**DATABASE MANAGEMENT SYSTEM**

1. **Introduction**

A database management system (DBMS) refers to the technology for creating and managing databases. DBMS is a software tool to organize (create, retrieve, update and manage) data in a database. The main aim of a DBMS is to supply a way to store up and retrieve database information that is both convenient and efficient.

1. **Characteristics of database**

The data in a database should have the following features:

1. **Organized/Related.** It should be well organized and related.
2. **Shared.** Data in a database are shared among different users and applications.
3. **Permanent or Persistence.** Data in a database exist permanently in the sense the data can live beyond the scope of the process that created it.
4. **Validity/integrity/Correctness.** Data should be correct with respect to the real world entity that they represent.
5. **Security.** Data should be protected from unauthorized access.
6. **Consistency.** Whenever more than one data element in a database represents related real world values, the values should be consistent with respect to the relationship.
7. **Non-redundancy:** No two data items in a database should represent the same real world entity.
8. **Independence.** Data at different levels should be independent of each other so that the changes in one level should not affect the other levels.
9. **Easily Accessible.** It should be available when and where it is needed i.e. it should be easily accessible.
10. **Recoverable.** It should be recoverable in case of damage.
11. **Flexible to change.** It should be flexible to change.
12. **Advantages**
	1. Reduced data redundancy
	2. Reduced updating errors and increased consistency
	3. Greater data integrity and independence from applications programs
	4. Improved data access to users through use of host and query languages
	5. Improved data security
	6. Reduced data entry, storage, and retrieval costs
	7. Facilitated development of new applications program
13. **Disadvantages**
	1. Database systems are complex, difficult, and time-consuming to design
	2. Substantial hardware and software start-up costs
	3. Damage to database affects virtually all applications programs
	4. Extensive conversion costs in moving form a file-based system to a database system
	5. Initial training required for all programmers and users
14. **Different Types of Database Users in DBMS**
	1. **A. Application Programmers**
	2. Application programmers are the one who writes application programs that uses the database. These application programs are written in programming languages like COBOL or PL (Programming Language 1)
	3. **B. End Users**
	4. End users are those who access the database from the terminal end. They use the developed applications and they don’t have any knowledge about the design and working of database. These are the second class of users
	5. **Casual User**
	6. These users have great knowledge of query language. Casual users access data by entering different queries from the terminal end. They do not write programs but they can interact with the system by writing queries.
	7. **Naive**
	8. Any user who does not have any knowledge about database can be in this category. There task is to just use the developed application and get the desired results. For example: Clerical staff in any bank is a naïve user. They don’t have any dbms knowledge but they still use the database and perform their given task.
	9. **DBA (Database Administrator)**
	10. DBA can be a single person or it can be a group of person. Database Administrator is responsible for everything that is related to database. He makes the policies, strategies and provides technical supports.
	11. **System Analyst**
	12. System analyst is responsible for the design, structure and properties of database. All the requirements of the end users are handled by system analyst. Feasibility, economic and technical aspects of DBMS is the main concern of system analyst.

**2. Database System Concepts and Architecture**

**2.1 Types of database models**

* Hierarchical **database** **model**.
* Relational **model**.
* Network **model**.
* Object-oriented **database** **model**.
* Entity-relationship **model**.
* Document **model**.
* Entity-attribute-value **model**.
* Star schema.

**2.2 Schema**

Schema is the overall view of database in dbms.in other words, schema is the plan design, scheme of database is known as schema. Schema is actually the frameset in which values of the data items are organized. At each level of architecture there is a separate schema.

1. **Types of schema:**

There are three types of schema that are explained individually below:

a. External schema

b. Conceptual schema, and

c. Internal schema

External schema: - External view of database is known as external schema. The external schema is linked with each end user of data.

Conceptual schema: - The conceptual view is defined by the conceptual schema which describes all the attributes, records, relationship etc.

Internal schema: - Internal view is defined by the internal schema which describes how data will be stored physically in the database. It also described how data will be accessed using the resources that are provided by dbms.

1. **Following are the three levels of database architecture**

1. Physical Level
2. Conceptual Level
3. External Level

**1. Physical Level**

* Physical level describes the physical storage structure of data in database.
* It is also known as Internal Level.
* This level is very close to physical storage of data.
* At lowest level, it is stored in the form of bits with the physical addresses on the secondary storage device.
* At highest level, it can be viewed in the form of files.
* The internal schema defines the various stored data types. It uses a physical data model.

**2. Conceptual Level**

* Conceptual level describes the structure of the whole database for a group of users.
* It is also called as the data model.
* Conceptual schema is a representation of the entire content of the database.
* These schema contains all the information to build relevant external records.
* It hides the internal details of physical storage.

**3. External Level**

* External level is related to the data which is viewed by individual end users.
* This level includes a no. of user views or external schemas.
* This level is closest to the user.
* External view describes the segment of the database that is required for a particular user group and hides the rest of the database from that user group.
1. **Mappings**

In computing and data management, data mapping is the process of creating data element mappings between two distinct data models. Data mapping is used as a first step for a wide variety of data integration tasks, including: Data transformation or data mediation between a data source and a destination.

1. **Data Independence**

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

****

Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.

1. **Logical Data Independence**

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

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

1. **Physical Data Independence**

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

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

1. **Database Languages and Interfaces**

Because a database supports a number of user groups, as mentioned previously, the DBMS must have languages and interfaces that support each user group.

1. **DBMS Languages**
* DDL – the data definition language, used by the DBA and database designers to define the conceptual and internal schemas.
* The DBMS has a DDL compiler to process DDL statements in order to identify the schema constructs, and to store the description in the catalogue.
* In databases where there is a separation between the conceptual and internal schemas, DDL is used to specify the conceptual schema, and SDL, storage definition language, is used to specify the internal schema.
* For a true three-schema architecture, VDL, view definition language, is used to specify the user views and their mappings to the conceptual schema. But in most DBMSs, the DDL is used to specify both the conceptual schema and the external schemas.
* Once the schemas are compiled, and the database is populated with data, users need to manipulate the database. Manipulations include retrieval, insertion, deletion and modification.
* The DBMS provides operations using the DML, data manipulation language.
* In most DBMSs, the VDL, DML and the DML are not considered separate languages, but a comprehensive integrated language for conceptual schema definition, view definition and data manipulation. Storage definition is kept separate to fine-tune the performance, usually done by the DBA staff.
* An example of a comprehensive language: SQL, which represents a VDL, DDL, DML as well as statements for constraint specification, etc.
1. **Data Manipulation Languages (DMLs)**

Two main types:

High-level/Non procedural

* Can be used on its own to specify complex database operations.
* DMBSs allow DML statements to be entered interactively from a terminal, or to be embedded in a programming language. If the commands are embedded in a general purpose programming language, the statements must be identified so they can be extracted by a pre-compiler and processed by the DBMS.

Low Level/Procedural

* Must be embedded in a general purpose programming language.
* Typically retrieves individual records or objects from the database and processes each separately.
* Therefore it needs to use programming language constructs such as loops.
* Low-level DMLs are also called record at a time DMLS because of this.
* High-level DMLs, such as SQL can specify and retrieve many records in a single DML statement, and are called set at a time or set oriented DMLs.
* High-level languages are often called declarative, because the DML often specifies what to retrieve, rather than how to retrieve it.

DML Commands

* When DML commands are embedded in a general purpose programming language, the programming language is called the host language and the DML is called the data sub-language.
* High-level languages used in a standalone, interactive manner is called a query language.
* Casual end users use high-level query language to specify requests, where programmers usually use embedded DML.
* Parametric end users usually interact with user-friendly interfaces, which can also be used by casual users who don’t want to learn the high-level languages.
1. **DBMS Interfaces**

Types of interfaces provided by the DBMS include:

Menu-Based Interfaces for Web Clients or Browsing

* Present users with list of options (menus)
* Lead user through formulation of request
* Query is composed of selection options from menu displayed by system.

Forms-Based Interfaces

* Displays a form to each user.
* User can fill out form to insert new data or fill out only certain entries.
* Designed and programmed for naïve users as interfaces to canned transactions.

Graphical User Interfaces

* Displays a schema to the user in diagram form. The user can specify a query by manipulating the diagram. GUIs use both forms and menus.

Natural Language Interfaces

* Accept requests in written English, or other languages and attempt to understand them.
* Interface has its own schema, and a dictionary of important words. Uses the schema and dictionary to interpret a natural language request.

Interfaces for Parametric Users

* Parametric users have small set of operations they perform.
* Analysts and programmers design and implement a special interface for each class of naïve users.
* Often a small set of commands included to minimize the number of keystrokes required. (I.e. function keys)

Interfaces for the DBA

* Systems contain privileged commands only for DBA staff.
* Include commands for creating accounts, setting parameters, authorizing accounts, changing the schema, reorganizing the storage structures etc.
1. DBMS tasks:
2. Managing large quantity of structured data
3. Efficient retrieval and modification: query processing and optimization
4. Sharing data: multiple users use and manipulate data
5. Controlling the access to data: maintaining the data integrity

 An example of a database (relational):

1. Relations (tables)
2. Attributes (columns)
3. Tuples (rows)
4. Example query: Salesperson='Mary' AND Price>100.

Database schema (e.g. relational):

* 1. Names and types of attributes
	2. Addresses
	3. Indexing
	4. Statistics
	5. Authorization rules to access data etc.

Data independence: separation of the physical and logical data

* + 1. Particularly important for distributed systems
		2. The mapping between them is provided by the schema

Architecture of a DBMS - three levels: external, conceptual and internal schema

1. **Types of DBMS**
	* 1. The data structures supported: tables (relational), trees, networks, objects
		2. Type of service provided: high level query language, programming primitives

 Levels



1. **Basic DBMS types**
* Linear files
	+ - Sequence of records with a fixed format usually stored on a single file
		- Limitation: single file
		- Example query: Salesperson='Mary' AND Price>100
	+ Hierarchical structure
		- Trees of records: one-to-many relationships
		- Limitations:
		- Requires duplicating records (e.g. many-to-many relationship)
		- Problems when updated
		- Retrieval requires knowing the structure (limited data independence): traversing the tree from top to bottom using a procedural language
* Network structure: similar to the hierarchical database with the implementation
	+ - of many-to-many relationships
	+ Relational structure
	+ Object-Oriented structure
		- Objects (collection of data items and procedures) and interactions between them.
		- Is this really a new paradigm, or a special case of network structure?
		- Separate implementation vs. implementation on top of a RDBMS

**3. Data Modeling using E.R. Model (Entity Relationship Model)**

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

**3.1 Entity**

An entity can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

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

**3.2 Attributes**

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

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

 **Types of Attributes**

* **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
* **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first\_name and last\_name.
* **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average\_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data\_of\_birth.
* **Single-value attribute** − Single-value attributes contain single value. For example − Social\_Security\_Number.
* **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email\_address, etc.

These attribute types can come together in a way like −

* simple single-valued attributes
* simple multi-valued attributes
* composite single-valued attributes
* composite multi-valued attributes

**3.3 Entity-Set and Keys**

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll\_number of a student makes him/her identifiable among students.

* **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
* **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
* **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

**3.4 Relationship**

The association among entities is called a relationship. For example, an employee **works\_at** a department, a student **enrolls** in a course. Here, Works\_at and Enrolls are called relationships.

**Relationship Set**

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

**Degree of Relationship**

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

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

**Mapping Cardinalities**

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

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



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



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



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



**4. Relational Model**

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

**Concepts**

**Tables** − In relational data model, relations are saved in the format of Tables. This format stores the relation among entities. A table has rows and columns, where rows represents records and columns represent the attributes.

**Tuple** − A single row of a table, which contains a single record for that relation is called a tuple.

**Relation instance** − A finite set of tuples in the relational database system represents relation instance. Relation instances do not have duplicate tuples.

**Relation schema** − A relation schema describes the relation name (table name), attributes, and their names.

**Relation key** − Each row has one or more attributes, known as relation key, which can identify the row in the relation (table) uniquely.

**Attribute domain** − Every attribute has some pre-defined value scope, known as attribute domain.

**4.1 Constraints**

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called **Relational Integrity Constraints**. There are three main integrity constraints −

* Key constraints
* Domain constraints
* Referential integrity constraints

**Key Constraints**

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called **key** for that relation. If there are more than one such minimal subsets, these are called ***candidate keys***.

Key constraints force that −

* in a relation with a key attribute, no two tuples can have identical values for key attributes.
* a key attribute can not have NULL values.

Key constraints are also referred to as Entity Constraints.

**Domain Constraints**

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

**4.2 Referential integrity Constraints**

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

**4.3 Foreign key in DBMS**

Foreign keys are the columns of a table that points to the [primary key](https://beginnersbook.com/2015/04/primary-key-in-dbms/) of another table. They act as a cross-reference between tables.

**4.4 Relational structure**

Relations, attributes, tuples

 Primary key (unique combination of attributes for each tuple)

 Foreign keys: relationships between tuples (many-to-many).

 Example: SUPPLIES defines relations between ITEM and SUPPLIER tuples.

 Advantages: many-to-many relationships, high level declarative query language (e.g. SQL)

 SQL example (retrieve all items supplied by a supplier located in Troy):

 SELECT ItemName

 FROM ITEM, SUPPLIES, SUPPLIER

 WHERE SUPPLIER.City = "Troy" AND

 SUPPLIER.Supplier# = SUPPLIES.Supplier# AND

 SUPPLIES.Item# = ITEM.Item#

 Programming language interfaces: including SQL queries in the code

* 1. **Retrieving and manipulating data: query processing**

Parsing and validating a query: data dictionary - a relation listing all relations and relations listing the attributes

Plans for computing the query: list of possible way to execute the query, estimated cost for each. Example:

SELECT ItemNames, Price

FROM ITEM, SALES

WHERE SALES.Item# = ITEM.Item# AND Salesperson="Mary"

Index: B-tree index, drawbacks - additional space, updating;

indexing not all relations (e.g. the keys only)

Estimating the cost for computing a query: size of the relation, existence/size of the indices. Example: estimating Attribute=value with a given number of tuples and the size of the index.

Query optimization: finding the best plan (minimizing the computational cost and

the size of the intermediate results), subsets of tuples, projection and join.

Static and dynamic optimization

**5. Normalization**

Normalization in DBMS: 1NF, 2NF, 3NF and BCNF in Database. By Chaitanya Singh | Filed Under: DBMS. Normalization is a process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly.

 If a database design is not perfect, it may contain anomalies, which are like a bad dream for any database administrator. Managing a database with anomalies is next to impossible.

* **Update anomalies** − If data items are scattered and are not linked to each other properly, then it could lead to strange situations. For example, when we try to update one data item having its copies scattered over several places, a few instances get updated properly while a few others are left with old values. Such instances leave the database in an inconsistent state.
* **Deletion anomalies** − We tried to delete a record, but parts of it was left undeleted because of unawareness, the data is also saved somewhere else.
* **Insert anomalies** − We tried to insert data in a record that does not exist at all.

Normalization is a method to remove all these anomalies and bring the database to a consistent state.

**5.1 Functional Dependency**

Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

**5.2 Trivial Functional Dependency**

* **Trivial** − If a functional dependency (FD) X → Y holds, where Y is a subset of X, then it is called a trivial FD. Trivial FDs always hold.
* **Non-trivial** − If an FD X → Y holds, where Y is not a subset of X, then it is called a non-trivial FD.
* **Completely non-trivial** − If an FD X → Y holds, where x intersect Y = Φ, it is said to be a completely non-trivial FD.

**5.3 First Normal Form**

First Normal Form is defined in the definition of relations (tables) itself. This rule defines that all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units.



We re-arrange the relation (table) as below, to convert it to First Normal Form.



Each attribute must contain only a single value from its pre-defined domain.

**5.4 Second Normal Form**

Before we learn about the second normal form, we need to understand the following −

* **Prime attribute** − An attribute, which is a part of the candidate-key, is known as a prime attribute.
* **Non-prime attribute** − An attribute, which is not a part of the prime-key, is said to be a non-prime attribute.

If we follow second normal form, then every non-prime attribute should be fully functionally dependent on prime key attribute. That is, if X → A holds, then there should not be any proper subset Y of X, for which Y → A also holds true.



We see here in Student\_Project relation that the prime key attributes are Stu\_ID and Proj\_ID. According to the rule, non-key attributes, i.e. Stu\_Name and Proj\_Name must be dependent upon both and not on any of the prime key attribute individually. But we find that Stu\_Name can be identified by Stu\_ID and Proj\_Name can be identified by Proj\_ID independently. This is called **partial dependency**, which is not allowed in Second Normal Form.



We broke the relation in two as depicted in the above picture. So there exists no partial dependency.

**5.5 Third Normal Form**

For a relation to be in Third Normal Form, it must be in Second Normal form and the following must satisfy −

* No non-prime attribute is transitively dependent on prime key attribute.
* For any non-trivial functional dependency, X → A, then either −
	+ X is a superkey or,
	+ A is prime attribute.



We find that in the above Student\_detail relation, Stu\_ID is the key and only prime key attribute. We find that City can be identified by Stu\_ID as well as Zip itself. Neither Zip is a superkey nor is City a prime attribute. Additionally, Stu\_ID → Zip → City, so there exists **transitive dependency**.

To bring this relation into third normal form, we break the relation into two relations as follows −



**5.6 Boyce-Codd Normal Form**

Boyce-Codd Normal Form (BCNF) is an extension of Third Normal Form on strict terms. BCNF states that −

* For any non-trivial functional dependency, X → A, X must be a super-key.

In the above image, Stu\_ID is the super-key in the relation Student\_Detail and Zip is the super-key in the relation ZipCodes. So,

Stu\_ID → Stu\_Name, Zip

and

Zip → City

Which confirms that both the relations are in BCNF.

**5.7 Denormalization**

Denormalization is a database optimization technique in which we add redundant data to one or more tables. This can help us avoid costly joins in a relational database. Note that denormalization does not mean not doing normalization. It is an optimization technique that is applied after doing normalization.

**Pros of Denormalization:-**

1. Retrieving data is faster since we do fewer joins
2. Queries to retrieve can be simpler(and therefore less likely to have bugs),
since we need to look at fewer tables.

**Cons of Denormalization:-**

1. Updates and inserts are more expensive.
2. Denormalization can make update and insert code harder to write.
3. Data may be inconsistent . Which is the “correct” value for a piece of data?
4. Data redundancy necessities more storage.

**6. Database Access and Security**

**Database security** concerns the use of a broad range of information security controls to protect databases (potentially including the data, the database applications or stored functions, the database systems, the database servers and the associated network links) against compromises of their confidentiality, integrity and availability. It involves various types or categories of controls, such as technical, procedural/administrative and physical.

**6.1 Creating and using indexes**

 Indexes are special lookup tables that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

The basic syntax of a **CREATE INDEX** is as follows.

CREATE INDEX index\_name ON table\_name;

**6.2 creating and using views**

A view is nothing more than a SQL statement that is stored in the database with an associated name. A view is actually a composition of a table in the form of a predefined SQL query.

A view can contain all rows of a table or select rows from a table. A view can be created from one or many tables which depends on the written SQL query to create a view.

Database views are created using the **CREATE VIEW** statement. Views can be created from a single table, multiple tables or another view.

The basic **CREATE VIEW** syntax is as follows −

CREATE VIEW view\_name AS

SELECT column1, column2.....

FROM table\_name

WHERE [condition];

**6.3 Database security**

Database security refers to the collective measures used to protect and secure a database or database management software from illegitimate use and malicious threats and attacks.

**6.4 Difference between Grant and Revoke**

In SQL, the DCL commands are used for assigning the different authorizations to the user, these type of authorizations is known as privilege. Grant and Revoke commands are the DCL commands. The GRANT command is used for conferring the authorization to the users whereas REVOKE command is used for withdrawing the authorization. Select, insert, update and delete are some of the privileges that are included in SQL standards.

**Definition of Grant**

The database administrator defines the **GRANT** command in SQL for giving the access or privileges to the users of the database. Three major components which are involved in the authorization are the users, privilege/s (operations) and a database object. The **user** is the one who triggers the execution of the application program. **Operations** are the component which is embedded in an application program. The **operations** are performed on database objects such as relation or view name.

**SYNTAX of GRANT Command:**

grant <privilege record>
on <relation title or view title>
to <user/role record>;

**Definition of Revoke**

The **REVOKE** command in SQL is defined to take away the granted privileges (authorizations) from the user of the database. The one who has the authority to withdraw the privileges is the database administrator.

**SYNTAX of REVOKE Command:**

revoke <privilege list>
on <relation name or view name>
from <user/role list>;

**Key Differences Between Grant and Revoke**

1. The Grant command confers the privileges to the user while Revoke command withdraws the privileges from the user.
2. In the centralized system, the DCL commands GRANT and REVOKE can be easily executed. When the control is decentralized, the queries are more flexible but complex. GRANT command is easy to deal with, but in case of REVOKE command, it is recursive in manner.

### Conclusion

The GRANT command gives the privileges or access to the users on the database objects. On the other hand, REVOKE command is used for removing the rights or privileges from the users on the database objects.

**7. MYSQL/SQL (Structured Query Language)**

Data Definition Language (DDL) is a standard for commands that define the different structures in a database. DDL statements create, modify, and remove database objects such as tables, indexes, and users. Common DDL statements are CREATE, ALTER, and DROP.

**7.1 Creating Tables**

The **CREATE TABLE** statement is used to **create** a **table** in SQL. We know that a **table** comprises of rows and columns. So while **creating tables** we have to provide all the information to SQL about the names of the columns, type of data to be stored in columns, size of the data etc.

**7.2 Creating a table with data from another table**

The most portable means of copying a table is to:

1. Create the new table with a CREATE TABLE statement
2. Use INSERT based on a SELECT from the old table:
3. INSERT INTO new\_table

SELECT \* FROM old\_table

In SQL Server, I'd use the INTO syntax:

SELECT \*

 INTO new\_table

 FROM old\_table

Because in SQL Server, the INTO clause creates a table that doesn't already exist.

1. **DDL(Data Definition Language) :** DDL or Data Definition Language actually consists of the SQL commands that can be used to define the database schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in database.

**Examples of DDL commands:**

* + [**CREATE**](https://www.geeksforgeeks.org/sql-create/) – is used to create the database or its objects (like table, index, function, views, store procedure and triggers).
	+ [**DROP**](https://www.geeksforgeeks.org/sql-drop-truncate/) – is used to delete objects from the database.
	+ [**ALTER**](https://www.geeksforgeeks.org/sql-alter-add-drop-modify/)-is used to alter the structure of the database.
	+ [**TRUNCATE**](https://www.geeksforgeeks.org/sql-drop-truncate/)–is used to remove all records from a table, including all spaces allocated for the records are removed.
	+ [**COMMENT**](https://www.geeksforgeeks.org/sql-comments/) –is used to add comments to the data dictionary.
	+ [**RENAME**](https://www.geeksforgeeks.org/sql-alter-rename/) –is used to rename an object existing in the database.
1. **DML(Data Manipulation Language) :** The SQL commands that deals with the manipulation of data present in database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

**Examples of DML:**

* + [**SELECT**](https://www.geeksforgeeks.org/sql-select-clause/) – is used to retrieve data from the a database.
	+ [**INSERT**](https://www.geeksforgeeks.org/sql-insert-statement/) – is used to insert data into a table.
	+ [**UPDATE**](https://www.geeksforgeeks.org/sql-update-statement/) – is used to update existing data within a table.
	+ [**DELETE**](https://www.geeksforgeeks.org/sql-delete-statement/) – is used to delete records from a database table.
1. **DCL(Data Control Language) :** DCL includes commands such as GRANT and REVOKE which mainly deals with the rights, permissions and other controls of the database system.

**Examples of DCL commands:**

* + **GRANT**-gives user’s access privileges to database.
	+ **REVOKE**-withdraw user’s access privileges given by using the GRANT command.
1. **TCL(transaction Control Language) :** TCL commands deals with the [transaction within the database](https://www.geeksforgeeks.org/sql-transactions/).

**Examples of TCL commands:**

* + **COMMIT**– commits a Transaction.
	+ [**ROLLBACK**](https://www.geeksforgeeks.org/sql-transactions/)– rollbacks a transaction in case of any error occurs.
	+ **SAVEPOINT**–sets a savepoint within a transaction.
	+ **SET TRANSACTION**–specify characteristics for the transaction.

**7.3 Various types of select commands**

**SQL SELECT Statement**

The most commonly used SQL command is SELECT statement. SQL SELECT statement is used to query or retrieve data from a table in the database. A query may retrieve information from specified columns or from all of the columns in the table. To create a simple SQL SELECT Statement, you must specify the column(s) name and the table name. The whole query is called SQL SELECT Statement.

**Syntax of SQL SELECT Statement:**

SELECT column\_list FROM table-name
[WHERE Clause]
[GROUP BY clause]
[HAVING clause]
[ORDER BY clause];

* table-name is the name of the table from which the information is retrieved.
* column\_list includes one or more columns from which data is retrieved.
* The code within the brackets is optional.

**Various types of joins**

SQL Joins are used to relate information in different tables. A Join condition is a part of the sql query that retrieves rows from two or more tables. A SQL Join condition is used in the SQL [*WHERE Clause*](http://beginner-sql-tutorial.com/sql-where-clause.htm) of select, update, delete statements.

**7.4 Joins in SQL**

The **SQL Syntax** for joining two tables is:

SELECT col1, col2, col3...
FROM table\_name1, table\_name2
WHERE table\_name1.col2 = table\_name2.col1;

SQL Joins can be classified into Equi join and Non Equi join.

**1) SQL Equi joins**

It is a simple sql join condition which uses the equal sign as the comparison operator. Two types of equi joins are SQL Outer join and SQL Inner join.

**For example:** You can get the information about a customer who purchased a product and the quantity of product.

**2) SQL Non equi joins**

It is a sql join condition which makes use of some comparison operator other than the equal sign like >, <, >=, <=

**Sub query**

A Subquery or Inner query or a Nested query is a query within another SQL query and embedded within the WHERE clause. A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

**Aggregate functions**

**Aggregate functions** in SQL. In database management an **aggregate function** is a **function** where the values of multiple rows are grouped together as input on certain criteria to form a single value of more significant meaning. Various **Aggregate Functions** 1) Count() 2) Sum() 3) Avg() 4) Min() 5) Max()

**Challenges of My SQL**

* 1. **Data quality**
	2. **Intellectual property rights**
	3. **Database survivability**
	4. **Introduction to Big Data**

Big Data is a collection of large datasets that cannot be adequately processed using traditional processing techniques. Big data is not only data it has become a complete subject, which involves various tools, techniques and frameworks.

* 1. **Importance of Big Data**

The importance of big data is how you utilize the data which you own. Data can be fetched from any source and analyze it to solve that enable us in terms of

1) Cost reductions

2) Time reductions,

3) New product development and optimized offerings, and

4) Smart decision making.

* 1. **Understanding Big Data with samples**

**Big Data Technologies**

* Accurate analysis carried out based on big data which helps to increase and optimizes operational efficiencies, enable cost reductions, and reduce risks for the business operations.
* In order to capitalize on big data one should require infrastructure that manages and processes huge volumes of structured and unstructured data in real-time and can ensure data privacy and security.
* Many technologies are available in the market from different vendors which includes Amazon, IBM, Microsoft, etc., to approach big data. To pick up a particular technology one must examine its classes, which areas are as follows

**Operational Big Data**

* It includes the applications such as MongoDB which provides operational capabilities for interactive and real time workloads where data is generally captured and stored.
* NoSQL Big Data systems are designed in such a way it capitalizes on new cloud computing architectures, to permit access on massive computations to be run reasonably and efficiently. Hence this builds operation on big data workloads much easier to manage, cheaper and faster to implement.

**Analytical Big Data**

* It owns the systems like Massively Parallel Processing database systems and MapReduce which provides the analytical capabilities for re collective and complex analysis.
* MapReduce provides a new method for analyzing the data that flaunts its capabilities provided by SQL, and based on a system called MapReduce that can be scaled up from single servers to thousands of high and low end machines.
	1. **Barriers**

Barriers that are imposed on big data are as follows:

* Capture data
* Storage Capacity
* Searching
* Sharing
* Transfer
* Analysis
* Presentation

**SECTION-A**

**Note:** Very Short Answer type questions. Attempt any 15 parts. (15x2=30)

Q.1 a) Name two design approach of control unit of CPU.

b) Define the function of SP (Stack pointer).

c) Write the equation that how instruction ADD R2, C, D is evaluated?

d) CISC computers are fast in speed as compared to RISC. (T/F).

e) Define direct addressing mode.

f) Define interrupt.

g) How many 128x8 RAM chips are needed to provide a memory capacity of 2048 bytes?

h) Define memory map.

i) What are the different memory connections with CPU?

j) Name the cache mapping schemes.

k) 1 M bytes = \_\_\_\_\_\_\_ Kbytes.

l) Define control commands.

m) Define baud rate.

n) Define burst transfer.

o) Expand CRC.

p) The memory is shared in tightly coupled multiprocessor (T/F).

q) Define bus arbitration.

r) Define write through policy.

**SECTION-B**

**Note:** Short answer type questions. Attempt any ten parts 10x4=40

Q.2 i) Differentiate hardwired control and micro programmed control.

ii) Explain one address and zero address instruction formats.

iii) Explain stack organization in memory.

iv) Explain register and indirect addressing modes with examples.

v) Explain the structure of a memory device used as backup memory in computers.

vi) Define the purpose of boot strap loader. Which memory device strores boot strap

loader and why?

vii) Draw a suitable diagram of ascociative memory of m word and n cells per word.

viii) Explain direct mapping technique of cache with suitable diagram.

ix) Explain any one page replacement policy.

x) Write the advantages of RISC computers.

xi) Write a note on I/O (Input/output) processor.

xii) Explain interrupt driven data transfer with suitable diagram.

xiii) Draw a suitable diagram that shows DMA data transfer in computer.

xiv) Explain serial (daisy- chain) bus arbitration suitable diagram multiprocessor.

xv) Explain cache coherence problem in multiprocessor system.

**SECTION-C**

**Note:** Long answer type questions. Attempt any three questions. 3x10=30

Q.3 Explain the characteristics of RISC and CISC computers.

Q.4 what is virtual memory? Explain the process that how virtual memory is used in the system.

Q.5 what is BIOS? Explain its functions in computer systems.

Q.6 Define addressing mode. Explain various addressing modes with suitable examples.

Q.7 Name the various interconnection structures used in multiprocessor. Explain the working of any two of them.

**Other Important Questions:-**

1. How is a DBMS distinguished from a file-based system?
2. What is data independence and why is it important?
3. What is the purpose of managing information?
4. Discuss the uses of databases in a business environment.
5. What is metadata?
6. What are the disadvantages of File processing?
7. What is DBMS? Explain advantages and disadvantages of DBMS.
8. What are the different components of DBMS?
9. Explain different types of databases.
10. Explain Data anomalies.
11. What are the functions of DBMS?
12. Explain the importance of Database models
13. Explain Hierarchical Data Model.
14. Explain Network Data Model.
15. Explain Relational Database Model.
16. Explain Entity Relationship Model.
17. Explain The Object Oriented Model.
18. Explain Data abstraction or 3 schema architecture.
19. What is DBMS? Advantages and Disadvantages of DBMS.
20. What are Strong and Weak Entity Sets in DBMS
21. What are the Components of DBMS?
22. Database Normalization
23. What are Data Models? Type of Data Models.
24. Traditional File Processing System
25. Advantages of Database
26. What is ER-Model? Advantages and Disadvantages of E-R Model.
27. What is Data Independence of DBMS?
28. Type of Functional Dependence (FD)
29. Database Languages
30. What is a Database Architecture
31. Relational Model
32. Database Approach
33. What are the Difference between DDL, DML and DCL Commands?
34. What are INTANCES, SCHEMAS AND SUBSCHEMA in DBMS?
35. Network Model
36. What are the Functions and Service of DBMS
37. What is a Database Instance
38. What is Difference between Relation and Relational Schema?
39. What is DBA?
40. What is a Database Server
41. What is the role of DBA, Data Manager, File Manager, and Disk Manager?
42. Type of Database System