



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David



DATABASE SECURITY

Week:
28-Nov-2023

Database Security: Database Systems (DBS)
Dr Suresh Shanmugasundaram

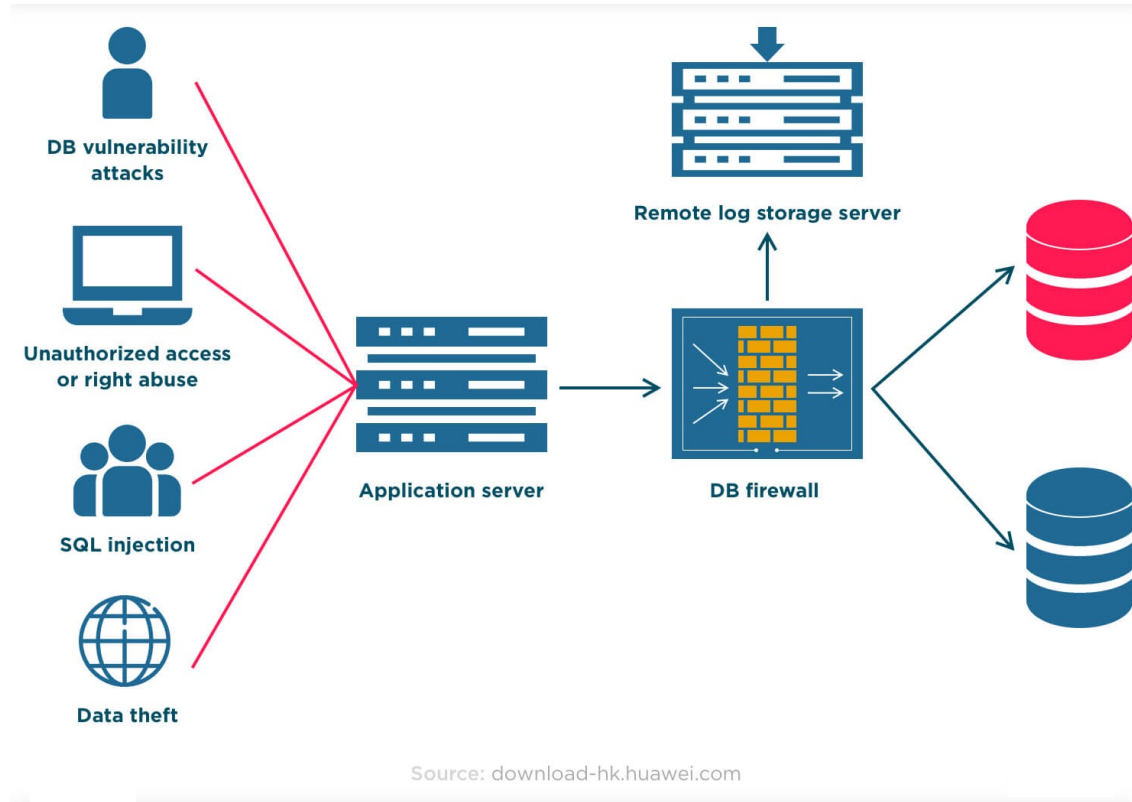
Database Security Objectives



- ❑ The scope of database security.
- ❑ Why database security is a serious concern for an organization.
- ❑ The type of threats that can affect a database system.
- ❑ How to protect a computer system using computer-based controls.
- ❑ Approaches for securing a DBMS on the Web.

Database Security

- Data is a valuable resource that must be strictly controlled and managed, as with any corporate resource.
- Part or all of the corporate data may have strategic importance and therefore needs to be kept secure and confidential.



Database Security

- ❑ Mechanisms that protect the database against intentional or accidental threats.
- ❑ 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.

Database Security

- ❑ Involves measures to avoid:
 - ❑ Theft and fraud
 - ❑ Loss of confidentiality (secrecy)
 - ❑ Loss of privacy
 - ❑ Loss of integrity
 - ❑ Loss of availability

Database Security

- Threat
 - ▣ Any situation or event, whether intentional or unintentional, that will adversely affect a system and consequently an organization.

Database Security - Confidentiality



A company maintains a database with employee information, including salaries. What measure ensures the confidentiality of this data?

- A. Regular database backups
- B. Strong encryption of sensitive fields
- C. Increased database availability
- D. Publicly accessible database links

Database Security - Confidentiality



In a financial database handling transactions, what protects sensitive customer information from unauthorized access?

- A. Frequent data updates
- B. Role-based access control
- C. Optimized database performance
- D. Unencrypted communication channels

Database Security - Integrity



How can a healthcare database ensure the integrity of patient records against unauthorized modifications?

- A. Regular database indexing
- B. Hashing and checksums
- C. Increased database replication
- D. Relaxed access controls

Database Security - Integrity



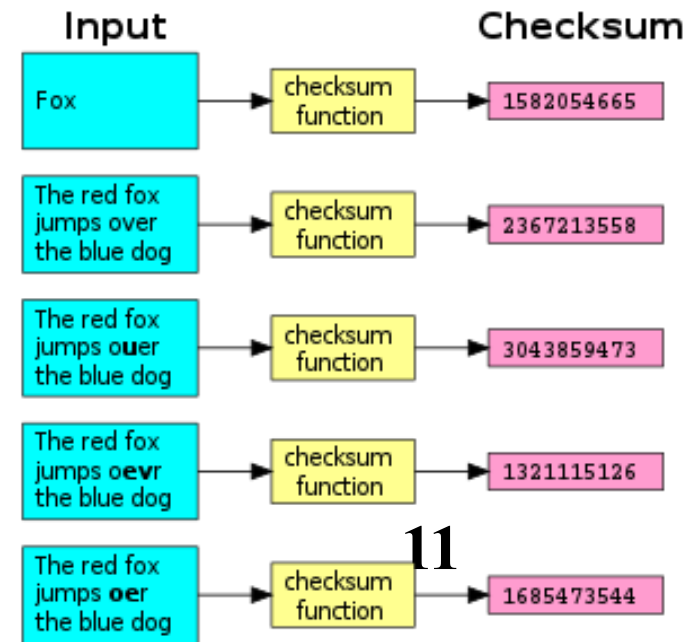
How can a healthcare database ensure the integrity of patient records against unauthorized modifications?

- A. Regular database indexing
- B. Hashing and checksums
- C. Increased database replication
- D. Relaxed access controls

Database Security - Integrity

How can a healthcare database ensure the integrity of patient records against unauthorized modifications?

- A. Regular database indexing
- B. Hashing and checksums
- C. Increased database replication
- D. Relaxed access controls



Database Security - Integrity



To prevent tampering with student grades in a university database, what is a recommended integrity measure?

- A. Open database access policies
- B. Real-time database monitoring
- C. Strong authentication mechanisms
- D. Regular database defragmentation

Database Security - Availability



In an online retail database, what ensures high availability, especially during peak shopping times?

- A. Efficient query optimization
- B. Distributed database architecture
- C. Encrypted database connections
- D. Frequent database schema changes

Database Security - Availability



In an online retail database, what ensures high availability, especially during peak shopping times?

- A. Efficient query optimization
- B. Distributed database architecture
- C. Encrypted database connections
- D. Frequent database schema changes

Database Security - Availability



How can the availability of an emergency services database be maintained during network disruptions?

- A. Regular database backups
- B. Geographically distributed database servers
- C. Weakened database authentication
- D. Reducing database redundancy

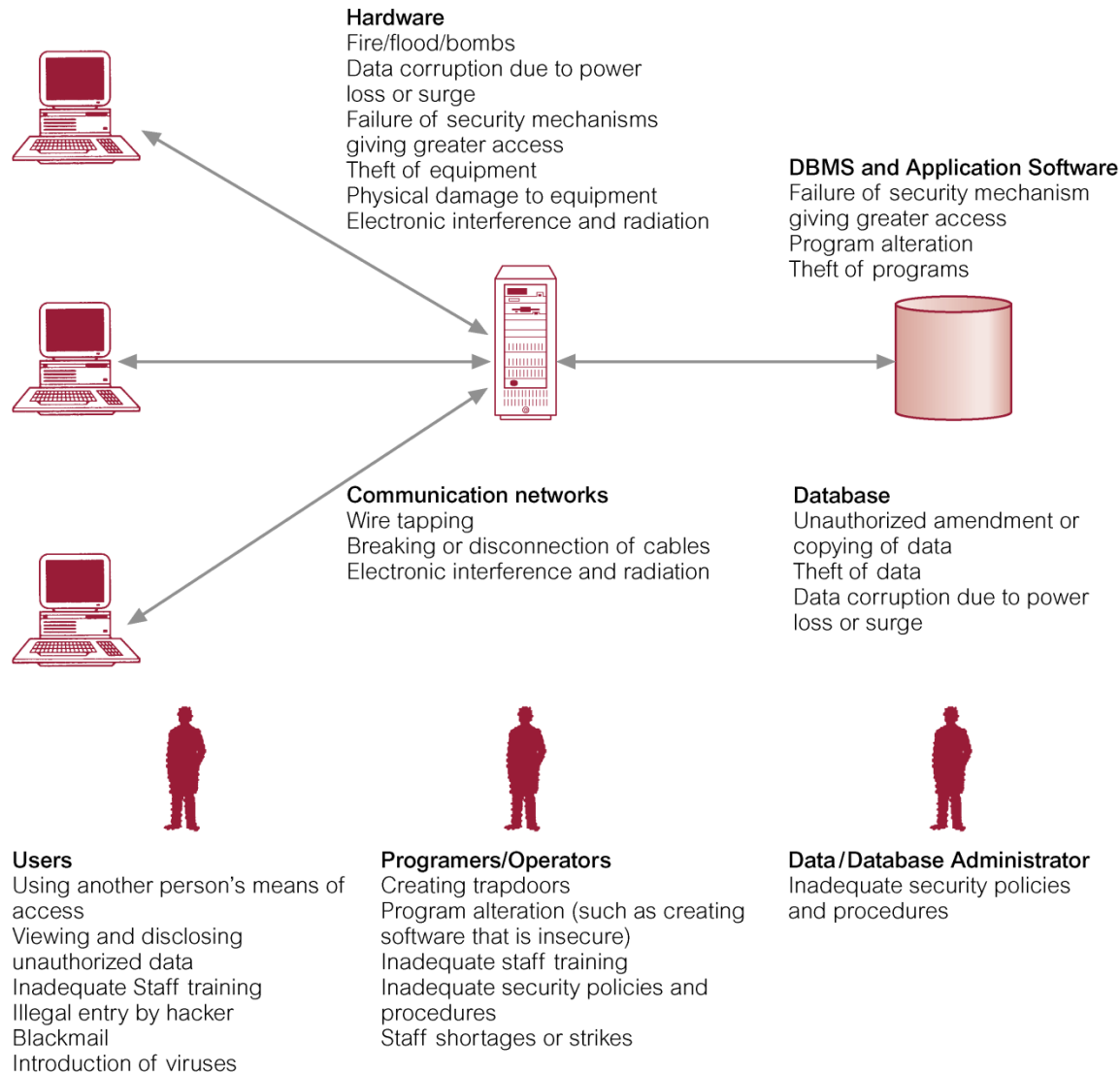
Database Security - CIA



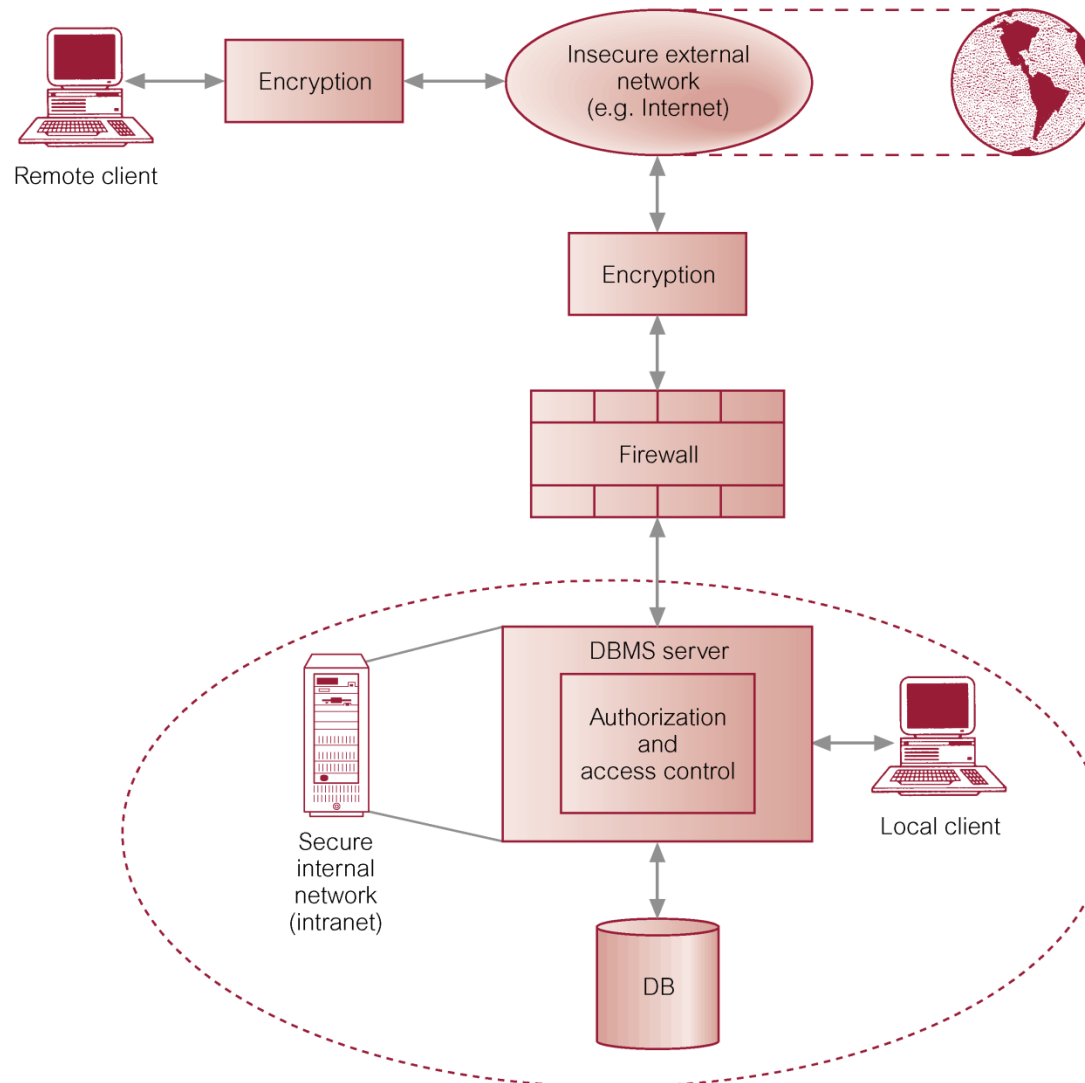
In an e-government database system, what is crucial to ensure the trustworthiness of the services provided?

- A. Limiting database backups
- B. Auditing and logging database activities
- C. Simplifying database access controls
- D. Delaying software updates for the database system

Summary of Threats to Computer Systems



Typical Multi-user Computer Environment



Countermeasures – Computer-Based Controls

- Concerned with physical controls to administrative procedures and includes:
 - ▣ Authorization
 - ▣ Access controls
 - ▣ Views
 - ▣ Backup and recovery
 - ▣ Integrity
 - ▣ Encryption
 - ▣ RAID technology

Physical and Logical Security

Physical Security

- facility monitoring (surveillance systems, cameras, guards, exterior lighting)
- access controls to facilities/data center/computers (biometrics, access cards)
- alarm systems (fire, burglar, water, humidity, power fluctuations)
- shred sensitive documents
- proper storage/disposal of hard drives and other electronic storage media
- secure storage of backup copies of data and master copies of critical software

Logical Security

- e-IDs and passwords
- system authentication
- biometrics
- logs of logon attempts
- application-level firewalls
- anti-virus and anti-spyware software
- intrusion detection systems
- encryption for data in transit
- smart cards

Design Level DB Security

- a. Database design should be simple (possibility of the data being corrupted by the authorized user is less)
- b. Database must be normalized (normalized database is almost free from update anomalies. Harder to impose normalization once the relations are created)
- c. Database designer should decide the privilege for each group of users (this will prevent illegal access by users to most extent)
- d. Create unique view for each user/group of users (view does provide some security, but not always adequate)

Maintenance Level DB Security

- DBA (Database Administrator)'s role.
 - a. Operating system issues and availability
 - b. Confidentiality and accountability through authorization rules
 - c. Encryption
 - d. Authentication Schemes

DB Security Through Access Control

- Two main mechanisms of access control:
 - a. **Discretionary Access Control:** regulates user access to named objects through **privileges**
 - b. **Mandatory Access Control:** based on system-wide **policies** that cannot be changed by individual users.

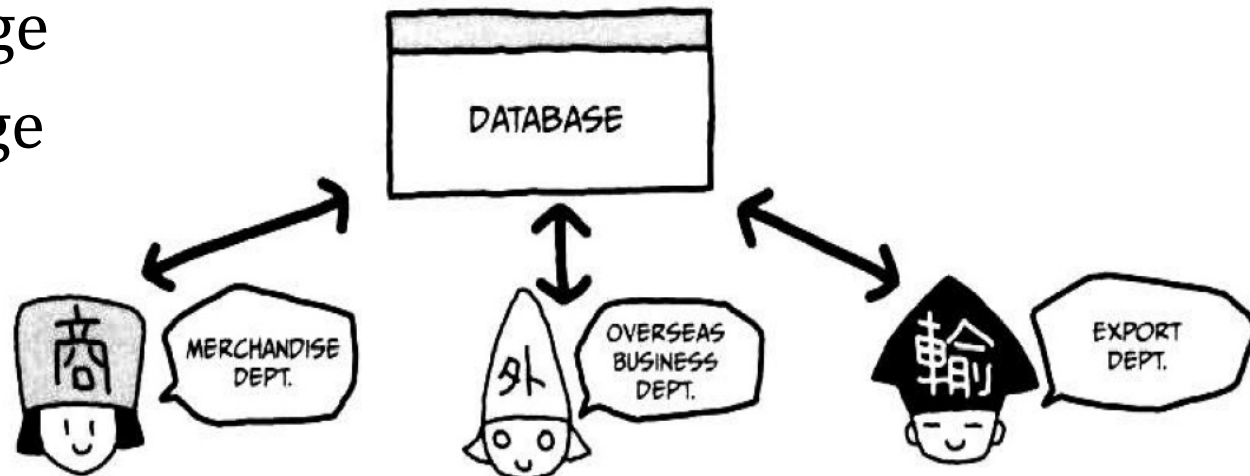
DBMS Security Support

- DBMS can provide some security
 - Each database user has an **account, username & password**
 - These are used to identify a user and control their access to information
- DBMS verifies password and checks a user's permissions when they try to
 - Retrieve data
 - Modify data
 - Modify the database structure

Permissions and Privilege

- ❑ SQL uses privileges to control access to tables and other database objects
 - ▣ SELECT privilege
 - ▣ INSERT privilege
 - ▣ UPDATE privilege
 - ▣ DELETE privilege

Granting privileges



Privileges in SQL

GRANT <privileges>

ON <object>

TO <users>

[WITH GRANT OPTION]



<privileges> is a list of

**SELECT <columns>, INSERT <columns>, DELETE, and
UPDATE <columns>, or simply ALL**

<users> is a list of usernames or PUBLIC

<object> is the name of a table or view

WITH GRANT OPTION means that the users can pass their privileges on to others

Privileges Examples



```
GRANT ALL ON Fruit  
TO King  
WITH GRANT OPTION;
```

```
GRANT SELECT, UPDATE(Quantity) ON  
Fruit TO Farmer;
```

Privileges Examples

GRANT ALL ON Fruit

TO King

WITH GRANT OPTION;

The user 'King' can do anything to the Fruit table, and can allow other users to do the same (by using **GRANT** statements)

GRANT SELECT, UPDATE(Quantity) ON

Fruit TO Farmer;

The user 'Farmer' can view the entire Fruit table, and can change Quantity values, but cannot change other values or pass on their privilege

Privileges Examples

GRANT SELECT ON Users TO 'Suresh' @ 'localhost';

GRANT SELECT, INSERT, DELETE, UPDATE ON Users TO 'Suresh'@'localhost';

GRANT SELECT ON Users TO '*'@'localhost';

What

privilege(s) is/are granted?

table(database object) on which these privileges apply?

usernames to which these were granted?

Removing Privileges

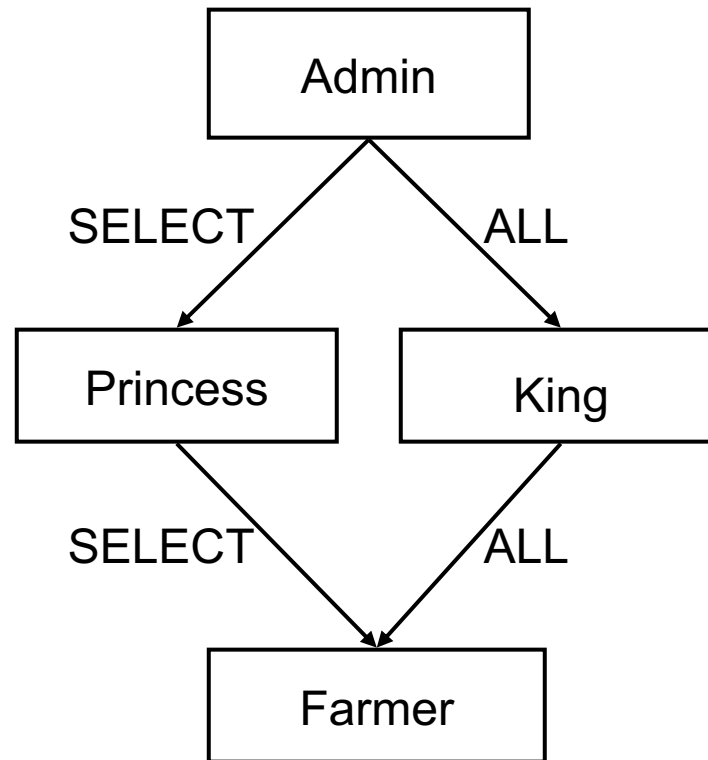
- If you want to remove a privilege you have granted you use

**REVOKE <privileges>
ON <object>
FROM <users>**

- If a user has the same privilege from other users then they keep it
- All privileges dependent on the revoked one are also revoked

Granting Privileges

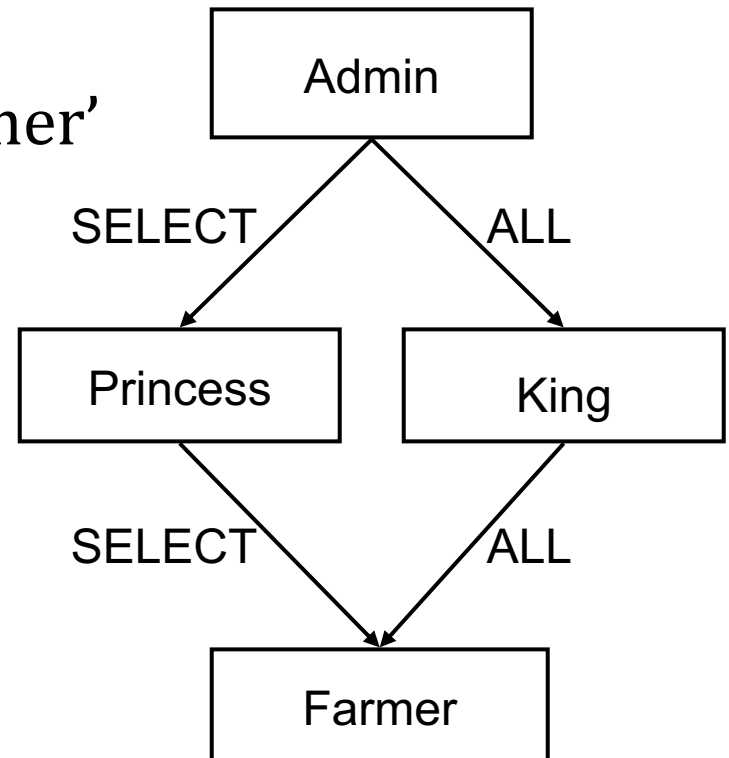
□ Example



Granting Privileges

□ Example

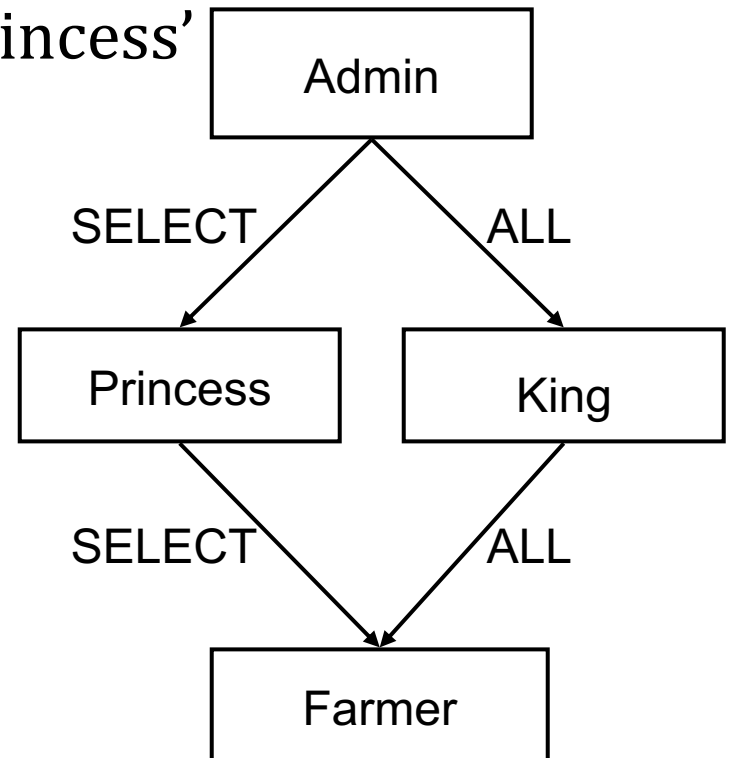
- 'Admin' grants ALL privileges to 'King', and SELECT to 'Princess' with grant option
- 'King' grants ALL to 'Farmer'
- 'Princess' grants SELECT to 'Farmer'



Removing Privileges

□ Example

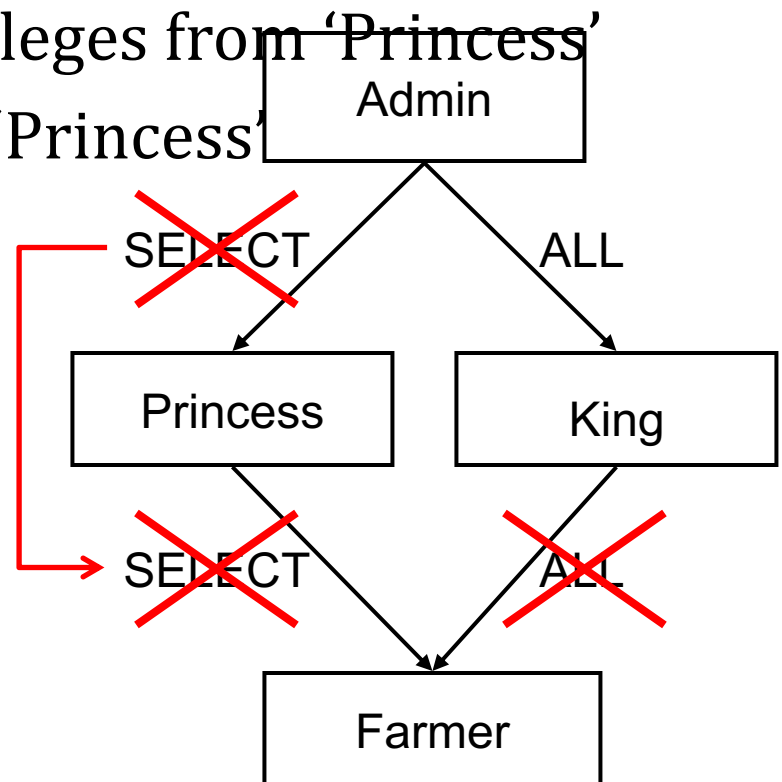
- 'King' revokes ALL from 'Farmer'
- 'Farmer' still has SELECT privileges from 'Princess'
- 'Admin' revokes SELECT from 'Princess'
- 'Farmer' loses SELECT also



Removing Privileges

□ Example

- ▣ 'King' revokes ALL from 'Farmer'
- ▣ 'Farmer' still has SELECT privileges from 'Princess'
- ▣ 'Admin' revokes SELECT from 'Princess'
- ▣ 'Farmer' loses SELECT also



Views

- Privileges work at the level of tables
 - ▣ You can restrict access by column
 - ▣ You cannot restrict access by row
- Views, along with privileges, allow for customised access
- Views provide 'derived' tables
 - ▣ A view is the result of a SELECT statement which is treated like a table
 - ▣ You can SELECT from (and sometimes UPDATE etc) views just like tables

Creating Views

CREATE VIEW <name>

AS <select stmt>

<name> is the name of the new view

<select stmt> is a query that returns the rows and columns of the view

□ Example

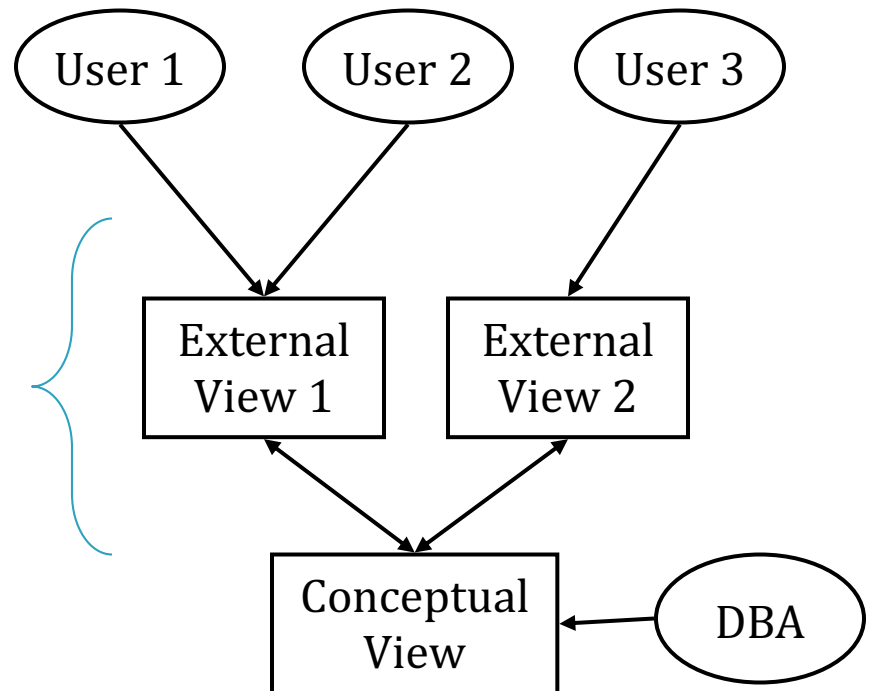
- We want each user to be able to view the names and phone numbers (only) of those employees in their own department
- In SQL, you can refer to the current user as **USER()**

View Example

```
CREATE VIEW OwnDept AS  
SELECT Name, Phone FROM Employee  
WHERE Department =  
  (SELECT Department FROM Employee  
   WHERE name = USER());  
  
GRANT SELECT ON OwnDept TO USER;
```

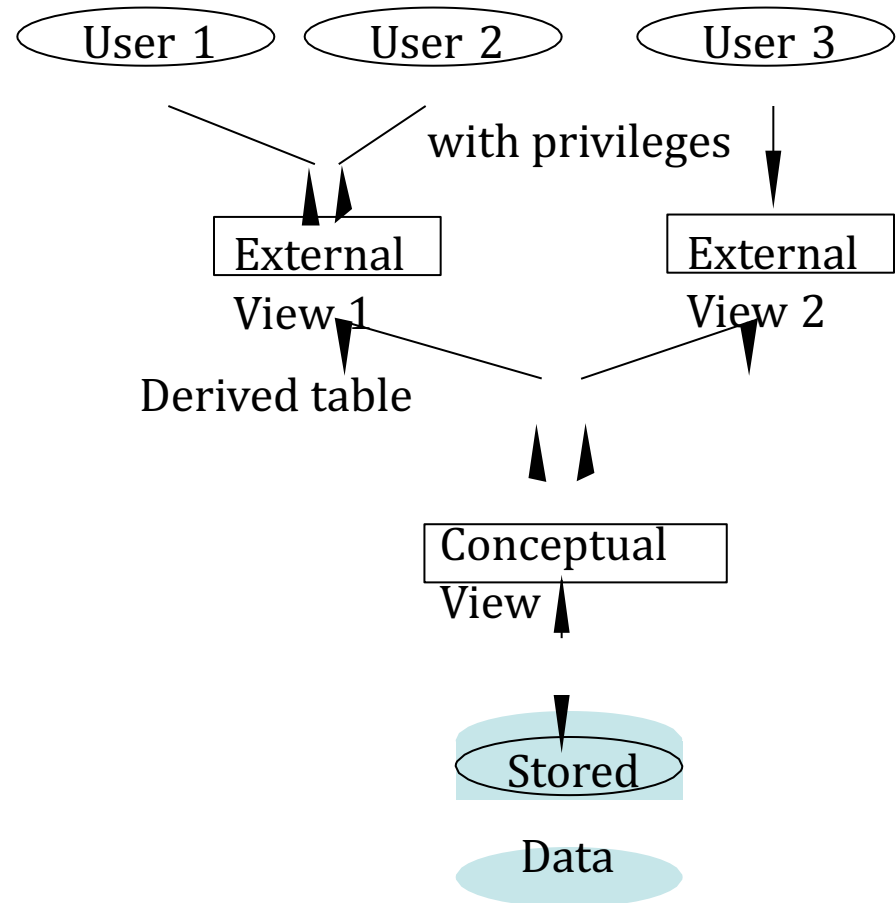
Using Views and Privileges

- ❑ Views and privileges are used together to control access
 - ▣ A view is made which contains the **necessary** information **only**
 - ▣ **Privileges** are granted to that **view**, rather than the **underlying tables**
 - ▣ **Revoke** any privileges they have on the original table



Using Views and Privileges

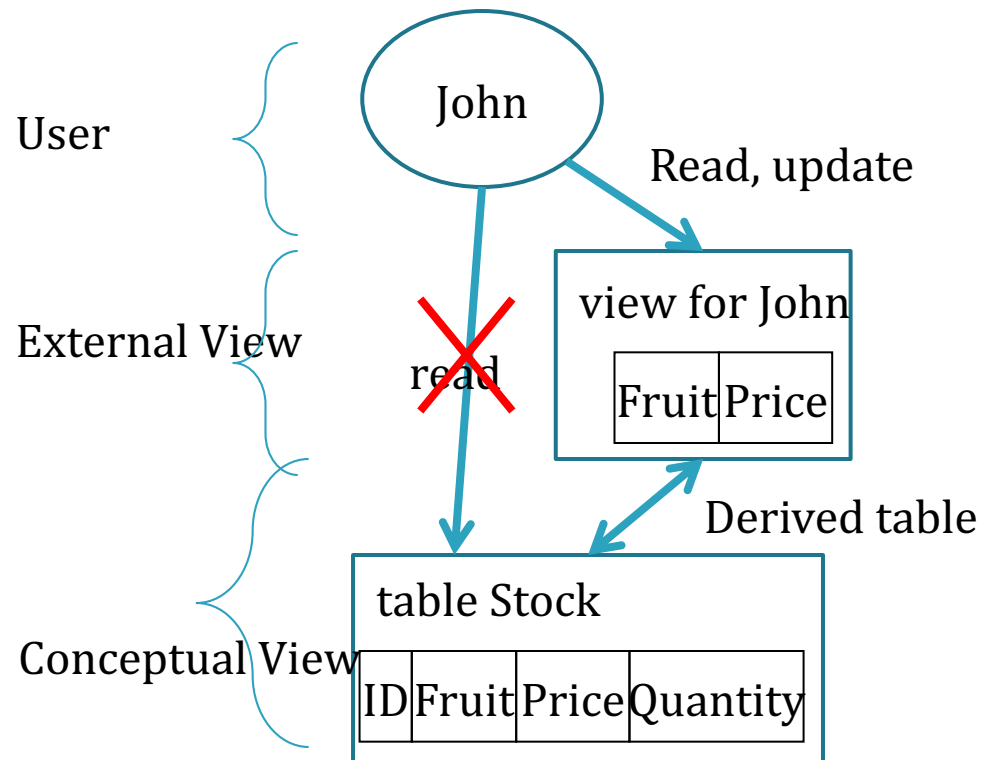
- Views and privileges are used together to control access
 - ▣ A **view** is made which contains the **necessary** information **only**
 - ▣ **Privileges** are granted to / revoke from **view**, rather than the underlying tables



Using Views and Privileges

Example

- User 'John' originally allowed to **read** from Stock
- We want to **revoke** 'John' access to Stock table, but let him **read** the fruit and price, and be able to **update** the price (only)



Using Views and Privileges

Example

1. Create a view

```
CREATE VIEW for John  
AS SELECT Fruit, Price  
FROM Stock
```

2. Set the privileges

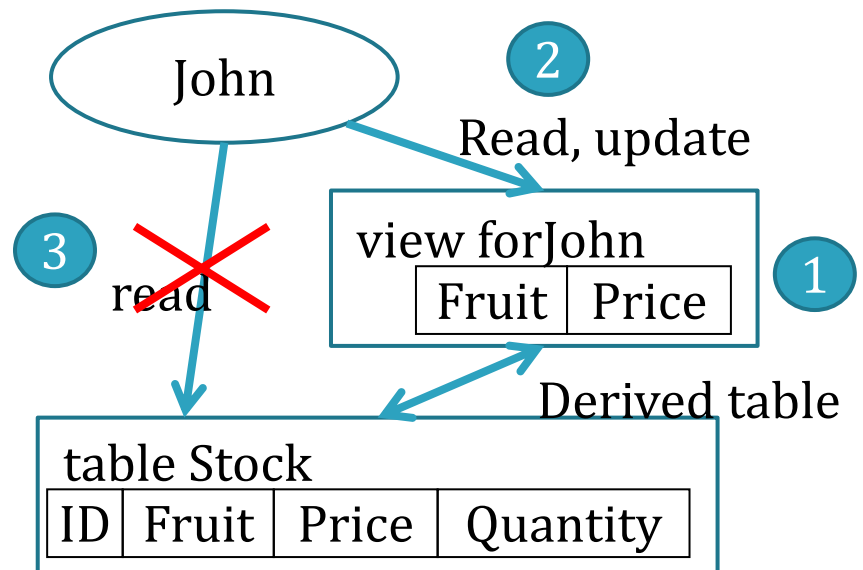
```
GRANT SELECT, UPDATE (Price)  
ON for John  
TO John
```

3. Revoke existing privilege

```
REVOKE ALL ON Stock FROM John
```

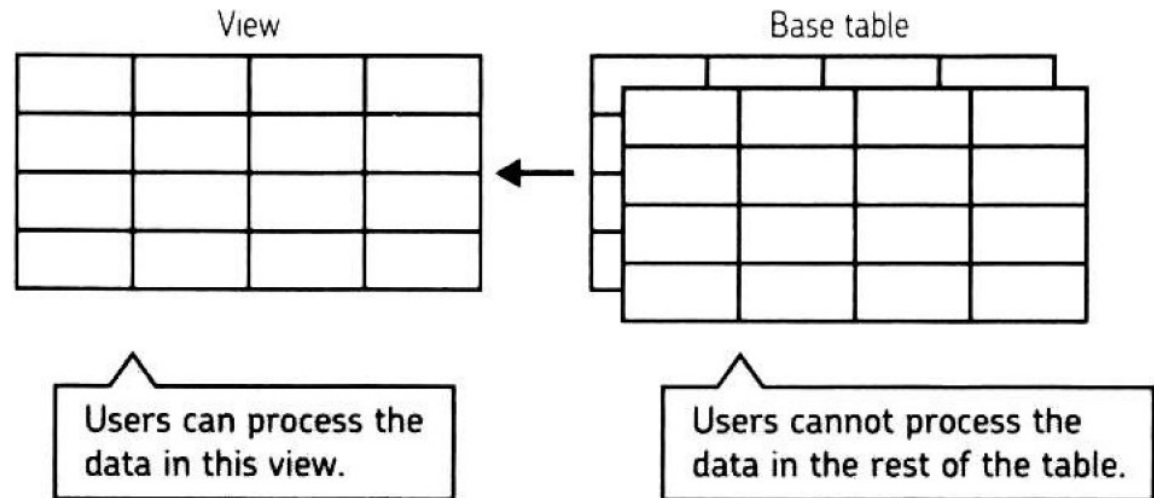
User 'John' originally allowed to **read** from Stock

We want to **revoke** 'John' access to Stock table, but let him **read** the fruit and price, and be able to **update** the price (only)



View Updating

- Views are like virtual tables
 - ▣ Their value depends on the 'base' tables that they are defined from
 - ▣ You can select from views just like a table



- ▣ What about update, insert, and delete?

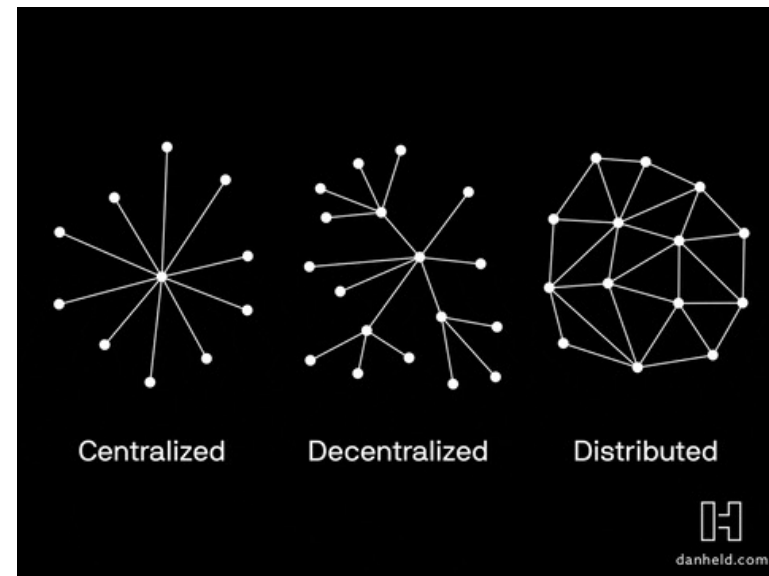
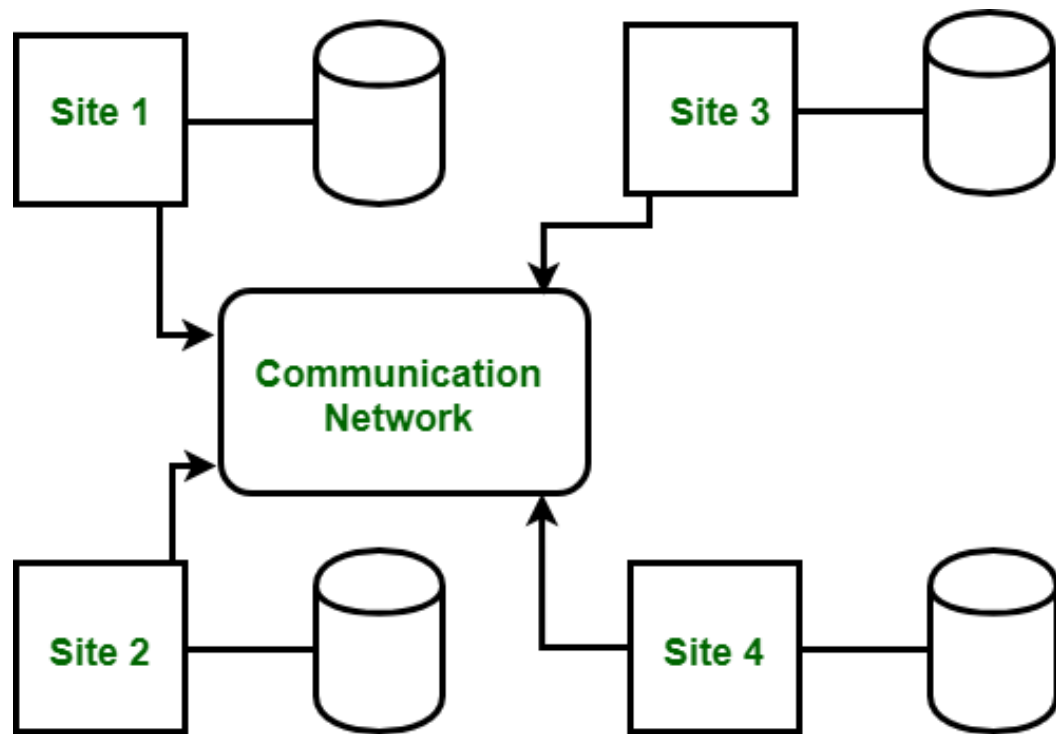
View Updating

- Updates to the base tables change the views => OK.
- Updates to the views change base tables?
 - ▣ The defining query of the view should **satisfy certain conditions**:
 - Every element in SELECT is a column name
 - Should not use DISTINCT
 - View should be defined on a single table (no join, union, etc. used in FROM)
 - WHERE should not have nested SELECTs
 - Should not use GROUP BY or HAVING

A horizontal bar at the top of the slide, divided into a red section on the left and a blue section on the right.

Distributed databases

Distributed Database

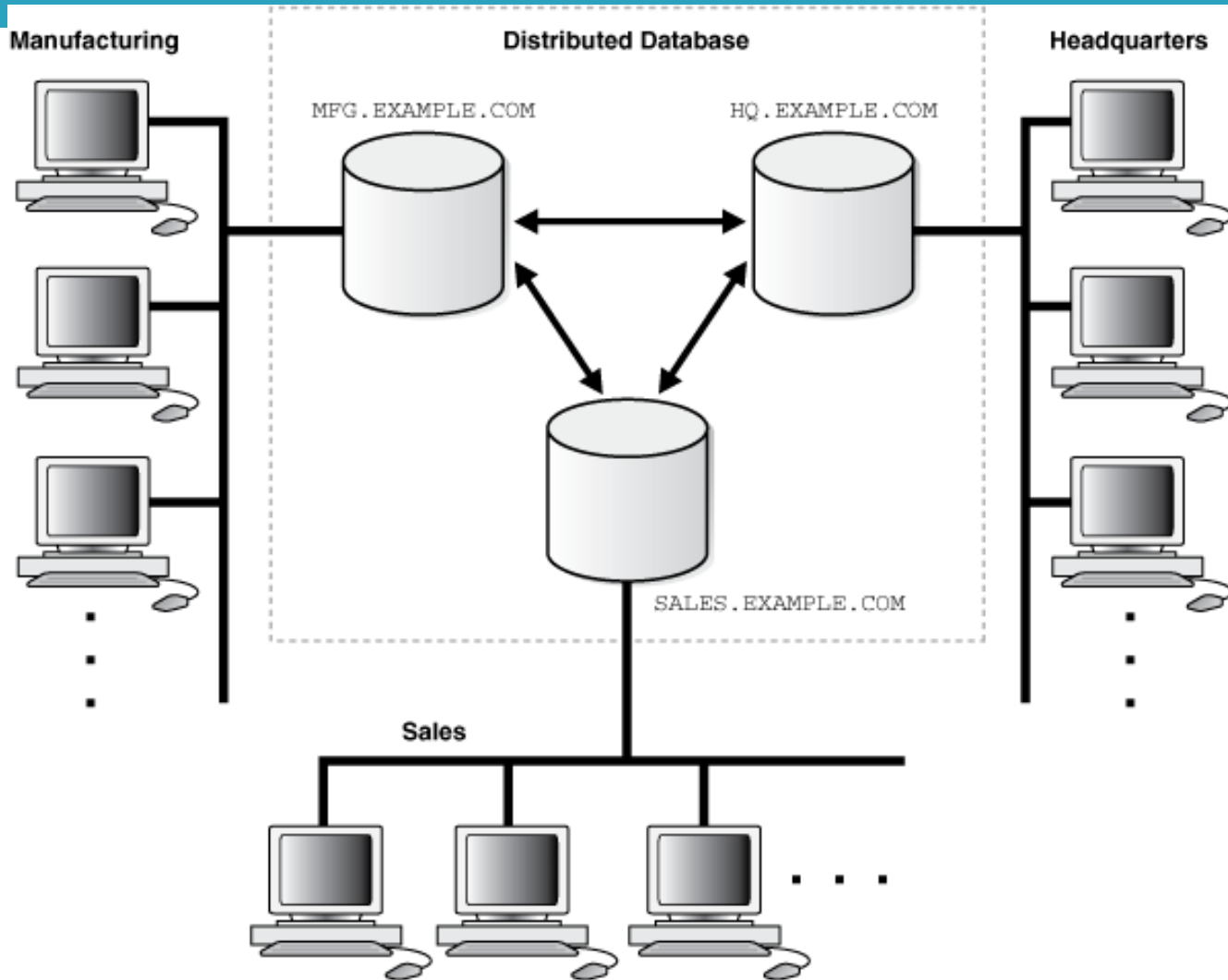


<https://www.geeksforgeeks.org/dates-twelve-rules-for-distributed-database-systems/>

Distributed Database

- ❑ **Distributed database (DDB)** - as a collection of multiple logically interrelated databases distributed over a computer network.
- ❑ **Distributed database management system (DDBMS)** - as a software system that manages a distributed database while making the distribution transparent to the user.
- ❑ Minimum conditions to be satisfied for a DB to be distributed:
 - a. Connection of database nodes over a computer network.
 - b. Logical interrelation of the connected databases.
 - c. Possible absence of homogeneity among connected nodes.

Homogenous Distributed Database



Distributed Databases – Properties

- **Transparency:** Data and Software are distributed over multiple nodes connected by a network, so additional transparencies like network transparency, location transparency, naming transparency, replication transparency, fragmentation transparency etc. are introduced.
- **Availability:** probability that the system is continuously available during a time interval, the system needs to address faults and recovery from faults.
- **Reliability:** probability that a system is running (not down) at a certain time point, recovery manager needs to handle failures from transaction, networks and hardware.

Distributed Databases – Properties

- **Scalability:** extent to which the system can expand its capacity while continuing to operate without interruption, horizontal scalability (add new nodes to the system) and vertical scalability (increase the capacity of a node in the system) Need to have partition tolerance to continue operating if the network is partitioned.
- **Autonomy:** extent to which individual nodes or DBs in a connected DDB can operate independently. A high degree of autonomy is desirable for increased flexibility and customized maintenance of an individual node. Autonomy can be applied to design, communication, and execution.
-

Distributed Databases – Properties

- **Transparency:** Data and Software are distributed over multiple nodes connected by a network, so additional transparencies like network transparency, location transparency, naming transparency, replication transparency, fragmentation transparency etc. are introduced.
- **Availability:** probability that the system is continuously available during a time interval, the system needs to address faults and recovery from faults.
- **Reliability:** probability that a system is running (not down) at a certain time point, recovery manager needs to handle failures from transaction, networks and hardware.
-

Distributed Databases – Properties

- ❑ **Scalability:** extent to which the system can expand its capacity while continuing to operate without interruption, horizontal scalability (add new nodes to the system) and vertical scalability (increase the capacity of a node in the system) Need to have partition tolerance to continue operating if the network is partitioned.
- ❑ **Autonomy:** extent to which individual nodes or DBs in a connected DDB can operate independently. A high degree of autonomy is desirable for increased flexibility and customized maintenance of an individual node. Autonomy can be applied to design, communication, and execution.
- ❑

Data Fragmenting and Sharding

- ❑ Decisions must be made regarding which site should be used to store which portions of the database.
- ❑ Identify the logical units of the database that are to be distributed.

Table T				
T1	T2	T3	...	TN

Horizontal fragmentation (Sharding)

Vertical fragmentation

Horizontal fragmentation (Sharding)

Eno	Ename	Design	Salary	Dep
101	A	abc	3000	1
102	B	abc	4000	1
103	C	abc	5500	2
104	D	abc	5000	2
105	E	abc	2000	2

EMP 1 = $\sigma_{\text{Dep} = 1}$ EMPLOYEE

EMP 2 = $\sigma_{\text{Dep} = 2}$ EMPLOYEE

Eno	Ename	Design	Salary	Dep
101	A	abc	3000	1
102	B	abc	4000	1

Eno	Ename	Design	Salary	Dep
103	C	abc	5500	2
104	D	abc	5000	2
105	E	abc	2000	2

$$T = T1 \cup T2$$

Vertical fragmentation

T1

T2

Eno	Ename	Design	Tuple_id	Salary	Dep	Tuple_id
101	A	abc	1	3000	1	1
102	B	abc	2	4000	2	2
103	C	abc	3	5500	3	3
104	D	abc	4	5000	1	4
105	E	abc	5	2000	4	5

Vertical fragmentation

T1

T2

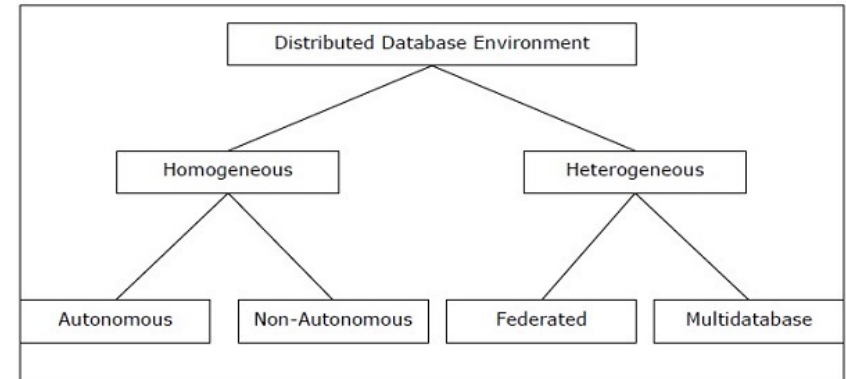
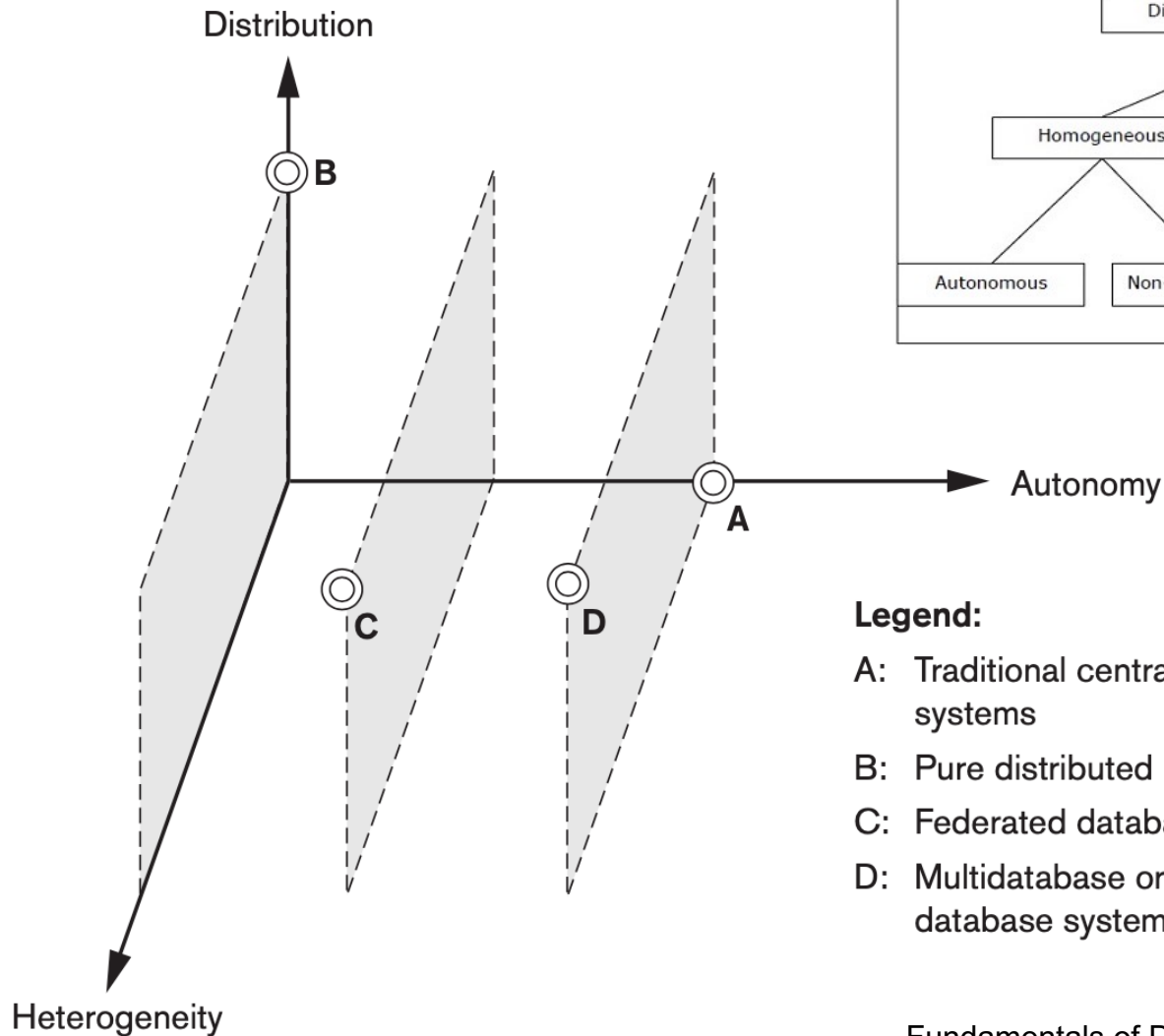
Eno	Ename	Design	Tuple_id	Salary	Dep	Tuple_id
101	A	abc	1	3000	1	1
102	B	abc	2	4000	2	2
103	C	abc	3	5500	3	3
104	D	abc	4	5000	1	4
105	E	abc	5	2000	4	5

$\pi_{\text{EMPLOYEE}} (\text{T1} \bowtie \text{T2})$

DDBMS Components

- ❑ **Data Replication and Allocation:** Fully replicated DDB – copy of whole DB in every site of the distributed system or partially replicated DDB – only some fragments are copied.
- ❑ **Concurrency Control & Recovery** – Tricker. Need to address distributed deadlocks, distributed commits, failure of communication links etc.
- ❑ **Transaction Management** – Still manages to achieve the ACID properties, uses 2 Phase and 3 Phase commit protocol.
- ❑ **Query Optimization** – Done using query mapping, localization, global query optimization and local query optimization.

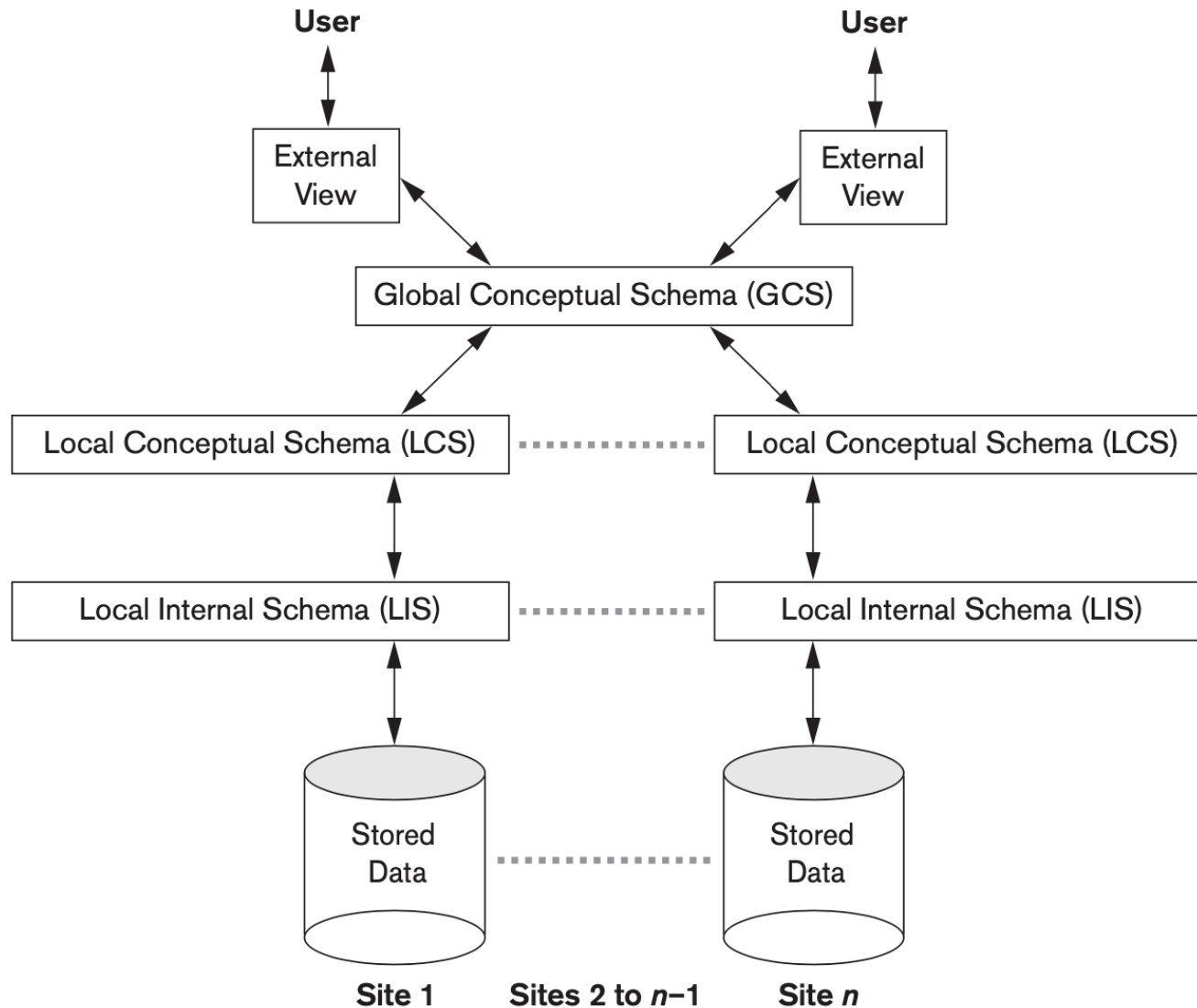
Classification of DDB



Legend:

- A: Traditional centralized database systems
- B: Pure distributed database systems
- C: Federated database systems
- D: Multidatabase or peer-to-peer database systems

Schema Architecture of DDB



Advantages of Distributed Databases

- Depending on the needs of the organization, distributed databases can have many advantages over a single, centralized system that provides remote access to users, including
 - increased local autonomy,
 - improved reliability,
 - better data availability,
 - increased performance,
 - reduced response time, and
 - lower communications cost.
- Distributed database systems include traditional SQL-based relational systems, object databases, XML- based databases, unstructured data systems, NoSQL, and NewSQL databases.

Coming up...

NoSQL
Databases