CHAPTER 3

Database Integrity and Security Concepts

Contents...

Objectives...

- To understand Concepts of Database Integrity and Integrity Constraints
- To understand Meaning of Database Security and Threats to Database Security
- To learn Methods for Database Security
- To know How Access Control can be Represented in Databases
- To know How View can be Enforced as Security in Databases
- To learn Overview of Encryption Techniques
- To understand Importance of Statistical Database Security

3.0 INTRODUCTION

- Data integrity and security are the most important aspects of the database. Integrity refers to the trustworthiness of data or resources and it is usually phrased in terms of preventing unauthorized access.

- Data integrity refers to the validity, consistency, correctness and accuracy of data in a database and is represented in terms of integrity constraints.
- Database security refers to the preventive measures for maintaining data integrity. Database security concerns the confidentiality, integrity and availability of data stored in a database.
- Data integration means protecting the data from unauthorized access, deletion and modification from the intruder. The goal of database security is the protection of data against threats such as accidental or intentional loss, destruction or misuse.
- In short, database security is protecting the database from unauthorized access, alteration or deletion while data integrity refers to accuracy or correctness or validation of the data.
- A database constraint is a type of rule or restriction placed on a database table/relation. These rules enforced on a data column(s) on table (coloum level constraints) and the whole table (table level constraints).
- Integrity rules (constraints) used to prevent invalid data from being entered into the database. This ensures the accuracy and reliability of the data in the database.

3.1 DOMAIN AND REFERENTIAL INTEGRITY CONSTRAINTS

- Data integrity is enforced using the following three integrity constraints:
  1. **Entity Integrity:** This is related to the concept of primary keys. All tables should have their own primary keys which should uniquely identify a row and not be NULL.
  2. **Referential Integrity:** Referential integrity ensures data in one table maintains the relationship with data to another table/relation. It maintains the dependent values between tables. Referential integrity enforced with a foreign key/primary key relationship. Referential integrity ensures that no row in a child table has a foreign key that doesn't exist as a primary key in the parent table. This means that when we enforce referential integrity, we cannot add a row to the child table with a foreign key that does not match a primary key in the parent table.
  3. **Domain Integrity:** This means that there should be a defined domain for all the columns in a database. Domain integrity refers to restricting the values that are valid for a column (conform to the domain description in place, meaning that values in the column must match a particular format and/or data type) and the allowance or absence of NULLs.
- Integrity constraints define the correct states of a database so that database integrity can be maintained before, during and after database transactions are completed.

3.1.1 Domain Integrity Constraint

- The domain integrity constraints are used to specify the valid values that a column defined over the domain can take. The domain integrity constraint ensures data validity in the database.
- The steps for domain integrity constraints are:

Step 1: In defining the domain integrity constraint is to determine the appropriate data type. The domain data types could be integer, real, Boolean, character, text, inet and so on. For example, the data type of first name and email address is text.

Step 2: After specifying the data type, check constraints, such as the email address pattern, need to be defined.

Check Constraint:

- A check constraint can be applied to a single attribute or a combination of many attributes in a tuple.
- For example, let us assume that customer_service schema is defined as (custorar_id, service_id, start_date, end_date, order_date). For this relation, we can have a check constraint to make sure that start_date and end_date are entered correctly by applying the following check (start_date < end_date).

Default Constraint:

- In default constraint the attribute can have a default value. The default value could be a fixed value such as the default hourly wage of the employees, for example, ₹ 90.
- It may also have a dynamic value based on a function such as random, current time and date. For example, in the customer_service relation, order_date can have a default value which is the current date.

Unique Constraint:

- A unique constraint guarantees that the attribute has a distinct value in each tuple.
- A unique constraint allows null values. For example, let us assume we have a relation player defined as player (player_id, player_nickname). The player uses his/her ID to play with others; he/she can also pick up a nickname which is not used by someone else.

Not Null Constraint:

- By default, the attribute value can be null. The not null constraint restricts an attribute from having a null value.
- For example, each person in the birth registry record should have a name.

3.1.2 Referential Integrity Constraint (April 15)

- In PL/pgSQL relations/tables are associated with each other via common attributes. Referential integrity constraints govern the association between two relations and ensure data consistency between tuples.
- If a tuple in one relation references a tuple in another relation, then the referenced tuple must exist. In the example of customer service, if a service is assigned to a customer, then the service and the customer must exist, as shown in the Fig. 3.1.

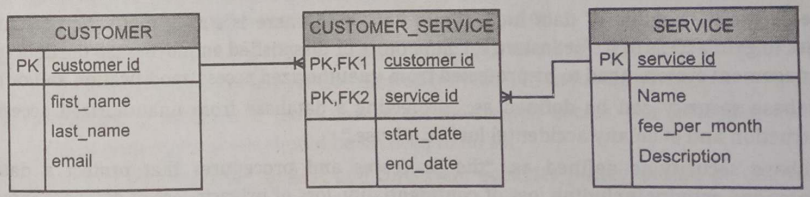

Fig. 3.1

- For example, in the customer service relation, we cannot have a tuple with values (5, 1, 01-01-2019, NULL), because we do not have a customer with customer_id equal to 5.
- The lack of referential integrity constraints can lead to many problems such as Invalid data in the common attributes, Invalid information during joining of data from different relations and Performance degradation.
- In PL/pgSQL the referential integrity constraints are achieved via foreign keys. A foreign key is an attribute or a set of attributes that can identify a tuple in the referenced relation.
- As the purpose of a foreign key is to identify a tuple in the referenced relation, foreign keys are generally primary keys in the referenced relation.
- Unlike a primary key, a foreign key can have a null value. It can also reference a unique attribute in the referenced relation. Allowing a foreign key to have a null value enables us to model different cardinality constraints and define the participation between two different relations.

- For example, a company acquired by another company as shown in Fig. 3.2.

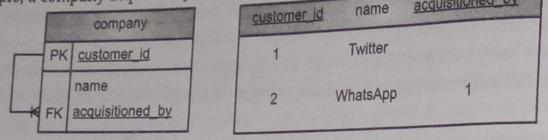

Fig. 3.2

- To ensure data integrity in referential integrity constraints, foreign keys can be used to define several behaviors when a tuple in the referenced relation is updated or deleted.
- The following behaviors in referential are called referential actions in referential integrity constraints:
  1. **Cascade:** When a tuple is deleted or updated in the referenced relation, the tuples in the referencing relation are also updated or deleted.
  2. **Restrict:** The tuple cannot be deleted or the referenced attribute cannot be updated if it is referenced by another relation.
  3. **No Action:** Similar to restrict, but it is deferred to the end of the transaction.
  4. **Set Default:** When a tuple in the referenced relation is deleted or the referenced attribute is updated, then the foreign key value is assigned the default value.
  5. **Set Null:** The foreign key attribute value is set to null when the referenced tuple is deleted.

3.2 INTRODUCTION TO DATABASE SECURITY CONCEPTS (April 15; Oct. 17)

- Security is an important issue in database management because information stored in a database is valuable and sensitive resource.
- Due to the high value of data incorporate databases, there is strong motivation for unauthorized users to gain access to it, for instance, competitors or dissatisfied employees. So the data in a database management system need to be protected from unauthorized access, modification and/or deletion.
- Database security can be defined as, "protecting a database from unauthorized access, malicious destruction and even any accidental loss or misuse."

OR

- Database security is defined as, "the processes and procedures that protect a database from unintended activity including loss of confidentiality, loss of privacy, loss of database integrity or loss of database availability to the users".

Threats to Database Security:

- Threat is any intentional or accidental event that may adversely affect the system. For example, using another person's log-in name to access data, unauthorized copying data, program/data alteration, Illegal entry by hacker, viruses etc.
- A threat is a potential violation of security. A database security violation takes place when someone carries out an unauthorized retrieval, modification or destruction of information in a database.
- The major database security threats are as follows:
  1. **Loss of Confidentiality:** Loss of confidentiality indicates that unauthorized users have been able to access information.
  2. **Loss of Privacy:** It includes loss of protection of an individual's data files
  3. **Loss of Integrity:** It refers to data corruption and invalid data.
  4. **Loss of Availability:** It includes the sabotage of hardware, the network or applications that may cause data to become unavailable to users.

  5. **Theft and Fraud:** It includes intentional security breaches of data and unauthorized data manipulation because of inadequate access controls, inadequate physical security of the database.
  6. **Accidental Losses:** It includes losses as a result of human error, software problems, and hardware problems.
- To protect the database, we must take security measures at several levels: (April 16)
  1. **Physical:** The sites containing the computer systems must be secured against armed or surreptitious entry by intruders.
  2. **Human:** Users must be authorized carefully to reduce the chance of any such user giving access to an intruder in exchange for a bribe or other favors.
  3. **Operating System:** No matter how secure the database system is, weakness in operating system security may serve as a means of unauthorized access to the database.
  4. **Network:** Since, almost all database systems allow remote access through terminals or networks, software-level security within the network software is as important as physical security, both on the Internet and in networks private to an enterprise.
  5. **Database System:** Some database-system users may be authorized to access only a limited portion of the database. Other users may be allowed to issue queries, but may be forbidden to modify the data. It is responsibility of the database system to ensure that these authorization restrictions are not violated.
- Security at all these levels must be maintained if database security is to be ensured. A weakness at a low level of security (physical or human) allows circumvention of strict high level (database) security measures.
- There are three main objectives to consider while designing a secure database application:
  1. **Secrecy**: Information should not be disclosed to unauthorized users. For example, a student should not be allowed to examine other students' grades.
  2. **Integrity**: Only authorized users should be allowed to modify data. For example, students may be allowed to see their grades, yet not allowed (obviously!) to modify them.
  3. **Availability**: Authorized users should not be denied access. For example, an instructor who wishes to change a grade should be allowed to do so.
- To achieve these objectives, a clear and consistent **security policy** should be developed to describe what security measures must be enforced. Next, the **security mechanisms** of the underlying DBMS must be utilized to enforce the policy.

3.3 METHODS FOR DATABASE SECURITY

- In today's world, data, applications and security requirements change while a system is in service. Thus, database systems require continuous monitoring for vulnerabilities, misuse, and other types of system weaknesses security threats can expose.
- There are different techniques to implement the database security as follows:

1. **Authorization:** A DBMS typically includes a database security and authorization sub-system that is responsible for ensuring the security of portions of a database against unauthorized access. It is now customary to refer to two types of database security mechanisms:
   (i) **Discretionary Security Mechanisms:** These are used to grant privileges to users, including the capability to access specific data files, records, or fields in a specified mode (such as read, insert, delete or update).

(ii) **Mandatory Security Mechanisms:** These are used to enforce multilevel security by classifying the data and users into various security classes (or levels) and then implementing the appropriate security policy of the organization.

2. **Access Control:** The security mechanism of a DBMS must include provisions for restricting access to the database as a whole. This function is called access control. An access control mechanism is a way to control the data that is accessible to a given user. It is handled by creating user accounts and passwords to control login process by the DBMS.
3. **Statistical Database Security:** The security problem associated with databases is that of controlling the access to a statistical database, which is used to provide statistical information or summaries of values based on various criteria. The countermeasures to statistical database security problem are called inference control measures.
4. **Database Encryption Techniques:** It is used to protect sensitive data (such as credit card numbers) that is being transmitted via some type communication network. The data is encoded using some encoding algorithm. An unauthorized user who access encoded data will have difficulty deciphering it, but authorized users are given decoding or decrypting algorithms (or keys) to decipher data.

Database Security and the DBA:

- The database administrator (DBA) is the central authority for managing a database system. The DBA's responsibilities include:
  1. Granting privileges to users who need to use the system, and
  2. Classifying users and data in accordance with the policy of the organization.
- The DBA is responsible for the overall security of the database system. The DBA has a DBA account in the DBMS. Sometimes, these are called a system or super user account. These accounts provide powerful capabilities such as:
  1. **Account Creation:** This action creates a new account and password for a user or a group of users to enable them to access the DBMS.
  2. **Privilege Granting:** This action permits the DBA to grant certain privileges to certain accounts.
  3. **Privilege Revocation:** This action permits the DBA to revoke (cancel) certain privileges that were previously given to certain accounts.
  4. **Security Level Assignment:** This action consists of assigning user accounts to the appropriate security classification level.
- Action 1 is access control, whereas 2 and 3 are discretionary security mechanisms and 4 is used to control mandatory authorization security mechanisms.
- If any tampering with the database is suspected, a database audit is performed. A database audit consists of reviewing the log to examine all accesses and operations applied to the database during a certain time period.
- A database log that is used mainly for security purposes is sometimes called an audit trail.

3.3.1 Discretionary Access Control Method

(April 15)

- Discretionary Access Control (DAC) method is based on the concept of access rights (also called privileges) and mechanism for giving users such privileges.
- The typical method of enforcing discretionary access control in a database system is based on the granting and revoking privileges.
- A privilege is an action, such as creating, executing, reading, updating or deleting that a user is permitted to perform on database objects.

3.3.1.1 Types of Discretionary Privileges

- The DBMS must provide selective access to each relation in the database based on specific accounts. Operations may also be controlled; thus having an account does not necessarily entitle the account holder to all the functionality provided by the DBMS.
- There are two levels for assigning privileges to use the database system:
  1. **Account Level:** At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.
  2. **Relation Level (or Table Level):** At this level, the DBA can control the privilege to access each individual relation or view in the database.
- The privileges at the **account level** apply to the capabilities provided to the account itself and can include:
  1. The **CREATE SCHEMA** or **CREATE TABLE** privilege, to create a schema or base relation.
  2. The **CREATE VIEW** privilege.
  3. The **ALTER** privilege, to apply schema changes such adding or removing attributes from relations.
  4. The **DROP** privilege, to delete relations or views.
  5. The **MODIFY** privilege, to insert, delete, or update tuples, and
  6. The **SELECT** privilege, to retrieve information from the database by using a SELECT query.
- The second level of privileges applies to the relation level which includes base relations and virtual (view) relations.
- The granting and revoking of privileges generally follow an authorization model for discretionary privileges known as the access matrix or authorization matrix model.
- In the access matrix model:
  1. The rows of a matrix M represents subjects (users, accounts, programs).
  2. The columns represent objects (relations, records, columns, views, operations).
  3. Each position M (i, j) in the matrix represents the types of privileges (read, write, update) that subject i holds on object j.
- To control the granting and revoking of relation privileges, each relation R in a database is assigned and owner account, which is typically the account that was used when the relation was created in the first place.
- The owner of a relation is given all privileges on that relation. The DBA can assign and owner to a whole schema by creating the schema and associating the appropriate authorization identifier with that schema, using the CREATE SCHEMA command.
- The owner account holder can pass privileges on any of the owned relation to other users by granting privileges to their accounts.
- In SQL the following types of privileges can be granted on each individual relation R:
  1. **SELECT** (retrieval or read) privilege on R:
     - Gives the account retrieval privilege.
     - In SQL this gives the account the privilege to use the SELECT statement to retrieve tuples from R.
  2. **MODIFY** privileges on R:
     - This gives the account the capability to modify tuples of R.
     - In SQL this privilege is further divided into UPDATE, DELETE, and INSERT privileges to apply the corresponding SQL command to R.

- In addition, both the INSERT and UPDATE privileges can specify that only certain attributes can be updated by the account.

3. **REFERENCES** privilege on R:
   - This gives the account the capability to reference relation R when specifying integrity constraints.
   - The privilege can also be restricted to specific attributes of R.
   - To create a view, the account must have SELECT privilege on all relations involved in the view definition.

Specifying Privileges using Views:

- The mechanism of views is an important discretionary authorization mechanism in its own right. For example,
  - If the owner A of a relation R wants another account B to be able to retrieve only some fields of R, then A can create a view V of R that includes only those attributes and then grant SELECT on V to B.
  - The same applies to limiting B to retrieving only certain tuples of R; a view V' can be created by defining the view by means of a query that selects only those tuples from R that A wants to allow B to access.

3.3.1.2 Granting and Revoking Privileges (April 16)

- A privilege is the ability to access a specific database resource or to perform a specific action within a database. Examples of privileges include deleting a row, creating a table or executing a procedure.
- A privilege is an access right to execute a particular type of PL/pgSQL statement or to access another user's object.
- There are two types of privileges as explained below:

1. **System Privileges:** A system privilege is the right to perfrom a particular action on a particular type of object. For example, the privileges to create tables and to delete the rows of table in a database are system privileges. System privilege indicate user to CREATE, ALTER, or DROP database elements.
2. **Object Privileges:** An object privilege is a privilege or right to perform a particular action on a specific table, view, sequence, procedure, function or package. For example, the privilege to delete rows from the table DEPT is an object privilege. Object privileges allows user to EXECUTE, SELECT, INSERT or DELETE data from database objects to which the privileges apply.

1. **GRANT Command:**

- GRANT command is used to provide access or privileges on the database objects to the users. Thus, grant command is used to confer authorization.
- GRANT command has the following **syntax:**

```
GRANT privilege_name
ON  <tablename | viewname>
TO {user list | role_name}
[WITH GRANT OPTION];
```

Where, privilege_name is the access right or privilege granted to the user like SELECT, INSERT, UPDATE, DELETE etc., WITH GRANT OPTION - allows a user to grant access rights to other users.

- Following example states that the user 'AMAR' is authorised to perform SELECT, DELETE and UPDATE operations on the table (or relation) EMPLOYEE with the capability to grant those privileges to other users on EMPLOYEE table.


```
GRANT SELECT, DELETE, UPDATE
ON EMPLOYEE
TO AMAR
WITH GRANT OPTION
```

2. ***REVOKE* Command:** (April 16)

- In some cases it is desirable to grant a privilege to a user temporarily. For example, the owner of a relation may want to grant the SELECT privilege to a user for a specific task and then revoke that privilege once the task is completed. Hence, a mechanism for revoking privileges is needed.
- In PL/pgSQL, a REVOKE command is included for the purpose of canceling privileges. The REVOKE command is used to remove or cancel the given privileges or access rigths from the user.

Syntax:

```
REVOKE privilege_name
ON <tablename | viewname>
FROM { user list | role_name} [restrict | cascade ]
```

- If we use RESTRICT keyword, the privilege will be revoked only from the specified user. If the specified user granted had the WITH GRANT OPTION and granted the same privilege to other users, they will retain the privilege.
- If we use CASCADE, it will revoke the privilege and any dependent privileges as a result of our grant. A dependent privilege is one that could exist, if we granted the privilege that we were trying to revoke, which is what we are trying to achieve as a result of our REVOKE statement.
- Following example states that the DELETE and UPDATE authority on the EMP-ID and EMP-SALARY attributes (columns) are removed from the user 'AMAR'.

```
REVOKE DELETE, UPDATE (EMP-ID, EMP-SALARY)
ON EMPLOYEE
FROM AMAR
```

Propagation of Privileges using the GRANT OPTION:

- Whenever, the owner A of a relation R grants a privilege on R to another account B, privilege can be given to B with or without the GRANT OPTION. If the GRANT OPTION is given, this means that B can also grant that privilege on R to other accounts.
  - Suppose that B is given the GRANT OPTION by A and that B then grants the privilege on R to a third account C, also with GRANT OPTION. In this way, privileges on R can propagate to other accounts without the knowledge of the owner of R.
  - If the owner account A now revokes the privilege granted to B, all the privileges that B propagated based on that privilege should automatically be revoked by the system.

Example: Suppose that A1 creates the two base relations EMPLOYEE and DEPARTMENT. A1 is then owner of these two relations and hence all the relation privileges on each of them.

EMPLOYEE Relation

NAME	SSN	BDATE	ADDRESS	SALARY	DNO

DEPARTMENT Relation

DNUMBER	DNAME	MGR_SSN

- Suppose that A1 wants to grant A2 the privilege to insert and delete tuples in both of these relations, but A1 does not want A2 to be able to propagate these privileges to additional accounts:

```
GRANT INSERT, DELETE ON
EMPLOYEE, DEPARTMENT TO A2;
```

- Suppose that A1 wants to allow A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts. A1 can issue the command:

```
GRANT SELECT ON EMPLOYEE, DEPARTMENT
TO A3 WITH GRANT OPTION;
```

- A3 can grant the SELECT privilege on the EMPLOYEE relation to A4 by issuing:

```
GRANT SELECT ON EMPLOYEE TO A4;
```

- Notice that A4 can not propagate the SELECT privilege because GRANT OPTION was not given to A4
- Suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 can issue:

```
REVOKE SELECT ON EMPLOYEE FROM A3 CASCADE;
```

- The DBMS must now automatically revoke the SELECT privilege on EMPLOYEE from A4, too, because A3 granted that privilege to A4 and A3 does not have the privilege any more.
- Suppose that A1 wants to give back to A3 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A3 to be able to propagate the privilege. The limitation is to retrieve only the NAME, BDATE and ADDRESS attributes and only for the tuples with DNO=5. A1 then create the view:

```
CREATE VIEW A3 EMPLOYEE AS
SELECT NAME, BDATE, ADDRESS
FROM EMPLOYEE
WHERE DNO = 5;
```

- After the view is created, A1 can grant SELECT on the view A3EMPLOYEE to A3 as follows:

```
GRANT SELECT ON A3EMPLOYEE TO A3 WITH GRANT OPTION;
```

- Suppose that A1 wants to allow A4 to update only the SALARY attribute of EMPLOYEE; A1 can issue:

```
GRANT UPDATE ON EMPLOYEE (SALARY) TO A4;
```

- The UPDATE or INSERT privilege can specify particular attributes that may be updated or inserted in a relation. Other privileges (SELECT, DELETE) are not attribute specific.

3.3.1.3 Audit Trails

- The DBMS must provide selective access to each relation in the database based on specific login accounts. Database auditing involves observing a database so as to be aware of the actions of database users.
- An audit trail is a log of all changes (inserts/deletes/updates) to the database along with information such as which user performed the change, and when the change was performed.
- A typical audit trail entry might contain the information such as request (source text), terminal from which the operation was evoked, user who evoked the operation, date and time of the operation, tuples, attributes, affected, old values and new values.
- A audit trail is used to track erroneous/fraudulent updates. The audit trail aids security to the database.
- For example, if the balance on a bank account is found to be incorrect, bank may wish to trace all the updates performed on the account, to find out incorrect updates, as well as the persons who carried out the updates. The bank could then also use the audit trail to trace all the updates performed by these persons, in order to find other incorrect updates.

3.3.2 Mandatory Access Control Method

(April 16, 18; Oct. 17)

- The discretionary access control techniques of granting and revoking privileges on relations have traditionally been the main security mechanism for relational database systems. DAC is an all-or-nothing method because a user either has or does not have a certain privilege.
- In many applications, and additional security policy is needed that classifies data and users based on security classes. This approach as mandatory access control would typically be combined with the discretionary access control mechanisms.
- Mandatory Access Control (MAC) is based on system-wide policies that cannot be changed by individual users.
- MAC is used to enforce multi-level security by classifying the data and users into various security classes or levels and then implementing the appropriate security policy of the organisation.
- In MAC method:
  - Typical security classes are Top Secret (TS), Secret (S), Confidential (C), and Unclassified (U), where TS is the highest level and U the lowest: $TS \geq S \geq C \geq U$.
  - The commonly used model for multilevel security, known as the Bell-LaPadula model, classifies each subject (user, account, program) and object (relation, tuple, column, view, operation) into one of the security classifications, T, S, C or U.
  - Clearance (classification) of a subject S as class (S) and to the classification of an object O as class (O).
  - Two restrictions are enforced on data access based on the subject/object classifications:
    - **Simple Security Property:** A subject S is not allowed read access to an object O unless class(S) ≥ class(O).
    - **Star Property (or * Property):** A subject S is not allowed to write an object O unless class(S) ≤ class(O).
- The first restriction enforces the rule that no subject can read an object whose security classification is higher than the subject's security clearance.
- The second restriction is less intuitive. It prohibits a subject from writing an object at a lower security classification than the subject's security clearance. Violation of this rule would allow information to flow from higher to lower classifications, which violates a basic tenet of multilevel security.
- To incorporate multilevel security notions into the relational database model, it is common to consider attribute values and tuples as data objects. Hence, each attribute A is associated with a classification attribute C in the schema, and each attribute value in a tuple is associated with a corresponding security classification. In addition, in some models, a tuple classification attribute TC is added to the relation attributes to provide a classification for each tuple as a whole. Hence, a multilevel relation schema R with n attributes would be represented as R (A_1, C_1, A_2, C_2, ..., A_n, C_n, TC) where, each C_i represents the classification attribute associated with attribute A_i. The value of the TC attribute in each tuple t – which is the highest of all attribute classification values within t – provides a general classification for the tuple itself, whereas each C_i provides a finer security classification for each attribute value within the tuple.

Multilevel Relations and Polyinstantiation:

- The apparent key of a multilevel relation is the set of attributes that would have formed the primary key in a regular (single-level) relation. A multilevel relation will appear to contain different data to subjects (users) with different clearance levels.

- In some cases, it is possible to store a single tuple in the relation at a higher classification level and produce the corresponding tuple at a lower-level classification through a process known as filtering.
- In other cases, it is necessary to store two or more tuples at different classification levels with the same value for the apparent key. This leads to the concept of polyinstantiation where several tuples can have the same apparent key value but have different attribute values for users at different classification levels.
- The presence of data objects that appear to have different values to users with different clearances is called polyinstantiation.

Comparing Discretionary Access Control and Mandatory Access Control:

- Discretionary Access Control (DAC) policies are characterized by a high degree of flexibility, which makes them suitable for a large variety of application domains.
- The main drawback of DAC models is their vulnerability to malicious attacks, such as Trojan horses embedded in application programs.
- By contrast, mandatory policies ensure a high degree of protection in a way; they prevent any illegal flow of information. Mandatory policies have the drawback of being too rigid and they are only applicable in limited environments.
- In many practical situations, discretionary policies are preferred because they offer a better trade-off between security and applicability.

3.3.3 Role Based Access Control for Multilevel Security

- The goal of MultiLevel Security (MLS) for a relational database is to prevent the unauthorized access of the data by preventing any user from accessing any data to which he/she has no access.
- Role Based Access Control (RBAC) emerged rapidly in the 1990s as a proven technology for managing and enforcing security in large-scale enterprise wide systems. Its basic notion is that permissions are associated with roles and users are assigned to appropriate roles.
- A role is a set of related privileges that are combined to provide a centralized unit from which to manage similar users or objects of a database.
- A role to serve one of the following two purposes:
  1. to manage the privileges for a database application and/or
  2. to manage the privileges for a user group.
- Roles are a collection of privileges or access rights. When there are many users in a database it becomes difficult to grant or revoke privileges to users.
- There are two types of roles namely, application roles and user roles as explained below:
  1. An **application role** is created by granting all the privileges necessary to run a given database application. Then, the application role is granted to other roles or to specific users.
  2. A **user role** is created to a group of database users with common privilege requirements. User privileges are managed by granting application roles and privileges to the user role and granting the user role to appropriate users.
- Roles can be created using the CREATE ROLE and delete or destroy a role using DROP ROLE command. The GRANT and REVOKE commands discussed under DAC can then be used to assign and revoke privileges from roles.
- RBAC appears to be a viable alternative to traditional discretionary and mandatory access controls; it ensures that only authorized users are given access to certain data or resources.
- Many DBMSs have allowed the concept of roles, where privileges can be assigned to roles. Role hierarchy in RBAC is a natural way of organizing roles to reflect the organization's lines of authority and responsibility.

- Another important consideration in RBAC systems is the possible temporal constraints that may exist on roles, such as time and duration of role activations, and timed triggering of a role by an activation of another role.
- Using an RBAC model is highly desirable goal for addressing the key security requirements of Web-based applications. In contrast, Discretionary Access Control (DAC) and Mandatory Access Control (MAC) models lack capabilities needed to support the security requirements emerging enterprises and Web-based applications.

Creating Roles:

- The **syntax** to create a role is:

```
CREATE ROLE role_name [IDENTIFIED BY password]
```

- **For example:** To create a role called "testing" with password as "test@12345", the code will be as follows:

```
CREATE ROLE testing [IDENTIFIED BY test@12345]
```

- It is easier to GRANT or REVOKE privileges to the users through a role rather than assigning a privilege directly to every user. If a role is identified by a password, then, when we GRANT or REVOKE privileges to the role, we definitely have to identify it with the password.
- We can GRANT or REVOKE privilege to a role as given below.
- **For example:** To grant CREATE TABLE privilege to a user by creating a testing role:
  1. First, create a testing role:

     ```
     CREATE ROLE testing [IDENTIFIED BY test@12345]
     ```

  2. Second, grant a CREATE TABLE privilege to the ROLE testing. We can add more privileges to the ROLE:

     ```
     GRANT CREATE TABLE TO testing;
     ```

  3. Third, grant the role to a user.

     ```
     GRANT testing TO user1;
     ```

     To revoke a CREATE TABLE privilege from testing ROLE, we can write:

     ```
     REVOKE CREATE TABLE FROM testing;
     ```

Droping Roles:

- The **syntax** to drop a role from the database is given below:

```
DROP ROLE role_name;
```

- **For example:** To drop a role called testing, we can write:

```
DROP ROLE testing;
```

3.4 USE OF VIEWS IN SECURITY ENFORCEMENT (April 15, 18)

- A view is a database object that is of a stored query. A view can be accessed as a virtual table in PostgreSQL.

3.4.1 Concept of View (April 15, 18)

- We can use views to restrict table access so that the users see only specific rows or columns of a table. A view is a subset of a real table, selecting certain columns or certain rows from an ordinary table.
- Views are pseudo-tables means they are not real tables. A view can be defined as, "a virtual relation or table that is derived from one or more base relations".
- When we create a view, we basically create a query and assign it a name, therefore a view is useful for wrapping a commonly used complex query.

- A view can be created from one or many tables, which depends on the written PostgreSQL query. Views, which are kind of virtual tables, allow users to do the following:
  1. Structure data in a way that users or classes of users find natural or intuitive.
  2. Restrict access to the data such that a user can only see limited data instead of complete table.
  3. Summarize data from various tables, which can be used to generate reports.
- A normal view does not store any data except the materialized view. Since views are not ordinary tables, we may not be able to execute a DELETE, INSERT, or UPDATE statement on a view.
- Note that in a PostgreSQL, we can create a special view called a materialized view that stores data physically and refreshes the data periodically from the base tables.

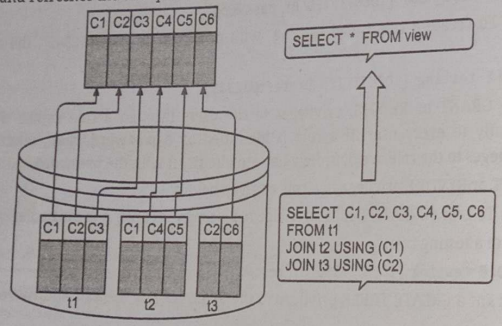

Fig. 3.3: Concept of View

3.4.2 Creating Views

(April 15, 16)

- The PostgreSQL (PL/pgSQL) views are created using the CREATE VIEW statement. The PostgreSQL views can be created from a single table, multiple tables or another view.

Syntax:

```
CREATE [TEMP | TEMPORARY] VIEW view_name AS
SELECT column1, column2.....
FROM table_name
WHERE [condition];
```

- We can include multiple tables in the SELECT statement in very similar way as we use them in normal PostgreSQL SELECT query.
- If the optional TEMP or TEMPORARY keyword is present, the view will be created in the temporary space. Temporary views are automatically dropped at the end of the current session.
- Consider following emp table:

id	name	age	address	salary
1	Paul	32	Pune	20000
2	Allen	25	Mumbai	15000
3	Teddy	23	Pune	20000
4	Mark	25	Bangalore	65000
5	David	27	Mumbai	85000

Example:

```
postgres=# CREATE VIEW COMPANY_VIEW AS
SELECT ID, NAME, AGE
FROM COMPANY;
```

- Now, we can query COMPANY_VIEW in a similar way as we query an actual table.

```
postgres =# SELECT * FROM COMPANY_VIEW;
```

id	name	age
1	Paul	32
2	Allen	25
3	Teddy	23
4	Mark	25
5	David	27

- To change the defining query of a view, we use the CREATE VIEW statement with OR REPLACE addition as follows:

```
CREATE OR REPLACE view_name
AS Query
```

- To change the definition of a view, use the ALTER VIEW statement. PosgreSQL allows us to set a default value for a column name, change the view's schema, set or reset options of a view.
- For example, we can change the name of the view from customer_master to customer_info by using the following statement:

```
ALTER VIEW customer_master RENAME TO customer_info
```

3.4.3 Dropping Views

(April 15, 16)

- To drop a view, simply use the DROP VIEW statement with the view_name.

Syntax:

```
DROP VIEW [IF EXISTS] view_name;
```

- We specify the name of the view that you want to remove after DROP VIEW clause. Removing a view that does not exist in the database will result an error. To avoid this, we normally add IF EXISTS option to the statement to instruct PostgreSQL to remove the view if it exists, otherwise do nothing.
- Following command will delete COMPANY_VIEW view, which previously created:

```
postgres=# DROP VIEW IF EXISTS COMPANY_VIEW;
```

3.4.4 Security Enforcement Using Views

- A view is a virtual relation that does not actually exist in the database, but is produced upon request by a particular user, at the time of request.
- It is the dynamic result of one or more relational operations operating on the base relations to produce another relation. The view is a widely used method for implementing access control in database applications.
- The view mechanism provides a powerful and flexible security mechanism by hiding parts of the database from certain users. The user is not aware of the existence of any attributes or rows that are missing from the view.

- Suppose that A1 wants to give a limited capability to A3 for SELECT from the EMPLOYEE relation and wants to allow A3 to be able to propagate the privilege.
- The limitation is to retrieve only the NAME, BDATE, and ADDRESS attributes and only for the tuples with DNO=5. A1 then create the view:

```
CREATE VIEW A3 EMPLOYEE AS
SELECT NAME, BDATE, ADDRESS
FROM EMPLOYEE
WHERE DNO = 5;
```

- After the view is created, A1 can grant SELECT on the view A3EMPLOYEE to A3 as follows:

```
GRANT SELECT ON A3EMPLOYEE TO A3 WITH GRANT OPTION
```

3.5 OVERVIEW OF ENCRYPTION TECHNIQUE FOR SECURITY (April 16)

- The primary purpose of encryption is to protect the confidentiality of data stored in database systems or transmitted via the Internet or other computer networks.
- Encryption is a means of maintaining secure data in an insecure environment. Encryption consists of applying an encryption algorithm to data using some pre-specified encryption key. The resulting data has to be decrypted using a decryption key to recover the original data.
- Encryption is a technique of encoding data so, that only authorized users can understand it. There are two advantages of encrypting data stored in the database:
  1. Encrypted data allows authorized users to access data without worrying about other users or the system administrator gaining any information.
  2. Encryption of data may simplify or even strengthen other authorization mechanisms. For example, distribution of the cryptographic key amongst only trusted users is both, a simple way to control read access, and an added layer of security above that offered by views.
- Data encryption technique is used to protect data from threats. Data encryption is a method of coding or scrambling of data so that humans cannot read them.

Basic Concept of Encryption:

- In encryption, the message to be encrypted is known as plaintext. The plaintext is transformed by a function that is parameterized by a key. The output of the encryption process is known as the cipher text.
- The process of converting the plaintext to ciphertext is called as encryption and process of converting the ciphertext to plaintext is called as decryption. Encryption is performed at the transmitting end and decryption is performed at the receiving end.
- For encryption process we need the encryption key and for decryption process we need decryption key as shown in Fig. 3.4.
- Without the knowledge of decryption key intruder cannot break the ciphertext to plaintext. This process is also called as Cryptography.

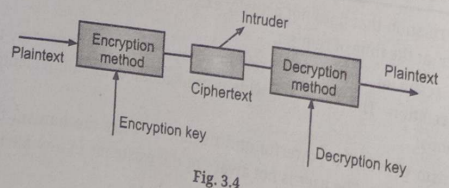

Fig. 3.4

- Today's encryption algorithms are divided into two categories i.e., symmetric and asymmetric. In symmetric algorithm key ciphers use the same key, or secret, for encrypting and decrypting a message or file. The **Data Encryption Standard (DES)** is symmetric-key method of data encryption.
- Asymmetric algorithm, also known as public-key cryptography, uses two different but mathematically linked keys, one public and one private. RSA is asymemetric algorithm.
- The **Data Encryption Standard (DES)** is a system developed by the U.S. government for use by the general public. It has been widely accepted as a cryptographic standard both in the United States and abroad.
- DES can provide end-to-end encryption on the channel between the sender A and receiver B. DES algorithm is a careful and complex combination of two of the fundamental building blocks of encryption namely, substitution and permutation (transposition).
- The DES algorithm derives its strength from repeated application of these two techniques for a total of 16 cycles.
- Plaintext (the original form of the message) is encrypted as blocks of 64 bits. After questioning the adequacy of DES, the National Institute of Standards (NIST) introduced the **Advanced Encryption Standards (AES)**. This algorithm has a block size of 128 bits and thus takes longer time to crack.
- In 1976 Diffie and Hellman proposed a new kind of cryptosystem, which they called **public key encryption**. Public key algorithms are based on mathematical functions rather than operations on bit patterns.
- They also involve the use of two separate keys in contrast to conventional encryption, which uses only one key.
- The use of two keys can have profound consequences in the areas of confidentiality, key distribution, and authentication.
- The two keys used for public key encryption are referred to as the public key and the private key. The private key is kept secret, but it is referred to as private key rather than a secret key.
- A public key encryption scheme or infrastructure has following ingredients:
  1. **Plaintext**: This is the data or readable message that is fed into the algorithm as input.
  2. **Encryption Algorithm**: The encryption algorithm performs various transformations on the plaintext to convert into unreadable format.
  3. **Public and Private Keys**: These are pair of keys that have been selected so that if one is used for encryption, the other is used for decryption.
  4. **Ciphertext**: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different cipher texts. (April 15)
  5. **Decryption Algorithm:** This algorithm accepts the cipher text and the matching key and produces the original plaintext. Public key is made for public and private key is known only by owner. A general-purpose public key cryptographic algorithm relies on One key for encryption and a different but related key for decryption.
- The essential steps for public key infrastructure are as follows:
  1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
  2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private (private key).
  3. If a sender wishes to send a private message to a receiver, the sender encrypts the message using the receiver's public key.
  4. When the receiver receives the message, he or she decrypts it using the receiver's private key. No other recipient can decrypt the message because only the receiver knows his or her private key.

- The RSA Public Key Encryption algorithm, one of the first public key schemes was introduced in 1978 by Ron Rivest (R), Adi Shamir (S), and Len Adleman (A) at MIT and is named after them.
- The RSA encryption algorithm incorporates results from number theory, such as the difficulty of determining the large prime factors of a large number.

3.6 STATISTICAL DATABASE SECURITY (Oct. 16)

- Statistical databases are used to provide statistical information or summaries of values based on various criteria. For example, a database for population statistics may provide statistics based on age groups, income levels, household size, education levels, and other criteria.
- A statistical database contains confidential information about individuals or events, which is used to answer statistical queries concerning sums, averages, and numbers with certain characteristics.
- A statistical databases contians confidential information about individuals. These databases are mainly used to generate statistical information about the stored information.
- The statistical databases accept only statistical queries, which involve stastistical functions such as SUM, AVG, COUNT, MIN, MAX and so on. However, users are not allowed to retrieve information about a particular individual.
- Implementing security in such a database raises new problems because it is possible to infer the information about specific individuals from a sequence of satistical queries.
- Statistical database users such as government statisticians (or market research firms) are allowed to access the database to retrieve statistical information about a population but not to access the detailed confidential information about specific individuals. Security for statistical databases must ensure that information about individuals cannot be accessed.
- Statistical database security system is used to control the access to a statistical database, which is used to provide statistical information or summaries of values based on various criteria.
- Statistical databases are mainly, used to produce statistics on various populations. The database may contain confidential data on individuals, which should be protected from user access.
- Users are permitted to retrieve statistical information on the populations, such as averages, sums, counts, maximums, minimums, and standard deviations.
- A population is a set of tuples of a relation (table) that satisfy some selection condition. Statistical queries involve applying statistical functions to a population of tuples.
- For example, we may want to retrieve the number of individuals in a population or the average income in the population. However, statistical users are not allowed to retrieve individual data, such as the income of a specific person.
- For example, if a user is permitted statistical access to an employee database, he or she is able to write queries such as:

```
SELECT SUM (Salary)
FROM Employee
WHERE Dept = 10;
```

but not:

```
SELECT Sarlary
FROM Employee
WHERE empId = 'E101';
```

- Statistical database security techniques must prohibit the retrieval of individual data. This can be achieved by prohibiting queries that retrieve attribute values and by allowing only queries that involve statistical aggregate functions such as COUNT, SUM, MIN, MAX, AVERAGE, and STANDARD DEVIATION. Such queries are sometimes called statistical queries.
- It is DBMS's responsibility to ensure confidentiality of information about individuals, while still providing useful statistical summaries of data about those individuals to users. Provision of privacy protection of users in a statistical database is paramount.
- In some cases it is possible to infer the values of individual tuples from sequence statistical queries. This is particularly true when the conditions result in a population consisting of a small number of tuples.

Example: Given the following database schema:

EMPLOYEE Relation

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	DNO

DEPARTMENT Relation

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE

DEPT_LOCATION Relation

DNUMBER	DLOCATION

PROJECT Relation

PNAME	PNUMBER	PLOCATION	DNUM

DEPENDENT Relation

ESSN	DEP-NAME	SEX	BDATE	RELATIONSHIP

- Suppose that all the relations were created by (and hence are owned by) user X, who wants to grant the following privileges to user accounts A, B, C, D, and E:
  1. Account A can retrieve or modify any relation except DEPENDENT and can grant any of these privileges to other users.

     ```
     GRANT SELECT, UPDATE ON EMPLOYEE, DEPARTMENT, DEPT_LOCATIONS, PROJECT, WORKS_ON TO
     ACCOUNTA WITH GRANT OPTION;
     ```

  2. Account B can retrieve all the attributes of EMPLOYEE and DEPARTMENT except for SALARY, MGRSSN, and MGRSTARTDATE.

     ```
     CREATE VIEW EMPS AS SELECT FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX SUPERSSN,
     DNO FROM EMPLOYEE;
     GRANT SELECT ON EMPS TO ACCOUNTB;
     CREATE VIEW DEPTS AS SELECT DNAME, DNUMBER FROM DEPARTMENT;
     GRANT SELECT ON DEPTS TO ACCOUNTB;
     ```

  3. Account C can retrieve or modify WORKS_ON but can only retrieve the FNAME, MINIT, LNAME, SSN attributes of EMPLOYEE and PNAME, PNUMBER attributes of PROJECT.

     ```
     GRANT SELECT, UPDATE ON WORKS_ON TO ACCOUNTC;
     CREATE VIEW EMP1 AS SELECT FNAME, MINIT, LNAME, SSN FROM EMPLOYEE;
     ```

```
GRANT SELECT ON EMP1 TO ACCOUNTC;
CREATE VIEW PROJ1 AS SELECT PNAME, PNUMBER FROM PROJECT;
GRANT SELECT ON PROJ1 TO ACCOUNTC;
```

4. Account D can retrieve any attribute of EMPLOYEE or DEPENDENT and can modify DEPENDENT.

```
GRANT SELECT ON EMPLOYEE, DEPENDENT TO ACCOUNTD;
GRANT UPDATE ON DEPENDENT TO ACCOUNTD;
```

5. Account E can retrieve any attribute of EMPLOYEE but only for EMPLOYEE tuples that have DNO = 3. Write SQL statements to grant these privileges. Use views where appropriate.

```
CREATE VIEW DNO3_EMPLOYEES AS SELECT * FROM EMPLOYEE WHERE DNO=3;
GRANT SELECT ON DNO3_EMPLOYEES TO ACCOUNTE;
```

PRACTICE QUESTIONS

Q.I Multiple Choice Questions:

1. The protection of data against unauthorized access is called as?
   (a) Data security (b) Data integrity
   (c) Both (a) and (b) (d) All of these
2. The validity, consistency, and accuracy of the data in a database is knows as?
   (a) Data security (b) Data integrity
   (c) Both (a) and (b) (d) All of these
3. Data integrity can be ensured by checking which of the following?
   (a) integrity (b) Entity integrity
   (c) Referential integrity (d) All of these
4. Which a potential violation of security?
   (a) Virus (b) Exception
   (c) Threat (d) None of these
5. Which integrity constraint ensures data validity?
   (a) Domain (b) Referential
   (c) Entity (d) None of these
6. Which access control is based on the concept of access rights and mechanism for giving users such privileges?
   (a) Role Based Access Control (RBAC) (b) Mandatory Access Control (MAC)
   (c) Discretionary Access Control (DAC) (d) All of these
7. Which command removes user access rights or privileges to the database objects?
   (a) GRANT (b) REVOKE
   (c) Both (a) and (b) (d) None of these
8. Which access control based on labeling data objects with security classes and assigning security clearances to users?
   (a) Role Based Access Control (RBAC) (b) Mandatory Access Control (MAC)
   (c) Discretionary Access Control (DAC) (d) All of these
9. Referential integrity constraints are achieved via?
   (a) super keys (b) foreign keys
   (c) candidate keys (d) None of these

10. Which access control relevant to database systems, is concerned with aggregating access privileges into named entities (called roles) and assigning such entities to individual users, groups of users, and other roles.
    (a) Role Based Access Control (RBAC) (b) Mandatory Access Control (MAC)
    (c) Discretionary Access Control (DAC) (d) All of these
11. A _____ can be granted system or object privileges.
    (a) priviledge (b) role
    (c) Both (a) and (b) (d) All of these
12. Which is a virtual relation/table that is derived from one or more base relations?
    (a) trigger (b) cursor
    (c) view (d) None of these
13. The method of coding or scrambling of data is called as?
    (a) Encryption (b) Decryption
    (c) Cryptography (d) None of these
14. Which database is used to provide statistical information or summaries of values based on various criteria?
    (a) Relational (b) Statistical
    (c) Object (d) None of these

Answers

1. (a)	2. (b)	3. (d)	4. (c)	5. (a)	6. (c)	7. (b)	8. (b)	9. (b)	10. (a)
11. (b)	12. (c)	13. (a)	14. (b)						

Q.II Fill in the Blanks:

1. Database _____ involves the security of a database against unauthorized access.
2. A _____ is a type of rule or restriction placed on a database table.
3. A _____ constraint can be applied to a single attribute or a combination of many attributes in a tuple.
4. Entity integrity implies that a primary key cannot accept _____ value.
5. The goal of database _____ is the protection of data against threats such as accidental or intentional loss, destruction or misuse.
6. _____ refers to the trustworthiness of data or resources.
7. _____ is the selective restriction of access to database resource and includes security mechanisms in a database management system to protect against unauthorized access allows us to encapsulate a query and process each individual row at a time.
8. A _____ policy is a means of assigning access rights based on rules specified by users.
9. _____ command is used to provide access or privileges on the database objects to the users.
10. An _____ role is created by granting all the privileges necessary to run a given database application.
11. The _____ provides data security and is possible to grant each user the privileges to access the database only through a small set of views that contain the appropriate data for the user, thereby restricting and controlling each user's access to the database.
12. Data _____ technique is used to protect from threats.

13. A _____ database contains confidential information about individuals or organisations, which is used to answer statistical queries concerning sums, averages and numbers with certain characteristics.
14. _____ security means monitoring effectiveness of database security policy.

Answers

1. security	2. constraint	3. CHECK	4. NULL	5. security
6. Integrity	7. Access Control (AC)	8. DAC	9. GRANT	10. application
11. view	12. Encryption	13. statistical	14. auditing	

Q.III State True or False:

1. Data integrity refers to the validity, consistency, correctness and accuracy of data in a database and is represented in terms of integrity constraints.
2. Roles are a collection of privileges or access rights.
3. CREATE ROLE command used for creating roles.
4. A view is a database object that is of a stored query. A view can be accessed as a virtual table in PostgreSQL.
5. The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).
6. Domain integrity is related to the concept of foreign keys.
7. Statistical database security focuses on the protection of confidential individual values stored in so-called statistical databases and used for statistical purposes.
8. Data security refers to the protection of data against unauthorized access or corruption and is necessary to ensure data integrity.
9. Availability refers to the ability to use the information or resource desired.
10. MAC is a type of security access control that grants or restricts object access via an access policy determined by an object's owner group and/or subjects.
11. A role can be granted system or object priviledge.
12. Database recovery gives protection of a database against unauthorized access.

Answers

1. (T)	2. (T)	3. (T)	4. (T)	5. (T)	6. (F)	7. (T)	8. (T)	9. (T)	10. (F)
11. (T)	12. (F)								

Q.IV Answer the following Questions:

(A) Short Answer Questions:

1. What is database integrity?
2. What is database security?
3. What is role? Enlist its types.
4. Enlist purpose of roles.
5. Define the term encryption.
6. What is statistical database?
7. What is MAC? Define it. a hardware address that uniquely identifies each node of network.
8. What is DAC?
9. What is RBAC? Role Back Acces control
10. How to create view? Give syntax.
11. State two purposes of roles.

12. Define the term privilege.
13. What is database constraint?
14. Define the term statistical database.

(B) Long Answer Questions:

1. Explain the term constraints in detail.
2. Describe domain integrity constraints with example.
3. Define the terms database integrity and security. Also explain its importance in database.
4. Discuss the database security threats.
5. State different levels of security.
6. Explain role of DBA with respect to database security.
7. Explain various methods for database security in brief.
8. Discuss the types of privileges at the account level and those at the relation level.
9. Which account is designated as the owner of a relation? What privileges does the owner of a relation have?
10. How is the view mechanism used as an authorization mechanism?
11. What is meant by granting a privilege?
12. What is meant by revoking a privilege?
13. List the types of privileges available in SQL.
14. What is the difference between discretionary and mandatory access control?
15. What are the typical security classifications? Discuss the simple security property and the * property, and explain the justification behind these rules for enforcing multilevel security.
16. Describe the multilevel relational data model. Define the following terms: apparent key, polyinstantiation, filtering.
17. Explain encryptions techniques used for database security.
18. What are two advantages of encrypting data stored in the database?
19. What is a statistical database? Discuss the problem of statistical database security.

UNIVERSITY QUESTIONS AND ANSWERS

April 2015

1. What is database security? **(1 M)**
Ans. Refer to Section 3.2.
2. Define ciphertext. **(1 M)**
Ans. Refer to Page 3.17, Point (4)
3. What is referential integrity? **(1 M)**
Ans. Refer to Section 3.1.2.
4. Explain discretionary access control mechanism. **(5 M)**
Ans. Refer to Section 3.1.1.
5. What is view? Explain different statements in views. **(5 M)**
Ans. Refer to Section 3.4.

April 2016

1. Give the types of security classes used in mandatory access control method. **(1 M)**
Ans. Refer to Section 3.3.2.
2. Write a short note on encryption. **(5 M)**
Ans. Refer to Section 3.5.
3. State the commands used to generate and destroy a view with their syntax. **(2 M)**
Ans. Refer to Sections 3.4.2 and 3.4.3.

4. Discuss in short a revoke statement with its syntax.

Ans. Refer to Section 3.3.1.2, Point (2).

April 2017

1. State the different levels of security.

Ans. Refer to Page 3.5.

October 2017

1. What is database security?

Ans. Refer to 3.2.

2. Explain mandatory access control method.

Ans. Refer to Section 3.3.2.

3. Explain statistical database security.

Ans. Refer to Section 3.6.

April 2018

1. What is view?

Ans. Refer to Section 3.4.

2. Write a short note on mandatory access control method.

Ans. Refer to Section 3.3.2.

Syllabus ...

1. Relational Database Design Using PLSQL **(8 Hrs.)**

1.1 Introduction to PLSQL.
1.2 PL/pgSqL: Datatypes, Language Structure.
1.3 Controlling the Program Flow, Conditional Statements, Loops.
1.4 Stored Procedures.
1.5 Stored Functions.
1.6 Handling Errors and Exceptions.
1.7 Cursors.
1.8 Triggers.

2. Transaction Concepts and Concurrency Control **(10 Hrs.)**

2.1 Describe a Transaction, Properties of Transaction, State of the Transaction.
2.2 Executing Transactions Concurrently Associated Problem in Concurrent Execution.
2.3 Schedules, Types of Schedules, Concept of Serializability, Precedence Graph for Serializability.
2.4 Ensuring Serializability by Locks, Different Lock Modes, 2PL and its Variations.
2.5 Basic Timestamp Method for Concurrency, Thomas Write Rule.
2.6 Locks with Multiple Granularity, Dynamic Database Concurrency (Phantom Problem).
2.7 Timestamps versus Locking.
2.8 Deadlock and Deadlock Handling - Deadlock Avoidance (Wait-Die, Wound-Wait), Deadlock Detection and Recovery (Wait for Graph).

3. Database Integrity and Security Concepts **(6 Hrs.)**

3.1 Domain Constraints.
3.2 Referential Integrity.
3.3 Introduction to Database Security Concepts.
3.4 Methods for Database Security.
  3.4.1 Discretionary Access Control Method.
  3.4.2 Mandatory Access Control.
  3.4.3 Role Base Access Control for Multilevel Security.
3.5 Use of Views in Security Enforcement.
3.6 Overview of Encryption Technique for Security.
3.7 Statistical Database Security.

4. Crash Recovery **(4 Hrs.)**

4.1 Failure Classification.
4.2 Recovery Concepts.
4.3 Log base recovery Techniques (Deferred and Immediate Update).
4.4 Checkpoints, Relationship between Database Manager and Buffer Cache. ARIES Recovery Algorithm.
4.5 Recovery with Concurrent Transactions (Rollback, Checkpoints, Commit).
4.6 Database Backup and Recovery from Catastrophic Failure.

5. Other Databases **(2 Hrs.)**

5.1 Introduction to Parallel and Distributed Databases.
5.2 Introduction to Object Based Databases.
5.3 XML Databases.
5.4 NoSQL Database.
5.5 Multimedia Databases.
5.6 Big Data Databases.