Database is used in almost all applications. Keeping it secure is a prime concern of organisations to ensure that the sensitive data is not exposed, altered or destroyed.

Let us begin with some of the high-level database security requirements.

Note : This section assumes that you have sufficient understanding of basic database concepts. The focus of this section is purely on security aspects of database.

2.3.1 Database Security Requirements

The database security requirements can be categorised into the following.

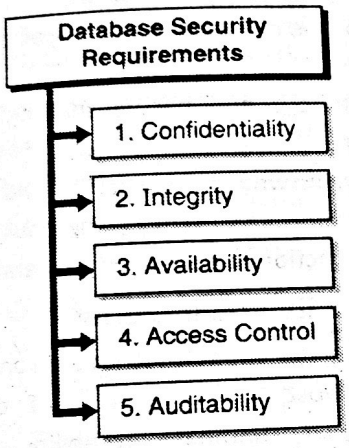

Fig. 2.3.1 : Database security requirements

1. Confidentiality

➢ **Definition :**
Confidentiality *of data ensures that it is not exposed to entities that do not have sufficient authorisation to view the data.*

- For example, who should be allowed to see your account details online? The banking application must ensure that it authenticates you and then lets you view only the account data that is tied to your account. Even if you are successfully authenticated to the banking application, you cannot see account details of any other user.
- There are several mechanisms using which confidentiality of data can be preserved. Some of them are as following.
  - (i) Data encryption
  - (ii) Database views
  - (iii) Access control and permissions

2. Integrity

- Integrity of database is perhaps the most crucial part of database security.

➢ **Definition :**
Integrity *of data ensures that the data stored in the database is not altered in any unauthorised way and its structure, value and meaning is preserved as it was desired to be.*

- The database should be protected from any damage arising out of physical, logical, structural or programmable aspects. For example, suppose your account has Rs.10,000 and you withdraw Rs.2,000 from ATM. As soon as the cash is dispensed, the power goes off. What would happen? If the bank's database is not updated with your transaction, it would still show that you hold Rs.10,000 in your account and bank would be in loss. It is crucial to ensure that the database records are accurate.
- There are several mechanisms using which integrity of data can be preserved. Some of them are as following.
  - (i) Save points
  - (ii) Two-phase commits
  - (iii) Backup and Restore
  - (iv) Error detection

3. Availability

- The database is crucial for delivering the data relevant to the user or the application.

➢ **Definition :**
Availability *of database ensures that the data contained in the database is usable when someone is required to use it.*

- If for some reason, the database is not available, the user would not be able to use the application. For example, you want to urgently make a payment for your online purchase, but the banking application is not available because there is some database maintenance activity going on. What do you do? Don't you feel frustrated?

- Similarly, if there are thousands of users requesting data at the same time, the database has to prioritize one request over the another or be capable of serving multiple requests in parallel. If the database response is slow, it could again lead to poor user experience.
- There are several mechanisms using which availability of database can be ensured. Some of them are as following.
  (i) High Availability
  (ii) Redundancy
  (iii) Replication
  (iv) Mirroring

4. Access Control

- By now, you understand the importance of access control.

> **Definition :**
> ***Access Control*** *in database ensures that only the entities authorised to use the required data from the database are allowed access.*

- Access Control is also one of the mechanisms to provide confidentiality and integrity of data in the database. Like OS, there are different users with different capabilities on database server. There could be administrators and general users. There could also be roles that define a pre-selected set of capabilities that someone assuming the role would have.
- For example, can the administrator of the banking database server increase the account balance amount for anyone? Can she delete the transaction records and rollback the database to a previous state? So, even though someone might be an administrator on the database, it does not mean that she has unrestricted rights over the data contained in the database. For example, the database administrator's job might be to just ensure that the database server is up and running, it is properly patched, and the desired network access is permitted. She may not have any rights on the data at all. How do you ensure this separation? Let's answer it later in the chapter. Access control in database could be little tricky.
- There are several mechanisms using which access control on database can be ensured. Some of them are as following.
  (i) Authentication and Authorization
  (ii) Roles and Permissions
  (iii) Data views
  (iv) Firewalls

5. Auditability

Auditability provides the capability of tracing a... evidences.

> **Definition :**
> ***Auditability*** *of database ensures that any access to... is appropriately logged and reported.*

- Audit trails are evidences and records of who has done... what on the database server. For example, if you find... that a table is missing from the database, then you can... check the audit logs and find out who has deleted the... table and when. Using audit logs, you can hold... someone accountable for her actions.
- Additionally, audit logs also help to maintain the... integrity of the database by logging all the actions that... are being taken on the database. You can periodically... review the logs and ensure that only the desired... actions are being carried out. If you find any... discrepancy in what you expected and what is being... done on the database system, you can take the... appropriate actions and restore the database integrity.

2.3.2 Reliability and Integrity

As you understand, database up time and accuracy are probably the most crucial aspects of delivering any application. Hence, database reliability and database integrity require special and careful attention. Let's begin with defining and understanding those terms and then diving deeper into understanding them.

> **Definition :**
> *The term* ***reliability*** *refers to the ability of a hardware or a software component to consistently perform according to its specifications.*

With respect to database,

> **Definition :**
> ***Database reliability*** *ensures that the database server can perform at the desired level consistently for a long time without requiring regular downtime and maintenance activities.*

Similarly, as you learnt earlier,

> **Definition:**
> *Integrity of data ensures that the data stored in the database is not altered in any unauthorised way and its structure, value and meaning is preserved as it was desired to be.*

2.3.2(A) Database Reliability and Integrity Requirements

From security perspective, database reliability and integrity requirements can be categorised as following.

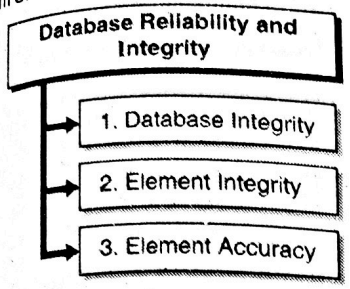

Fig. 2.3.2 : Database reliability and integrity

1. Database Integrity

Database integrity requires that the database maintains both the physical as well as the logical integrity. The OS, on which the database server is running, must ensure that the files related to the database are not accessed by unauthorised entities and also, they are free from disk errors and provided general OS level protection and correction mechanisms. Additionally, the logical structure of database, tables within it, permissions, roles, etc. must all be sufficiently preserved to avoid any integrity errors.

2. Element integrity

Element integrity requires that the values of various elements in the database tables are changed, updated or modified by only authorised entities. It is not altered outside the application (by directly connecting to the database) or by following random and unapproved processes. Setting up adequate access control mechanisms could ensure element integrity.

3. Element Accuracy

Element accuracy ensures that only correct and desired values are written to elements in the table. For example, if the balance field in the table can take numeric values, only numeric values should be allowed and not characters. Database itself enforces several integrity checks and constraints to ensure that elements are accurate.

(i) **Domain Constraints :** Domain constraints specify a valid set of values for an attribute. For example, age can only be numeric, and it cannot take character values.

(ii) **Entity Integrity Constraints :** The entity integrity constraint states that the primary key value cannot be null for any record.

(iii) **Referential Integrity Constraints :** Referential integrity ensures that all foreign keys have an existing and referenceable primary key.

(iv) **Primary Key Constraints :** Primary key in a table must be unique.

2.3.2(B) Database Integrity Protection Mechanism

Let's discuss some of the database integrity protection mechanisms.

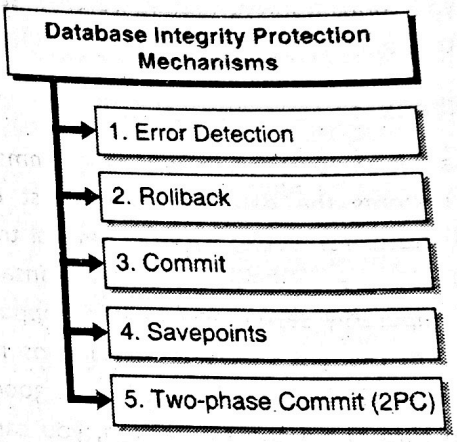

Fig. 2.3.3 : Database integrity protection mechanisms

1. Error Detection

There could be several errors that a database system can encounter.

Some of them could be:

(i) **Application errors :** Application wrongly updated the database

(ii) **Database internal errors :** Database had some data processing errors internally

(iii) **OS errors :** OS crashed during data processing

(iv) **Hardware errors :** Some hardware hosting the database failed during data processing

(v) **Network errors :** Database lost connectivity during data processing

(vi) **Disaster :** A natural disaster such as flood completely wiped off all data on the database server.

- Error detection mechanism such as Cyclic Redundancy Codes (CRC), hash values, parity bits etc. can be computed for crucial data. Such codes can be computed at the field level, table level or for the entire database. Before any read or write operation is carried out, these codes can be checked to ensure that the element value is accurate and has not been manipulated or altered from the last known good value (based on the codes).
- If and when errors are detected, the last known good value can be restored either from the backup or any other preferred mechanism of correcting data based on your organisation's preference such as processing the transaction logs.

2. Rollback

- Rollback is an operation (as well as a command in SQL) that restores the database to the last committed (good) state. You could carry out several transactions (updates) on a database, but those transactions are only temporarily saved. You have an opportunity to cancel the changes that the transactions made and restore the database to the last known good state. If you are sure about the transaction, you can commit (permanently save) the transaction changes into the database.

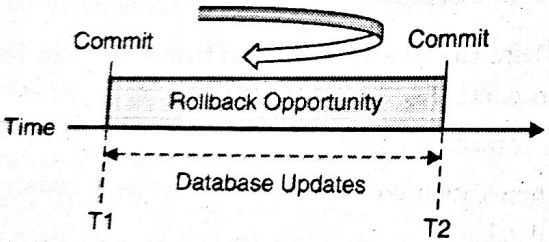

Fig. 2.3.4

- Rollback is immensely helpful if the transaction fails due to any errors or if the user herself cancels the transaction (for example, deciding to not purchase after putting thing in cart and then backing out from the payment page). Rollback helps to maintain the database integrity by cancelling out the undesired changes and restoring the good state.

Let's take an example.

```
INSERT INTO product VALUES(1, 'Biscuit');
INSERT INTO product VALUES(2, 'Tea');
INSERT INTO product VALUES(3, 'Coffee');
INSERT INTO product VALUES(4, 'Chocolate');
INSERT INTO product VALUES(5, 'Cake');

COMMIT;

SELECT * FROM product;

UPDATE product SET name = 'Ketchup' WHERE
id = '3';

SELECT * FROM product;

ROLLBACK;

SELECT * FROM product;
```

The output from the first SELECT statement would be as following.

Id	Name
1	Biscuit
2	Tea
3	Coffee
4	Chocolate
5	Cake

- The database state is committed with these values.
- Now, you make a change to the database entry. The output from the second SELECT statement would be as following.

Id	Name
1	Biscuit
2	Tea
3	Ketchup
4	Chocolate
5	Cake

- If this change is undesired, you can restore the previous state (last committed state) by issuing the ROLLBACK command. The third SELECT statement shows that the last good state is restored.

Id	Name
1	Biscuit
2	Tea
3	Coffee
4	Chocolate
5	Cake

3. Commit

- Commit is an operation (as well as a command in SQL) that permanently saves the changes made to the database. The time at which the changes are committed becomes a reference point for a known good state of the database. Any transactional changes (INSERT, UPDATE, DELETE) made after the commit until the next commit can be rolled back if undesired.

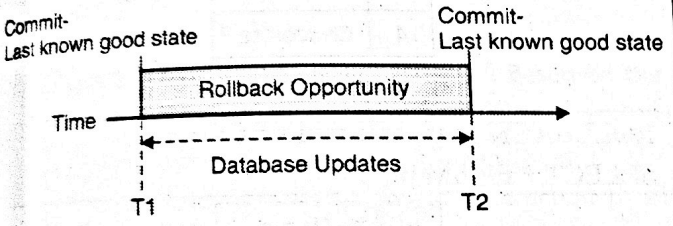

Fig. 2.3.5

- Commit ensures that the database is in good state and preserves the database state until a next good state is reached. This way the database integrity can be assured even if there are errors or undesired changes to the database. Once the changes are committed, the new values are available to all the applications and users.
- Let's take an example.

```
INSERT INTO product VALUES(1, 'Biscuit');
INSERT INTO product VALUES(2, 'Tea');
INSERT INTO product VALUES(3, 'Coffee');
INSERT INTO product VALUES(4, 'Chocolate');
INSERT INTO product VALUES(5, 'Cake');

COMMIT;

SELECT * FROM product;

UPDATE product SET name = 'Ketchup' WHERE id = '3';

SELECT * FROM product;

COMMIT;

SELECT * FROM product;
```

The output from the first SELECT statement would be as following.

Id	Name
1	Biscuit
2	Tea
3	Coffee
4	Chocolate
5	Cake

- The database state is committed with these values.
- Now, you make a change to the database entry. The output from the second SELECT statement would be as following.

Id	Name
1	Biscuit
2	Tea
3	Ketchup
4	Chocolate
5	Cake

- If the change looks good, you can COMMIT and save the change. The third SELECT statement then shows the changed value as per the last COMMIT.

Id	Name
1	Biscuit
2	Tea
3	Ketchup
4	Chocolate
5	Cake

4. Save points

- Save points let you keep reference placeholders for your changes so that you can go back to a particular state if desired.

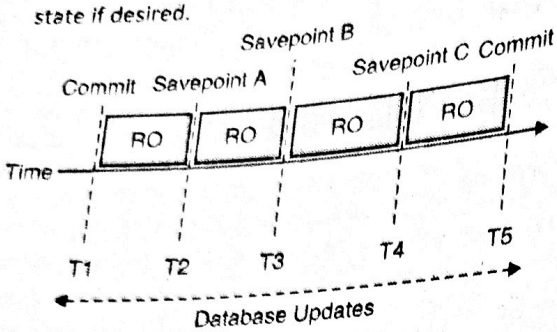

Fig. 2.3.6

- So, instead of one big rollback opportunity between two commits, save points provide several smaller rollback opportunities. This is very useful if you want to selectively rollback the changes instead of all the changes.

Let's see an example.

```
INSERT INTO product VALUES(1, 'Biscuit');
INSERT INTO product VALUES(2, 'Tea');
INSERT INTO product VALUES(3, 'Coffee');
INSERT INTO product VALUES(4, 'Chocolate');
INSERT INTO product VALUES(5, 'Cake');

COMMIT;

SELECT * FROM product;

UPDATE product SET name = 'Ketchup' WHERE id = '3';
SAVEPOINT A;

UPDATE product SET name = 'Juice' WHERE id = '3';
SAVEPOINT B;

UPDATE product SET name = 'Chips' WHERE id = '3';
SAVEPOINT C;
```

In this example, you have created three save points. Each save point can be considered as a reference placeholder to which you can take the database state to.

For example,

```
ROLLBACK TO SAVEPOINT A;
SELECT * FROM product;
```

Id	Name
1	Biscuit
2	Tea
3	Ketchup
4	Chocolate
5	Cake

```
ROLLBACK TO SAVEPOINT B;
SELECT * FROM product;
```

Id	Name
1	Biscuit
2	Tea
3	Juice
4	Chocolate
5	Cake

```
ROLLBACK TO SAVEPOINT C;
SELECT * FROM product;
```

Id	Name
1	Biscuit
2	Tea
3	Chips
4	Chocolate
5	Cake

Now, if you `ROLLBACK TO SAVEPOINT B` and then issue `COMMIT`, the database would save that change permanently.

```
ROLLBACK TO SAVEPOINT B;
COMMIT;
SELECT * FROM product;
```

Id	Name
1	Biscuit
2	Tea
3	Juice
4	Chocolate
5	Cake

Two-phase Commit (2PC)

When you carry out a transaction, often multiple databases are required to be updated.

Definition :

Two-phase Commit (2PC) *is a mechanism to ensure that in a transaction either all updates are carried out successfully or none.*

There is a coordinator that manages the transaction and commit process. The database (workers) follow instructions from the coordinator and either commit or rollback the transaction. The core objective is to ensure that all databases are synchronized (updated) with the transaction. No database is in a state where its data is inaccurate.

The two phases of 2PC are as following.

(i) **Phase 1 (Vote or Pre-commit phase) :** The coordinator sends a request to all workers asking if each one can agree to commit on the transaction. The workers either respond Yes or No.

(ii) **Phase 2 (Commit or Rollback Phase) :** Based on the response from each worker

(a) If all workers have responded as "Yes" : The coordinator issues a "Commit" instruction to all workers. All workers need to commit the changes and send acknowledgement to the coordinator.

(b) If any worker has responded as "No" : The coordinator issues a "Rollback" instruction to all workers. All workers need to rollback the changes and send acknowledgement to the coordinator.

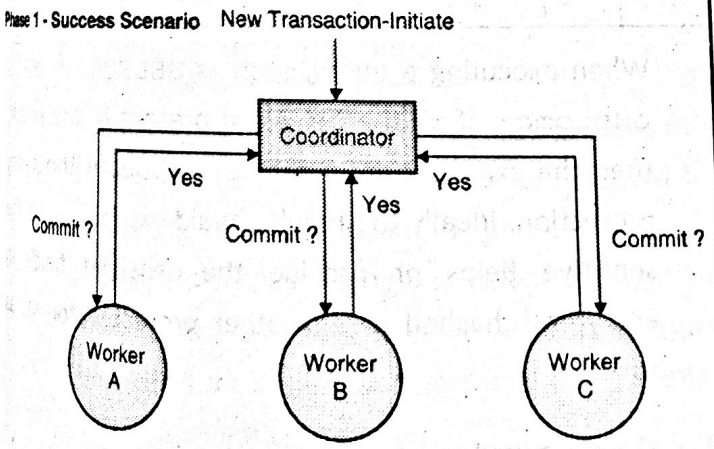

Fig. 2.3.7

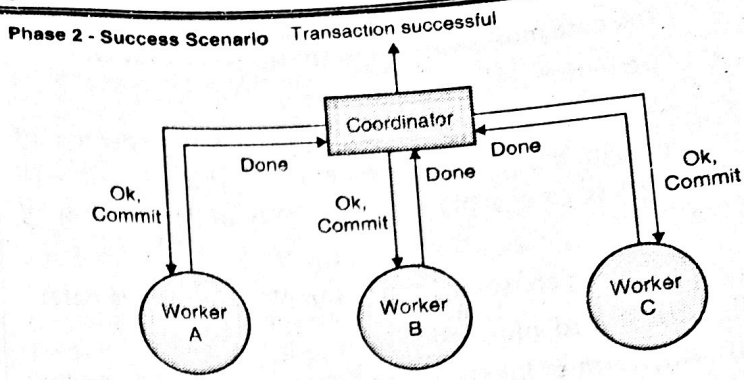

Fig. 2.3.8

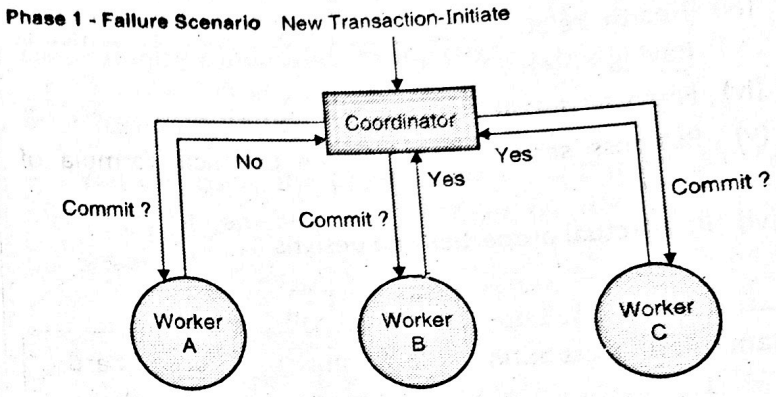

Fig. 2.3.9

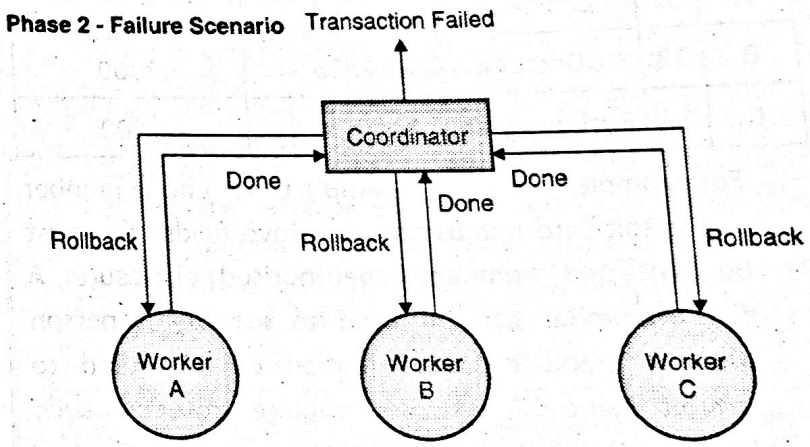

Fig. 2.3.10

This ensures that a transaction is considered complete and successful only if all the databases are updated and are consistent. If any of the database fails to update or be consistent, the transaction fails and any changes that occurred are rolled back.

2.3.3 Sensitive Data

Definition :

Sensitive data *is the information whose loss, misuse, unauthorised access or modification could adversely affect the national, organisational, societal or personal interest.*

The data must be protected as per its sensitivity.

> **Definition :**
> ***Sensitivity level of data*** *defines the degree of caution and protection that you need to exercise with respect to confidentiality, integrity and availability of data.*

Following are some of the examples of sensitive data:

(i) Credit card information
(ii) Any form of identification information such as Aadhar Card or Passport number
(iii) Health related information (for example, someone having HIV+)
(iv) Financial details
(v) Business secrets (for example chemical formula of Coke)
(vi) Intellectual properties and designs

Table 2.3.1

Name	Age	Occupation	Phone Number	Credit Card Number
A	32	Engineer	1234	9000
B	33	Doctor	1235	8000
C	34	Business man	2314	7000

- For example, in the given sample table, phone number and credit card number are sensitive fields that must be protected from any unauthorised disclosure. A phone number can be used to identify a person whereas credit card information can be used to commit financial frauds. Both require protection such that only the authorised application can use them as and when desired.
- Databases often store sensitive information along with non-sensitive information. If all information in a database is non-sensitive or all information in the database is sensitive, it is comparatively easier to either open up or lock down the entire database. When the database has mixed type of information (sensitive as well as non-sensitive) it becomes tricky to safeguard the sensitive information from disclosure – either directly or indirectly.
- It is important to understand the types of probable disclosures around sensitive data. Later you would see that how these types of disclosures can be made using various inference attack techniques.

2.3.3(A) Types of Disclosure of Sensitive Data

There could be several types of disclosure of sensitive data as shown in Fig 2.3.11.

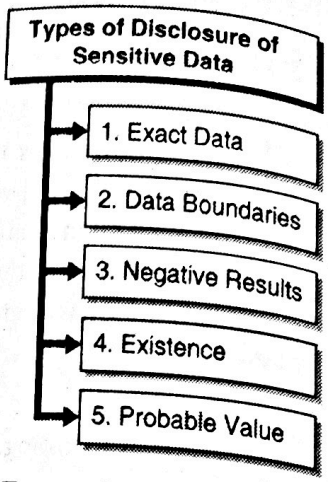

Fig. 2.3.11 : Types of disclosure of sensitive data

1. Exact Data

- Disclosure of exact sensitive data is most damaging. This could be because of errors or because of lack of adequate security controls around the protection of sensitive data. For example, consider the following sample table named `customer`.

Name	Age	Occupation	Phone Number	Credit Card Number
A	32	Engineer	1234	9000
B	33	Doctor	1235	8000
C	34	Business man	2314	7000

- When executing a query, such as `SELECT * FROM customer`, if all the fields are present in the result, then the exact sensitive data is disclosed without any protection. Ideally the result should explicitly omit the sensitive fields or provide the sensitive fields in encrypted, hashed or any other protected format as suitable.

2. Data Boundaries

In this type of disclosure, the actual data is not disclosed but the range of data could be disclosed. The range itself could give you a fair idea of how the distribution of various sensitive values could be in the actual table. Let's consider the following example.

Name	Income Tax	Income
A	1,00,000	10,00,000
B	2,00,000	20,00,000
C	3,00,000	30,00,000
D	5,00,000	50,00,000

Consider that the example table consists of records of employees and their respective income tax and income field is sensitive and is not disclosed to everyone. If you know that the minimum income tax paid by an employee is 1,00,000 and the maximum income tax paid by an employee is 5,00,000, you can fairly estimate the income range of employees of a company. Consider a flat tax of 10%, it would mean that the lowest paid employee earns around 10,00,000 whereas the highest paid employee earns around 50,00,000. This salary range disclosure could be sensitive information for a company. The competitor company could structure their employment offers around this information and attract talented employees.

3. Negative Results

Consider the following Table 2.3.2.

Table 2.3.2

Location	Weapon Count
A	0
B	10
C	200
D	50

If you can just find whether the count of weapons in a military location is either 0 or not 0, that itself could be sensitive disclosure. The actual count of weapons does not matter here. A location not having any weapons might be easier to target and attack. Negative result disclosure proves or disproves an assumption and can help to infer (guess) sensitive information.

4. Existence

Very similar to negative results, just the existence or non-existence of a record or a field could itself be a sensitive information. Consider the following table.

Location	Missiles
A	10
B	10
C	200
D	50

Just knowing that only the locations *A*, *B*, *C* and *D* are present in the missiles table, it could mean that other locations such as *E* and *F* do not have missiles yet. The existence of location information itself is sensitive and the missile count, even though is sensitive, may not be the only sensitive information worth protecting.

5. Probable Value

- You can query the database to know the probability value of an assumption. Consider the following example.

Name	Preference	Political Party
A	Coffee	X
B	Coffee	Y
C	Tea	X
D	Tea	X

- Your goal here could be to determine how many people are with a particular political party. The association of an individual name with a political party is sensitive and protected information but the association of preference and political party is not. Now, you can execute the following queries.

```
SELECT COUNT(*) FROM people WHERE
preference = 'Coffee';
SELECT COUNT(*) FROM people WHERE
preference = 'Coffee' AND party = 'X';

SELECT COUNT(*) FROM people WHERE
preference = 'Tea;
SELECT COUNT(*) FROM people WHERE
preference = 'Tea' AND party = 'X';
```

- The result of respective queries is 2, 1, 2, 2. So, you can safely assume the following probable values
  - Around 50% of people who prefer coffee are with the party X
  - Majority of people who prefer tea are with the party X

- There is a 50-50 chance of someone preferring coffee or tea

– Now, assume that there are 1,000 records in the table. Based on the probability values you inferred earlier, the following statements look to be very convincing, isn't it?

- Around 500 people would prefer coffee
- Around 200 of coffee drinking people would be with party X
- Around 500 people would prefer tea
- Around 400 of tea drinking people would be with part X
- So, 200 + 400 = 600, means that around 60% of the people are with political party X

– This value of 60% probability of a political party association could be sensitive but could be easily disclosed.

2.3.4 Inference Attacks

Inference refers to the ability to deduce information (guess or logically arrive at a conclusion).

➢ **Definition :**

Inference attacks *target non-sensitive information to deduce sensitive information.*

The attack analyses several information and combination of facts associated with the sensitive information to make smart guesses about the sensitive information. You have already learnt about the several types of disclosure that are possible for sensitive data in the previous section. Let's now learn about inference attacks.

2.3.4(A) Types of Inference Attacks

At a high level, there are three types of inference attacks that can be done on databases to reveal (or infer) sensitive information.

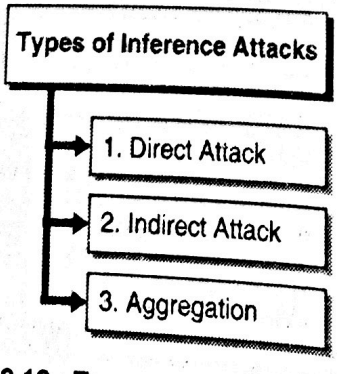

Fig. 2.3.12 : Types of inference attacks

1. Direct Attack

➢ **Definition :**

Direct attacks *try to reveal the sensitive information by executing direct queries on the database.*

The queries are formed such that it is not-obvious that the attacker is trying to get sensitive information. Let's take an example.

Name	Age	Gender	Role	Promotion
A	32	M	Designer	No
B	33	F	Programmer	Yes
C	34	F	Manager	No
D	31	M	Architect	No
E	35	M	Programmer	Yes
F	34	M	QA	No

```
SELECT name FROM employee WHERE promotion='Yes';
```

The database might reject the query because the attacker is trying to access the sensitive field and the employee names, whose promotions are due, must not be revealed.

Let's modify the query little bit.

```
SELECT name FROM employee WHERE (gender = 'M' AND gender = 'F') OR promotion='Yes' OR Age > 50 OR name = 'X';
```

In this query, the database might assume that the query would result into several records (names – both who are due for promotion and who are not due for promotion) and the list of names would not provide the sensitive information about who is about to get promoted. But that does not actually happen. The query still results into only revealing names who are due for promotion because other conditions in the query evaluate to false.

– A gender cannot be a male and a female at the same time

– There are no employees aged above 50

– Neither there is an employee whose name is X

The sensitive information might be revealed by making the query more complex and assuming that the produced results are not sensitive enough to be protected.

2. **Indirect Attack**

➢ **Definition :**

Indirect inference attacks *do not actually get the sensitive data but makes guesses based on facts regarding the data.*

It could be based on statistical approaches that you learnt earlier – sum, count, mean, median, etc. The examples are disclosures such as

- Data boundaries
- Negative results
- Existence and
- Probable values

3. **Aggregation**

➢ **Definition :**

Aggregation *is the collection of various facts associated with the data to make the guessed information (inference) more realistic.*

- Note here that unlike other attacks which are purely done on databases, aggregation can be done outside the database. The results from database are correlated with other information that the attacker might have about the data, environment, people or system.
- For example, the employee vacation database might not show the vacation details (from which date to which date the employee was absent from work), but her Facebook account details might have status updates, photos and other comments regarding the vacation. The attacker could then use this information to carry out attacks in the absence of employee.
- Another example could be that a general military clerk may not have information about how many soldiers are getting placed where, but she might have food or shelter arrangement tasks assigned that would indirectly exhibit the number of people who are likely to be stationed at a base.
- Aggregation is increasingly becoming powerful as more and more companies collect your online activities, locations where you go, your purchase choices and other consumer behaviour. The companies then use your data to sell customized services to you or for other purposes and motives.

2.3.4(B) Protection against Inference Attacks

There are several ways to protect from inference attacks.

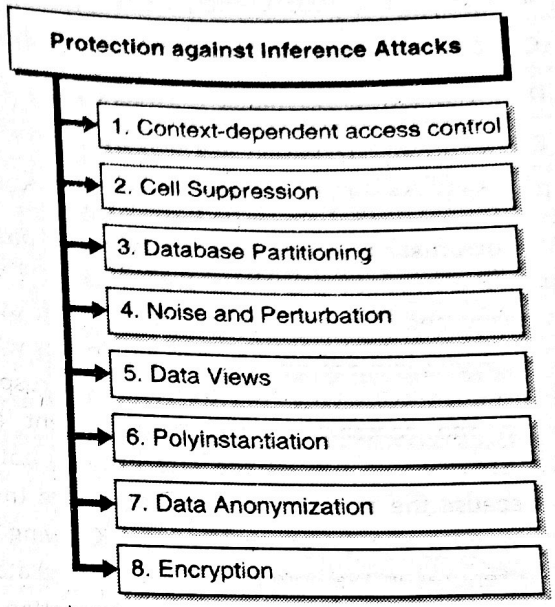

Fig. 2.3.13 : Protection against inference attacks

1. Context-dependent access control

– Databases are content driven, meaning that they respond with the content (records) as queried. No active state of the system is maintained to decide on returning the records based on what has been previously accessed or shared.

➢ **Definition :**

Context-dependent access control *is a sensitive data protection technique where the previous state of access requests is maintained and the future access decisions are taken based on what the user already knows from the previous access requests.*

– Based on the context (situation), further information access decisions are taken. If the user already knows enough, such that letting her to know more information would lead to inferring or disclosing sensitive data either directly or indirectly, then the further information is not provided to the user.

Let's take an example.

Name	Age	Gender	Role	Tenure	Promotion
A	32	M	Designer	2 years	No
B	33	F	Programmer	3 years	Yes
C	34	F	Manager	6 months	No
D	31	M	Architect	2 years	No
E	35	M	Programmer	7 years	Yes
F	34	M	QA	1 year	No

Suppose that the attacker knows from the past data that programmers get a promotion around their 3rd year in the company and 7th year in the company. The attacker might try to first find out the name of employees who are programmers and then try to find out their respective tenures in the company. A context-dependent access control system would not let the attacker run both the queries because the attacker already knows from the first query that who the programmers are. Knowing their respective tenures in the company would let the attacker infer who all are due for promotion. Promotion is a sensitive field and its inference needs to be protected as well.

2. Cell Suppression

➢ **Definition :**

Cell Suppression *is a sensitive data protection technique where the table cells containing the potentially sensitive information are hidden.*

- The table cells that contain sensitive information or that can make the inference possible are hidden. Let's take an example.

Sales (units sold)	Asia	America	Europe	Total
Product A	10	25	35	70
Product B	20	75	80	175
Product C	30	10	10	50
Total	60	110	125	295

- It is important to understand that which cells should be suppressed such that the overall sales data for each product and each region is not disclosed.

Consider the below cell suppression pattern.

Sales (units sold)	Asia	America	Europe	Total
Product A	10	25	35	70
Product B	-	-	-	-
Product C	30	10	10	50
Total	60	110	125	295

- Can you still figure out the suppressed values? Yes, t... quick arithmetic. For example, if the total sales in A... is 60 units, then product B must have sold (60 – 30 – 10) 20 units.
- So, it is important to understand which cells should ... actually suppressed to avoid revealing the sensitive information. Now, consider the below cell suppressio... pattern. Is this better? Can you determine sales pe... product in each region?

Sales (units sold)	Asia	America	Europe	Total
Product A	10	-	35	-
Product B	-	-	-	175
Product C	-	10	-	-
Total	60	110	125	295

3. Database Partitioning

➢ **Definition :**

Database Partitioning *is a sensitive data protection technique where the table containing the sensitive information is split into multiple tables such that the attacker would have to know all the related tables to infer the sensitive data.*

The problem with keeping both the insensitive and sensitive data together in a table is that the attacker has multiple ways to infer sensitive information either directly or indirectly. It is hard to protect the sensitive data against manipulated queries in a mixed (insensitive with sensitive data) table. Database partitioning splits the sensitive and other related data into multiple tables to make it hard for the attacker to make inference attack on just one table.

Let's see an example. Consider the following table.

ID	Name	Age	Gender	Role	Tenure	Promotion
E001	A	32	M	Designer	2 years	No
E002	B	33	F	Programmer	3 years	Yes

ID	Name	Age	Gender	Role	Tenure	Promotion
E003	C	34	F	Manager	6 months	No
E004	D	31	M	Architect	2 years	No
E005	E	35	M	Programmer	7 years	Yes
E006	F	34	M	QA	1 year	No

The table can be partitioned as the following.

ID	Name	Age	Gender
E001	A	32	M
E002	B	33	F
E003	C	34	F
E004	D	31	M
E005	E	35	M
E006	F	34	M

Role ID	Role
R001	Designer
R002	Programmer
R003	Manager
R004	Architect
R005	QA

ID	Role ID
E001	R001
E002	R002
E003	R003
E004	R004
E005	R002
E006	R005

ID	Tenure	Promotion
E001	2 years	No
E002	3 years	Yes
E003	6 months	No
E004	2 years	No
E005	7 years	Yes
E006	1 year	No

The attacker would now require working with four different tables and the ability to carry out more detailed analysis for inference. If the Role ID to Role mapping table is completely blocked access to along with the Employee ID to Tenure and Promotion mapping table, it becomes very hard to infer anything from the other two tables. You have split the insensitive data and sensitive data and can now apply access controls more rigorously to avoid inference attacks.

4. Noise and Perturbation

➤ **Definition :**

Noise and Perturbation *is a sensitive data protection technique where fake information (records) is inserted into the table to make the inference misleading.*

The bogus data inserted into the table makes inference non-conclusive or uncertain. It rather confuses the attacker whether the data is accurate enough to be trusted.

Let's take an example.

Name	Age	Gender	Role	Tenure	Promotion
A	32	M	Designer	2 years	No
B	33	F	Programmer	3 years	Yes
K	29	F	Programmer	8 years	Yes
C	34	F	Manager	6 months	No
D	31	M	Architect	2 years	No
L	44	M	Programmer	4 years	No
E	35	M	Programmer	7 years	Yes
F	34	M	QA	1 year	No

The highlighted records in the given example are fake records. Programmer *L*, even though has spent 4 years in the company, is not chosen for promotion. Programmer *K*, with the age of just 29, is shown 8 years in the company and ready for promotion. There is no clear inference to make here.

5. Data Views

➤ **Definition :**

Data View *is a sensitive data protection technique where the data is logically partitioned as per the access requirements for various groups.*

– Data Views are created on top of existing tables to contain only rows and columns that are specifically allowed for a group or an entity.

– Let's take an example. Consider that the following Table 2.3.3 is an employee salary database

Table 2.3.3 : Employee salary

Manager Name	Employee Name	Salary
A1	A	S1
	B	S2
	C	S3
B1	D	S4
	E	S5
	F	S6
	G	S7
	H	S8
C1	J	S9
	K	S10

Each manager is only allowed to view the salary of their direct staff and not others in the company. So, each manager has access to the partial records (records of her direct reports only) and not all the records in the entire table. But HR and Payroll is allowed to see all the records because they need to process salary every month for each employee. So, basically, you have 5 groups that have varied access requirements for the same table –

(i) **HR** : requiring full access

(ii) **Payroll** : requiring full access

(iii) **Manager A1** : requiring access to employees A, B and C

(iv) **Manager B1** : requiring access to employees D, E, F, G and H

(v) **Manager C1** : requiring access to employees J and K

So, you can create one database view for each manager.

Manager Name	Employee Name	Salary
A1	A	S1
	B	S2
	C	S3

Manager Name	Employee Name	Salary
B1	D	S4
	E	S5
	F	S6
	G	S7
	H	S8

Manager Name	Employee Name	Salary
C1	J	S9
	K	S10

This way, the respective managers can only access records of their staff and no one else.

6. Polyinstantiation

Definition :

***Polyinstantiation** is a sensitive data protection technique where what data to show is explicitly decided based on the entity that is requesting access.*

Simply put, you can restrict access to sensitive data. But a direct restriction may give an attacker a clue that there is something which is sensitive and hence it is restricted. The attacker then might try other ways of disclosing the data. To avoid this, when an attacker queries the data, she is shown manipulated records and not the real ones. This satisfies the attacker to some extent by assuring her that her actions are actually resulting into the sensitive data disclosure.

Polyinstantiation thus allows creating multiple instances of data as per the entity. It could also create instances as per the sensitivity labels that you learnt in the MAC-based access policy. Let's see an example.

Sensitive Label	Name	Age	Gender	Role	Tenure	Promotion
Confidential	A	32	M	Designer	2 years	No
Unclassified	A	32	M	Designer	4 years	Yes
Confidential	B	33	F	Programmer	3 years	Yes
Unclassified	B	33	F	Programmer	1 year	No

So, an entity that has a clearance level of Confidential would see the actual records of the employee.

Sensitive Label	Name	Age	Gender	Role	Tenure	Promotion
Confidential	A	32	M	Designer	2 years	No
Confidential	B	33	F	Programmer	3 years	Yes

An entity that does not match the Confidential level and is below it, would see manipulated records of the same employee.

Sensitive Label	Name	Age	Gender	Role	Tenure	Promotion
Unclassified	A	32	M	Designer	4 years	Yes
Unclassified	B	33	F	Programmer	1 year	No

7. Data Anonymization

➢ **Definition :**

Data Anonymization *is a sensitive data protection technique where the sensitive information (usually personal information) from the data is removed or randomized.*

- These days a lot of companies use machine learning and artificial intelligence technology to make meaningful inferences to predict the consumer behaviour or future. The technology could be used to predict chances of someone getting diagnosed with cancer or predicting match result of a football game. Large datasets, from the past studies or results, are collected to train the machine learning models so that the models can learn the past patterns and use those patterns for future prediction.
- To ensure that such datasets do not provide personal information such that to expose someone's privacy, the identity information from the data is taken out (removed) or replaced with something that is meaningless or not so obvious. It then becomes difficult to infer or pinpoint individuals behind the data and the data can be safely used for training purposes.
- Let's take an example. The following Table 2.3.4 shows the medical records of a group of people.

Table 2.3.4

Name	Age at which first consumed tobacco	Age at which diagnosed with cancer	Died at the age of
A	22	37	41
B	17	28	32
C	16	47	51
D	34	41	46

- For research purpose, the name of the person is irrelevant. Data anonymization removes this information or replaces it with something else so that individual information is not disclosed.

Age at which first consumed tobacco	Age at which diagnosed with cancer	Died at the age of
22	37	41
17	28	32
16	47	51
34	41	46

Name	Age at which first consumed tobacco	Age at which diagnosed with cancer	Died at the age of
R001	22	37	41
R002	17	28	32
R003	16	47	51
R004	34	41	46

8. Encryption

➢ **Definition :**

Encryption *is a sensitive data protection technique where the sensitive information is randomized to make it unreadable by unauthorised entities.*

- You must already be aware of encryption from your prior courses. Encryption ensures that the data, that is to be protected, is not stored in plaintext and is rather stored in cipher text (as encrypted). So, even if the encrypted text is revealed, it does not cause much harm because it is useless until decrypted with the right key to provide the real information.
- So, the Table 2.3.4 from previous examples, could possibly encrypt and store the sensitive data as shown in the following Table 2.3.5.

Table 2.3.5

Name	Age	Gender	Role	Tenure	Promotion
A	32	M	Designer	2 years	Weg2e212763t
B	33	F	Programmer	3 years	Rgrngooi22288
C	34	F	Manager	6 months	Fef3ijfi912e222

Comparison between Inference Protection Techniques

Table 2.3.6

Technique	Complexity	Performance	Protection	Used
Context-dependent	Very High	Low	Medium	Less Frequently
Cell Suppression	Very High	Low	Medium	Less Frequently
Database Partitioning	Low	High	High	Frequently
Noise	High	Low	Medium	Less Frequently
Data Views	Low	High	High	Most often
Polyinstantiation	Medium	Medium	High	Less Frequently
Encryption	Medium	High	Very High	Most often

2.3.5 Multilevel Database Security

Similar to the MAC-based access model that you learnt earlier,

➢ **Definition :**

Multilevel databases *store data that have different sensitivity requirements.*

- A particular cell in the table might be more sensitive than the entire record.
- The values resulting from the arithmetic operations on the individual values in the table might be more sensitive than the individual values themselves.
- Categorizing the data as either sensitive or insensitive may not be sufficient.
- Multilevel databases allow you to assign specific sensitivity labels to the data. The sensitivity label might be for a table, for a record or for a cell. Based on the clearance level of the entity requesting access, the access decisions are evaluated and taken.

2.3.5(A) Approaches for Designing Multilevel Databases

There are several approaches for designing multilevel databases.

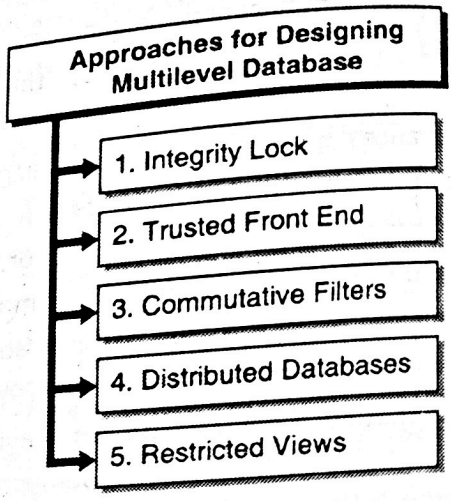

Fig. 2.3.14 : Approaches for designing multilevel database

1. Integrity Lock

➢ **Definition :**

Integrity Lock mechanism *for providing multilevel database security assigns the sensitivity label to each data item.*

Based on the clearance level of the entity requesting access, the access decisions are taken. The sensitivity label is protected from getting changed or disclosed.

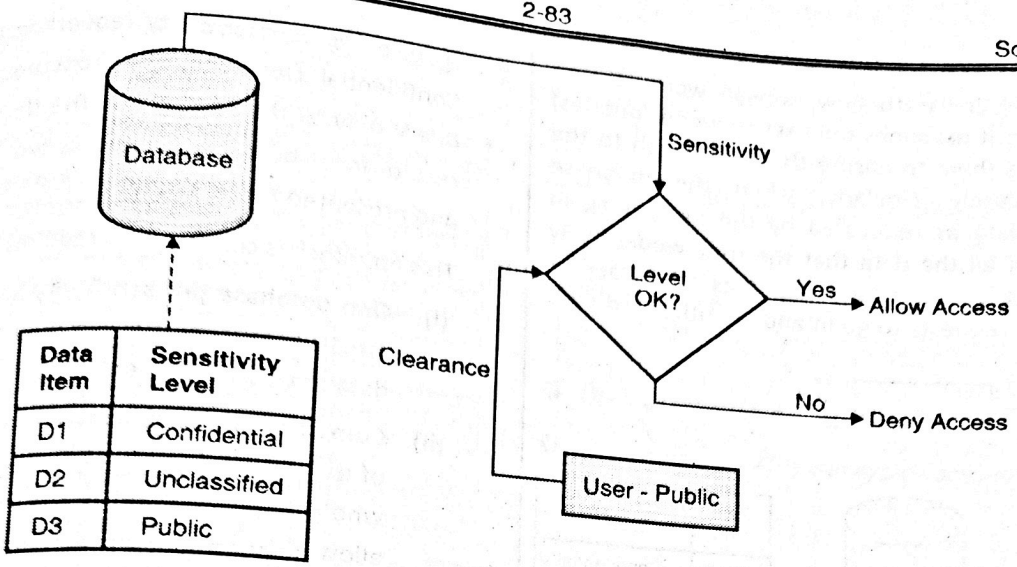

Fig. 2.3.15

The problem with this approach is that it severely impacts performance.

- Each access requests requires comparing the sensitivity level with clearance level
- Extra space is required to store the sensitivity level itself for each data item
- Sensitivity level itself requires protection from disclosure and alteration

2. Trusted Front End

Definition :

Trusted Front End mechanism *for providing multilevel database security relies on an external system to validate access decisions and control access to the sensitive data.*

- Databases are best suited for storing and providing information. Implementing security for each data item might add a significant overhead in terms of operations. The databases are not often used as a standalone system. Rather they are used with applications that provide a user interface where the user interacts with the required data.
- For example, you do not directly interact with the Flipkart database for your purchase. Instead you go through the Flipkart portal.

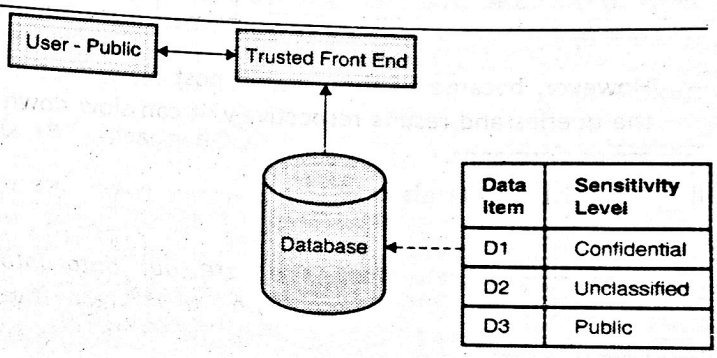

Fig. 2.3.16

- Front end is the application that actually interacts with the database and the items stored therein. The database relies on this front end to make good access decisions based on the user or entity that it is serving. This way, there is no need to carry out access decisions at the database layer. Instead sensitivity-based access control decisions can be taken right from the front-end application. All the logic for evaluating access decisions is built in the front end. The database performance is not hit.

3. Commutative Filters

Definition :

Commutative filters *examine both incoming requests and outgoing data to enforce appropriate access control.*

- It works very similar to how firewall works at the network layer. It examines the user requests (queries) and reformats them to ensure that they reach to the database securely. Similarly, when the database returns the data as requested by the user, it again examines that all the data that the user would likely see is actually permitted. It does not permit inappropriate requests to go in and unauthorised data to come out.

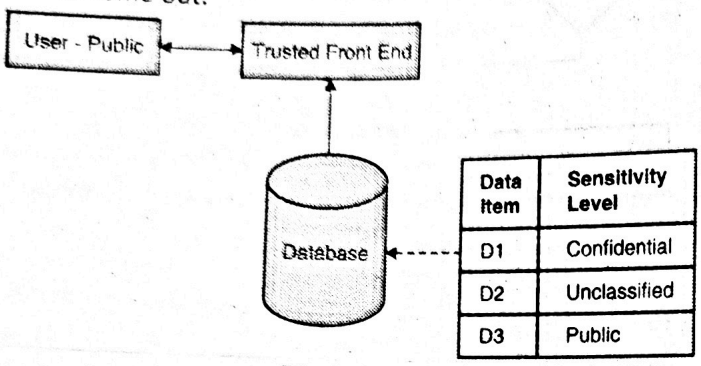

Fig. 2.3.17

- However, because of the pre and post processing of the queries and results respectively, it can slow down the performance.

4. Distributed Databases

➤ **Definition :**

***Distributed databases separate** out data into multiple databases based on their respective sensitivity levels.*

- The data is stored as per the respective sensitivity level of each data item. Front end has to build logic to query the appropriate databases based on the entity requesting the data.
- For example, consider that there are three databases.
  - **(i) Most Sensitive :** Top Secret
  - **(ii) Medium Sensitive :** Confidential
  - **(iii) Insensitive :** Public

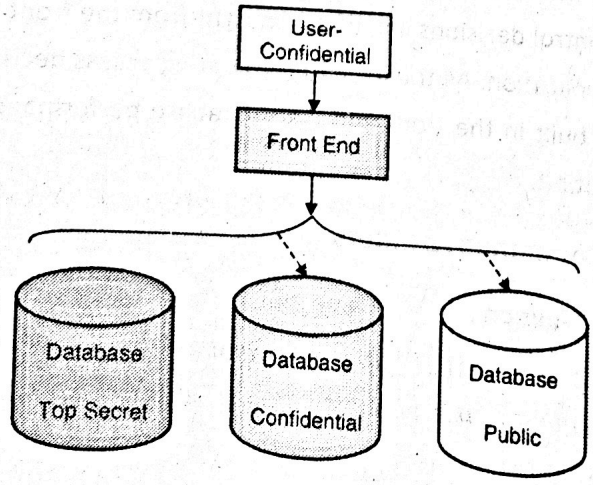

Fig. 2.3.18

- Now, if there is an entity requesting access that ... Confidential clearance, the front end would su... query only to the second and the third database ... results from both the databases would be comb... and presented to the entity.
- This approach is complex and requires
  - **(i) One database per sensitivity level :** So, if th... are 10 sensitivity levels, it would require split... data into 10 databases.
  - **(ii) Complex Front end :** Front end has to build a ... of logic to re-format and design queries based ... who is requesting access and which databases ar... allowed to be queried as per the clearance lev... of the entity.

5. Restricted Views

- You learnt earlier how database views can be used ... protect against the inference attack. Similarly, you ca... create views based on sensitivity levels.

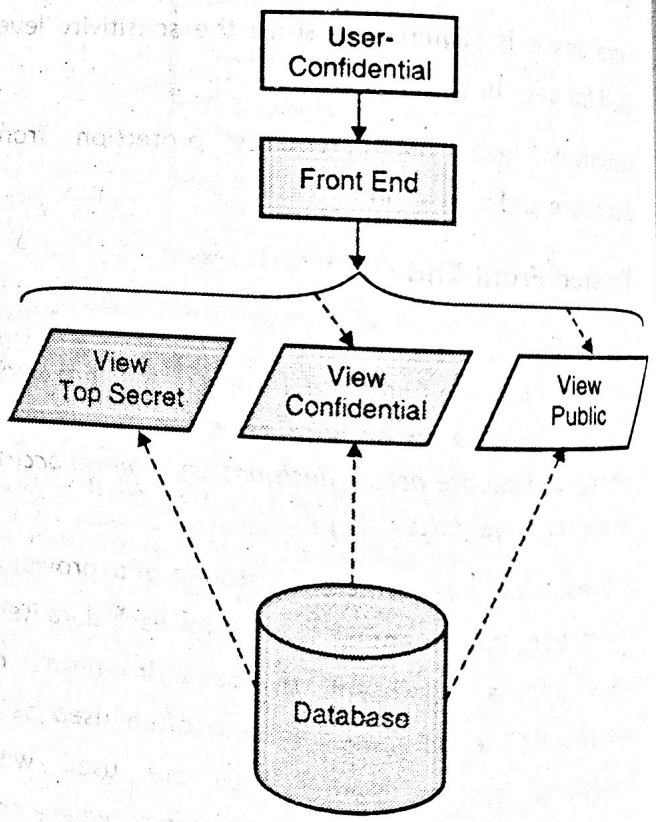

Fig. 2.3.19

- The entity requesting access to the data would be tied ... to the views that are allowed as per clearance level ... the entity.

Comparison between approaches for designing multilevel databases

Sr. No.	Approach	Complexity	Performance
1.	Integrity Lock	Medium	Low
2.	Trusted Front End	Low	High
3.	Commutative Filters	Medium	Medium
4.	Distributed Databases	High	Medium
5.	Restricted Views	Medium	High

Review Questions

Here are a few review questions to help you gauge your understanding of this chapter. Try to attempt these questions and ensure that you can recall the points mentioned in the chapter.

Software Vulnerabilities

Q.1 With a simple code example, explain buffer overflow. **(6 Marks)**

Q.2 You are reviewing code written by a colleague. She has used gets() in her code. What do you suggest and why? **(4 Marks)**

Q.3 What can cause buffer overflow? **(7 Marks)**

Q.4 Write a short note on ASLR. **(4 Marks)**

Q.5 How Data Execution Prevention (DEP) can be used to provide protection from buffer overflow? **(5 Marks)**

Q.6 Classify buffer overflow protection techniques. **(6 Marks)**

Q.7 Julius is interested in buffer overflow protection. He comes to you asking about the advantages and disadvantages of OS-based and compiler-based protection mechanisms. What do you tell him? **(6 Marks)**

Q.8 What is a canary? How and where it is used? **(6 Marks)**

Q. 9 Draw a block diagram of StackShield and explain. **(6 Marks)**

Q. 10 Explain any three types of buffer overflow. **(6 Marks)**

Q. 11 Write a short note on Format String vulnerability. **(4 Marks)**

Q. 12 Compare the types of XSS. **(6 Marks)**

Q. 13 Persisted XSS can exploit any visitor. Comment. **(6 Marks)**

Q. 14 Alex is worried about XSS. You tell him to relax and follow some guidelines on preventing XSS. He asks you which guidelines you would recommend. What do you tell him? **(6 Marks)**

Q. 15 How SQLi works? **(6 Marks)**

Q. 16 Classify SQLi. **(8 Marks)**

Q. 17 Write a short note on tautology. **(4 Marks)**

Q. 18 Database programmers are joining your company today. You are sent to explain company guidelines on preventing SQLi. What do you explain? **(7 Marks)**

Q. 19 Compare various types of malware on the basis of Purpose, Trigger, Replication and suitable Countermeasure. **(6 Marks)**

Q. 20 Draw the virus classification diagram (no explanation required). **(6 Marks)**

Q. 21 What are the components of bots? **(6 Marks)**

Q. 22 Write a short note on botnet architectures. **(5 Marks)**

Q. 23 What are the ways of detecting rootkits? **(6 Marks)**

Q. 24 Ron wants to ensure that everyone in the company is aware of malware protection guidelines. You are required to make a presentation covering the malware protection guidelines. Which guidelines would you include in your presentation? **(4 Marks)**