Lecture 10 – General Database Issues and Security

1) Transactions

A transaction is a *logical unit of work*. For example if Person X asks their bank to transfer £1000 from their current account to their deposit account. If after debitting their current account with £1000, the system crashes before crediting their deposit account with the same amount, then the balance will be incorrect. So either both steps have to take place or neither. This is termed a logical unit of work or *transaction*.

Operations of a Transaction

A database is treated as a set of data items and disk blocks which are accessed by transactions. A transaction accesses or modifies the contents of a database. A transaction with only read operations is known as a *read-only* transaction. A transaction with write operations is known as an *update* transaction.
Additional operations of a transaction, which can be explicitly specified, are:

- **COMMIT** - the transaction is successful and the data items value must be changed (if any).
- **ROLLBACK/ABORT** - the transaction is not successful do not change any of the data item values.

Why do you need the concept of Transaction?

- to maintain database consistency over time (atomicity).
- to evaluate if multiple users can simultaneously access and modify the data (concurrency)
- to make the changes to data permanent (durability).

2) Concurrency

Why is Concurrency Control needed?

In general, a transaction consists of a set of data items that are accessed (read set), and a set of data items that are modified (write set). Several problems can occur when concurrent transactions (ie several transactions executing at the same time on the same data) execute in an uncontrolled manner. For example:

(a) Lost Update Problem:
A transaction *overwrites* a data item modified by other transactions.

(b) Temporary Update or Dirty Read:
A transaction reads *uncommitted* modified data item values updated by other transactions.

(c) Incorrect Summary Problem:
A transaction reads *partially updated* data item values from other transactions.

3) Why Recovery is Needed

When a transaction is submitted to a DBMS for execution, the system is responsible for making sure that either

- all the operations in the transaction are completed successfully and their effect is recorded permanently in the database, or
- the transaction has no effect whatsoever on the database or on any other transactions

The DBMS must not permit some operations of a transaction to be applied to the database while other operations of the transaction are not. This may happen if a transaction fails after executing some of its operations but before executing all of them.

Types of Failures

1 A computer failure (system crash)
2 A transaction or system error
3 Local errors or exception conditions detected by the transaction
4 Concurrency control enforcement
5 Disk failure
6 Physical problems and catastrophes

Types 1-4 are more common types of failures, whenever a failure of type 1 through 4 occurs, the system must keep sufficient information to recover from the failure.

Failures in DBMSs

Transaction Failures

- Transaction aborts due to software bugs, or deadlocks
- Average of 3% of transactions abort abnormally (not user-intended)

Media Failures

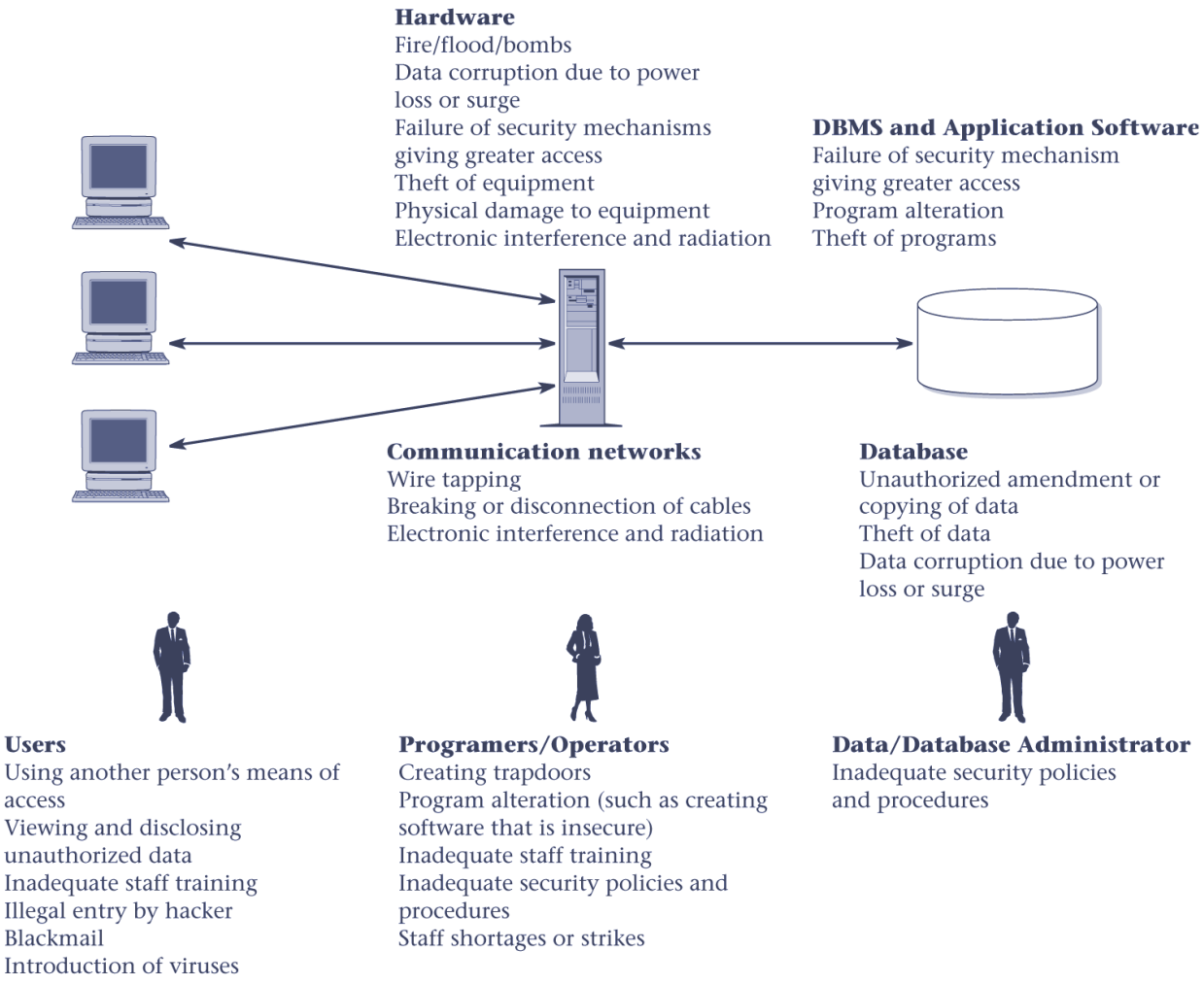
- Failure of the secondary storage devices.

4) Database Security

Purpose And Scope Of Database Security

Security considerations do not only apply to the data held in a database. Breaches of security may affect other parts of the system, which may in turn affect the database. Consequently, database security encompasses hardware, software, people, and data. To effectively implement security requires appropriate controls, which are defined in specific mission objectives for the system. This need for security, while often having been neglected or overlooked in the past, is now increasingly recognized by organizations. The reason for this turn-around is due to the increasing amounts of crucial corporate data being stored on computer and the acceptance that any loss or unavailability of this data could be potentially disastrous.

The main types of threat that could affect a database system and the possible outcomes for an organization.



A summary of the potential threats to computer systems.

Means Of Providing Security For A Database

1) Authorization

Authorization is the granting of a right or privilege that enables a subject to have legitimate access to a system or a system's object. Authorization controls can be built into the software, and govern not only what database system or object a specified user can access, but also what the user may do with it. The process of authorization involves authentication of a subject requesting access to an object, where 'subject' represents a user or program and 'object' represents a database table, view, procedure, trigger, or any other object that can be created within the database system.

2) Views

A view is a *virtual table* that does not necessarily exist in the database but can be produced upon request by a particular user, at the time of request. The view mechanism provides a powerful and flexible security mechanism by hiding parts of the database from certain users. The user is not aware of the existence of any columns or rows that are missing from the view. A view can be defined over several tables with a user being granted the appropriate privilege to use it, but not to use the base tables. In this way, using a view is more restrictive than simply having certain privileges granted to a user on the base table(s).

3) Backup and recovery

Backup is the process of periodically taking a copy of the database and log file (and possibly programs) onto offline storage media. A DBMS should provide backup facilities to assist with the recovery of a database following failure. To keep track of database transactions, the DBMS maintains a special file called a log file (or journal) that contains information about all updates to the database. It is always advisable to make backup copies of the database and log file at regular intervals and to ensure that the copies are in a secure location. In the event of a failure that renders the database unusable, the backup copy and the details captured in the log file are used to restore the database to the latest possible consistent state. Journaling is the process of keeping and maintaining a log file (or journal) of all changes made to the database to enable recovery to be undertaken effectively in the event of a failure.

4) Integrity constraints

Contribute to maintaining a secure database system by preventing data from becoming invalid, and hence giving misleading or incorrect results.

5) Encryption

Is the encoding of the data by a special algorithm that renders the data unreadable by any program without the decryption key. If a database system holds particularly sensitive data, it may be deemed necessary to encode it as a precaution against possible external threats or attempts to access it. Some DBMSs provide an encryption facility for this purpose. The DBMS can access the data (after decoding it), although there is degradation in performance because of the time taken to decode it. Encryption also protects data transmitted over communication lines. There are a number of techniques for encoding data to conceal the information; some are termed irreversible and others reversible. *Irreversible techniques*, as the name implies, do not permit the original data to be known. However, the data can be used to obtain valid statistical information. *Reversible techniques* are more commonly used. To transmit data securely over insecure networks requires the use of a cryptosystem, which includes:

- an encryption key to encrypt the data (plaintext);
- an encryption algorithm that, with the encryption key, transforms the plain text into ciphertext;
- a decryption key to decrypt the ciphertext;
- a decryption algorithm that, with the decryption key, transforms the ciphertext back into plain text.

6) Redundant Array of Independent Disks (RAID)

RAID works by having a large disk array comprising an arrangement of several independent disks that are organized to improve reliability and at the same time increase performance. The hardware that the DBMS is running on must be *fault-tolerant*, meaning that the DBMS should continue to operate even if one of the hardware components fails. This suggests having redundant components that can be seamlessly integrated into the working system whenever there is one or more component failures. The main hardware components that should be fault-tolerant include disk drives, disk controllers, CPU, power supplies, and cooling fans. Disk drives are the most vulnerable components with the shortest times between failures of any of the hardware components.

One solution is the use of Redundant Array of Independent Disks (RAID) technology. RAID works by having a large disk array comprising an arrangement of several independent disks that are organized to improve reliability and at the same time increase performance.

Database Security and the DBA

DBA has privilege account, with the ability to use privileged commands to creating and managing users accounts by:

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

Types of Database Security

Discretionary security mechanisms: grant previleges to users (create, read, write previleges).
Mandatory security mechanisms: multilevel security by classifying data and users into various security classes (or levels) and then implementing the appropriate security policy of the organization.
A DBMS typically has a database security and authorization subsystem that is reponsible for ensuring security of portions of a database against unauthorized access.
We shall cover only discretionary security mechanisms.

Access Control

- Based on the granting and revoking of privileges.
- A privilege allows a user to create or access (that is read, write, or modify) some database object (such as a relation, view, and index) or to run certain DBMS utilities.
- Privileges are granted to users to accomplish the tasks required for their jobs.
- Most DBMS provide an approach called Discretionary Access Control (DAC).
- SQL standard supports DAC through the GRANT and REVOKE commands.
- The GRANT command gives privileges to users, and the REVOKE command takes away privileges.
- DAC while effective has certain weaknesses. In particular an unauthorized user can trick an authorized user into disclosing sensitive data.
- An additional approach is required called Mandatory Access Control (MAC).
- DAC is based on system-wide policies that cannot be changed by individual users.
- Each database object is assigned a security class and each user is assigned a clearance for a security class, and rules are imposed on reading and writing of database objects by users.
- DAC determines whether a user can read or write an object based on rules that involve the security level of the object and the clearance of the user. These rules ensure that sensitive data can never be ‘passed on’ to another user without the necessary clearance.
- The SQL standard does not include support for MAC.

GRANT

- **GRANT** gives privileges to users for specified objects.

```
GRANT      {PrivilegeList | ALL PRIVILEGES}
ON         ObjectName
TO         {AuthorizationIdList | PUBLIC}
[WITH GRANT OPTION]
```

- PrivilegeList consists of one or more of above privileges separated by commas.
- **ALL PRIVILEGES** grants all privileges to a user.
- **PUBLIC** allows access to be granted to all present and future authorized users.
- ObjectName can be a base table, view, domain, character set, collation or translation.
- **WITH GRANT OPTION** allows privileges to be passed on.
- Example - Give all users **SELECT** access on Branch table.

```
GRANT SELECT
ON   Branch
TO PUBLIC;
```

REVOKE

- **REVOKE** takes away privileges granted with **GRANT**.

```
REVOKE [GRANT OPTION FOR]
       {PrivilegeList | ALL PRIVILEGES}
ON ObjectName
FROM {AuthorizationIdList | PUBLIC}
       [RESTRICT | CASCADE]
```

- **ALL PRIVILEGES** refers to all privileges granted to a user by user revoking privileges.
- **GRANT OPTION FOR** allows privileges passed on via **WITH GRANT OPTION** of **GRANT** to be revoked separately from the privileges themselves.
- **REVOKE** fails if it results in an abandoned object, such as a view, unless the **CASCADE** keyword has been specified.
- Privileges granted to this user by other users are not affected.
- Example - Revoke privilege SELECT on Branch table from all users.

```
REVOKE SELECT
ON     Branch
FROM PUBLIC;
```

References and Further Reading:

T CONNOLLY & C BEGG. *Database Systems – A Practical Approach to Design, Implementation and Management,* 4th Edition. Addison-Wesley, 2005.

- Chapters on Security and SQL Data Definition.

T CONNOLLY & C BEGG. *Database Solutions – A step-by-step guide to building databases,* 2nd Edition. Addison-Wesley, 2004.

- Chapter on Database Administration and Security.

Notes on *3ISY402 - Database Systems* module by Dr. Kamalasen Rajalingham.