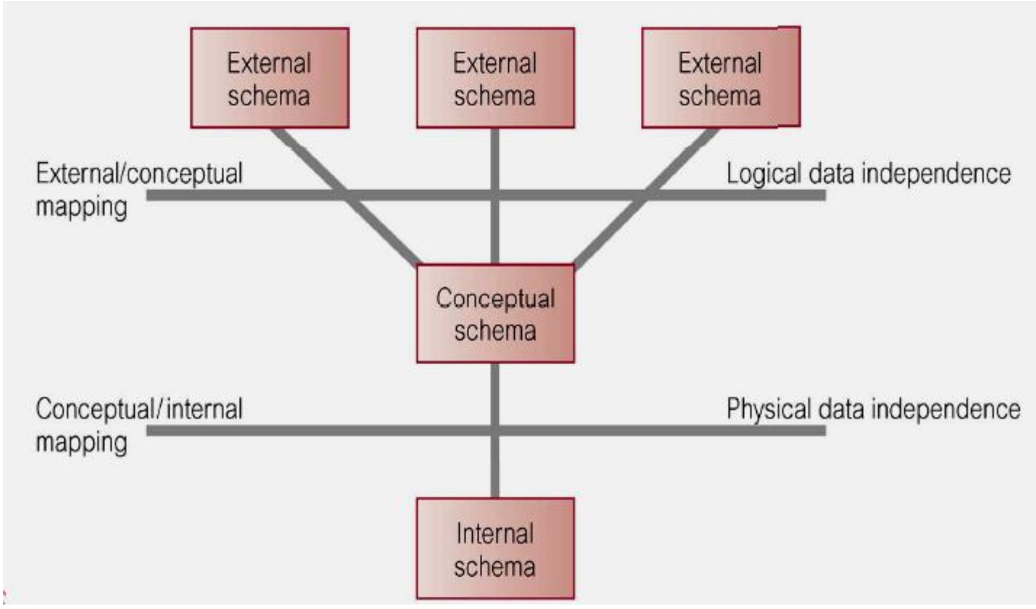
8. Database Views and Security

Database is consisted of various Database Constructs such as Tables, Domains, Views, Constraints, and Indexes. You can use DDL of SQL to define/create Views.

- ❑ CREATE VIEW – Add a View
- ❑ DROP VIEW – Remove a View

The Concept of View Tables was originated with the Three Level Architecture.


External schema
External schema
External schema
External/conceptual mapping
Logical data independence
Conceptual schema
Conceptual/internal mapping
Physical data independence
Internal schema

Conceptual Schema Consisted of Base Tables.
External Schema Consisted of View Tables.

Base/Source Tables in the Conceptual Schema	View Tables in the External Schema
Contained Actually Stored Data	Do not contain actual data. Only View Definition/Structure stored. Data of the View is retrieved from base tables when the query of the View executed.
Actual Tables	Virtual Tables
Occupy Storage Space.	Do not occupy any Storage Space.
The Tables defined to meet the need of the entire Company	Views defined for individual user needs.
Defined within a single schema.	Defined in various schemas.
Complex query needed to retrieve necessary data.	Query can be simplified.
Performance of Query Response is high.	Performance is low.

View can be derived by

□ Projecting from Base Tables or Another View. (Column Selection)
□ Selecting from Base Tables or Another View. (Row/Record Selection)
□ Both Projecting and Selecting from Base Tables or Another Views.
□ Joining Tables or another View.
□ Projecting and Selecting of Joining Base Tables or Another Views.
□ Deriving new attributes using SQL Functions.

```
CREATE  VIEW  Girls  AS
        SELECT  name, age, tpNumber  FROM  Student  WHERE  sex = ‘female’ ;
```

Name of the View – Girls
Columns – Same as the Selection Query. Domain/Data types are derived from the source table
Name of the source table – Student
Selection Query – SELECT name , age , tpNumber FROM Student WHERE sex = ‘female’

```
CREATE  VIEW  Girls ( gName, gAge , gTpNumber)  AS
        SELECT  name , age , tpNumber FROM Student WHERE sex = ‘female’ ;
```

Name of the View – Girls
Columns – Renamed set of columns (gName, gAge , gTpNumber)
Name of the source table – Student
Selection Query – SELECT name , age , tpNumber FROM Student WHERE sex = ‘female’

```
CREATE  VIEW  ViewName  AS  SubQuery  ;
CREATE  VIEW  ViewName (column1 , column2)  AS  SubQuery  ;
```

Sub Query – Any Selection Query which output a Table with one or more columns and one or more rows.

Example – 1 : Consider the following Base Tables,

Employee (empNo , empName , department)
Department (depNo , depName , headNo)

Create a View Table called “EmployeeOfExamDepartment” which defines only Employees in Exam Department.

```
CREATE  VIEW  EmployeeOfExamDepartment  AS
                    SELECT empName , department
                    FROM   Department, Employee
                    WHERE  department = depNo AND  depName = 'Exam' ;
```

Example – 2 : Consider the following Base Tables,

Employee (empNo , empName , depNo)
Department (depNo , depName , headNo)

Create a View Table called "EmployeeOfExamDepartment" which defines only Employees in Exam Department.

```
CREATE  VIEW  EmployeeOfExamDepartment AS
                    SELECT empName ,  Employee.depNo
                    FROM  Department, Employee
                    WHERE  Employee.depNo = Department.depNo AND  depName = 'Exam' ;
```

```
CREATE VIEW EmployeeOfExamDepartment AS
                    SELECT empName , E.depNo
                    FROM   Department D,  Employee E
                    WHERE  E.depNo = D.depNo  AND  depName = 'Exam' ;
```

Example – 1 : Consider the following Base Tables,

Employee (empNo , empName , Dno , Salary)
Department (Dnumber , Dname)

Consider the following View Table called "Dept_Info".

```
CREATE VIEW Dept_Info ( Dept_Name , No_Of_Emps ,  Total_Sal ) AS
                    SELECT  Dname ,  COUNT(*) ,  SUM(Salary)
                    FROM   Department ,  Employee
                    WHERE  Dnumber  =  Dno
                    GROUP  BY  Dname ;
```

Name of the View ....
Column Names of the View ....
Identify the sub query ....
....
....

Derived pattern
Select Project
Select and Project Joined
Joined and Select and Project Joined and Select and Project and
Functions

What are the joined tables? ....
What is the joined condition? ....
What is the Execution order? ....
....
What is the Description of the View? ....

Benefits of Views

Security

Protect data from unauthorized access.
Each user is given permission to access the database via only a small set of views that contain specific data the user is authorized to see.

Query Simplicity

Turning multiple table queries to single table queries against views, by drawing data from several tables.
It provides flexible and powerful data access capabilities.
It also improves productivity of end-user and programmers by:

– Simplifying database access by presenting the structure of data that is most natural to the user.
– Simplifying the use of routine and repetitive statements

Natural Interface

"Personalized" view of database structure that makes sense for the user.
Restructure or tailor the way in which tables are seen, so that different users see it from different perspectives, thus allowing more natural views of the same enterprise database.

Insulation from change

Logical Data independence - maintain independence among different user views and between each user view and the physical constructs.
A view can present a consistent image of the database structure, even if the underlying source tables are changed.

Change unit of measurement

Show age in months which is actually stored in Years in the DB.

Derive new attributes

Age is derived from the Date of Birth in the Database.

Limitation of Views

Update Difficulties

If a View has been constructed using a single base table, it will be easy to update the underline base table.
Hear, the Update Statement should be converted into an Update Statement against the Source Table.

CREATE VIEW Girls AS SELECT name , age , tpNumber FROM Student WHERE sex = 'female' ;

UPDATE Girls SET age = 23 WHERE name = 'Ganga' ; Should be converted into ,
UPDATE Student SET age = 23 WHERE name = 'Ganga' ; (Because actual data in the Student table)

If the View table has been constructed using several Source Tables in a Complex way, then it is difficult to do the UPDATE operation.
To accomplish the modification, you need to create series of SQL statements over base tables.
Which will be executed in a various ways (ambiguous) resulting unpredictable modifications.

Creating Migrating rows

When a row is altered such that it no longer satisfies WHERE condition then it will disappear from the view.
New rows will appear within the as a result of update or insert statement.
Rows that enter or leave a view are called migrating rows.

WITH CHECK OPTION clause of the CREATE VIEW statement prohibits a row migrating out of the view.

Ensures that if a row fails to satisfy WHERE clause of defining query, it is not added to underlying base table.

CREATE VIEW Emp_View AS SELECT * FROM Employee WHERE Dept = 'SAL' ;

UPDATE Emp_View SET Dept = 'FIN' WHERE Emp_no = 179 ;

Suppose that he worked in SAL department earlier. So he is in the above View.
He will be removed from the Emp_View as the updating change his department from SAL to FIN.
His record has migrated out due to update.
This can be avoided using the "WITH CHECK OPTION"
If any update affect the WHERE condition of the VIEW, it will not be allowed.

CREATE VIEW Emp_View AS SELECT * FROM Employee WHERE Dept = 'SAL' WITH CHECK OPTION

Then, it is impossible to do, ***UPDATE Emp_View SET Dept = 'FIN' WHERE Emp_no = 179 ;***

Reducing Performance
DBMS must translate queries against the view to queries against the source tables.
These disadvantages mean that we cannot indiscriminately define and use views instead of source tables.

How a View can Gives Data/ View Implementation.

Two main approaches have been suggested for efficiently implementing a view for querying.

- ❑ Query modification
- ❑ View materialization

Query modification
Involves modifying the view query into a query on the underlying base tables.

```
CREATE VIEW Girls ( gName, gAge , gTpNumber) AS
        SELECT name , age , tpNumber FROM Student WHERE sex = 'female' ;

SELECT gName FROM Girls WHERE gAge > 18 ;   should be converted into
SELECT name FROM Students WHERE age > 18 ;
```

The disadvantage of this approach is that it is inefficient for views defined via complex queries that are time consuming to execute.

View materialization
Involves physically creating a temporary view table in the Memory when the view is first queried and keeping the table on the assumption that other queries on the view will follow.
An efficient strategy for automatically updating the view table when the base tables are updated must be developed (incremental update)
The view is kept as long as it is being queried.

Security

What are the Threats to databases?

- ❑ Loss of integrity
- ❑ Loss of availability
- ❑ Loss of confidentiality

What are the Counter Measures that could be taken against above Threats?

- ❑ Access control
- ❑ Inference control
- ❑ Flow control
- ❑ Encryption

The DBA is the central authority for managing a database system.
DBA's responsibilities include granting privileges to users who need to use the system and classifying users and data in accordance with the policy of the organization.

DBA should do,

- ❑ Account creation
- ❑ Privilege granting
- ❑ Privilege revocation
- ❑ Security level assignment

A DBMS offers two main approaches to access control:

1. Discretionary/Optional access control

Govern the access of users to information on the basis of user's identity and predefined discretionary "rules" defined by the security administrator. The rules specify, for each user and object in the system, the types of access the user is allowed for the object

2. Mandatory access control

Govern the access to the information by the individuals on the basis of the classification of subjects and objects in the system.

Discretionary Access Control

The typical method of enforcing discretionary access control in a database system is based on the granting and revoking privileges.

The account level At this level, the DBA specifies the particular privileges that each account holds independently of the relations in the database.

CREATE SCHEMA or CREATE TABLE privilege, to create a schema or base relation.
CREATE VIEW privilege.
ALTER privilege, to apply schema changes such as adding or removing attributes from relations.
DROP privilege, to delete relations or views.
MODIFY privilege, to insert, delete, or update tuples.
SELECT privilege, to retrieve information from the database by using a SELECT query.

(Not defined as a part of SQL 2)

The relation (table level) At this level, the DBA can control the privilege to access each individual relation or view in the database.

1. SELECT (retrieval or read) privilege on R :
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.

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.

A user may be assigned all, none or all combination of these types of authorization.

In SQL2, 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.

Suppose that the DBA creates four accounts A1, A2, A3, and A4.
He wants only A1 to be able to create base relations.
Then the DBA must issue the following GRANT command in SQL,

GRANT CREATETAB TO A1 ;

In SQL2 the same effect can be accomplished by having the DBA issue a CREATE SCHEMA command as follows,

CREATE SCHEMA Earth AUTHORIZATION A1 ;

User account A1 can create tables under the schema called Earth.
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.
So, he can grant the privileges to other accounts.

GRANT UPDATE(Designation) ON Employee TO A2 , A3 ;

What is the table : ....
Who can grant the privileges : ....
To whom the privileges has granted: ....
What are the Privileges granted above : ....

GRANT SELECT, INSERT ON Employee TO A2 ;

What is the table : ....
Who can grant the privileges : ....
To whom the privileges has granted: ....
What are the Privileges granted above : ....

GRANT <privilege list> ON <relation or view> TO <user list>

REVOKE SELECT ON Employee FROM A2 ;
Now what are the privileges A2 have? ....

REVOKE UPDATE(Designation) ON Employee FROM A2 ;
Now what are the privileges A2 have? ....

REVOKE <privilege list> ON <relation or view> FROM <user list>

The revocation of a privilege from a user may cause other users also to lose that privilege.
This behavior is called cascading of the revoke.
The REVOKE statement may also specify RESTRICT.
In this case, an error is returned if there are any cascading revokes, and the revoke action is not carried out.

REVOKE SELECT ON Employee FROM A2 RESTRICT ;

Propagation of Privileges

Whenever the owner A of a relation R grants a privilege on R to another account B, the privilege can be given to B with or without the GRANT OPTION.

If the GRANT OPTION is given to B can also grant that privilege on R to other accounts.

GRANT SELECT ON Employee , Department TO B WITH GRANT OPTION ;
GRANT SELECT ON Employee TO C ; (This can be issued by the B as he has the Granting Power)

Example – 1

Suppose that A1 wants to give to A4 a limited capability to SELECT from the EMPLOYEE relation and wants to allow A4 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 has to create a view:

```
CREATE VIEW A4EMPLOYEE AS
        SELECT NAME, BDATE, ADDRESS
        FROM EMPLOYEE
        WHERE DNO = 5 ;
```

After the view is created, A1 can grant SELECT on the view A3EMPLOYEE to A4 as follows,
GRANT SELECT ON A4EMPLOYEE TO A4 WITH GRANT OPTION ;

Finally, 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 ;***

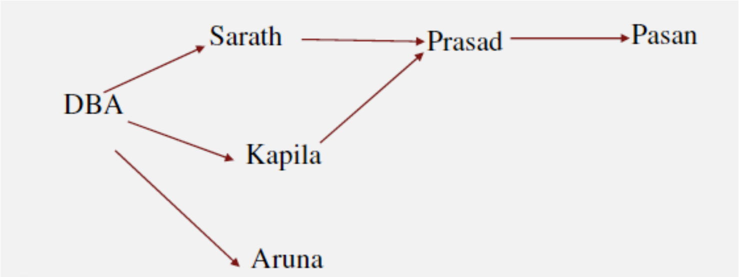
What are the privileges that the A4 have?
....
....

Example – 2

A bank maintains a database to keep track of customer, branch, account and loan information. Corresponding bank relational schema is given below.

Branch (BranchName , Code , City , Assets)
Account (AcctNo , Balance , Acc-type , BranchName)
Loan (LoanNo , Amount , Loan-type , BranchName)
Customer (CustomerNo , Name , Address , Phone)
C_A (CustomerNo , AcctNo)
C_L (CustomerNo , LoanNo)

Privileges are granted to users of the database as described in the questions given below and as shown in the authorization diagram.



1. User Sarath is able to retrieve the customer details of each customer at the Colombo branch along with his AcctNo and Balance and is able to update phone and address information of each customer. Write SQL statements to allow this.

   ....

2. Kapila is able to retrieve the name of each customer who has a loan but does not have an account at the bank.

   ....

3. Prasad is able to retrieve the following:

   - AcctNo, the name and the account balance of each customer who has a bank balance greater than Rs: 10,000 at the Colombo branch in such account and
   - LoanNo, the name and the loan amount of each customer at the Colombo branch who only has loans but no accounts at the bank.

   ....

4. To revoke the privileges of Sarath, the following command is issued

   REVOKE SELECT, UPDATE(Address, Phone) ON customer_account FROM Sarath RESTRICT ;

   Explain the impact of it on the users Prasad and Pasan.

   ....
   ....

RBAC - Role-based access control

Permissions are associated with roles, and users are made members of appropriate roles thereby acquiring the roles' permissions. This greatly simplifies management of permissions.

Roles are created for the various job functions in an organization and users are assigned roles based on their responsibilities and qualifications.

Users can be easily reassigned from one role to another.

Roles can be granted new permissions as new applications and systems are incorporated, and permissions can be revoked from roles as needed.

CREATE ROLE product_manager ;
CREATE ROLE hr_manager;

A layer of security can be added to roles by specifying a password, as in the following example:

CREATE ROLE overall_manager IDENTIFIED BY manager_password ;

Granting Privileges to Roles

Both system and object privileges can be granted to a role.
A role can be granted to another role.

GRANT SELECT, INSERT, UPDATE, DELETE ON Product TO product_manager ;
GRANT CREATE USER TO hr_manager ;
GRANT product_manager , hr_manager TO overall_manager ;

Identify Privileges of each Role ;

(1) product_manager :...
(2) hr_manager :...
(3) overall_manager :...

Granting Roles to a User

GRANT overall_manager TO A1 ;

Privileges of A1 : ....

To grant a role, you must
– have been granted the role with the ADMIN OPTION or
– have been granted the GRANT ANY ROLE system privilege or
– you must have created the role.

GRANT overall_manager TO A1 WITH ADMIN OPTION ;

Privileges of A1 : ....

Revoking a Role from a User
The following statement revokes the role overall_manager from the user A1:

REVOKE overall_manager FROM A1 ;

Revoking a System Privilege from a Role
The following statement revokes the CREATE TABLE system privilege from the overall_manager role:

REVOKE CREATE TABLE FROM overall_manager ;

Revoking a Role from a Role
To revoke the role hr_manager from the role overall_manager, issue the following statement:

REVOKE hr_manager FROM overall_manager ;

Dropping a Role

Specify the name of the role to be dropped. ***DROP ROLE hr_manager ;***

Setting a Role

A user who has been granted one or more roles has to invoke the SET ROLE command to enable or disable roles for the current user session.

If the role has a password, you must also specify the password to enable the role by using the IDENTIFIED BY clause.

Example 1 : To activate the role accountant having a password
SET ROLE accountant IDENTIFIED BY acct ;
Name of the Role .... Password ....

Example 2 : To disable all roles granted to you for the current session, issue the following statement:
SET ROLE NONE ;

Default Roles

It is possible to provide a facility to set up a default list of roles to be activated at the time of user login. This facility is enabled through the use of DEFAULT ROLE clause of the ALTER USER command.

Example : To make accountant role as the default active role for A1
ALTER USER A1 DEFAULT ROLE accountant ;

Example : To make all authorized roles for A1 part of his default active role set except the auditor role
ALTER USER Scott DEFAULT ROLE ALL EXCEPT auditor;

Example : To remove all roles from A1's default active role set
ALTER USER A1 DEFAULT ROLE NONE ;

Mandatory access control

This is an all-or-nothing method:
A user either has or does not have a certain privilege.
In many applications, 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.

Typical security classes are

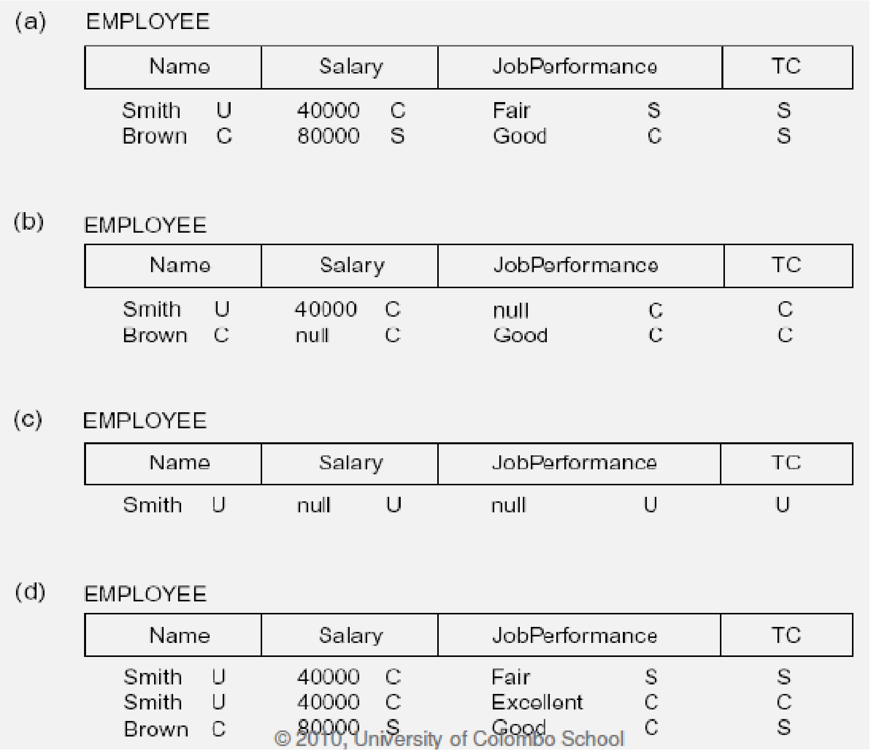
- □ Top secret (TS)
- □ Secret (S)
- □ Confidential (C)
- □ Unclassified (U)

Where TS is the highest level and U the lowest : $TS \geq S \geq C \geq U$

A multilevel relation schema R with n attributes would be represented as

R (A1 ,C1 ,A2 ,C2 , …, An ,Cn ,TC)

where each Ci represents the classification attribute associated with attribute Ai.



(a) EMPLOYEE

Name		Salary		JobPerformance		TC
Smith	U	40000	C	Fair	S	S
Brown	C	80000	S	Good	C	S

(b) EMPLOYEE

Name		Salary		JobPerformance		TC
Smith	U	40000	C	null	C	C
Brown	C	null	C	Good	C	C

(c) EMPLOYEE

Name		Salary		JobPerformance		TC
Smith	U	null	U	null	U	U

(d) EMPLOYEE

Name		Salary		JobPerformance		TC
Smith	U	40000	C	Fair	S	S
Smith	U	40000	C	Excellent	C	C
Brown	C	80000	S	Good	C	S

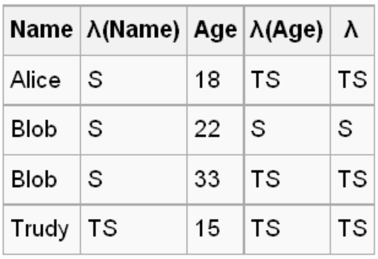
A multilevel relation will appear to contain different data to subjects (users) with different clearance levels.
In some cases, it is necessary to store two or more tuples at different classification levels with the same value for the apparent key. This leads to the concept of polyinstantiation where several tuples can have the same apparent key value but have different attribute values for users at different classification levels.

Polyinstantiation

Polyinstantiation is a database technique that allows the database to contain different tuples with the same key but with different classifications.
Polyinstantiation occurs because of mandatory policy.

In databases, polyinstantiation is database-related SQL (structured query language) terminology. It allows a relation to contain multiple rows with the same primary key; the multiple instances are distinguished by their security levels. It occurs because of mandatory policy. Depending on the security level established, one record contains sensitive information, and the other one does not, that is, a user will see the record's information depending on his/her level of confidentiality previously dictated by the company's policy.

Consider the following table, where primary key is Name and λ(x) is the security level:



Name	λ(Name)	Age	λ(Age)	λ
Alice	S	18	TS	TS
Blob	S	22	S	S
Blob	S	33	TS	TS
Trudy	TS	15	TS	TS

Although useful from a security standpoint, polyinstantiation raises several problems:

- Moral scrutiny, since it involves lying.
- Providing consistent views.
- Explosion in the number of rows.

Database Security (Inference Control)

Statistical databases are used mainly 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.

Statistical users are not allowed to retrieve individual data, such as the income of a specific person.
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 a sequence statistical queries.
This is particularly true when the conditions result in a population consisting of a small number of tuples.