Chapter 12. Database Security

Table of contents

Objectives

At the end of this chapter you should be able to:

- Understand and explain the place of database security in the context of security analysis and management.
- Understand, explain and apply the security concepts relevant to database systems.
- Understand, identify and find solutions to security problems in database systems.
- Understand the basic language of security mechanisms as applied to database systems.
- Analyse access control requirements and perform fairly simple implementations using SQL.
- Appreciate the limitations of security subsystems.

Introduction

In parallel with this chapter, you should read Chapter 19 of Thomas Connolly and Carolyn Begg, "Database Systems A Practical Approach to Design, Implementation, and Management", (5th edn.).

Security is a large subject and one that, because it touches every activity of an information system, appears everywhere. In the main text you will start with a thumbnail introduction to security, while the extension reading contains references for you to pursue when you wish.

In earlier chapters in this module you have met concepts and techniques which can be regarded as security measures. For example, the process of recovery, whether from partial or total failure, should be considered as having a security dimension. Nearly all the work on concurrency (see chapter 13) is directed at another aspect of security. Again, a thumbnail introduction is given.

The main work you do in this chapter, however, is directed to database security rather than security in general, and to the principles of security theory and practice as they relate to database security. These are technical aspects of security rather than the big picture.

The chapter is organised into two parts. The first part covers security principles and models itself in two parts moving from the softer principles (setting the universe of discourse) through to some specific technical issues of database security. The second part is about logical access control in SQL databases.

The major practical area you will cover is the area of access control. After a discussion of the principles, you will quickly be getting into some detail of access

control in SQL databases. Extension reading, both textbooks and websites, is given for you to pursue the detail further.

What is not covered in this chapter, but is covered elsewhere in the module, are the subjects of database administration, transaction recovery and catastrophe recovery. The first of these is directly related to management controls on operation and development. The second is directly related to database integrity and consistency, thus being largely an internal matter. The third is easier to follow as an extension of the first and second. But all three are security based.

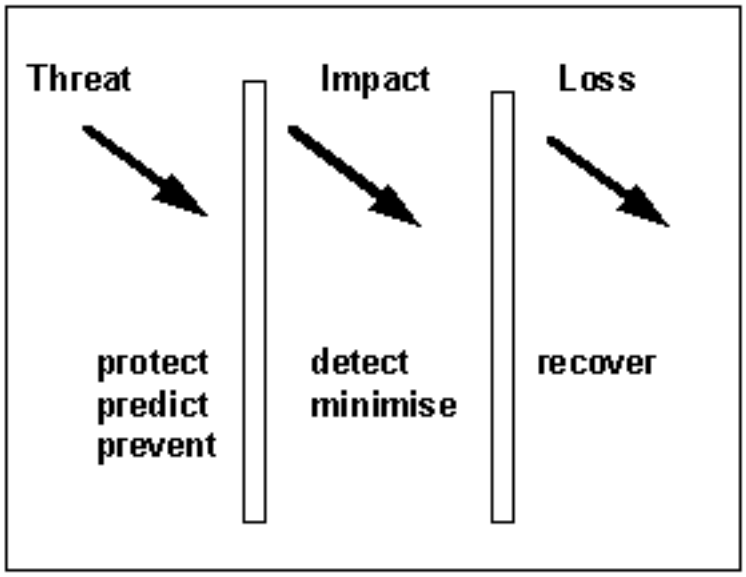
The scope of database security

Overview

All systems have ASSETS and security is about protecting assets. The first thing, then, is to know your assets and their value. In this chapter, concentrate on database objects (tables, views, rows), access to them, and the overall system that manages them. Note that not all data is sensitive, so not all requires great effort at protection. All assets are under threat.

The second thing to know is what THREATs are putting your assets at risk. These include things such as power failure and employee fraud. Note that threats are partly hypothetical, always changing and always imperfectly known. Security activity is directed at protecting the system from perceived threats.

If a threat is potential, you must allow for it to become an actuality. When it becomes actual there is an IMPACT. Impact you can consider and plan for. But in the worst case, there will be a LOSS. Security activity here is directed at minimising the loss and recovering the database to minimise the loss as well as further protecting from the same or similar threats.



An outlined development mechanism is:

1. Document assets (what they are, what their value is).
2. Identify treats (what they are, how likely they are, what the impact is if they occur).
3. Associate threats with each asset.
4. Design mechanisms to protect each asset appropriate to its value and the cost of its protection, to detect a security breach against each asset, to minimise the losses incurred and to recover normal operation.

Threats to the database

You will build your security skills from two directions. One is from the appreciation and awareness of changing threats, and the other from the technical remedies to them. Threats include:

- Unauthorised modification: Changing data values for reasons of sabotage, crime or ignorance which may be enabled by inadequate security mechanisms, or sharing of passwords or password guessing, for example.
- Unauthorised disclosure: When information that should not have been disclosed has been disclosed. A general issue of crucial importance, which can be accidental or deliberate.

- Loss of availability: Sometimes called denial of service. When the database is not available it incurs a loss (otherwise life is better without the system!). So any threat that gives rise to time offline, even to check whether something has occurred, is to be avoided.

The rest of this section is an overview of the categories of specific regulatory threats to database systems.

- **Commercial sensitivity:** Most financial losses through fraud arise from employees. Access controls provide both protection against criminal acts and evidence of attempts (successful or otherwise) to carry out acts detrimental to the organisation, whether fraud, extraction of sensitive data or loss of availability.
- **Personal privacy and data protection:** Internationally, personal data is normally subject to legislative controls. Personal data is data about an identifiable individual. Often the individual has to be alive but the method of identification is not prescribed. So a postal code for a home may in some cases identify an individual, if only one person is living at an address with the postal code. Such data needs careful handling and control.

  For more information see Data Protection later in the chapter. The issues are too extensive to be discussed here but the implications should be noted. Personal data needs to be identified as such. Controls must exist on the use of that data (which may restrict ad-hoc queries). Audit trails of all access and disclosure of the information need to be retained as evidence.
- **Computer misuse:** There is also generally legislation on the misuse of computers. Misuse includes the violation of access controls and attempts to cause damage by changing the database state or introducing worms and viruses to interfere with proper operation. These offences are often extraditable. So an unauthorised access in Hong Kong using computers in France to access databases in Germany which refer to databases in America could lead to extradition to France or Germany or the USA.
- **Audit requirements:** These are operational constraints built around the need to know who did what, who tried to do what, and where and when everything happened. They involve the detection of events (including CONNECT and GRANT transactions), providing evidence for detection, assurance as well as either defence or prosecution. There are issues related to computer-generated evidence not covered here.

In considerations of logical access to the database, it is easy to lose sight of the fact that all system access imposes risks. If there is access to operating system utilities, it becomes possible to access the disk storage directly and copy or damage the whole database or its components. A full consideration has to take all such access into account. Most analysts would be looking to minimise communications (direct, network and telecommunications) and isolate

the system from unnecessary threats. It is also likely that encryption would be used both on the data and the schema. Encryption is the process of converting text and data into a form that can only be read by the recipient of that data or text, who has to know how to convert it back to a clear message.

You will find it easier to consider security and auditing as issues separate from the main database functions, however they are implemented. Visualise the security server and audit servers as separate functional modules.

Principles of database security

To structure thoughts on security, you need a model of security. These come in various forms that depend on roles, degree of detail and purpose. The major categories are areas of interest (threats, impact and loss) as well as the actions involved in dealing with them.

Security risks are to be seen in terms of the loss of assets. These assets include:

- Hardware
- Software
- Data
- Data quality
- Credibility
- Availability
- Business benefit

Here we are primarily concerned with threats to the data and data quality but, of course, a threat to one asset has consequential impact on other assets. What is always important is that you are very clear on just what asset needs protection.

So as a summary:

Problem	Tool	Technique
Reliability (operational security)	Recover from corruption, loss and damage	Back-up, logging, checkpoints.
Access Security	Control Access	Passwords, dialogues.
Integrity (schema security)	Ensure internal consistency	Validation rules, constraints.

You need to accept that security can never be perfect. There always remains an element of risk, so arrangements must be made to deal with the worst eventuality - which means steps to minimise impact and recover effectively from loss or damage to assets. Points to bear in mind:

1. Appropriate security - you do not want to spend more on security than the asset is worth.
2. You do not want security measures to interfere unnecessarily with the proper functioning of the system.

Security models

A security model establishes the external criteria for the examination of security issues in general, and provides the context for database considerations, including implementation and operation. Specific DBMSs have their own security models which are highly important in systems design and operation. Refer to the SeaView model for an example.

You will realise that security models explain the features available in the DBMS which need to be used to develop and operate the actual security systems. They embody concepts, implement policies and provide servers for such functions. Any faults in the security model will translate either into insecure operation or clumsy systems.

Access control

The purpose of access control must always be clear. Access control is expensive in terms of analysis, design and operational costs. It is applied to known situations, to known standards, to achieve known purposes. Do not apply controls without all the above knowledge. Control always has to be appropriate to the situation. The main issues are introduced below.

Authentication and authorisation

We are all familiar as users with the log-in requirement of most systems. Access to IT resources generally requires a log-in process that is trusted to be secure. This topic is about access to database management systems, and is an overview of the process from the DBA perspective. Most of what follows is directly about Relational client-server systems. Other system models differ to a greater or lesser extent, though the underlying principles remain true.

For a simple schematic, see Authorisation and Authentication Schematic.

Among the main principles for database systems are authentication and authorisation.

Authentication

The client has to establish the identity of the server and the server has to establish the identity of the client. This is done often by means of shared secrets (either a password/user-id combination, or shared biographic and/or biometric data). It can also be achieved by a system of higher authority which has previously established authentication. In client-server systems where data (not necessarily the database) is distributed, the authentication may be acceptable from a peer system. Note that authentication may be transmissible from system to system.

The result, as far as the DBMS is concerned, is an authorisation-identifier. Authentication does not give any privileges for particular tasks. It only establishes that the DBMS trusts that the user is who he/she claimed to be and that the user trusts that the DBMS is also the intended system.

Authentication is a prerequisite for authorisation.

Authorisation

Authorisation relates to the permissions granted to an authorised user to carry out particular transactions, and hence to change the state of the database (write-item transactions) and/or receive data from the database (read-item transactions). The result of authorisation, which needs to be on a transactional basis, is a vector: Authorisation (item, auth-id, operation). A vector is a sequence of data values at a known location in the system.

How this is put into effect is down to the DBMS functionality. At a logical level, the system structure needs an authorisation server, which needs to co-operate with an auditing server. There is an issue of server-to-server security and a problem with amplification as the authorisation is transmitted from system to system. Amplification here means that the security issues become larger as a larger number of DBMS servers are involved in the transaction.

Audit requirements are frequently implemented poorly. To be safe, you need to log all accesses and log all authorisation details with transaction identifiers. There is a need to audit regularly and maintain an audit trail, often for a long period.

Access philosophies and management

Discretionary control is where specific privileges are assigned on the basis of specific assets, which authorised users are allowed to use in a particular way. The security DBMS has to construct an access matrix including objects like relations, records, views and operations for each user - each entry separating create, read, insert and update privileges. This matrix becomes very intricate as authorisations will vary from object to object. The matrix can also become very large, hence its implementation frequently requires the kinds of physical

implementation associated with sparse matrices. It may not be possible to store the matrix in the computer's main memory.

At its simplest, the matrix can be viewed as a two-dimensional table:

Database User	Database Object	**Access Rights**
Pay-clerk1	Pay-table-view1	Read, Update
Pay-clerk2	Pay-table	Read, Insert
Pay-auditor	Pay-table	Read

When you read a little more on this subject, you will find several other rights that also need to be recorded, notably the owners' rights and the grant right.

Mandatory control is authorisation by level or role. A typical mandatory scheme is the four-level government classification of open, secret, most secret and top secret. The related concept is to apply security controls not to individuals but to roles - so the pay clerk has privileges because of the job role and not because of personal factors.

The database implication is that each data item is assigned a classification for read, create, update and delete (or a subset of these), with a similar classification attached to each authorised user. An algorithm will allow access to objects on the basis of less than or equal to the assigned level of clearance - so a user with clearance level 3 to read items will also have access to items of level 0, 1 and 2. In principle, a much simpler scheme.

The Bell-LaPadula model (2005) defines a mandatory scheme which is widely quoted:

- A subject (whether user, account or program) is forbidden to read an object (relation, tuple or view) unless the security classification of the subject is greater or equal to that of the object.
- A subject is forbidden to write an object unless the security classification of the subject is less than or equal to that of the object.

Note that a high level of clearance to read implies a low level of clearance to write - otherwise information flows from high to low levels. This is, in highly secure systems, not permitted.

Mandatory security schemes are relatively easy to understand and, therefore, relatively easy to manage and audit. Discretionary security is difficult to control and therefore mistakes and oversights are easy to make and difficult to detect. You can translate this difficulty into costs.

There are perhaps two other key principles in security. One is disclosure, which is often only on a need-to-know basis. This fits in better with discretionary

security than mandatory, as it implies neither any prior existing level nor the need for specific assignment.

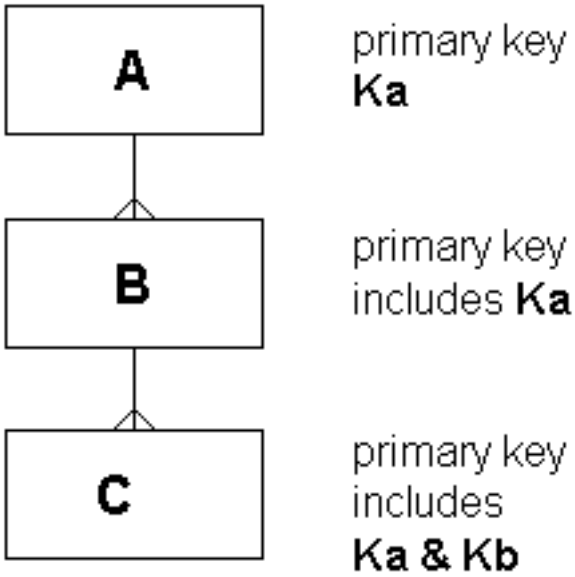
The other principle is to divide responsibilities. The DBA responsible for security management is him/herself a security risk. Management approaches that involve one person or a group of people that have connections in their work represent a similar risk. This emphasises the importance of security auditing and the importance of related procedure design.

Database security issues

This section reviews some of the issues that arise in determining the security specification and implementation of a database system.

Access to key fields

Suppose you have a user role with access rights to table A and to table C but not to table B. The problem is that the foreign key in C includes columns from B. The following questions arise:



Do you have access to the foreign key in C?

If you do, you know at least that a tuple exists in B and you know some information about B that is restricted from you.

Can you update the foreign key columns?

If so, it must cascade, generating an update to B for which no privileges have been given.

These problems do not directly arise where the database is implemented by internal pointers - as a user, you need have no knowledge of the relationships between the data you are accessing. They arise because relationships are data values. Often, knowing the foreign key will not be sensitive in itself. If it is, then the definition of a view may solve the problem.

Access to surrogate information

It is not difficult to conceive of cases where the view of the data provided to a user role extends to the external world.

An example should make the problem clear.

In a retail environment, there are frequent problems with pilferage. To deal with these, private detectives work undercover. They are to all intents and purposes employees of the business and assigned to normal business activities as other members of staff. They get pay checks or slips at the same time as everyone else, they appear in management information (such as the salary analysis) in the same manner. They have a job title and participate in the system as someone they are not. The store manager is unaware of the situation, as is everybody else except the corporate security manager. When the store manager accesses the database, the detective should look like a normal employee. Queries might include:

"What leave is due to …?"

The security staff have different queries:

"Do we have someone in …?"

You can probably envisage all kinds of complications. The detective should receive a pay slip with everyone else, but should not actually be paid (or perhaps he/she should be paid something different from the normal pay for the job).

You may want to handle these situations on separate databases. As a solution it may be appropriate, but the larger the problem the more scope there is for confusion. One suggested solution is the polyinstantiation of tuples - one individual is represented by more than one tuple. The data retrieved will depend on the security classification of the user. Tuples will have the same apparent primary key but different actual primary keys, and all applications will need to be carefully integrated with the security system.

Problems with data extraction

Where data access is visualised directly, the problem can be seen clearly enough: it is to ensure that authenticated users can access only data items which they are authorised to use for the purpose required. When the focus shifts from the data to the implications that can be drawn from that data, more problems arise.

Again, an example should make things clear.

You want to know the pay of the chief executive. You have access rights to the table, except for the MONTHLY-PAY field in this tuple. So you issue an SQL query SUM (MONTHLY-PAY) across the whole table. You then create a view SELECT MONTHLY-PAY … and issue a SUM on this view. Should you get the same answer in both cases?

If not, you can achieve your objective by subtracting the two sums. If you listed the monthly pay for all, what would you expect to see - all the tuples except the one restricted? Would you expect to be notified by asterisks that data was missing which you were not allowed to see?

Another example.

You are trying to trace an individual but have limited information. You feed your limited information into the statistical database (e.g. male, age over 40, white, red car, lives in North London) and retrieve the tuples for all that meet these categories. As you get more information, the number of tuples reduces until only one is left. It is possible to deduce personal information from a statistical database if you have a little information about the structure, even if no conventional personal identifiers are available (i.e. no date of birth, social security number or name).

Some solutions to this security problem are to prevent access to small numbers of tuples and/or to produce inaccurate data (not very inaccurate but sufficiently inaccurate to prevent inferences being drawn).

Access control in SQL

This section is about the implementation of security within SQL. The basics are given in SQL-92 but, as you will realise, much security is DBMS- and hardware-specific. Where necessary, any specifics are given in the SQL of Oracle. For some ideas on Object database management systems (ODBMS) as distinct from Relational, refer to the later chapter on Object databases.

Your first objective is to learn the specifics. The access requirements specification will be implemented using these statements. Your second objective is to extend your understanding of the problem through to the management and audit functions of an operating system.

The basic statements come first, and the management functions are discussed second. In the first part you will learn the SQL needed to manage a user; in the second you will learn a little of the SQL to manage a system.

Discretionary security in SQL

This section introduces the SQL statements needed to implement access control. You should aim at having sufficient knowledge of this area of SQL to translate a simple specification into an SQL script. You should also be conscious of the limitations implicit in this script which hardwires passwords into text.

The basics of SQL are inherently discretionary. Privileges to use a database resource are assigned and removed individually.

The first issue is who is allowed to do what with the security subsystem. You need to have a high level of privilege to be able to apply security measures. Unfortunately, such roles are not within the SQL standard and vary from DBMS to DBMS. A role is defined as a collection of privileges.

As an example, the supplied roles in Oracle include (among others):

- **SYSOPER:** Start and stop the DBMS.
- **DBA:** Authority to create users and to manage the database and existing users.
- **SYSDBA:** All the DBA's authority plus the authority to create, start, stop and recover.

The role of the DBA has been covered in other chapters. The point here is that you realise there are a large number of predefined roles with different privileges and they need to be controlled. It is important to be certain that the SQL defaults do not act in ways you do not anticipate.

Schema level

The first security-related task is to create the schema. In the example below, the authorisation is established with the schema. The authorisation is optional and will default to the current user if it is not specified.

Only the owner of the schema is allowed to manipulate it. Below is an example where a user is given the right to create tables. The creator of the table retains privileges for the tables so created. Similarly, synonyms are only valid for the creator of that synonym.

CREATE SCHEMA student_database AUTHORISATION U1;

The U1 refers to the authorisation identifier of the user concerned, who has to have the right to create database objects of this type – in this case, the schema for a new database.

Provided the authorisation is correct, then the right to access the database using the schema can be granted to others. So to allow the creation of a table:

GRANT CREATETAB TO U1 ;

The topic of schema modifications will not be taken up here.

Authentication

Using the client/server model (see chapter 15), it is necessary first to connect to the database management system, effectively establishing both authentication and the complex layers of communication between the local (client DBMS) and the server.

GRANT CONNECT TO student_database AS U1,U2,U3 IDENTIFIED BY P1,P2,P3;

U1,U2,U3 are user names, P1,P2,P3 are passwords and student_database is the database name.

GRANT CONNECT TO student_database AS U4/P4 ;

Connect rights give no permission for any table within the database. U4/P4 are the identifiers known to this database security services.

Note

Users, roles and privilege levels can be confusing. The following are the key distinctions:

- A user is a real person (with a real password and user account).
- A role, or a user-role, is a named collection of privileges that can be easily assigned to a given or new user. A privilege is a permission to perform some act on a database object.
- A privilege level refers to the extent of those privileges, usually in connection with a database-defined role such as database administrator.

Table level

The authority level establishes some basic rights. The SYSDBA account has full rights and can change everything. Rights to access tables have to be GRANTed separately by the DBA or SYSADM.

The following example assigns a read privilege to a named table (note only a read privilege). The privilege extends to creating a read-only view on the table:

GRANT SELECT ON TABLE1 TO U1;

And that which may be given can be removed. REVOKE is used generally to remove any specific privilege.

REVOKE SELECT ON TABLE1 FROM U1;

The main part of this aspect of security, though, is providing access to the data. In a Relational database we have only one data structure to consider, so if we can control access to one table we can control access to all. And as tables are two dimensional, if we can control access to rows and columns, we can deal with any request for data – including schema data. We still have to know what is allowed and what is not but, given the details, the implementation is not in itself a problem.

Remember that a VIEW is created by an SQL SELECT, and that a view is only a virtual table. Although not part of the base tables, it is processed and appears to be maintained by the DBMS as if it were.

To provide privileges at the level of the row, the column or by values, it is necessary to grant rights to a view. This means a certain amount of effort but gives a considerable range of control. First create the view:

'the first statement creates the view'

CREATE VIEW VIEW1

AS SELECT A1, A2, A3

FROM TABLE1

WHERE A1 < 20000;

'and the privilege is now assigned'

GRANT SELECT ON VIEW1 TO U1

WITH GRANT OPTION;

The optional "with grant option" allows the user to assign privileges to other users. This might seem like a security weakness and is a loss of DBA control. On the other hand, the need for temporary privileges can be very frequent and it may be better that a user assign temporary privileges to cover for an office absence, than divulge a confidential password and user-id with a much higher level of privilege.

The rights to change data are granted separately:

GRANT INSERT ON TABLE1 TO U2, U3;

GRANT DELETE ON TABLE1 TO U2, U3;

GRANT UPDATE ON TABLE1(salary) TO U5;

GRANT INSERT, DELETE ON TABLE1 TO U2, U3;

Notice in the update, that the attributes that can be modified are specified by column name. The final form is a means of combining privileges in one expression.

To provide general access:

GRANT ALL TO PUBLIC;

SQL system tables

The DBMS will maintain tables to record all security information. An SQL database is created and managed by the use of system tables. These comprise a relational database using the same structure and access mechanism as the main database. Examples below show the kind of attributes in some of the tables involving access control. The key declarations have been removed.

The examples are from Oracle database.

Command	Description
ALL_ARGUMENTS	Arguments in object accessible to the user
ALL_CATALOG	All tables, views, synonyms, sequences accessible to the user
ALL_COL_COMMENTS	Comments on columns of accessible tables and views
ALL_CONSTRAINTS	Constraint definitions on accessible tables
ALL_CONS_COLUMNS	Information about accessible columns in constraint definitions
ALL_DB_LINKS	Database links accessible to the user
ALL_ERRORS	Current errors on stored objects that user is allowed to create
ALL_INDEXES	Descriptions of indexes on tables accessible to the user
ALL_IND_COLUMNS	COLUMNs comprising INDEXes on accessible TABLES
ALL_LOBS	Description of LOBs contained in tables accessible to the user
ALL_OBJECTS	Objects accessible to the user
ALL_OBJECT_TABLES	Description of all object tables accessible to the user
ALL_SEQUENCES	Description of SEQUENCEs accessible to the user
ALL_SNAPSHOTS	Snapshots the user can access
ALL_SOURCE	Current source on stored objects that user is allowed to create
ALL_SYNONYMS	All synonyms accessible to the user
ALL_TABLES	Description of relational tables accessible to the user
ALL_TAB_COLUMNS	Columns of user's tables, views and clusters
ALL_TAB_COL_STATISTICS	Columns of user's tables, views and clusters
ALL_TAB_COMMENTS	Comments on tables and views accessible to the user
ALL_TRIGGERS	Triggers accessible to the current user
ALL_TRIGGER_COLS	Column usage in user's triggers or in triggers on user's tables
ALL_TYPES	Description of types accessible to the user
ALL_UPDATABLE_COLUMNS	Description of all updatable columns
ALL_USERS	Information about all users of the database
ALL_VIEWS	Description of views accessible to the user
DATABASE_COMPATIBLE_LEVEL	Database compatible parameter set via init.ora
DBA_DB_LINKS	All database links in the database
DBA_ERRORS	Current errors on all stored objects in the database
DBA_OBJECTS	All objects in the database
DBA_ROLES	All Roles which exist in the database
DBA_ROLE_PRIVS	Roles granted to users and roles
DBA_SOURCE	Source of all stored objects in the database
DBA_TABLESPACES	Description of all tablespaces
DBA_TAB_PRIVS	All grants on objects in the database
DBA_TRIGGERS	All triggers in the database
DBA_TS_QUOTAS	Tablespace quotas for all users
DBA_USERS	Information about all users of the database
DBA_VIEWS	Description of all views in the database
DICTIONARY	Description of data dictionary tables and views
DICT_COLUMNS	Description of columns in data dictionary tables and views
GLOBAL_NAME	global database name
NLS_DATABASE_PARAMETERS	Permanent NLS parameters of the database
NLS_INSTANCE_PARAMETERS	NLS parameters of the instance
NLS_SESSION_PARAMETERS	NLS parameters of the user session
PRODUCT_COMPONENT_VERSION	version and status information for component products
ROLE_TAB_PRIVS	Table privileges granted to roles
SESSION_PRIVS	Privileges which the user currently has set
SESSION_ROLES	Roles which the user currently has enabled.
SYSTEM_PRIVILEGE_MAP	Description table for privilege type codes. Maps privilege type numbers to type names
TABLE_PRIVILEGES	Grants on objects for which the user is the grantor, grantee, owner, or an enabled role or PUBLIC is the grantee
TABLE_PRIVILEGE_MAP	Description table for privilege (auditing option) type codes. Maps privilege (auditing option) type numbers to type names

Mandatory security in SQL

The topic was introduced above. First, classify the subjects (users and their agents) and the database objects concerned. Classify the means by which each has to be assigned a number indicating the security level, as it will be enforced by the applied rules (e.g. the Bell-LaPadula model (2005)).

The classification has to apply by table, by tuple, by attribute and by attribute value as appropriate to the requirement. Separate classifications may be needed to deal with create (INSERT), read (SELECT), UPDATE and DELETE permissions - though this will depend on the rules to be applied. The rules themselves may relate these options as they do in the model quoted.

Additional controls may be needed to deal with statistical security - these might restrict the number of tuples that can be retrieved, or add inaccuracies to the retrieved data, or provide controls on allowed query sequences (to identify and restrict users unnecessarily performing extractive analysis). Design considerations need to decide whether sensitive tuples should be maintained on the database and, if so, how.

So this converts mandatory security to the design of a security database (all the above information has to be stored in tables) with the associated transaction design, to implement the access rules and to control access to the security tables themselves. Mandatory security has to be built from the discretionary SQL tools.

Review question 1

- Distinguish discretionary from mandatory security.
- Describe the log-in process step by step.
- Explain the nature and use of surrogate information.
- Explain the implementation of access control in a distributed database.

Review question 2

What are the threats in the case below? Explain the nature of the threats. Write these down (one paragraph) before you read the notes. If you feel you need more information explain what you need to know?

Case

A senior manager is recorded as being in his office late one night. Subsequently at the time he was in his office the audit trail records several unsuccessful attempts to access database objects using a password of a member of clerical staff to objects to which the manager had no rights of access.

Notes pages

Data protection

Treat the following principles as abstract. Every company that has implemented data protection has followed these guides but, as usual, 'the devil is in the detail'. If you can be sure your database system complies with these you have done well. (And are you happy with storing personal images?)

- The information to be contained in personal data shall be obtained and personal data shall be processed fairly and lawfully.
- Personal data shall be held only for one or more specified and lawful purposes.
- Personal data held for any purpose or purposes shall not be used or disclosed in any manner incompatible with that purpose or those purposes.
- Personal data held for any purpose or purposes shall be adequately relevant and not excessive in relation to that purpose or those purposes.
- Personal data shall be accurate and, where necessary, kept up-to-date.
- Personal data held for any purpose or purposes shall not be kept for longer than is necessary for that purpose or purposes.
- An individual shall be entitled - at reasonable intervals and without undue delay or expense -

1. to be informed by any data user if he/she holds personal data of which the individual is the subject;
2. to access any such data held by a data user; and
3. where appropriate, to have such data corrected or erased.

- Appropriate security measures shall be taken against unauthorised access to, or alteration, disclosure or destruction of, personal data and against loss or destruction of personal data.

Computer misuse

Hacking offences

These are:

- Simple unauthorised access – merely accessing data to which you are not entitled. The law is not concerned with the systems control per se.
- Unauthorised access (with intent to commit an offence) – so you don't actually need to have succeeded, but rather just to have intended, to do something to the database.

- Unauthorised modification – no one can even attempt access without making some changes, but the purpose is to penalise outcomes.

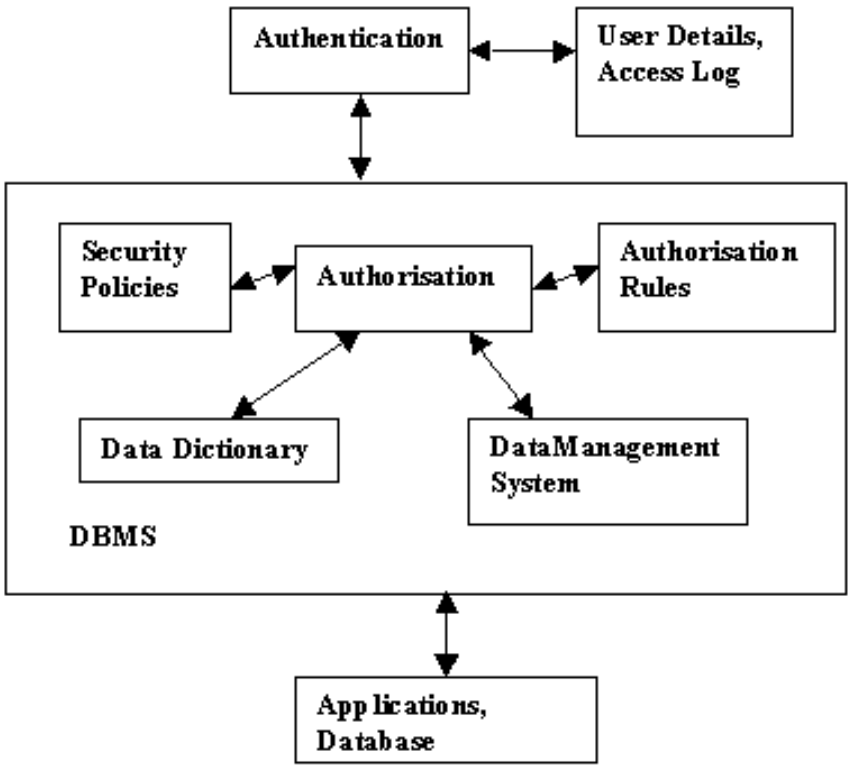
Clearly systems must record all accesses and attempted accesses, whether remote or local. Failure to do so may be seen as negligence, conspiracy or condoning on the part of system managers and designers.

Offences of introducing viruses are often covered in misuse legislation but are not included here. Yet viruses that attack DBMSs are of course possible.

Security plan

- Identify the user community.
- Gather the database information.
- Determine the types of user account (i.e. associate database objects and user roles).
- Undertake a threat analysis.
- Establish DBA authorities and procedures.
- Establish policies for managing (creating, deleting, auditing) user accounts.
- Determine the user tracking policy.
- Establish the user identification method.
- Define security incidents and reporting procedure.
- Assess the sensitivity of specific data objects.
- Establish standards and enforcement procedures (as well as back-up and recovery plans, of course).

Authentication and authorisation schematic



Authentication and authorisation

Generally speaking, when you log into a system, you want to be satisfied that you have logged into the right system and the system equally wants to be satisfied that you are who you claim to be. Think about this with an Internet Banking System as an example. Could a line be intercepted and the person on the other side pretend to be the bank while you disclosed passwords and account numbers? Could someone access the bank and empty your accounts? The part of the process that deals with this area is the authentication server.

There are only two ways to establish authentication: by means of shared secrets – things known only to you and the system - or by appealing to another system that has gone through the same process and is trusted. It is the second point that enables distributed transactions, at least without multiple log-ins. Shared secrets include passwords and personal information as reported to and stored

by the system. In each case, a dialogue appropriate to the situation has to be developed, monitored and managed. In the latter case, notice the need for secure communication between different system components, as secret information or its surrogate has to be transmitted between systems.

The result of authentication is a vector that contains an authentication identifier, usually with other information including date and time. Note that authentication is quite separate from access to database resources. You need to have obtained an authentication identifier before you start accessing the database. See the extension reading for some further insight on authentication.

In an SQL database system, the authentication process is initiated by the CONNECT statement. After successful execution of CONNECT, the resources of the database become potentially available.

Authorisation is permission given by the DBMS to use defined database resources and is based on the authentication identifier and a record of the permissions the owner of that identifier has. The identifier is recorded with the objects the user is allowed to access and the actions that can be performed on those objects. These are recorded either as a separate security database or as part of the normal system tables – accessible only to those with DBA privileges and higher.

Each SQL object has an owner. The owner has privileges on the objects owned. No other user can have any privileges (or even know the object exists) unless the owner supplies the necessary permission. In normal development, the DBA or system administrator will be the owner of the major assets. The scheme is basically discretionary. You need to look at security in SQL more as a toolkit than a solution.

Access control in SQL is implemented by transactions using the GRANT statement. These associate privileges with users and assets. Assets include data items (tables, views) and the privileges are the responsibility of the asset owner.

Access control activities

Overview

The exercise here is a simple implementation from a problem statement. You will produce the schema for a simple database (three tables), populate it with data, and then establish the access controls, test them and review the effect. Suggestions are then made for you to investigate some of the system tables.

The problem

You have received the following account of the access security requirements for an SQL database system.

The database has three tables: CUSTOMER (keyed on name and street number); ORDER (keyed on date-of-receipt and the foreign key from CUSTOMER), and ORDER-ITEM (keyed on item-name and the foreign key from ORDER).

The system is to process sales orders. Entry and access to documents and the progressing of orders is handled by the sales office. There are currently seven staff in this office. Tracey, the supervisor, needs to be able to see and change everything. Bill, Sheila and Govind do most of the routine work, but especially they cannot create new customers.

There are a few other things. Make sure that large orders, say above £1000, cannot be handled by anyone except Tracey and Govind. Also ensure that a temporary member of staff can process orders but cannot see the customer details.

1. Produce an analysis and the necessary SQL statements to handle the problems.
2. Implement the database, populate it with test data and apply your security statements.
3. Test your access control mechanism thoroughly.
4. If, when you see the sample solution, either your analysis or testing are faulty, you need to find out why.
5. Indicate any additional SQL security measures that could be taken.
6. Comment on the strengths and weaknesses of the measures you have taken.

The following activities will help you through solving the problem, step by step.

Activity 1 – Creating the database schema

Using the data definition statements from SQL, produce a script to create three tables (table 1 is related to many rows of table 2 which is related to many rows of table 3).

You may already have a database schema which can be used for this purpose, but in any case you must ensure that the primary and foreign keys are correctly declared.

Activity 2 – Populating the database

This is a security exercise so the data itself need not be to normal test data standards. Make sure that the data is easily recognisable – use meaningful values and remember that foreign key fields should be present.

Activity 3 – Analysing the problem

Treat the names given as user-roles (otherwise the problem becomes too obscure for a first attempt). Develop a matrix of user against database resource. For this exercise, the database resources in question are the base tables and views (virtual tables).

Your matrix will make it clear where access is to be permitted and to whom. These requirements translate directly to an SQL script of GRANT statements. Produce this script on paper and check it manually.

Activity 4 – Executing the security script (if you have a DBMS that permits this)

If you don't have an executing environment, then get a friend to critique your work so far.

Activity 5 – Testing the access control (if you have a DBMS that permits this)

Formulate SELECT statements from the problem specification and issue SELECT statements to check that they have been correctly implemented. If you find problems, correct them at the point they occurred.

Activity 6 – Conclusion

Indicate any additional SQL security measures that could be taken, and comment on the strengths and weaknesses of the measures you have taken.

Activity 7 – Postscript

Well, how well did you do?

Remember Tracey?

After your work, she thought she should have a raise. She asked and was refused and then returned to her desk. To answer these questions, refer to your database design and security script.

1. Tracey then tried to delete the CUSTOMER table. Did she succeed?
2. I hope not, but if so, why? Did you not inadvertently give her SYSADM privileges?
3. She then tried to delete some customers. Did she succeed? Did the deletes cascade?

4. She tried to insert a line in all orders over £1000 for 500 coffee machines. Did she succeed?
5. And how was the problem detected?
6. She tried to change her password? Did she succeed?
7. How much privilege can any one individual ever be given?