Computer Security: Principles and Practice

Fourth Edition

By: William Stallings and Lawrie Brown

Chapter 5

Database and
Data Center Security

5.1Database Security

Reasons database security has not kept pace with the increased reliance on databases are:

1. There is a dramatic imbalance between the complexity of modern database management systems (DBMS) and the security technique used to protect these critical systems
2. Databases have a sophisticated interaction protocol, Structured Query Language (SQL), which is complex
3. Effective database security requires a strategy based on a full understanding of the security vulnerabilities of SQL
4. The typical organization lacks full-time database security personnel
5. Most enterprise environments consist of a heterogeneous mixture of database platforms (Oracle, IBM DB2 and Informix, Microsoft, Sybase, etc.), , enterprise platforms (Oracle E-Business Suite, PeopleSoft, SAP, Siebel, etc.), , and OS platforms (UNIX, Linux, z/OS, and Windows, etc.). , creating an additional complexity hurdle for security personnel
6. The increasing reliance on cloud technology to host part or all of the corporate database

Enterprise Environments

- Most enterprise environments consist of a heterogeneous mixture of:
- Database Platforms
  - Oracle, IBM DB2 And Informix, Microsoft, Sybase, etc.
- Enterprise Platforms
  - Oracle E-business Suite, Peoplesoft, SAP, Siebel, etc.
- OS platforms
  - UNIX, Linux, Z/OS, And Windows, Etc.

5.2 Database Management System (DBMS)

- Structured collection of data stored for use by one or more applications
- Contains the relationships between data items and groups of data items
- Can sometimes contain sensitive data that needs to be secured

Query language

- Provides a uniform interface to the database for users and applications

Database management system (DBMS)

- Suite of programs for constructing and maintaining the database
- Offers ad hoc query facilities to multiple users and applications

DBMS Architecture

What type of users?

What type of users ?

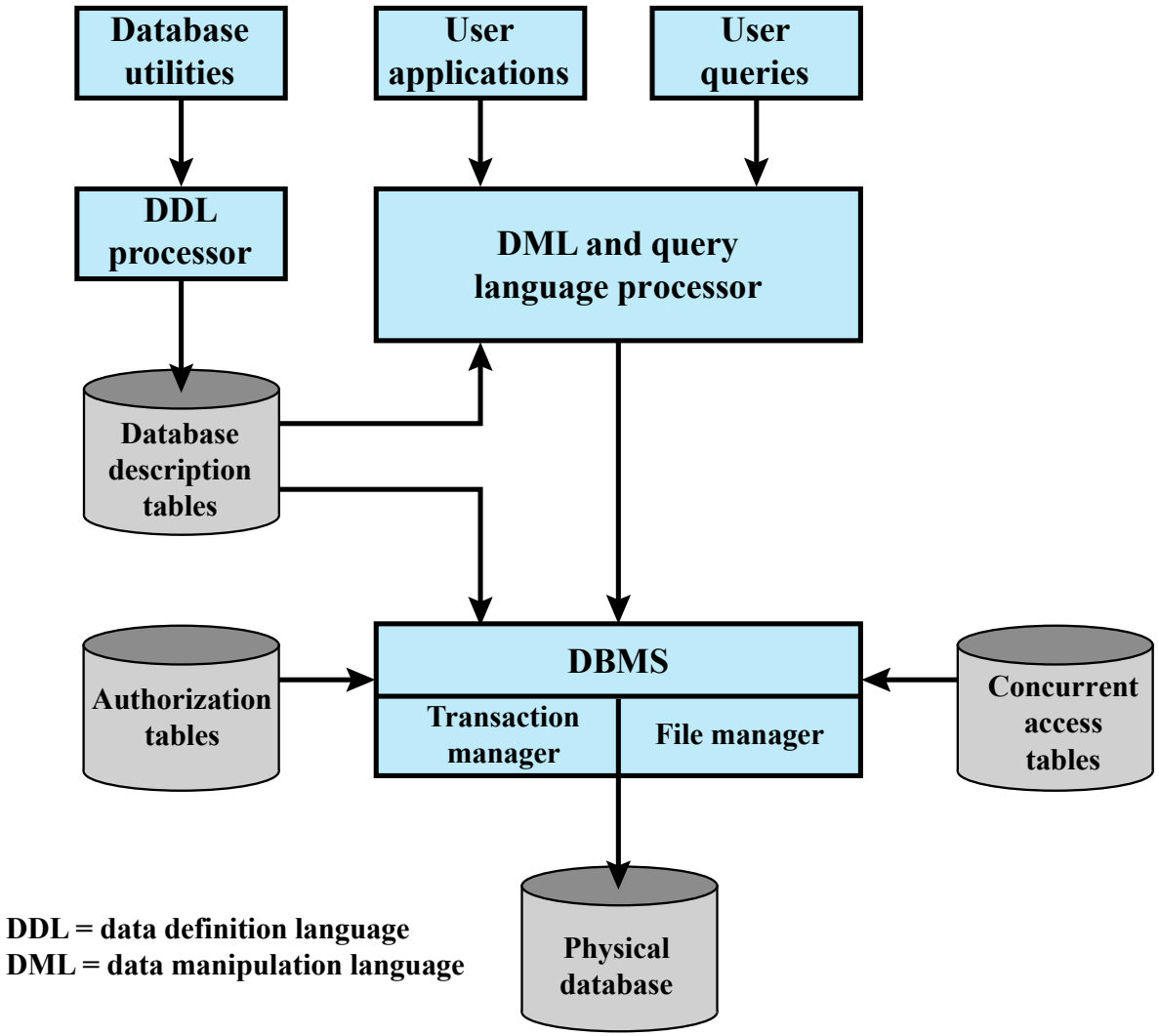


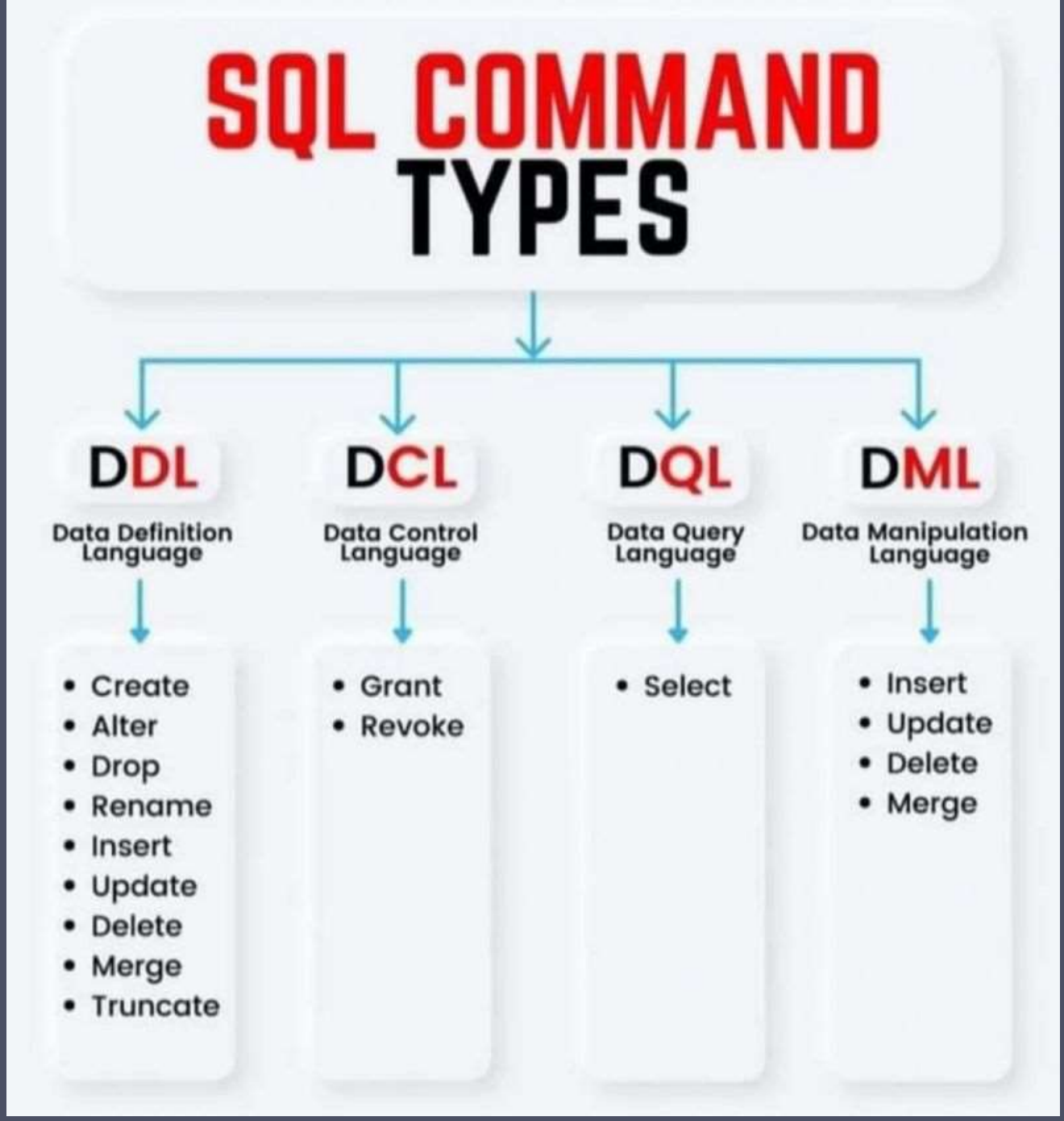
Figure 5.1 DBMS Architecture

Database Utilities

- Tools used by administrators to maintain, back up, and optimize the database system.
- Tools for backup, recovery, tuning, auditing, data loading,
- file reorganization, and integrity checks

DDL Processor

- Handles Data Definition Language commands such as CREATE, ALTER, DROP and updates metadata.


SQL COMMAND
TYPES
DDL
Data Definition Language
Create
Alter
Drop
Rename
Insert
Update
Delete
Merge
Truncate
DCL
Data Control Language
Grant
Revoke
DQL
Data Query Language
Select
DML
Data Manipulation Language
Insert
Update
Delete
Merge

Database Description Tables

- Metadata repository storing schema definitions, structures, and constraints.

User Applications

- External programs that interact with the DBMS via queries and data manipulation operations.

User Queries

- SQL commands issued directly by users or applications for retrieving or modifying data.

DML & Query Processor

- • Interprets and executes DML operations (INSERT, UPDATE, DELETE) and SELECT queries.

Authorization Tables

- Access control information determining which users can perform specific operations.

Concurrent Access Tables

- • Mechanisms ensuring multiple users can safely access data without conflict.

Concurrent Access Tables

- Mechanisms ensuring multiple users can safely access data without conflict.

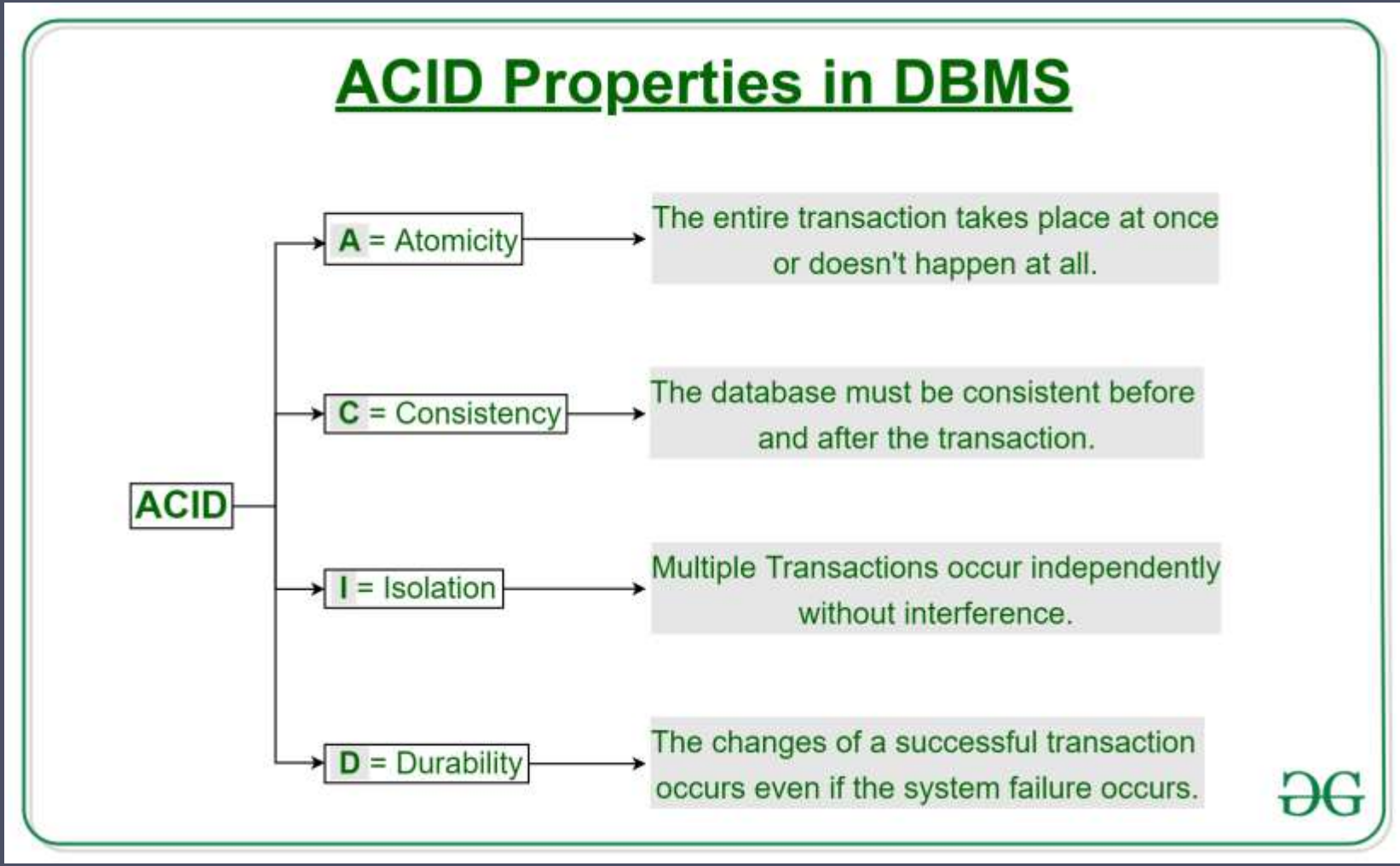
DBMS Core: Transaction Manager

- Ensures ACID properties: atomicity, consistency, isolation, durability.

ACID Properties in DBMS

- A transaction is a single logical unit of work that accesses and possibly modifies the contents of a database.
- Transactions access data using read and write operations.
- In order to maintain consistency in a database, before and after the transaction, certain properties are followed.
- These are called ACID properties.

ACID Properties in DBMS



DBMS Core: File Manager

- Manages storage, retrieval, and organization of data at the physical level.

Physical Database

- The actual stored data on disk, SSD, or other storage media.

https://java-questions.com/cap-theorem.html

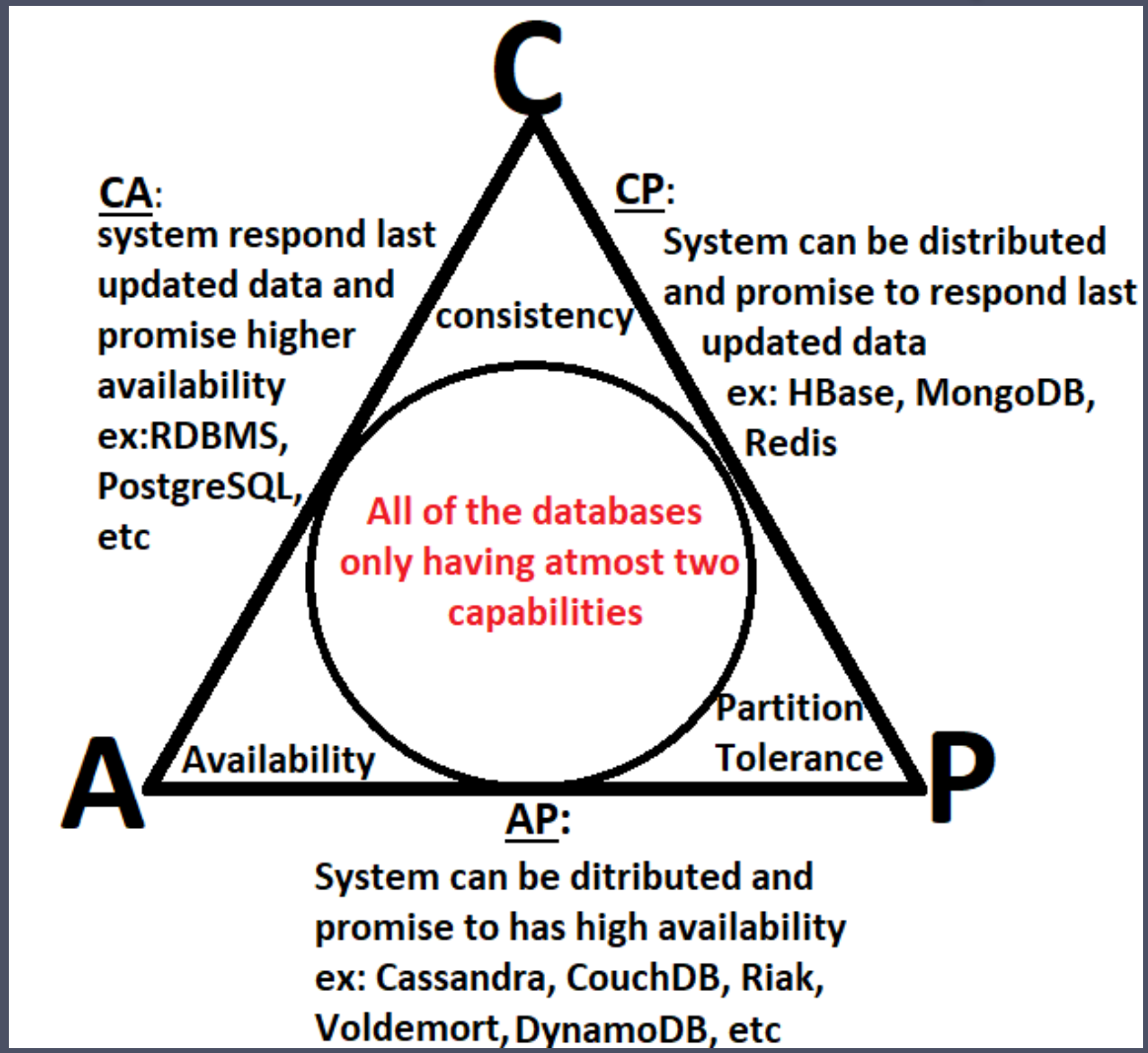
CAP theorem (1/3)

- Distributed systems are prone to outages and not safe from network failures.
- As system are being scaled up designing a resilient system had to deal with complexity incurred in the system.
- One of the fallacy of the distributed system is that network are reliable. This is where the CAP theorem comes into picture.

CAP theorem (2/3)

- The CAP theorem, also known as Brewer's theorem, states that it is impossible for a distributed computer system to simultaneously provide all three of the following guarantees:
- Consistency: all nodes see the same data at the same time.
- Availability: a guarantee that every request receives a response about whether it was successful or failed.
- Partition tolerance: the system continues to operate despite arbitrary message loss or failure of part of the system)
- According to the theorem, a distributed system cannot satisfy all three of these guarantees at the same time.

CAP theorem (3/3)



https://www.geeksforgeeks.org/the-cap-theorem-in-dbms/

5.3 Relational Databases

- Table of data consisting of rows and columns
  - Each column holds a particular type of data
  - Each row contains a specific value for each column
  - Ideally has one column where all values are unique, forming an identifier/key for that row
- Enables the creation of multiple tables linked together by a unique identifier that is present in all tables
- Use a relational query language to access the database
  - Allows the user to request data that fit a given set of criteria

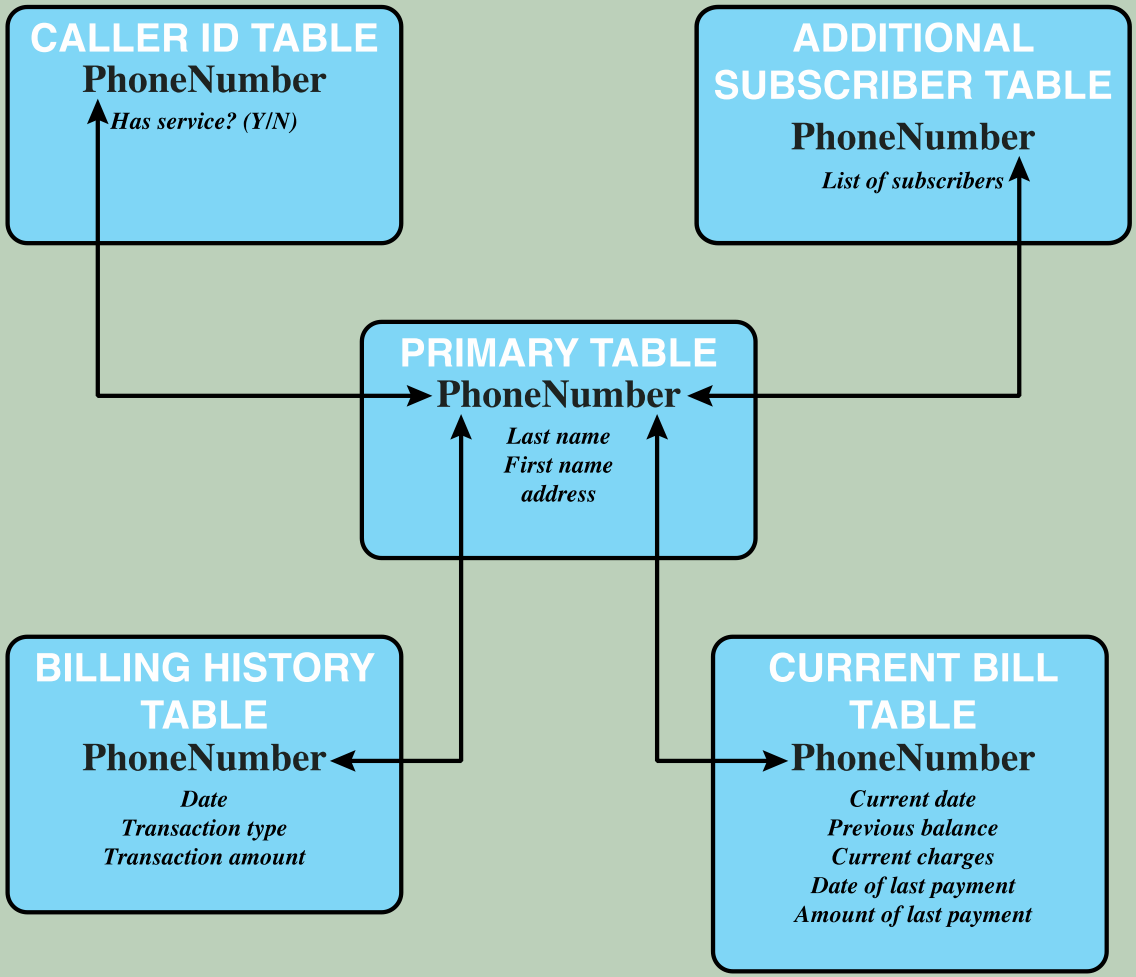


Figure 5.2 Example Relational Database Model. A relational database uses multiple tables related to one another by a designated key; in this case the key is the PhoneNumber field.

5.3.1 Relational Database Elements

- Relation
  - Table/file
- Tuple
  - Row/record
- Attribute
  - Column/field

Primary key

- Uniquely identifies a row
- Consists of one or more column names

Foreign key

- Links one table to attributes in another

View/virtual table

- Result of a query that returns selected rows and columns from one or more tables
- Views are often used for security purposes

Table 5.1
Basic Terminology for Relational Databases

Formal Name	Common Name	Also Known As
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

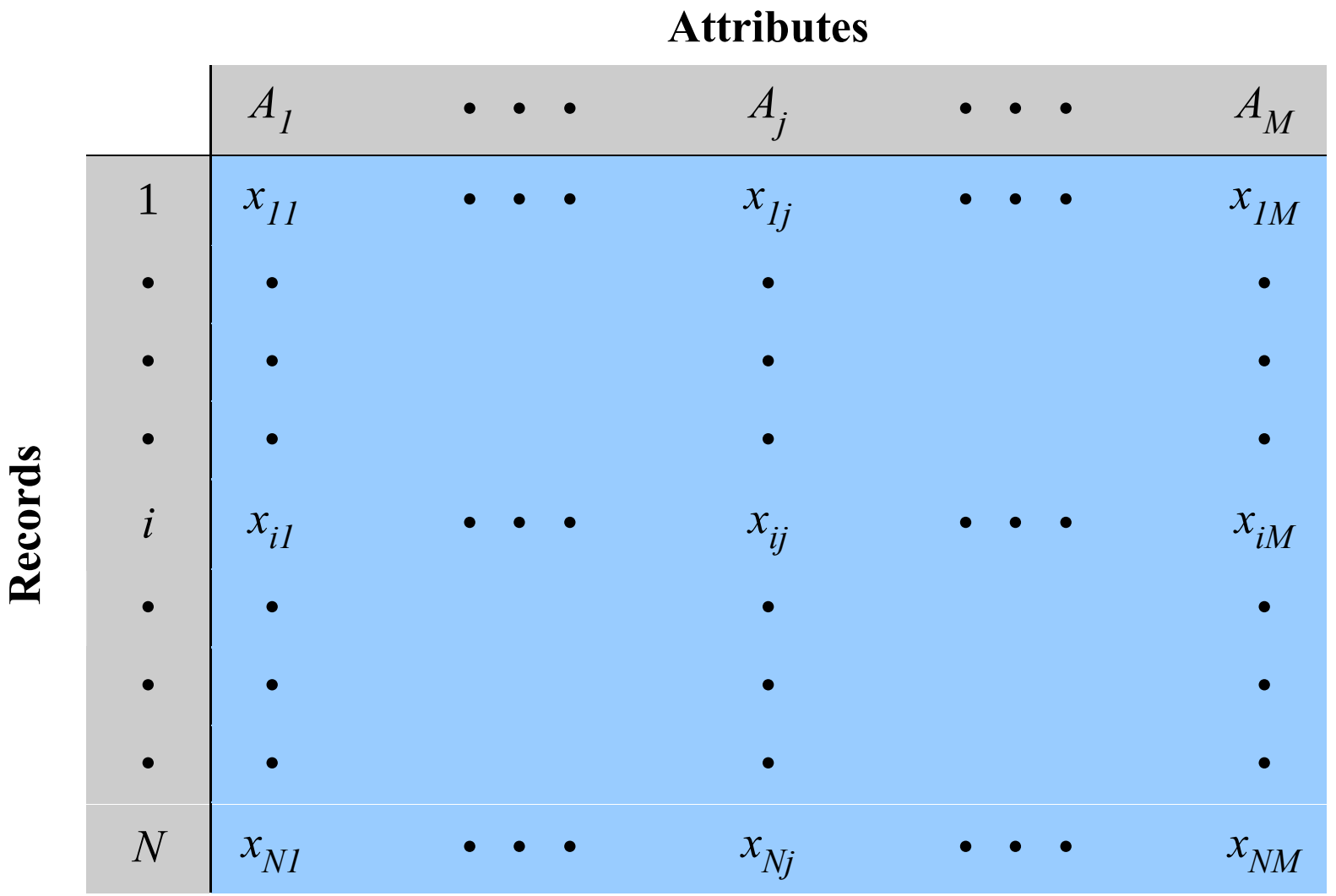


Figure 5.3 Abstract Model of a Relational Database

Department Table

Did	Dname	Dacctno
4	human resources	528221
8	education	202035
9	accounts	709257
13	public relations	755827
15	services	223945

primary key

Employee Table

Ename	Did	Salarycode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

foreign key

primary key

(a) Two tables in a relational database

Dname	Ename	Eid	Ephone
human resources	Jasmine	7712	6127099348
education	Holly	3054	6127092729
education	Robin	2976	6127091945
accounts	Smith	4490	6127099380
public relations	Neil	5088	6127092246
services	Robin	2345	6127092485
services	Cody	9664	6127093148

(b) A view derived from the database

Figure 5.4 Relational Database Example

5.3.2 Structured Query Language (SQL)

- Standardized language to define schema, manipulate, and query data in a relational database
- Several similar versions of ANSI/ISO standard
- All follow the same basic syntax and semantics

SQL statements can be used to:

- Create tables
- Insert and delete data in tables
- Create views
- Retrieve data with query statements

Department Table

Did	Dname	Dacctno
4	human resources	528221
8	education	202035
9	accounts	709257
13	public relations	755827
15	services	223945

primary key

Employee Table

Ename	Did	Salarycode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

foreign key

primary key

(a) Two tables in a relational database

```
CREATE TABLE department (
   Did INTEGER PRIMARY KEY,
   Dname CHAR (30),
   Dacctno CHAR (6) )
CREATE TABLE employee (
   Ename CHAR (30),
   Did INTEGER,
   SalaryCode INTEGER,
   Eid INTEGER PRIMARY KEY,
   Ephone CHAR (10),
   FOREIGN KEY (Did) REFERENCES department (Did) )
```

```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 15
```

Dname	Ename	Eid	Ephone
human resources	Jasmine	7712	6127099348
education	Holly	3054	6127092729
education	Robin	2976	6127091945
accounts	Smith	4490	6127099380
public relations	Neil	5088	6127092246
services	Robin	2345	6127092485
services	Cody	9664	6127093148

(b) A view derived from the database

Figure 5.4 Relational Database Example

```
CREATE VIEW newtable (Dname, Ename, Eid, Ephone)
AS SELECT D.Dname E.Ename, E.Eid, E.Ephone
FROM Department D Employee E
WHERE E.Did = D.Did
```

5.4 SQL Injection Attacks (SQLi)

- One of the most prevalent and dangerous network-based security threats
- Designed to exploit the nature of Web application pages
- Sends malicious SQL commands to the database server
- Most common attack goal is bulk extraction of data
- Depending on the environment SQL injection can also be exploited to:
  - Modify or delete data
  - Execute arbitrary operating system commands
  - Launch denial-of-service (DoS) attacks

A Typical SQLi Attack

1. Hacker finds a vulnerability in a custom Web application and injects an SQL command to a database by sending the command to the Web server. The command is injected into traffic that will be accepted by the firewall.

2. The Web server receives the malicious code and sends it to the Web application server.

3. The Web application server receives the malicious code from the Web server and sends it to the database server.

4. The database server executes the malicious code on the database. The database returns data from credit cards table.

5. The Web application server dynamically generates a page with data including credit card details from the database.

6. The Web server sends the credit card details to the hacker.

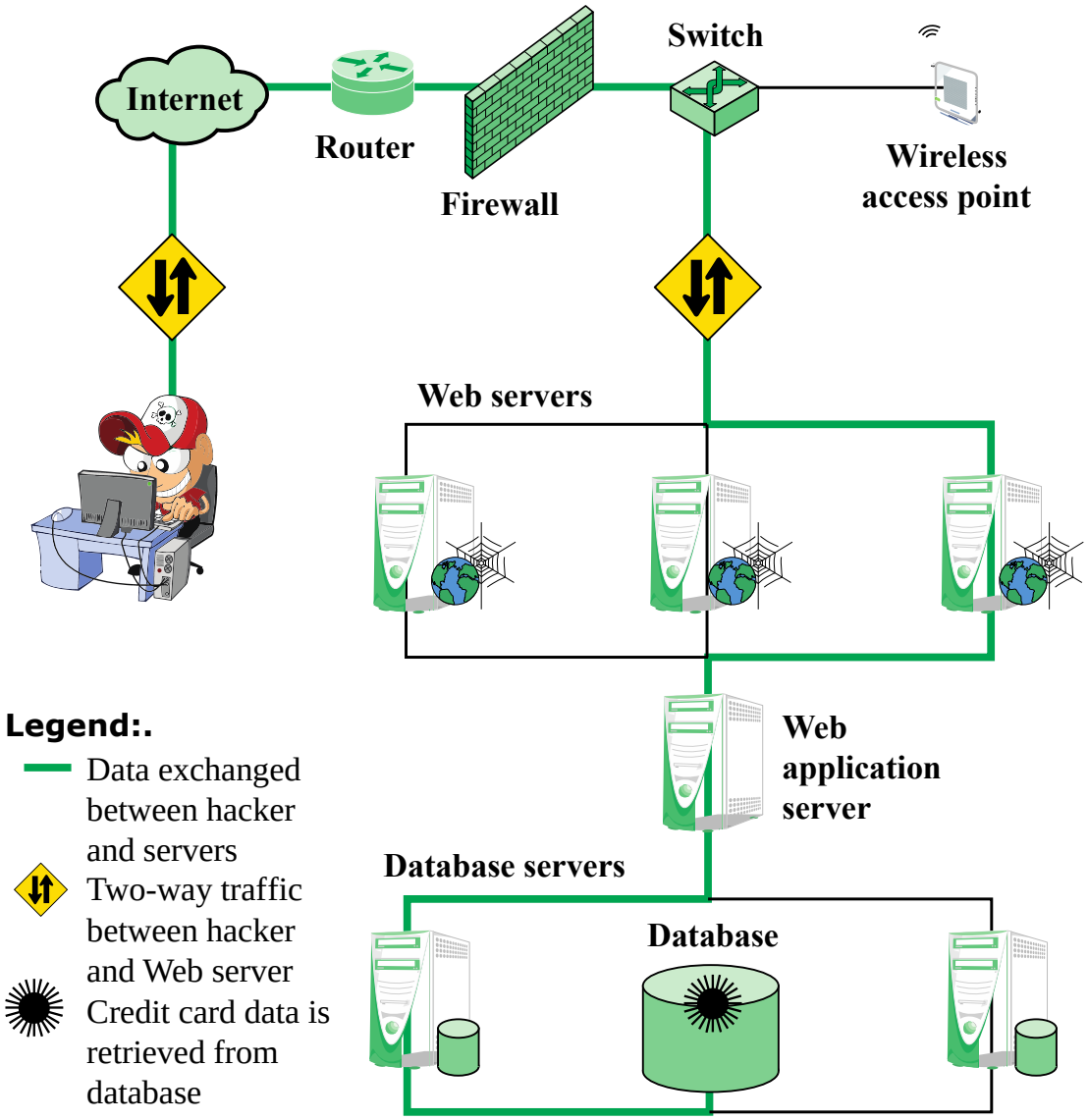
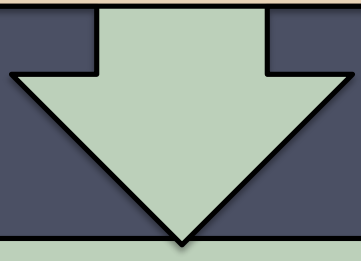


Figure 5.5 Typical SQL Injection Attack

Injection Technique

The SQLi attack typically works by prematurely terminating a text string and appending a new command

Because the inserted command may have additional strings appended to it before it is executed the attacker terminates the injected string with a comment mark "- -"



Subsequent text is ignored at execution time

Injection Technique

- script that build an SQL query by combining predefined strings with text entered by a user:

```
var Shipcity;
ShipCity = Request.form ("ShipCity");
var sql = "select * from OrdersTable where ShipCity = '" +
ShipCity + "'";
```

- The intention of the script's designer is that a user will enter the name of a city. e.g. When prompted user to enter a city, and if he enters Redmond, SQL generated query is :

```
SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond'
```

- Suppose, however, the user enters the following:

```
Boston'; DROP table OrdersTable--
```

- The result is

```
SELECT * FROM OrdersTable WHERE ShipCity =
'Redmond'; DROP table OrdersTable--
```

??

5.4.3.1 SQLi Attack Avenues (1/2)

- User input
  - Attackers inject SQL commands by providing suitable crafted user input e.g. form submission application via HTTP GET or POST
- Server variables
  - Server variables consists HTTP headers, network protocol headers, and environmental variables.
  - Web applications use these server variables in a variety of ways, such as logging usage statistics and identifying browsing trends.
  - If these variables are logged to a database without sanitization, this could create an SQL injection vulnerability.
  - Attackers can forge the values that are placed in HTTP and network headers and exploit this vulnerability by placing data directly into the headers

SQLi Attack Avenues (1/2)

- Second-order injection
  - A malicious user could rely on data already present in the system or database to trigger an SQL injection attack, so when the attack occurs, the input that modifies the query to cause an attack does not come from the user, but from within the system itself
- Cookies
  - An attacker could alter cookies such that when the application server builds an SQL query based on the cookie's content, the structure and function of the query is modified
- Physical user input
  - Applying user input that constructs an attack outside the realm of web requests e.g. barcodes, RFID tags, or even paper forms which are scanned using OCR passed to DBMS

5.4.3.2.1 Inband Attacks (1/2)

- Uses the same communication channel for injecting SQL code and retrieving results
- The retrieved data are presented directly in application Web page Include:
- Tautology
  - This form of attack injects code in one or more conditional statements so that they always evaluate to true e.g.

```
$query = "SELECT info FROM user WHERE name =
'$_GET["name"]' AND pwd = '$_GET["pwd"]'";
```

  - Attacker submits " ' OR 1=1 --" for the name field and there resulting query would look like this:

```
SELECT info FROM users WHERE name = ' ' OR 1=1 -- AND pwpd = ' '
```

Inband Attacks (1/2)

- End-of-line comment
  - After injecting code into a particular field, legitimate code that follows are nullified through usage of end of line comments.
  - An example would be to add ”- -” after inputs so that remaining queries are not treated as executable code, but comments e.g. preceding tautology
- Piggybacked queries
  - The attacker adds additional queries beyond the intended query, piggy-backing the attack on top of a legitimate request

5.4.3.2.2 Inferential Attack

- There is no actual transfer of data, but the attacker is able to reconstruct the information by sending particular requests and observing the resulting behavior of the Website/database server
- Include:
  - Illegal/logically incorrect queries
    - This attack lets an attacker gather important information about the type and structure of the backend database of a Web application
    - The attack is considered a preliminary, information-gathering step for other attacks
  - Blind SQL injection
    - Allows attackers to infer the data present in a database system even when the system is sufficiently secure to not display any erroneous information back to the attacker
    - The attacker asks the server true/false questions.
      - If the injected statement evaluates to true, the site continues to function normally. If the statement evaluates to false, although there is no descriptive error message, the page differs significantly from the normally functioning page.

5.4.3.2.3 Out-of-Band Attack

- Data are retrieved using a different channel
- This can be used when there are limitations on information retrieval, but outbound connectivity from the database server is lax

SQLi Countermeasures

1. Defensive coding
2. Detection
3. Run-time prevention

SQLi Countermeasures - Defensive coding

- Manual defensive coding practices
  - due insufficient input validation
  - e.g. type checking, numeric contain no characters other than digits.
  - another example is pattern matching to try to distinguish abnormal input.
- Parameterized query insertion
  - pass the value parameters to it separately such that any unsanitary user input is not allowed to modify the query structure.
- SQL DOM
  - DOM is a set of classes that enables automated data type validation and escaping
    - Encapsulation of database queries to provide a safe and reliable way to access databases
  - This changes the query-building process from an unregulated one that uses string concatenation to a systematic one that uses a type-checked API.
  - Within the API, developers are able to systematically apply coding best practices such as input filtering and rigorous type checking of user input

SQLi Countermeasures - Detection

- Signature based
  - This technique attempts to match specific attack patterns.
  - Such an approach must be constantly updated and may not work against self-modifying attacks.
- Anomaly based
  - define normal behavior and then detect behavior patterns outside the normal range.
  - A number of approaches have been used. In general terms, there is a training phase, in which the system learns the range of normal behavior, followed by the actual detection phase.
- Code analysis
  - involve the use of a test suite to detect SQLi vulnerabilities. The test suite is designed to generate a wide range of SQLi attacks and assess the response of the system.

SQLi Countermeasures - Run-time prevention

- Run-time prevention techniques have been developed as SQLi countermeasures.
- These techniques check queries at runtime to see if they conform to a model of expected queries.
- Various automated tools are available for this purpose

https://www.serverwatch.com/guides/sql-injection-detection-tools/

Top SQLi Detections Tools

- There are many SQLi detection tools available that can help identify and mitigate these vulnerabilities.

1. sqlmap: Best open-source SQLi detection tool
2. Invicti: Best for security scanning visibility
3. Burp Scanner: Best for combining manual and automated testing
4. jSQL Injection: Best for Java developers
5. AppSpider: Best for Windows OS users
6. Acunetix: Best for scanning script-heavy web apps
7. Qualys WAS: Best for regular vulnerability assessments
8. HCL AppScan: Best for managing on-premise and cloud environment vulnerabilities
9. Imperva: Best for automated mitigation

5.5 Database Access Control

Database access control system determines:

If the user has access to the entire database or just portions of it

What access rights the user has (create, insert, delete, update, read, write)

Can support a range of administrative policies

Centralized administration

- Small number of privileged users may grant and revoke access rights

Ownership-based administration

- The creator of a table may grant and revoke access rights to the table

Decentralized administration

- The owner of the table may grant and revoke authorization rights to other users, allowing them to grant and revoke access rights to the table

5.5.1A SQL-Based Access Definition

- Two commands for managing access rights:
  - Grant
    - Used to grant one or more access rights or can be used to assign a user to a role
  - Revoke
    - Revokes the access rights
- Typical access rights are:
  - Select
  - Insert
  - Update
  - Delete
  - References

5.5.1B Cascading Authorizations (1/2)

- The grant option enables an access right to cascade through a number of users e.g.

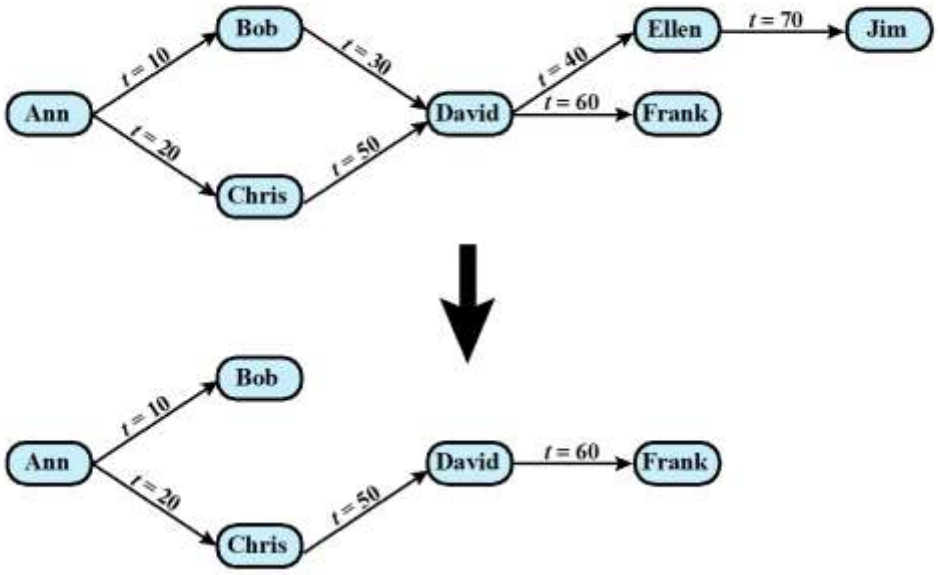


Figure 5.6 Bob Revokes Privilege from David

Cascading Authorizations-Convention (2/2)

- To generalize, the convention followed by most implementations is as follows.
- When user A revokes an access right, any cascaded access right is also revoked, unless that access right would exist even if the original grant from A had never occurred.

5.5.2 Role-Based Access Control (RBAC)

- Role-based access control eases administrative burden and improves security
- A database RBAC needs to provide the following capabilities:
  - Create and delete roles
  - Define permissions for a role
  - Assign and cancel assignment of users to roles
- Categories of database users:

Application owner	End user	Administrator
• An end user who owns database objects as part of an application	• An end user who operates on database objects via a particular application but does not own any of the database objects	• User who has administrative responsibility for part or all of the database

Table 5.2

Fixed Roles in Microsoft SQL Server

Role	Permissions
Fixed Server Roles	
sysadmin	Can perform any activity in SQL Server and have complete control over all database functions
serveradmin	Can set server-wide configuration options, shut down the server
setupadmin	Can manage linked servers and startup procedures
securityadmin	Can manage logins and CREATE DATABASE permissions, also read error logs and change passwords
processadmin	Can manage processes running in SQL Server
dbcreator	Can create, alter, and drop databases
diskadmin	Can manage disk files
bulkadmin	Can execute BULK INSERT statements
Fixed Database Roles	
db_owner	Has all permissions in the database
db_accessadmin	Can add or remove user IDs
db_datareader	Can select all data from any user table in the database
db_datawriter	Can modify any data in any user table in the database
db_ddladmin	Can issue all Data Definition Language (DDL) statements
db_securityadmin	Can manage all permissions, object ownerships, roles and role memberships
db_backupoperator	Can issue DBCC, CHECKPOINT, and BACKUP statements
db_denydatareader	Can deny permission to select data in the database
db_denydatawriter	Can deny permission to change data in the database

(Table is on page 165 in the textbook)

Fixed Server Roles

- sysadmin – Complete control over SQL Server
- serveradmin – Configure server & shutdown
- setupadmin – Manage linked servers/startup
- securityadmin – Manage logins & permissions
- processadmin – Control server processes
- dbcreator – Create/alter/drop databases
- diskadmin – Manage disk files
- bulkadmin – Perform BULK INSERT

Fixed Database Roles

- db_owner – Full control of the database
- db_accessadmin – Manage users
- db_datareader – Read all tables
- db_datawriter – Write to all tables
- db_ddladmin – Execute DDL statements
- db_securityadmin – Manage DB permissions
- db_backupoperator – Run backups
- db_denydatareader – Deny SELECT
- db_denydatawriter – Deny INSERT/UPDATE/DELETE

5.6 Inference

- Inference, as it relates to database security, is the process of performing authorized queries and deducing unauthorized information from the legitimate responses received.
- The inference problem arises when the combination of a number of data items is more sensitive than the individual items, or when a combination of data items can be used to infer data of a higher sensitivity
- The information transfer path by which unauthorized data is obtained is referred to as an inference channel .

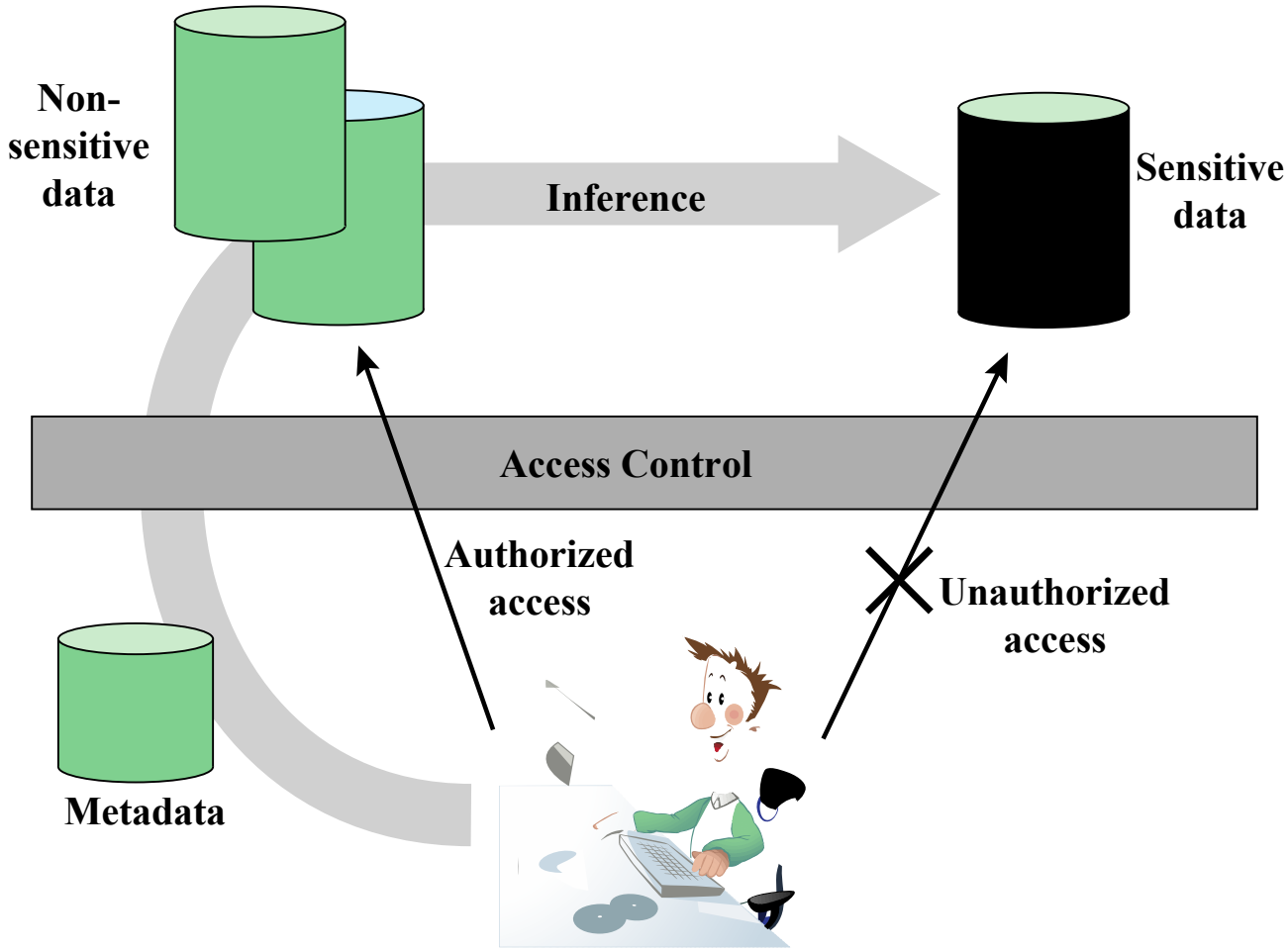


Figure 5.7 Indirect Information Access Via Inference Channel

Inference

- Figure 5.8b shows two views, defined in SQL as follows:

- For View 1

```
CREATE view V1 AS
SELECT Availability, Cost
FROM Inventory
WHERE Department = "hardware"
```

- For View 2

```
CREATE view V2 AS
SELECT Item, Department
FROM Inventory
WHERE Department = "hardware"
```

Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware
Cake pan	online only	12.99	housewares
Shower/tub cleaner	in-store/online	11.99	housewares
Rolling pin	in-store/online	10.99	housewares

(a) Inventory table

Availability	Cost ($)
in-store/online	7.99
online only	5.49
in-store/online	104.99

Item	Department
Shelf support	hardware
Lid support	hardware
Decorative chain	hardware

(b) Two views

Item	Availability	Cost ($)	Department
Shelf support	in-store/online	7.99	hardware
Lid support	online only	5.49	hardware
Decorative chain	in-store/online	104.99	hardware

(c) Table derived from combining query answers

Figure 5.8 Inference Example

Inference

- Users of these views are not authorized to access the relationship between Item and Cost.
- A user who has access to either or both views cannot infer the relationship by functional dependencies.
  - That is, there is not a functional relationship between Item and Cost such that knowing Item and perhaps other information is sufficient to deduce Cost.
- However, suppose the two views are created with the access constraint that Item and Cost cannot be accessed together.
- A user who knows the structure of the Inventory table and who knows that the view tables maintain the same row order as the Inventory table is then able to merge the two views to construct the table shown in Figure 5.8c.
- This violates the access control policy that the relationship of attributes Item and Cost must not be disclosed.

Inference Detection Example (1/2)

- Individually, the name, address, and salary information is available to a subordinate role, such as Clerk, but the association of names and salaries is restricted to a superior role, such as Administrator.
- Three tables solution

Employees (Emp#, Name, Address)
Salaries (S#, Salary)
Emp-Salary (Emp#, S#)

The Employees table and the Salaries table are accessible to the Clerk role, but the Emp-Salary table is only available to the Administrator role.

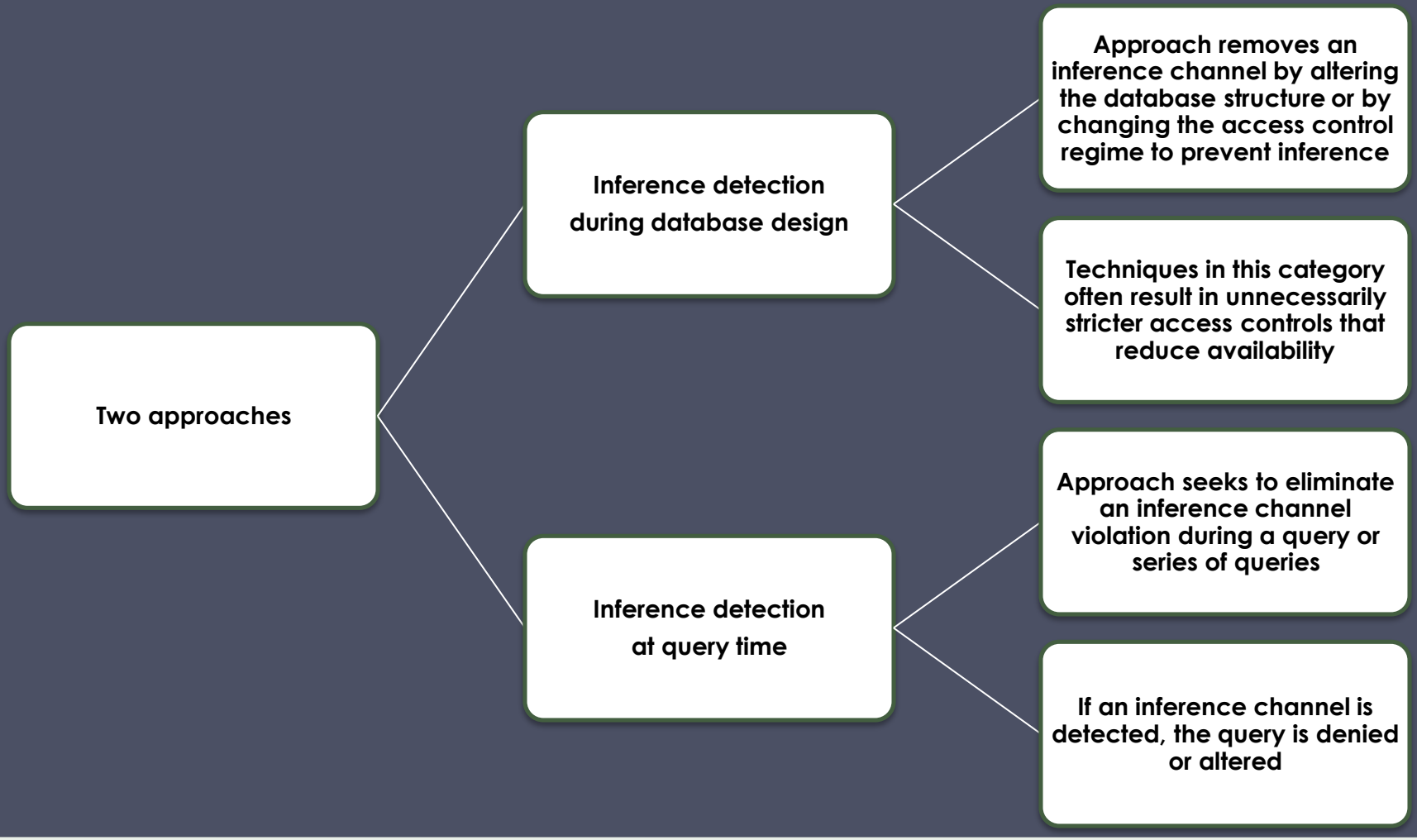
- Want to add a new attribute, employee start date, which is not sensitive

Employees (Emp#, Name, Address)
Salaries (S#, Salary, Start-Date)
Emp-Salary (Emp#, S#)

Inference Detection Example (2/2)

- However, an employee's start date is an easily observable or discoverable attribute of an employee.
- Thus, a user in the Clerk role should be able to infer (or partially infer) the employee's name.
- This would compromise the relationship between employee and salary.
- A straightforward way to remove the inference channel is to add the start-date column to the Employees table rather than to the Salaries table.

Inference Detection



- Some inference detection algorithm is needed for either of these approaches
- Progress has been made in devising specific inference detection techniques for multilevel secure databases and statistical databases

5.7 Database Encryption (1/2)

- The database is typically the most valuable information resource for any organization
  - Protected by multiple layers of security
    - Firewalls, authentication, general access control systems, DB access control systems, database encryption
    - Encryption becomes the last line of defense in database security
  - Can be applied to the entire database, at the record level, the attribute level, or level of the individual field

Database Encryption (2/2)

- Disadvantages to encryption:
  - Key management
    - Authorized users must have access to the decryption key for the data for which they have access
  - Inflexibility
    - When part or all of the database is encrypted it becomes more difficult to perform record searching

Database Encryption Scheme

A Database Encryption Scheme- 4 Entities

- Data owner: An organization that produces data to be made available for controlled release, either within the organization or to external users.
- User: Human entity that presents requests (queries) to the system. The user could be an employee of the organization who is granted access to the database via the server, or a user external to the organization who, after authentication, is granted access.
- Client: Front end that transforms user queries into queries on the encrypted data stored on the server.
- Server: An organization that receives the encrypted data from a data owner and makes them available for distribution to clients. The server could in fact be owned by the data owner but, more typically, is a facility owned and maintained
- by an external provider.

Overview

- Data stays encrypted on the server.
- Client handles all encryption/decryption.
- User sees plaintext; server never does.

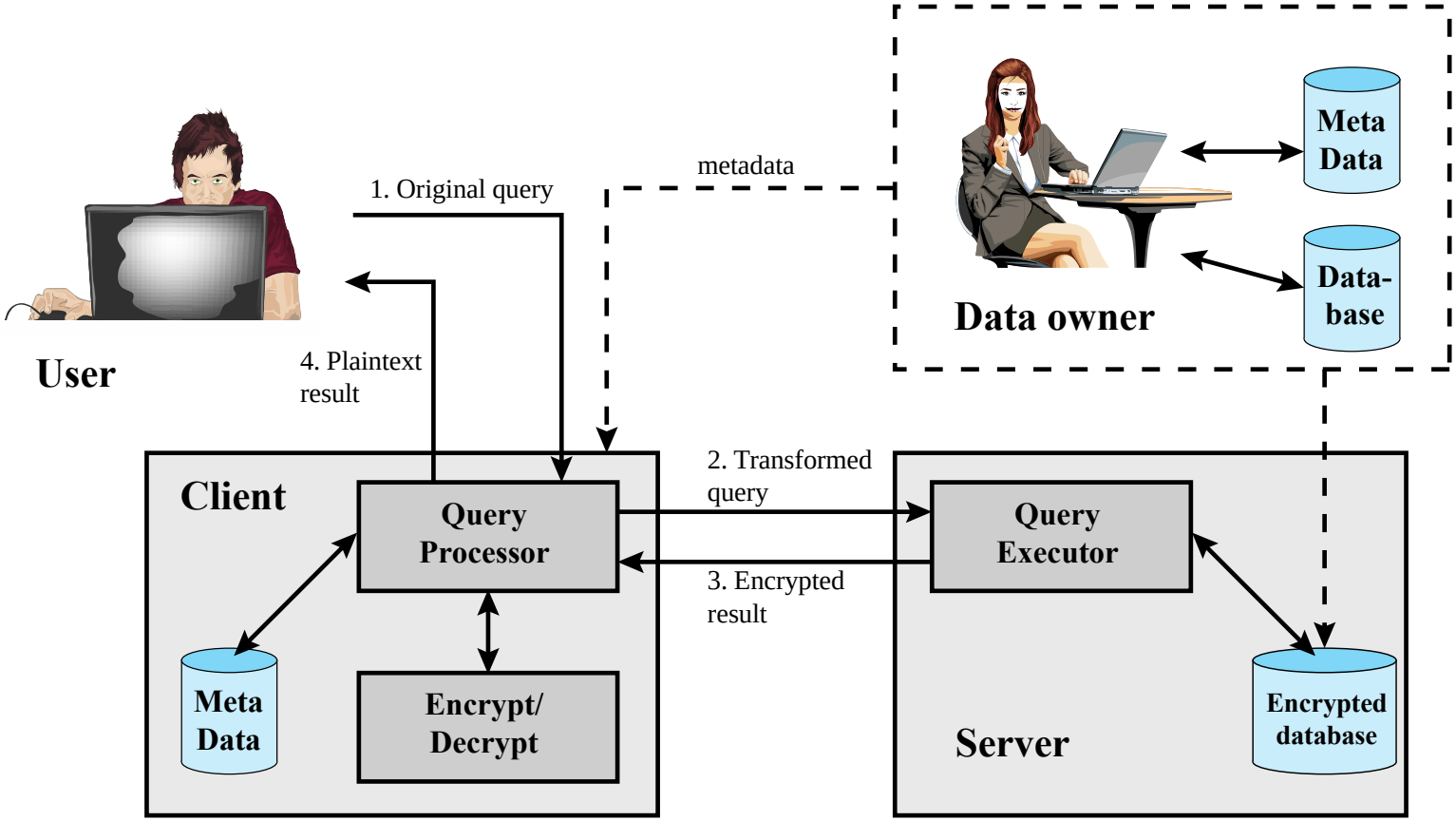


Figure 5.9 A Database Encryption Scheme

Step 1: User Query

- User submits a normal SQL query.
- Query goes to the client's Query Processor.

Step 2: Query Processor + Metadata

- Client uses metadata (Enc Algo, Key etc) from data owner.
- Converts query into encrypted-compatible form.
- Produces a transformed query.

Step 3: Server Execution

- Server receives transformed query.
- Executes it on the encrypted database.
- Returns encrypted result.

Step 4: Client Decrypts

- Client receives encrypted result.
- Decrypts it using metadata and keys.

Step 5: User Receives Plaintext

- Client outputs plaintext result to user.
- Server never sees unencrypted information.

Role of Data Owner

- Supplies metadata and encryption rules.
- Controls keys and policies.
- Ensures encrypted DB stays consistent.

Issues with the Database Encryption Diagram

High-Level Conceptual Issues

- Data owner role unclear
- Metadata meaning not defined
- No explanation of key management

Encryption Model Problems

- Encryption type unspecified
- Query transformation oversimplified
- Client-side crypto oversimplified

Query Processing Issues

- How queries become 'transformed' not explained
- Server execution model unrealistic
- No mention of encrypted indexes or operators

Metadata Issues

- Metadata seems magical and undefined
- Client & owner both shown holding metadata
- Server has none (unrealistic in real encrypted DBs)

Security Misconceptions

- Server shown as totally blind, which is not always true
- No mention of key rotation or secure distribution
- Simplifies encrypted DB security to a fault

Performance & Practical Issues

- No depiction of overhead from encrypted operations
- Ignores index, join, and sort limitations
- Hides real-world constraints

Example Query

Employee Table

Ename	Did	Salarycode	Eid	Ephone
Robin	15	23	2345	6127092485
Neil	13	12	5088	6127092246
Jasmine	4	26	7712	6127099348
Cody	15	22	9664	6127093148
Holly	8	23	3054	6127092729
Robin	8	24	2976	6127091945
Smith	9	21	4490	6127099380

foreign key — primary key

```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 15
```

- Assume the encryption key *k* is used and the encrypted value of the department id 15 is $E(k, 15) = 1000110111001110$.
- Then, the query processor at the client could transform the preceding query into

```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 1000110111001110
```

Example Query Issues

- This method is certainly straightforward but, as was mentioned, lacks flexibility.
- For example, suppose the Employee table contains a salary attribute and the user wishes to retrieve all records for salaries less than $70K.
- There is no obvious way to do this, because the attribute value for salary in each record is encrypted.
- The set of encrypted values do not preserve the ordering of values in the original attribute.

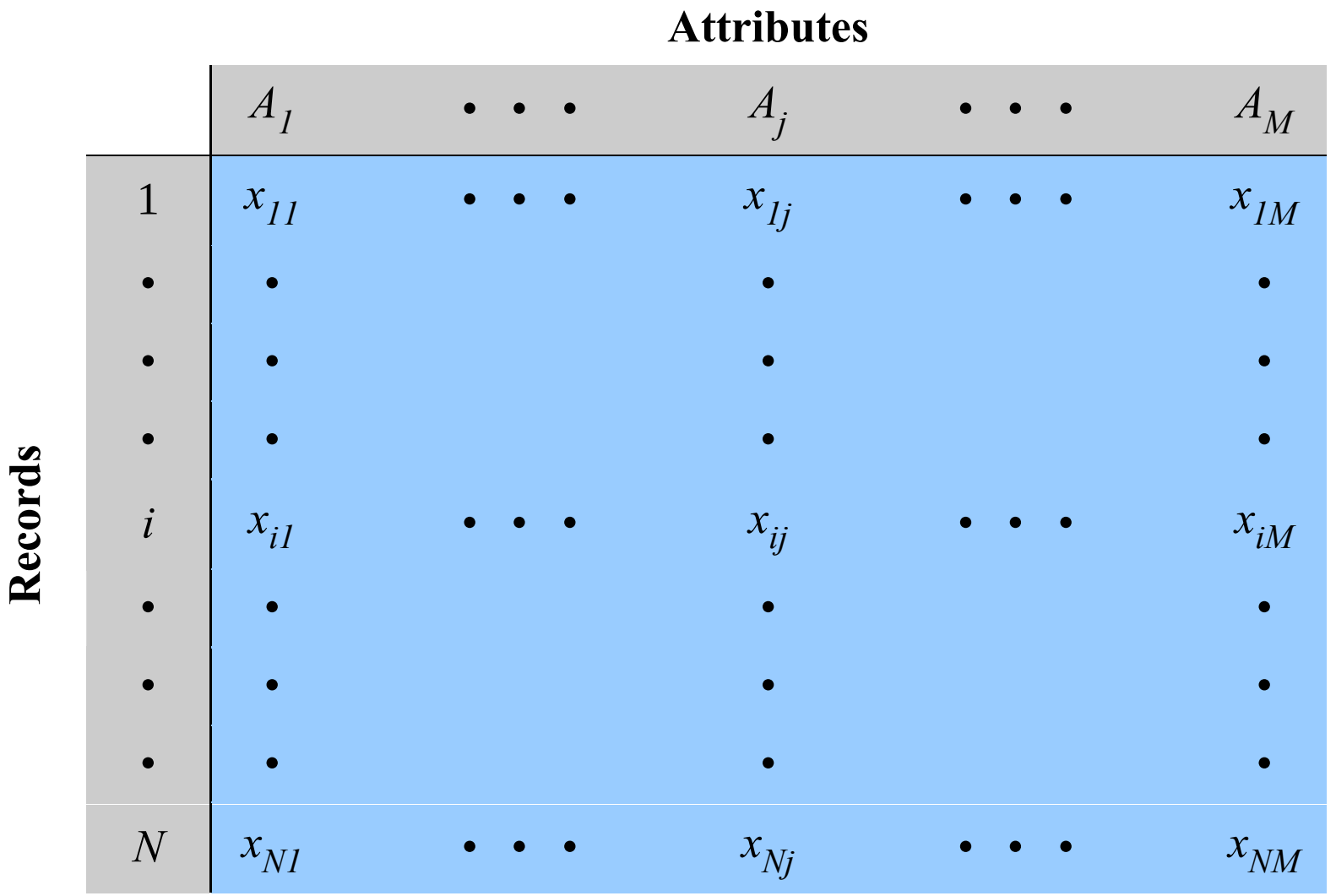
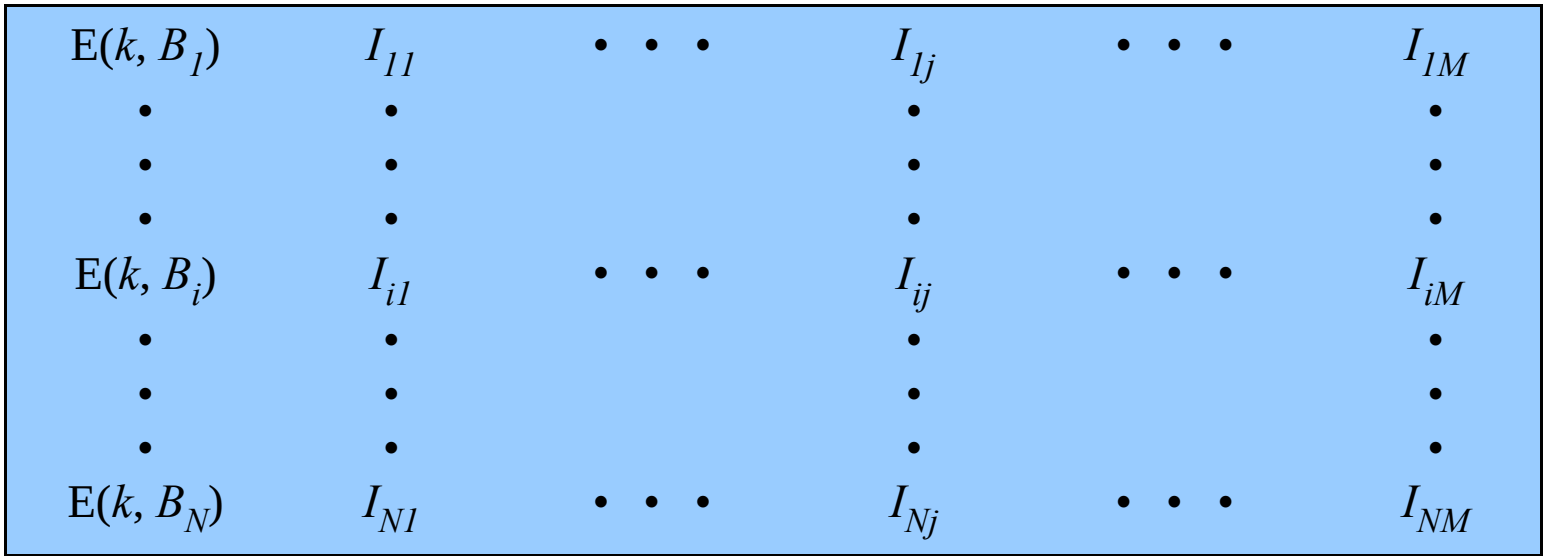


Figure 5.3 Abstract Model of a Relational Database



$B_i = (x_{i1} \| x_{i2} \| \ldots \| x_{iM})$

Figure 5.10 Encryption Scheme for Database of Figure 5.3

Table 5.3 Encrypted Database Example

(a) Employee Table

eid	ename	salary	addr	did
23	Tom	70K	Maple	45
860	Mary	60K	Main	83
320	John	50K	River	50
875	Jerry	55K	Hopewell	92

(b) Encrypted Employee Table with Indexes

E(*k*, *B*)	I(eid)	I(ename)	I(salary)	I(addr)	I(did)
1100110011001011…	1	10	3	7	4
0111000111001010…	5	7	2	7	8
1100010010001101…	2	5	1	9	5
0011010011111101…	5	5	2	4	9

A Flexible Approach (1/5)

- To provide more flexibility, the following approach is taken. Each record (row) of a table in the database is encrypted as a block. each row R_i is treated as a contiguous block

$$B_i = (x_{i1} \| x_{i2} \| \ldots \| x_{iM}).$$

- The entire row is encrypted, expressed as

$$E(k, B_i) = E(k, (x_{i1} \| x_{i2} \| \ldots \| x_{iM}))$$

- For some or all of the attributes an index value is created. For each row *Ri* of the unencrypted database, the mapping is as follows (see Figure 5.10):

$$(x_{i1}, x_{i2}, \ldots, x_{iM}) \rightarrow [E(k, B_i), I_{i1}, I_{i2}, \ldots, I_{iM}]$$

A Flexible Approach (2/5)

- Table 5.3 provides an example of this mapping. Suppose employee ID (eid)values lie in the range [1, 1000].
- We can divide these values into five partitions: [1, 200], [201, 400], [401, 600], [601, 800], and [801, 1000]; then assign index values 1, 2, 3, 4, and 5, respectively.
- For a text field, we can derive an index from the first letter of the attribute value. For the attribute *ename* , let us assign index 1 to values starting with A or B, index 2 to values starting with C or D, and so on.
- Similar partitioning schemes can be used for each of the attributes. Table 5.3b shows the resulting table.

A Flexible Approach (3/5)

- The values in the first column represent the encrypted values for each row.
- The actual values depend on the encryption algorithm and the encryption key. The remaining columns show index values for the corresponding attribute values.
- The mapping functions between attribute values and index values constitute metadata that are stored at the client and data owner locations but not at the server.
- This arrangement provides for more efficient data retrieval. Suppose, for example, a user requests records for all employees with eid 6 < 300. The query processor requests all records with I (eid) = 2.
- These are returned by the server. The query processor decrypts all rows returned, discards those that do not match the original query, and returns the requested unencrypted data to the user.

A Flexible Approach (4/5)

- The indexing scheme just described does provide a certain amount of information to an attacker, namely a rough relative ordering of rows by a given attribute.

- To obscure such information, the ordering of indexes can be randomized. For example, the eid values could be partitioned by mapping [1, 200], [201, 400], [401, 600], [601, 800], and [801, 1000] into 2, 3, 5, 1, and 4, respectively.

- Because the metadata are not stored at the server, an attacker could not gain this information from the server.

- Other features may be added to this scheme. To increase the efficiency of accessing records by means of the primary key, the system could use the encrypted value of the primary key attribute values, or a hash value.
- In either case, the row corresponding to the primary key value could be retrieved individually.

A Flexible Approach (5/5)

- Different portions of the database could be encrypted with different keys, so users would only have access to that portion of the database for which they had the decryption key.

- This latter scheme could be incorporated into a *role-based access* control system.

Data Center Security

- Data center:
  - An enterprise facility that houses a large number of servers, storage devices, and network switches and equipment
  - The number of servers and storage devices can run into the tens of thousands in one facility
  - Generally includes redundant or backup power supplies, redundant network connections, environmental controls (A.C. and fire suppression), and various security devices
  - Can occupy one room of a building, one or more floors, or an entire building
- Examples of uses include:
  - Cloud service providers
  - Search engines
  - Large scientific research facilities
  - IT facilities for large enterprises

Data Center Elements

1. Servers and Storage modules
2. Ethernet switches
3. Cabling

4. Switch Types
    A. Top-of-Rack (ToR) switches
        A. also known as Edge/Access/Server Access Switches
    B. Aggregation/distribution switches

    C. Core Switches

Data Center Cabling

- Cross connect: A facility enabling the termination of cables, as well as their interconnection with other cabling or equipment.
- Horizontal cabling: Any cabling that is used to connect a floor's wiring closet to wall plates in the work areas to provide local area network (LAN) drops for connecting servers and other digital equipment to the network. The term horizontal is used because such cabling is typically run along the ceiling or floor.
- Backbone cabling: Run between data center rooms or enclosures and the main
- cross-connect point of a building

Important Threats To Data Center

1. Denial of service
2. Advanced persistent threats from targeted attacks
3. Privacy breaches
4. Application exploits such as SQL injection
5. Malware
6. Physical security threats

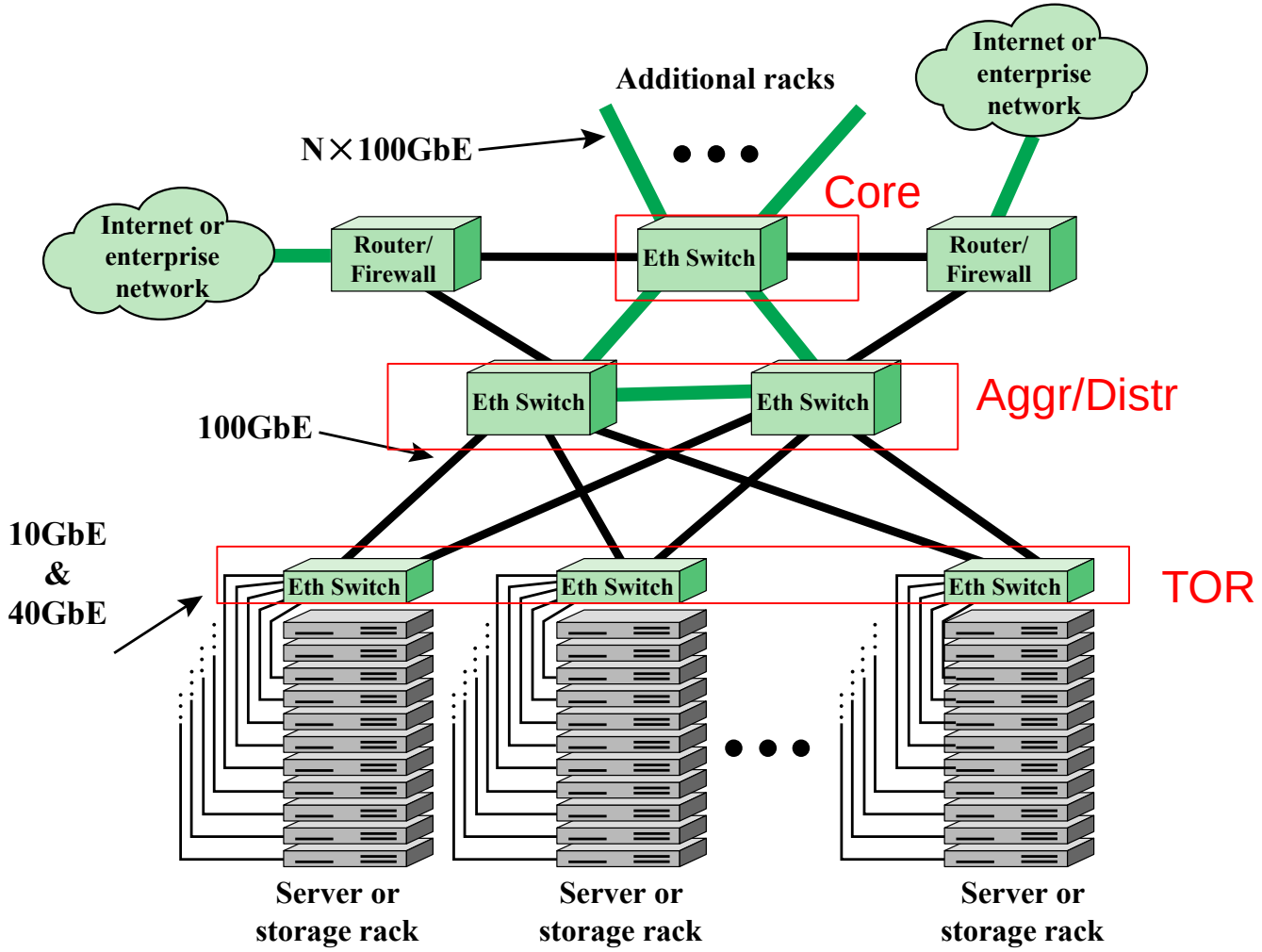


Figure 5.11 Key Data Center Elements

Data Center Security Aspects (1/2)

- Site security refers primarily to the physical security of the entire site including the building that houses the data center, as well as the use of redundant utilities.
- Physical security of the data center itself includes barriers to entry, such as a mantrap (a double-door single-person access control space) coupled with authentication techniques for gaining physical access. Physical security can also include security personnel, surveillance systems, and other measures (see Chapter 16)

Data Center Security Aspects (2/2)

- Network security is extremely important in a facility in which such a large collection of assets are concentrated in a single place and accessible by external network connections. Typically, a large data center will employ all of the network security techniques discussed in this text.

- Data security as opposed to the systems they reside on, involves techniques discussed in the remainder of this chapter.

Data Security	**Encryption, Password policiy, secure IDs, Data Protection (ISO 27002), Data masking, Data retention, etc.**
Network Security	**Firewalls, Anti-virus, Intrusion detection/prevention, authentication, etc.**
Physical Security	**Surveillance, Mantraps, Two/three factor authentication, Security zones, ISO 27001/27002, etc.**
Site Security	**Setbacks, Redundant utilities Landscaping, Buffer zones, Crash barriers, Entry points, etc.**

Figure 5.12 Data Center Security Model

TIA-492

- The Telecommunications Industry Association (TIA)
- TIA-492 (*Telecommunications Infrastructure Standard for Data Centers*) specifies the minimum requirements for telecommunications infrastructure of data centers
- Includes topics such as:
  - Network architecture
  - Electrical design
  - File storage, backup, and archiving
  - System redundancy
  - Network access control and security
  - Database management
  - Web hosting
  - Application hosting
  - Content distribution
  - Environmental control
  - Protection against physical hazards
  - Power management

Data Center Functional Areas (1/3)

1. Computer room:
   - Portion of the data center that houses date processing equipment.
2. Entrance room:
   - One or more entrance rooms house external network access provider equipment, plus provide the interface between the computer room equipment and the enterprise cabling systems.
   - Physical separation of the entrance room from the computer room provides better security.

Data Center Functional Areas (2/3)

3. Main distribution area:

- A centrally located area that houses the main crossconnect as well as core routers and switches for LAN and SAN (storage area network) infrastructures.

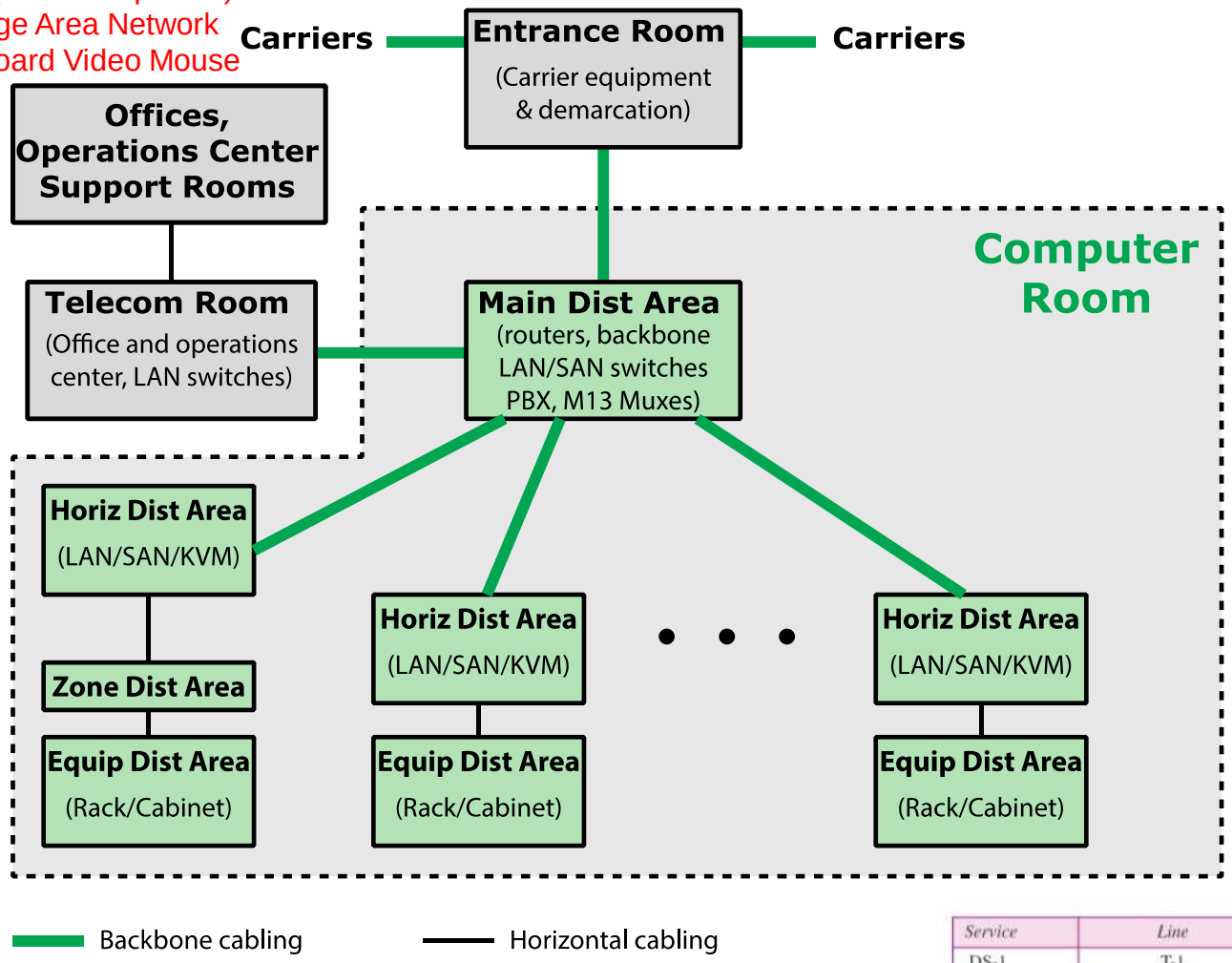
4.Horizontal distribution area (HDA):

- Serves as the distribution point for horizontal cabling and houses cross-connects and active equipment for distributing cable to the equipment distribution area.

Data Center Functional Areas (2/3)

- 5. Equipment distribution area (EDA): The location of equipment cabinets and racks, with horizontal cables terminating with patch panels.
- 5. Zone distribution area (ZDA): An optional interconnection point in the horizontal cabling between the HDA and EDA. The ZDA can act as a consolidation point for reconfiguration flexibility or for housing freestanding equipment such as mainframes.

PBX -private branch exchange
M13 Mux (M13 Multiplexer)
SAN-Storage Area Network
KVM-Keyboard Video Mouse



Service	*Line*	*Rate (Mbps)*	*Voice Channels*
DS-1	T-1	1.544	24
DS-2	T-2	6.312	96
DS-3	T-3	44.736	672
DS-4	T-4	274.176	4032

Figure 5.13 TIA-942 Compliant Data Center Showing Key Functional Areas

Tier	System design	Availability /Annual Downtime
1	•Susceptible to disruptions from both planned and unplanned activity •Single path for power and cooling distribution, no redundant components •May or may not have raised floor, UPS, or generator •Takes 3 months to implement •Must be shut down completely to perform preventive maintenance	99.671%/ 28.8 hours
2	•Less susceptible to disruptions from both planned and unplanned activity •Single path for power and cooling distribution, includes redundant components •Includes raised floor, UPS, and generator •Takes 3 to 6 months to implement •Maintenance of power path and other parts of the infrastructure require a processing shutdown	99.741%/ 22.0 hours
3	•Enables planned activity without disrupting computer hardware operation but unplanned events will still cause disruption •Multiple power and cooling distribution paths but with only one path active, includes redundant components •Takes 15 to 20 months to implement •Includes raised floor and sufficient capacity and distribution to carry load on one path while performing maintenance on the other	99.982%/ 1.6 hours
4	•Planned activity does not disrupt critical load and data center can sustain at least one worst-case unplanned event with no critical load impact • Multiple active power and cooling distribution paths, includes redundant components •Takes 15 to 20 months to implement	99.995%/ 0.4 hours

Table 5.4

Data Center Tiers Defined in TIA-942

(Table is on page 177 in textbook)

Summary

- The need for database security
- Database management systems
- Relational databases
  - Elements of a relational database system
  - Structured Query Language
- SQL injection attacks
  - A typical SQLi attack
  - The injection technique
  - SQLi attack avenues and types
  - SQLi countermeasures
- Database access control
  - SQL-based access definition
  - Cascading authorizations
  - Role-based access control
- Inference
- Database encryption
- Data center security
  - Data center elements
  - Data center security considerations
  - TIA-492