# UNIT 3 DATABASE SECURITY AND AUTHORISATION

**Structure** **Page Nos.**



## 3.0 INTRODUCTION

As the use of Internet technology is growing for both the Intranet and the Internet, information security is becoming exceedingly crucial for organisations. The World Wide Web provides a convenient, cheap, easily accessible and instantaneous way of information dissemination. It makes the dissemination of very easy but, it is equally important to ensure that information should only be accessible information to rightful users who have access rights to it.

With many organisations using database based dynamic web pages, corporate information security has become extremely important. Earlier, strict database access or specialised client software was required for viewing the data, but today a simple web browser is sufficient to view data in a database that is not properly protected. Thus, information security is at a vulnerable stage. Hence, the more a computing firm moves from the mainframe of the client/server to the Internet the more possibilities of security penetration.

Security database specialists have to rely on network administrators for implementing firewalls or other mechanisms to protect local data since the nature of Intranet/ Internet information access is such; however, the database administrator (DBA) has to perform many security function. This unit will examine the primary security areas that fall within the domain of the DBA, who then has to create database orient solutions.

## 3.1 OBJECTIVES

After going through this unit, you should be able to:

- define the various levels of database security;
- define different access control mechanism;
- explain the importance of security in statistical databases;
- define multi-level security, and
- identify the use of audit trails in database security.

## 3.2 DATABASE SECURITY : THE SCENARIO

You must realise that security is a journey, and not the final destination. We cannot assume a product/technique is absolutely secure, we may not be aware of fresh/new attacks on that product/technique. Many security vulnerabilities are not even published as attackers want to delay a fix, and manufacturers do not want the negative publicity. There is an ongoing and unresolved discussion over whether highlighting security vulnerabilities in the public domain encourages or helps prevention of further attacks.

The most secure database you can think of must be found in a most securely locked bank, or nuclear-proof bunker, installed on a standalone computer without an Internet or network connections, and under guard for 24×7×365. However, that is not a likely scenario with which we would like to work. A database server is to keep up with services, which often contain security problems, and you should be realistic about possible threats. You must assume failure at some point, and never store truly sensitive data in a database that unauthorised users may easily infiltrate/access.

A major point to consider here is that most data loss occurs because of social exploits and not technical ones. Thus, personnel procedures may more than encryption algorithms need to be looked into.

You will be able to develop an effective database security, if you realise that securing data is essential to the market reputation, profitability and business objectives. For example, as personal information such as credit card or bank account numbers are now commonly available in many databases, therefore, there are more opportunities for identity theft. As per an estimate, more than half of all identity theft cases are committed by employees who have access to large financial databases. Banks, companies that take credit cards services externally must place greater emphasis on safeguarding and controlling access to this proprietary database information.

## 3.3 LEVELS OF DATABASE SECURITY

Securing the database is a fundamental tenet for any security personnel while developing his or her security plan. The database is a collection of useful data and can be treated as the most essential component of an organisation and its economic growth. Therefore, any security effort must keep in mind that they need to provide the strongest level of control for the database.

As is true for any other technology, the security of database management systems depends on many other systems. These primarily include the operating system, the applications that use the DBMS, services that interact with the DBMS, the web server that makes the application available to end users, etc. However, please note that most importantly, DBMS security depends on us, the-users.

**Common Database Security Failures**

Database security is of paramount importance for an organisation, but many organisations do not take this fact into consideration, till an eventual problem occurs. The common pitfalls that threaten database security are:

**Weak User Account Settings:** Many of the database user accounts do not contain the user settings that may be found in operating system environments. For example, the user accounts name and passwords, which are commonly known, are not disabled or modified to prevent access.

The user account settings allow limited capabilities for security, without password controls on dictionary checks or account controls supporting expiration of user account.

**Insufficient Segregation of Duties:** No established security administrator role is defined in the database management of the organisation. This results in database administrators (DBAs) performing both the functions of the administrator (for users accounts), as well as the performance and operations expert. This may result in management inefficiencies.

**Inadequate Audit Trails:** The auditing capabilities of databases since it require keeping track of additional requirements, are often ignored for enhanced performance or disk space. Inadequate auditing results in reduced accountability. It also reduces the effectiveness of data history analysis. The audit trails records information about the actions taken on certain critical of data. They log events directly associated with the data, thus, they are essential for monitoring the access and the activities on a database system.

**Unused DBMS Security Features:** The security of an individual application is usually independent of the security of the DBMS. Please note that security measures that are built into an application apply to users of the client software only. The DBMS itself and many other tools or utilities that can connect to the database directly through ODBC or any other protocol, may bypass this application level security completely. Thus, you must try to use security restrictions that are reliable, for instance, try using security mechanism that are defined within the database.

Basically database security can be broken down into the following levels:

- Server Security
- Database Connections
- Table Access Control
- Restricting Database Access.

### 3.3.1 Server Security

Server security is the process of controlling access to the database server. This is the most important aspect of security and should be carefully planned.

The basic idea here is "You cannot access what you do not see".

For security purposes, you should never let your database server be visible to the world. If a database server is supplying information to a web server then it should be configured in such a manner that it is allowed connections from that web server only. Such a connection would require a trusted IP address.

**Trusted IP Addresses**

To connect to a server through a client machine, you would need to configure the server to allow access to only trusted IP addresses. You should know exactly who should be allowed to access your database server. For example, if it is the back end of a web server, then only that web server address should be allowed access to the database server. If the database server is the back end of a local application that is running on the internal network, then it should only talk to addresses from within the internal network.

### 3.3.2 Database Connections

With the ever-increasing number of Dynamic Applications, an application may allow immediate unauthenticated updates to some database. If you are going to allow users make updates to some database via a web page, please ensure that you validate all such updates. This will ensure that all updates are desirable and safe. For example, you may remove any possible SQL code from a user-supplied input. If a normal user does not input SQL code then we need not allow such data to be submitted.

### 3.3.3 Table Access Control

Table access control is probably one of the most overlooked but one of the very strong forms of database security because of the difficulty in applying it. Using a table access control properly would require the collaboration of both the system administrator as well as the database developer. In practise, however such "collaboration" is relatively difficult to find.

### 3.3.4 Restricting Database Access

By now we have defined some of the basic issues of database security, let us now look into the specifics of server security, from the point of view of network access of the system. Internet based databases have been the most recent targets of security attacks. All web-enabled applications listen to a number of ports. Cyber criminals often perform a simple "port scan" to look for ports that are open from the popular default ports used by database systems. How can we address this problem? We can address this problem **"by default",** that is, we can change the default ports a database service would listen into. Thus, this is a very simple way to protect the DBMS from such criminals.

There are many ways of preventing open access from the Internet. Each DBMS and OS has its own set of unique features for the same. Let us discuss some general methods of preventing open access from the Internet.

- **Trusted IP addresses :** UNIX servers are configured to answer ping requests only from a list of trusted hosts. In UNIX, this can be accomplished by configuring the *rhosts* file. Thus, it restricts server access to a list of specific users only.

- **Server Account Disabling :** It may be a good idea to suspend the server ID after three failed password attempts. This may thwart attackers. If such a scheme is not implemented, then an attacker can run a brute force program that generates millions of passwords. Such a program ultimately would break the combination of the user ID and password.

- **Special Tools :** Some customised tools, for example, *Real Secure,* send an alert when an external server is attempting to breach your system security. There are many such similar products available for the protecting of the DBMS from unauthorised Internet access.

## 3.4 ACCESS CONTROL

All relational database management systems provide some sort of intrinsic security mechanisms that are designed to minimise security threats as stated in the previous section. These mechanism range from the simple password protection offered in Microsoft Access to the complex user/role structure supported by advanced relational databases like Oracle and Microsoft SQL Server. But can we define access control for all these DBMS using a single mechanism? SQL provides that interface for access control. Let us discuss the security mechanisms common to all databases using the Structured Query Language (SQL).

An excellent practice is to create individual user accounts for each database user. Although, sharing of user accounts among various users is possible or even one user

account can be created for each type of user, however, such a practice should be discouraged. Why? It could be because of the following reasons:

**It will eliminate individual accountability**: *If any one of the users make a change in the database, we will not be able to trace it back to a specific individual even after going through audit logs. Imagine what would happen when a specific user leaves the organisation and his or her access from the database is to be removed? It will require change in the password and this will cause inconvenience to other users.*

Thus, it is important that we provide separate user accounts for separate users. Does this mechanism have any drawbacks? If the expected number of database users are small then it is all right to give them individual user name and passwords and all the database access privileges that they need to have on the database items. However, consider a situation when there are a large number of users. Specification of access rights to all these users individually will take a long time. That is still manageable as it may be a one time effort, however, the problem will be compounded if we need to change the access right for a particular type of users. Such an activity would require a huge maintenance cost. This cost can be minimised if we use a specific concept called "Roles". A database may have hundreds of users but their access rights may be categorised in specific roles for example, teachers, student in a university database. Such roles would require specification of access rights only once for the **role.** The users then can be assigned username, password and specific role. Thus, the maintenance of user accounts becomes easier as now we have limited roles to be maintained.

Let us explain SQL related security commands in more details.

### 3.4.1 Granting Permissions

You would need to create the users or roles before you grant them permissions. Then permissions can be granted to a user or a role. This can done with the use of the SQL GRANT statement.

The syntax of this statement is:

**GRANT <permissions>**
**[ON <table>]**
**TO <user/role>**
**[WITH GRANT OPTION]**

Now, let us define this statement line-by-line. The first line, GRANT <permissions>, allows you to specify the specific permissions on a table. These can be either relation-level data manipulation permissions (such as SELECT, INSERT, UPDATE and DELETE) or data definition permissions (such as CREATE TABLE, ALTER DATABASE and GRANT). More than one permission can be granted in a single GRANT statement, but data manipulation permissions and data definition permissions may not be combined in a single statement.

The second line, ON <table>, is used to specify the table on which permissions are being given. This line is not needed if we are granting data definition permissions.

The third line specifies the user or role that are being granted permissions.

Finally, the fourth line, WITH GRANT OPTION, is optional. If this line is included *in the statement, the user* is also permitted to grant the same permissions that s/he has received to other users. Please note that the WITH GRANT OPTION cannot be specified when permissions are assigned to a *role.*

Let us look at a few examples of the use of this statement.

**Example 1:** Assume that you have recently hired a group of 25 data entry operators who will be adding and maintaining student records in a university database system. They need to be able to access information in the STUDENT table, modify this information and add new records to the table. However, they should not be able to entirely delete a record from the database.

**Solution:** First, you should create user accounts for each operator (please refer to MCS-023, Block 2, Unit-1) and then add them all to a new role-Dataentry. Next, we would need to use the following SQL statement to grant them the appropriate permissions:

```
GRANT SELECT, INSERT, UPDATE
ON STUDENT
TO Dataentry
```

And that is all that you need to do. Let us now examine a case where we are assigning data definition permissions.

**Example 2**: We want to allow members of the DBA role to add new tables to our database. Furthermore, we want them to be able to grant other users permission to do the same.

**Solution:** The SQL statement to do so is:

```
GRANT CREATE TABLE
TO DBA
WITH GRANT OPTION
```

Notice that we have included the WITH GRANT OPTION line to ensure that our DBAs can assign this permission to other users.

At this point, we have discussed how to assign permissions to users and roles as necessary. We will now look at the methods for removing permissions from users.

### 3.4.2 Removing Permissions

Once we have granted permissions, it may be necessary to revoke them at a later date. SQL provides us with the REVOKE command to remove granted permissions. The following is the syntax of this command:

```
REVOKE [GRANT OPTION FOR] <permissions>
ON <table>
FROM <user/role>
```

Please notice that the syntax of this command is almost similar to that of the GRANT command. Please also note that the WITH GRANT OPTION is specified on the REVOKE command line and not at the end of the command as was the case in GRANT. As an example, let us imagine we want to revoke a previously granted permission to the user Usha, such that she is not able to remove records from the STUDENT database. The following commands will solve the problem:

```
REVOKE DELETE
ON STUDENT
FROM Usha
```

There is one additional mechanism supported by some commercial DBMS that is worth discussing – the DENY command. This command can be used to *explicitly* deny permission to a user that s/he might otherwise have received because of his/her membership of a role. Here is the syntax:

```
DENY <permission>
ON <table>
TO <user/role>
```

Consider the last problem again, let us imagine that Usha was also a member of the Teachers role that also had access to the STUDENT table. The previous REVOKE

statement would not be sufficient to deny her access to the table. It will remove the permission granted to her through a GRANT statement, but would not affect the permissions gained through her membership in the Teachers role. However, if we use a DENY statement it will block permission for any role. Here is the command:

```
DENY DELETE
ON STUDENT
TO Usha
```

Thus DENY command creates a "NOT PERMITTED" statement in the database access controls. If we later want to give Usha permission to remove student records again from the STUDENT table, we cannot simply use the GRANT command. This is because of the fact that the GRANT command permission to DELETE record would be overridden by the existing DENY. Thus, first we use the REVOKE command to remove the DENY Not permission as:

```
REVOKE DELETE
ON STUDENT
FROM Usha
```

Please notice that this command is exactly the same as the REVOKE used to remove a granted permission. Thus, the DENY and GRANT commands both work in a similar fashion -- they both create permissions in the database access control mechanism. The REVOKE command removes all such permissions for the specified user. Once this command has been issued, Usha will be able to delete student records from the table if she is a member of a role that possesses that permission. You can also issues a GRANT command to provide the DELETE permission to Usha's account.

The access control mechanisms supported by the Standard Query Language is a good starting point, but you must look into the DBMS documentation to locate the enhanced security measures supported by your system. You will find that many DBMS support more advanced access control mechanisms, such as granting permissions on specific attributes.

### ☞ Check Your Progress 1

1) The most secure database is found in ……

2) On what systems does the security of a Database Management System depend?

   ……

   ……

3) ……. is the process of limiting Access to the Database Server.

   ……

   ……

4) What are the different ways of preventing open access to the Internet?

   ……

   ……

5) Write the syntax for granting permission to alter database.

   ……

   ……

   ……

6) Write the syntax for 'Revoke Statement' that revokes with grant option.

……

……

……

7) What does DENY command do?

……

……

……

# 3.5 STATISTICAL DATABASE SECURITY

By now we have discussed the basic security measures in both DBMS and SQL commands that provide necessary permissions to the users. However, in practice there are many database systems where information can be determined without having the access rights to do so. This is a breach of data confidentiality. In the subsequent subsection we discuss this problem and the way to resolve it.

## 3.5.1 Types of Disclosures

Data of an organisation is a very sensitive resource. Even the characteristics of data are quite sensitive. For example, existence of a piece of data such as "use of health related drugs" is sensitive information and is a form of disclosure. The various type of disclosure may be:

- **Exact Data:** It may be defined as the determination of the value of an item by using a sequence of complex queries.
- **Bounds:** It is defined as finding the value of data item between two values. The bounds in such cases may be lowered using binary search. This may lead to a very narrow range.
- **Negative Result:** Sometimes it may be possible to determine a negative result. This is also a form of disclosure.
- **Existence:** The existence of data value in itself is a sensitive piece of information, regardless of the actual value. For example, existence of a record regarding defence expenses may be a disclosure.

Please note that such disclosure of data can be obtained without making a direct query to the database but rather a set of queries. We will explain it with the help of an example in the next sub-section. Please remember "A good security scheme needs to protect data using access control while avoiding such indirect disclosures".

## 3.5.2 Security vs. Decisions

Disclosure of data as indicated in the previous section is a major problem as disclosure may result in breach of security in some form or the other, and thus, is not acceptable. Thus, the first step in this direction would be to reject any query that directly asks for sensitive information that is hidden. But, how about a sequence of queries that are raised for the purpose of statistics (management information)? For example, we may be able to determine the average marks obtained in a class of 50 student, but if only 2 students have opted for a subject then the first student who knows his/her marks can find the marks of the other student by issuing the average marks query. Thus, statistical queries should be permitted only when some minimum number of records satisfies a condition. Thus, the overall objectives are to make sure that security is not compromised.

Let us discuss some of the queries that may result in the disclosure of sensitive data. Consider the relation in the following *Table*:

| Enrolment No. | Name | Gender | Grade | Fee paid | Age | Hostel |
|---|---|---|---|---|---|---|
| 0601 | Anil | M | C | 10000 | 25 | Ganga |
| 0602 | Dev | M | B | 0 | 20 | Yamuna |
| 0603 | Sunil | M | A | 9000 | 22 | Kaveri |
| 0604 | Esha | F | B | 10000 | 27 | Ganga |
| 0605 | Nagma | F | C | 9000 | 20 | Ganga |
| 0606 | Fiza | F | C | 10000 | 20 | Kaveri |
| 0607 | Ganesh | C | 9000 | 24 | 3 | Kaveri |
| 0608 | Himani | C | 0 | 21 | 0 | Yamuna |

Assume that a student can not only view his/her details in the *Table*, but also the names of his/her colleagues, and that s/he is allowed to make queries as well. Let us see how s/he can attack the database.

**Simple Attacks:** A simple form of direct attack may be by using the following type of query:

```
SELECT name
FROM STUDENT
WHERE (Hostel = 'Yamuna' AND Gender = 'F').
```

The query above will show the result as 'Himani'. Thus, the information that Himani stays in Yamuna hostel has been disclosed. Please note this is a direct query and can be caught, but a person can hide this query in many ways, without changing the basic meaning of the query.

But how do we stop such disclosures? A very simple solution in this case may be: If a query that processes N records, however produces very few records (M) such that N>>M then this query may compromise security and should not be allowed to produce results.

**Inference Based Attacks:** Such an attack is very interesting case for determination of data through mathematical logic. For example, to determine the age of the female candidate, a series of commands can be used for determining the required information. These commands may be:

(i) Determine the sum of age of all the female candidates,

(ii) Determine the sum age of female candidates not residing in Hostel Yamuna.

Since all these queries will result in a sizable number of records they would be answered, however, they can be used to determine the age of 'Himani' who stays in Yamuna and is a sole female student staying there as:

Age of Himani= Result of Query (i) – Result of Query (ii).

Similarly many statistical functions like average, count, sum and their combinations can be used on various combinations of records. This sequence may be used to disclose the sensitive database values.

Thus, the database security has been compromised. In such situation the solution may be – not to allow consecutive queries whose output record set intersection is very small.

But how actually these problems be solved? Let us discuss the solution to such problems in the next subsection.

### 3.5.3 Security and Statistical Queries

The two possible ways of securing database against such attacks may be:

i) Controlling the queries themselves, (such solutions may help in solving the queries that can be identified as causing problems).

ii) The queries that result in limited value can be checked by either rejecting a query. If the result set size is too small, it is also called 'suppression' or if the information that is to be displayed as a result of the query is changed slightly so that it does not match the actual value it is also called concealing.

### ☞ Check Your Progress 2

1)

| Subject Maximum Marks = 100 | Neha | Richa | Neelam | |
|---|---|---|---|---|
| Maths | 50 | 70 | 90 | |
| Science | 60 | 80 | 50 | |
| Total | 110 | 150 | 140 | |

Write a sequence of queries that would disclose the name of the person who scored the highest marks.

2) Write two ways of protecting against inference attacks.

……

……

……

3) With …… sensitive data values are not provided; the query is rejected without response.

## 3.6 MULTILEVEL SECURITY

In certain applications data items can be classified under various levels of security. Some such security levels may be Secret, Classified, etc. The database system that supports such security features are known as Multilevel Sensitive Databases. Such systems may have the following three security features:

- Security of different objects may be different from the other attribute values of that tuple or security may be different from the other values of the attributes.
- Thus, every individual element is the micro item for security.
- In addition, security against statistical queries may also need to be provided.
- A number of security level needs to be defined for such security.

But how do we enforce the multilevel security?

There are many techniques to support multi-level security. Two important methods of these are:

**Partitioning**

In this approach the original database is divided into partitions. Each of the partitions has a different level of security.

However, the major drawback of this approach is that the database looses the advantage of a relation.

**Encryption**

Encryption of critical information may help in maintaining security as a user who accidentally receives them cannot interpret the data. Such a scheme may be used when a user password file is implemented in the database design. However, the simple encryption algorithms are not secure, also since data is available in the database it may also be obtained in response to a query.

There are many more schemes to handle such multi-level database security. These references are given in the further reading.

## 3.7 AUDIT TRAILS IN THE DATABASES

One of the key issues to consider while procuring a database security solution is making sure you have a secure audit-trail. An audit trail tracks and reports activities around confidential data. Many companies have not realised the potential amount of risk associated with sensitive information within databases unless they run an internal audit which details who has access to sensitive data and have assessed it. Consider the situation that a DBA who has complete control of database information may conduct a security breach, with respect to business details and financial information. This will cause tremendous loss to the company. In such a situation database audit helps in locating the source of the problem. The database audit process involves a review of log files to find and examine all reads and writes to database items during a specific time period, to ascertain mischief if any, banking database is one such database which contains very critical data and should have the security feature of auditing. An audit trail is a log that is used for the purpose of security auditing

Database auditing is one of the essential requirements for security especially, for companies in possession of critical data. Such companies should define their auditing strategy based on their knowledge of the application or database activity. Auditing need not be of the type “all or nothing”. One must do intelligent auditing to save time and reduce performance concerns. This also limits the volume of logs and also causes more critical security events to be highlighted.

More often then not, it is the insiders who makes database intrusions as they often have network authorisation, knowledge of database access codes and the idea about the value of data they want to exploit. Sometimes despite having all the access rights and policies in place, database files may be directly accessible (either on the server or from backup media) to such users. Most of the database applications, store information in ‘form text’ that is completely unprotected and viewable.

As huge amounts are at stake, incidents of security breaches will increase and continue to be widespread. For example, a large global investment bank conducted an audit of its proprietary banking data. It was revealed that more than ten DBAs had unrestricted access to their key sensitive databases and over hundred employees had administrative access to the operating systems. The security policy that was in place was that proprietary information in the database should be denied to employees who did not require access to such information to perform their duties. Further, the bank’s database internal audit also reported that the backup data (which is taken once every day) was also cause for concern as tapes could get stolen. Thus the risk to the database was high and real and that the bank needed to protect its data.

However, a word of caution, while considering ways to protect sensitive database information, please ensure that the privacy protection process should not prevent authorised personnel from obtaining the right data at the right time.

The credit card information is the single, most common financially traded information that is desired by database attackers. The positive news is that database misuse or unauthorised access can be prevented with currently available database security products and audit procedures.

# 3.8 EXAMPLES OF VENDOR-SPECIFIC E-SECURITY

Individual vendors largely determine the security schemes that may be implemented to provide the link between the database and its interfaces. Following are some of the security components provided by the major vendors. Like client/server systems, security solutions must be combined together using multiple vendor products.

**Oracle**

Oracle, provides SSL and S-HTTP security. Oracle uses Java as a basic component of its security model. The company created its Oracle Web Server to work most effectively with Oracle clients such as the solutions created with the Developer/2000 or other development tools.

Oracle modified the HTTP protocol to allow a straight/direct connection between the client and the server. This connection defines a session in which the user is identified by a generated ID.

These enhancements are also present in the Secure Network Server (SNS) that is included in the Oracle Universal Database a single login permits access to any Oracle database in an enterprise system.

The Java security classes are used by the oracle development tools to give complete security integration to the client.

**Sybase**

Sybase provides a rather elegant way of protecting data access through the Web. It extends the logon security present in the Web server to the database server for authentication. Thus, it takes advantage of the native security present in the database. Sybase provides a middleware called Web.sql that is used to interface with the Netscape Web servers.

**Informix**

Informix, like Sybase, relies on the logon security present in the Web server. Therefore, any database access is specified through traditional ODBC-type login channels that passes the user and password information through the connectivity middleware. Specific drivers known as Universal Web Connect integrates Informix database security with Microsoft Web servers.

**Microsoft**

Microsoft has included most of the key security technologies with Internet Information Server (IIS). For user authentication, Microsoft provides its tried-and-true challenge/response mechanism. Traditional login on the Web presents the same security as in the basic Windows NT login. Unfortunately, only Microsoft Internet Explorer browser supports this login approach.

For database access, IIS security has been integrated with Microsoft SQL Server through the Internet Database Connector. Although, users must login through an HTML login form, the information may be verified by a SQL Server stored procedure.

Microsoft has also integrated the Kerberos security architecture into Windows NT Server. By releasing the server, Microsoft hopes to integrate the Kerberos native to the NT Server with public key security. Microsoft has already released a Certificate Server API in an attempt to create a Certificate Server standard.

**Netscape**

Netscape Communications intends its suite of servers as a complete system for security on the Internet. Login occurs originally through the browser and then, as in the Novell Directory Services, all certification is unified in this model. Therefore,

once login to the browser occurs, any resources that are permitted to the user are now accessible.

Currently, user authentication occurs by passing information to the data source via the middleware. Most companies, including Sybase, Oracle, and Informix, provide the necessary connectors for this process.

### ☞ Check Your Progress 3

1) What is multilevel Sensitive Database?

   …….

   ……..

   …….

2) Name the different techniques for multilevel security.

   …….

   ……..

   …..……..……

3) ……., while providing SSL & S-HTTP security, its using java as a basic component of its security model.

4) …… & …… currently relies on the logon security present in the Web Server.

5) For user Authentication, Microsoft provides its ……. mechanism.

## 3.9 SUMMARY

This unit provides basic information about database security. It introduces various levels of database security. Some of the information that is covered in this unit include the commands to GRANT and REVOKE assess rights. In addition, a command DENY has also been discussed. Please note DENY is not available in all DBMSs. We have then discussed security in databases with statistical queries. We have also suggested certain mechanism to ensure data base security, even through statistical queries may be allowed-definitely with some restriction. We have also defined the concept of audit trail and given few DBMS security support. You must go through any commercial DBMS documents for more details on security implemented in them.

## 3.9 SOLUTIONS/ANSWERS

### Check Your Progress 1

1) Second graded Bunkers.
2) Operating System,
   Application that use the database,
   Service that interact, and
   The Web Server.
3) Server Security
4) Trusted IP address.
   Server Account Disabling
   Special Tools, Example: Real source by ISS.

5) GRANT ALTER DATABASE TO Usha

6) REVOKE GRANT OPTION FOR <permissions>
ON <table>
FROM <user/role>

7) It creates a 'NOT PERMITTED' condition in the database Access control.

### Check Your Progress 2

1) The highest marks can be found as:
Find the total marks
Find the max. marks
Find the names of student whose total is in the set of maximum marks.

2) a) Control to the Queries.
b) Control individual items that are being displayed.

3) Suppression.

### Check Your Progress 3

1) The database with more than one i.e., various levels of security.

2) i) Partitioning
ii) Encryption

3) Oracle

4) Sybase and Informix

5) tried-and-true challenge/response mechanism.

## 3.10 FURTHER READING

1) B. Sudhir Kumar Reddy, "*Aspects of Database and Program Security*" November 2003, available on the internet.