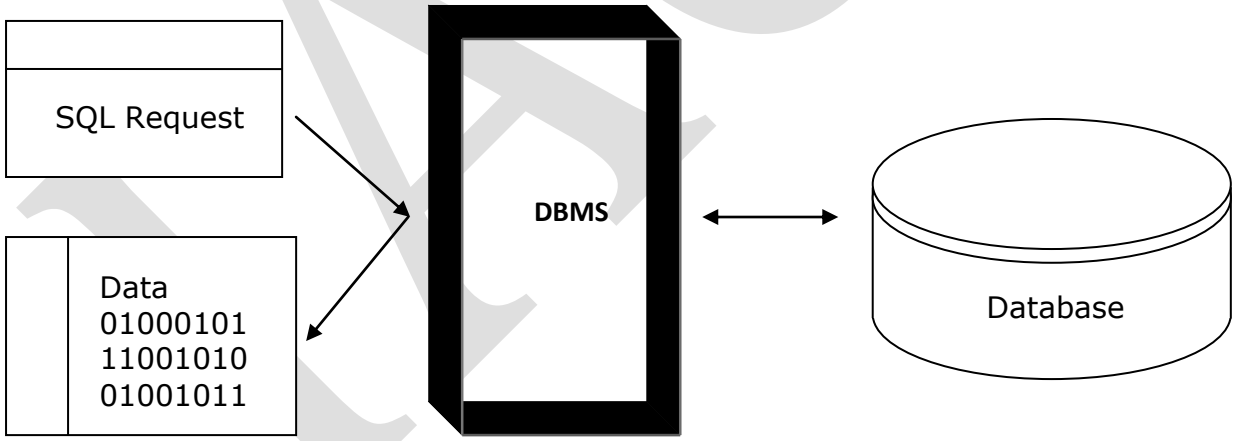
1. Normalization is based on mathematical foundation.
2. Removes the redundancy to the greater extent. After 3NF, data redundancy is reduced to the extent of foreign keys.
3. Removes the anomalies present in Inserts, Updates and Deletes.

Disadvantages of Normalization: Data retrieval (Select) operation performance will be severely affected.

STRUCTURE QUERY LANGUAGE (SQL)

Why SQL?

SQL is used to retrieve data from the database. The DBMS processes the SQL request, retrieves the requested data from the database and returns it. This process of requesting data from the database and receiving back the results is called a database query and hence the name Structured Query Language.



Computer System

SQL is used to control all the functions that a DBMS provides for its users, including:

(i) **Data Definition:** SQL allows a user to define the structure and the organization of the data to be stored and the relationships among the stored data items.

(ii) **Data Retrieval:** SQL allows a user or an application program to retrieve the stored data from the database.

(iii) **Data Manipulation:** SQL lets a user or an application program update the database by allowing to add new data, delete the existing data, and modify the existing data.

(iv) **Access Control:** SQL can be used to restrict a user's ability to retrieve, add, and modify data, thus protecting the stored data against unauthorized access.

DATA DEFINITION LANGUAGE (DDL): DDL statements help us in defining the table structure.

- Define and create a new table
- Remove a table that is no longer needed
- Change the definition of an existing table
- Define a virtual table (view) of data
- Build an index to access a table faster

Constraints: Data types help us to specify the nature or the kind of data that can be stored in a table.

Type of SQL Statement	**SQL Keywords**	**Function**
Data Definition Language (DDL)	CREATE ALTER DROP TRUNCATE	Used to define, change and drop the structure of a table Used to remove all rows from a table
Data Manipulation Language (DML)	SELECT INSERT INTOUPDATE DELETE FROM	Used to enter, modify, delete and retrieve data from a table
Data Control Language (DCL)	GRANT REVOKE COMMIT ROLLBACK	Used to provide control over the data in a database Used to define the end of a transaction

- **Types of Constraints:**

(i) **Column Constraint:** A constraint specified at the column level and is applied only to a specific column in addition to the column definition.

(ii) **Table Constraint:** A constraint specified at the table level after completion of all column definitions. This constraint is applied when we want to specify a constraint which involves more than one column in a table.

- **Create Table Statement:** The CREATE TABLE statement can:

(i) Create a table (ii) Define column constraints (iii) Define Table constraints

- **Alter Table Statement:**

(i) To add a new column definition to an existing table (ii) Drop a column from an existing table
(iii) Add or drop a primary key to / from an existing table
(iv) Add or drop a foreign key to / from an existing table
(v) Add or drop a unique constraint to / from an existing table
(vi) Add or drop a check constraint to / from an existing table

- **Drop Table Statement:**

(i) The DROP TABLE statements is used to drop or remove a table permanently from the database.
(ii) Both the schema / structure of the table and all of its contents are lost when DROP table command is used.
(iii) There is no way to recover data.

Truncate Table Statement: TRUNCATE TABLE statement is used to remove / delete all rows from a table.

When the TRUNCATE TABLE statement is used, all the contents of the specified table are lost but its definition remains intact. There is no way to recover the data. It releases the secondary memory occupied by the contents of the specified table.

Create Index Statement: An index is a structure which provides quick access to the rows of a table, based on the values of one or more columns. The index stores the data values and pointers (physical address information) to the rows where those data values occur. In the index, the data values are arranged either in descending or in ascending order, so that the RDBMS can quickly lookup the index to find a particular value. It then follows the pointer to locate the row containing the value.

Advantages of having an INDEX:

(i) Referring to indexed column(s) in search conditions speeds up the execution of SQL statements.
(ii) It is most appropriate when retrieval of data from tables is more frequent than inserts and updates

Disadvantages of having an INDEX:

(i) It consumes additional disk space
(ii) The INDEX must be updated whenever a row is added to the table or whenever updation of indexed column happens in an existing row. This imposes additional, overhead on INSERT and UPDATE statements for the table.

DATA MANIPULATION LANGUAGE (DML):

The DML statements are used to:
(i) Insert data into the table (ii) Delete data from the table
(iii) Retrieve / Fetch data from the table (iv) Modify update data in the table.

INSERT Statement:

(i) Single-row insert adds a single record (new row) of date to the table.
(ii) The number of columns mentioned in the column list and its data type must exactly match with the data values specified in the values clause or else an error will occur.

DELETE Statement:

(i) The DELETE statement can delete one or more rows from a table.

Difference between TRUNCATE and DELETE Statement:

(i) TRUNCATE is a Data Definition Language (DDL) statement whereas DELETE is a Data Manipulation Language (DML) statement.
(ii) TRUNCATE deletes all records from the table whereas DELETE can be used to selectively delete records from a table by using the WHERE clause

(iii) TRUNCATE releases the secondary storage occupied by the records of the table whereas DELETE does not do so.

(iv) Data removed using TRUNCATE cannot be recovered whereas data removed using DELETE can be recovered (a DCL statement called ROLLBACK can be used)

UPDATE Statement:

(i) One or more column values can be modified in the selected rows of a table using UPDATE statement.

SELECT Statement:

(i) It helps us in retrieving data from the database and returns the resultant set of records in the form of query results.

(ii) The asterisk (*) is a character that is used to denote all columns i.e. Select * will give all data of table.

1. **DISTINCT (Avoiding duplicates)**
   - Use of DISTINCT keyword in the column list before all columns help us in eliminating duplicate rows in the result set returned by the SELECT statement.

2. **WHERE CLAUSE (Row Selection):** The WHERE clause specifies a selection criteria or condition that limits the number of rows retrieved. It is a row wise operation. The WHERE clause can be used with any comparison operators such as =, >, <, >=, <=, <> or the logical operators (AND, OR, NOT).

VIEWS:

(i) A view is a VIRTUAL TABLE in the database defined by a query. A view does not exist in database as a stored set of data values. The rows and columns of data visible through view are produced by the query that defines the view.

DATA CONTROL LANGUAGE (DCL):

(i) DCL statements are used to control access to the database and the data in it. It is used to enforce data security.

(ii) **Granting Priviledges:** The GRANT statement is used to grant security privileges on database objects to specific users. Normally, the GRANT statement is used by the owner of the table or view to give other users access to the data.

(iii) **Revoking Priviledges:** The Revoke statement is used to Revoke priviledges previously granted with the GRANT statement.

******* IACE *******