Information Security
CS 3002

Dr. Haroon Mahmood

Assistant Professor

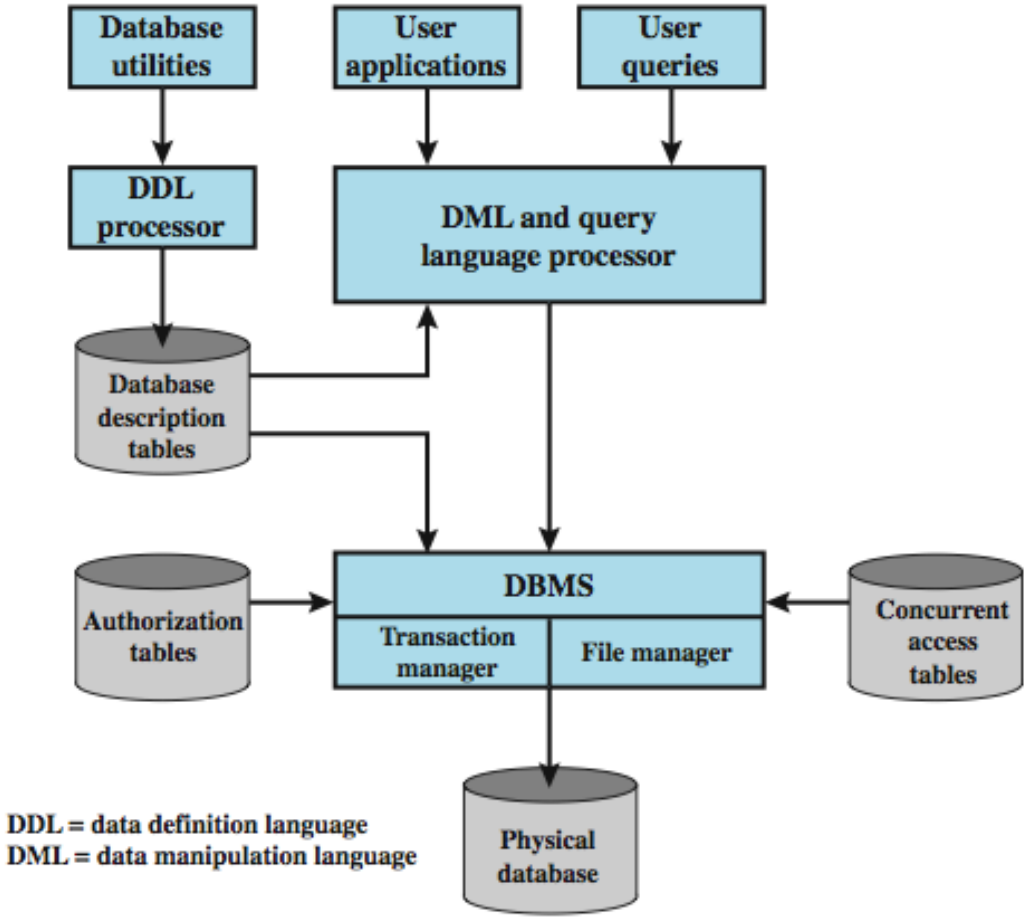
NUCES Lahore

Disclaimer: The contents of these slides have been taken from the book of Computer Security by William Stallings.

Database systems

- **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**

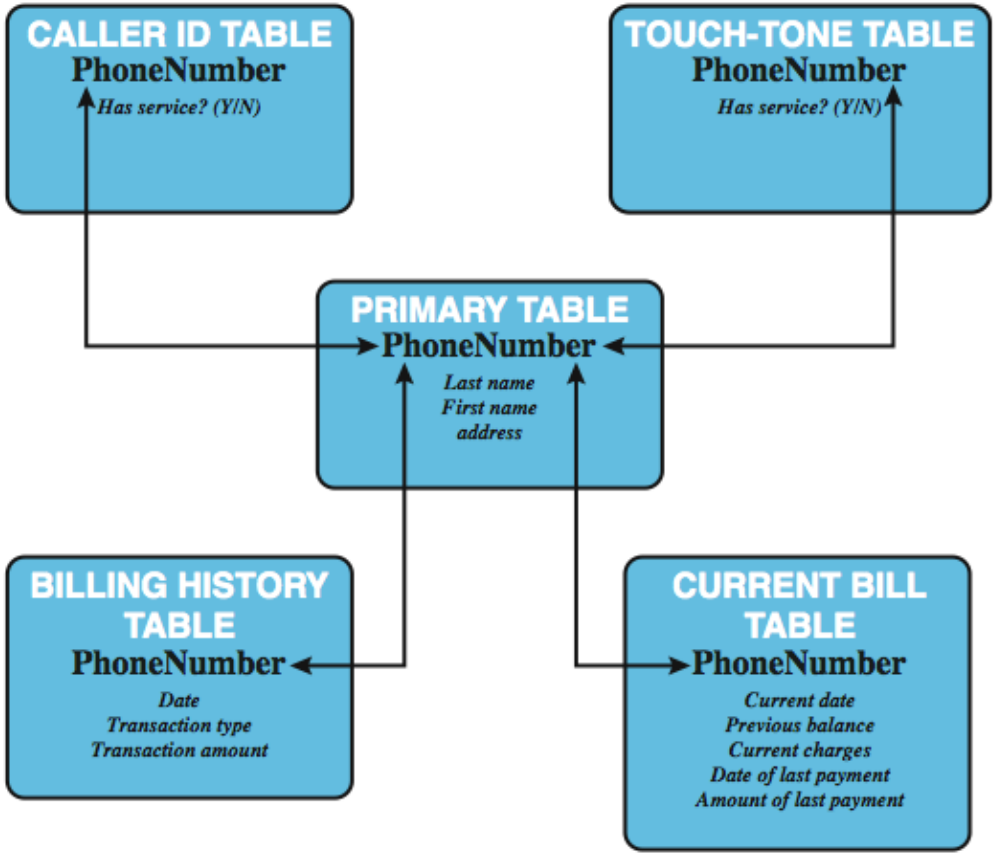
Database Architecture



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

A relational database example



Relational Database Elements

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

Structured Query Language

- **Structure Query Language (SQL)**
  - **originally developed by IBM in the mid-1970s**
  - **standardized language to define, manipulate, and query data in a relational database**
  - **several similar versions of ANSI/ISO standard**

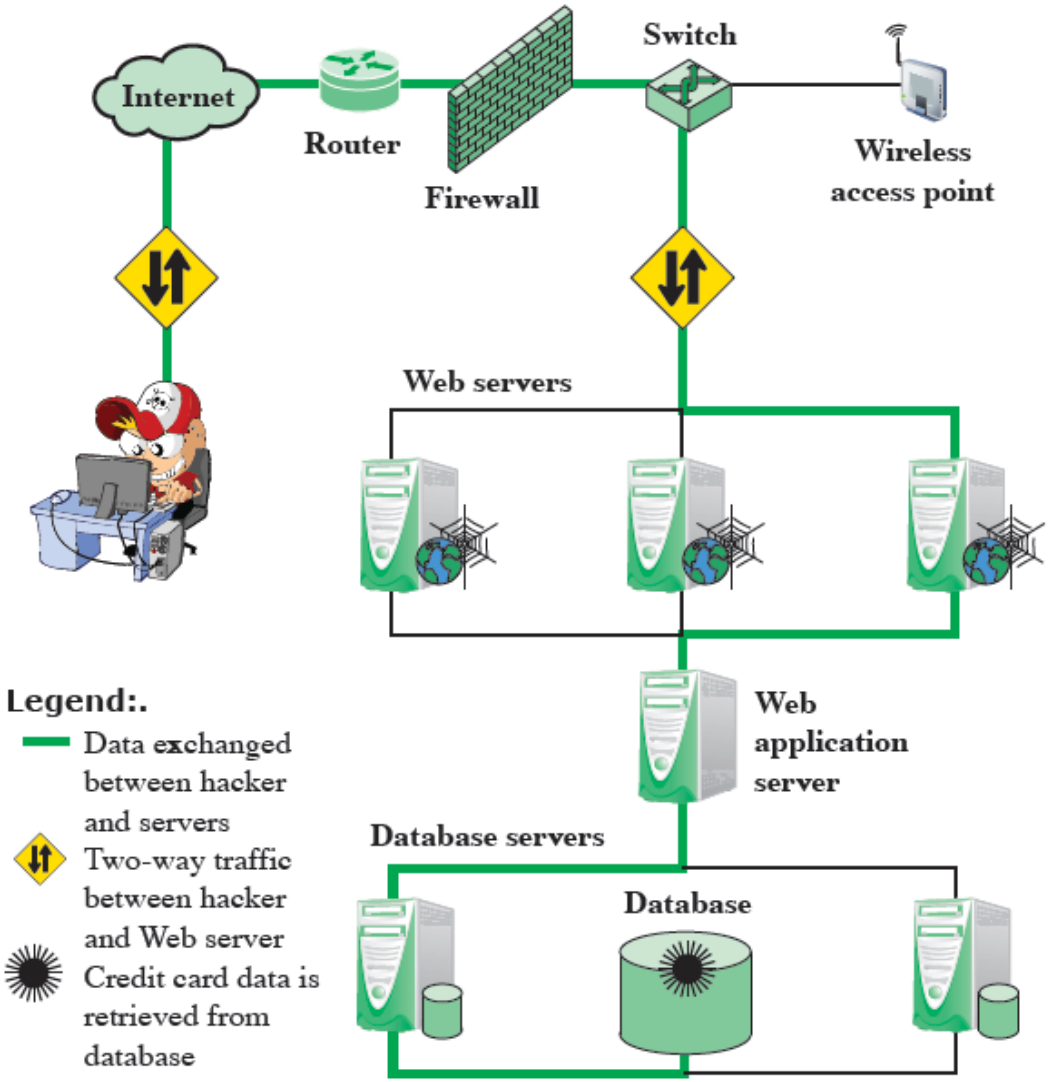
```
CREATE TABLE department (
        Did INTEGER PRIMARY KEY,
        Dname CHAR (30),
        Dacctno CHAR (6) )

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

```
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
```

SQL injection attacks

- One of the most prevalent and dangerous network-based security threats
- Sends malicious SQL commands to the database server
- 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 injection attack



Injection attack steps

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

Sample SQL injection

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

SELECT fname

FROM student

where fname is `'user prompt';`

What if user enters the following input?

User: John'; DROP table Course;--

Sample SQL injection

- var Shipcity;
  ShipCity = Request.form (“ShipCity”);
  var sql = “select * from OrdersTable where ShipCity = ‘ ” +
  ShipCity + “ ’ ”;

- SELECT * FROM OrdersTable WHERE ShipCity = ‘Redmond’

- Suppose, however, the user enters the following:
  Redmond’; DROP table OrdersTable --

- This results in the following SQL query:
  SELECT * FROM OrdersTable WHERE ShipCity =
  ‘Redmond’; DROP table OrdersTable--

In-band attacks

- **Tautology:** This form of attack injects code in one or more conditional statements so that they always evaluate to true

- **End-of-line comment:** After injecting code into a particular field, legitimate code that follows are nullified through usage of end of line comments

- **Piggybacked queries:** The attacker adds additional queries beyond the intended query, piggy-backing the attack on top of a legitimate request

Sample SQL injection: tautology

$query= “

SELECT info FROM user WHERE name =

`$_GET[“name”]’ AND pwd = `GET[“pwd”]`

”;

Attacker enters: ’ OR 1=1 --

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

Inferential attack (gathering info)

- 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
  - Illegal/logically incorrect queries: lets an attacker gather important information about the type and structure of the backend database of a Web application
    - The vulnerability leveraged by this attack is that the default error page returned by application servers is often overly descriptive.
  - 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 to observe the functionality in each case.

SQLi countermeasures

- **Defensive coding:** Stronger data validation
  - type checking, to check that inputs that are supposed to be numeric contain no characters other than digits
  - pattern matching to try to distinguish normal input from abnormal input
- **Detection**
  - Signature based
  - Anomaly based
  - Code analysis
- **Runtime prevention:** Check queries at runtime to see if they conform to a model of expected queries

Database Access Control

- DBMS provide access control for database
- It operates on assumption that user is already authenticated

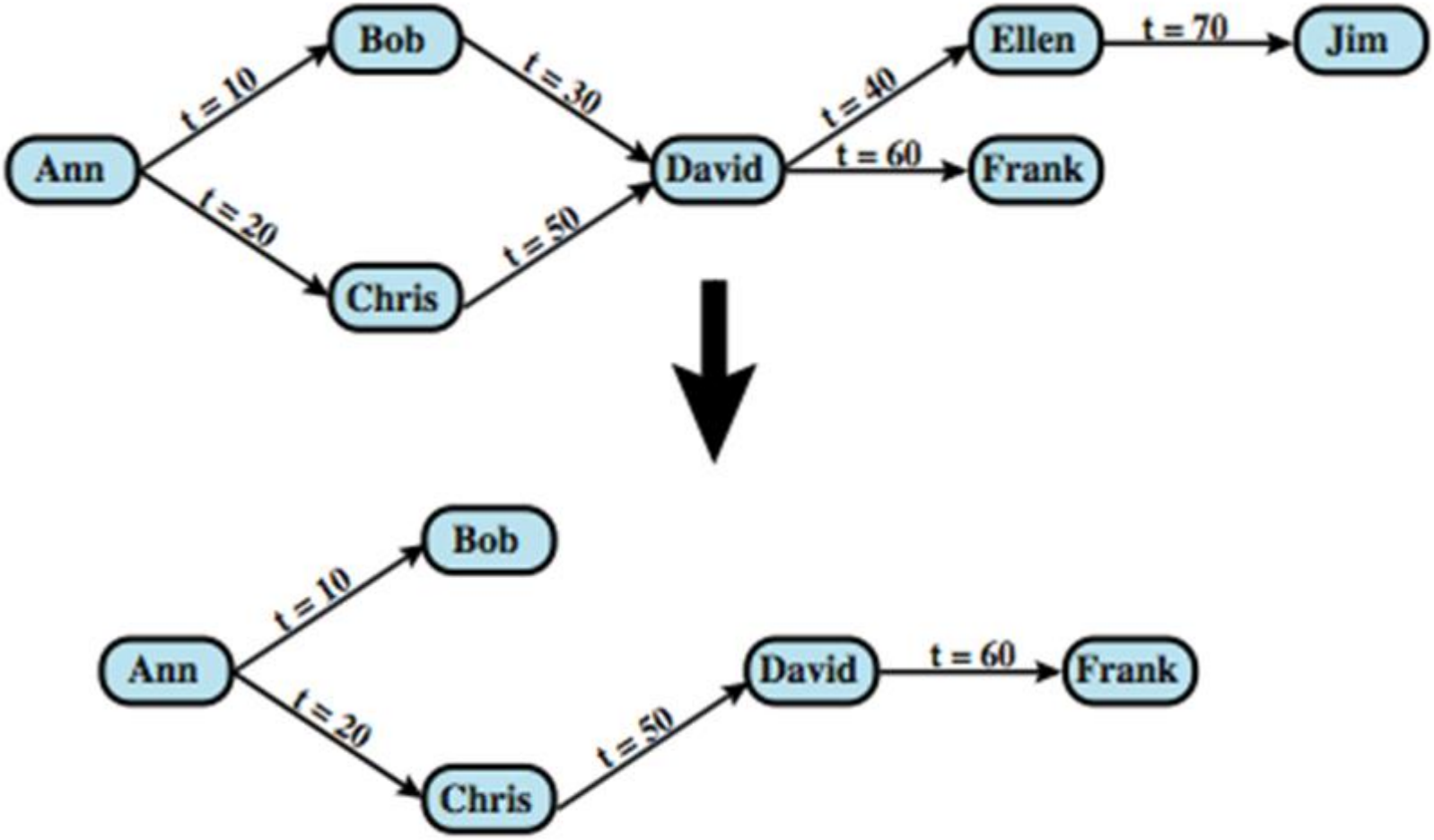
- DBMS provides specific access rights to portions of the database
  - e.g. create, insert, delete, update, read, write
  - to entire database, tables, selected rows or columns
  - possibly dependent on contents of a table entry

- can support a range of policies:
  - centralized administration
  - ownership-based administration
  - decentralized administration

SQL Access Controls

- If the user has access to the entire database or just portions of it
- Two commands:
  - `GRANT {privileges | role} [ON table] TO {user | role | PUBLIC} [IDENTIFIED BY password] [WITH GRANT OPTION]`
    - e.g. GRANT SELECT ON ANY TABLE TO john
  - `REVOKE {privileges | role} [ON table] FROM {user | role | PUBLIC}`
    - e.g. REVOKE SELECT ON ANY TABLE FROM john
  - `WITH GRANT OPTION: whether grantee can grant "GRANT" option to other users`
- Typical access rights are:
  - `SELECT, INSERT, UPDATE, DELETE, REFERENCES`

Cascading Authorizations



Role-Based Access Control (RBAC)

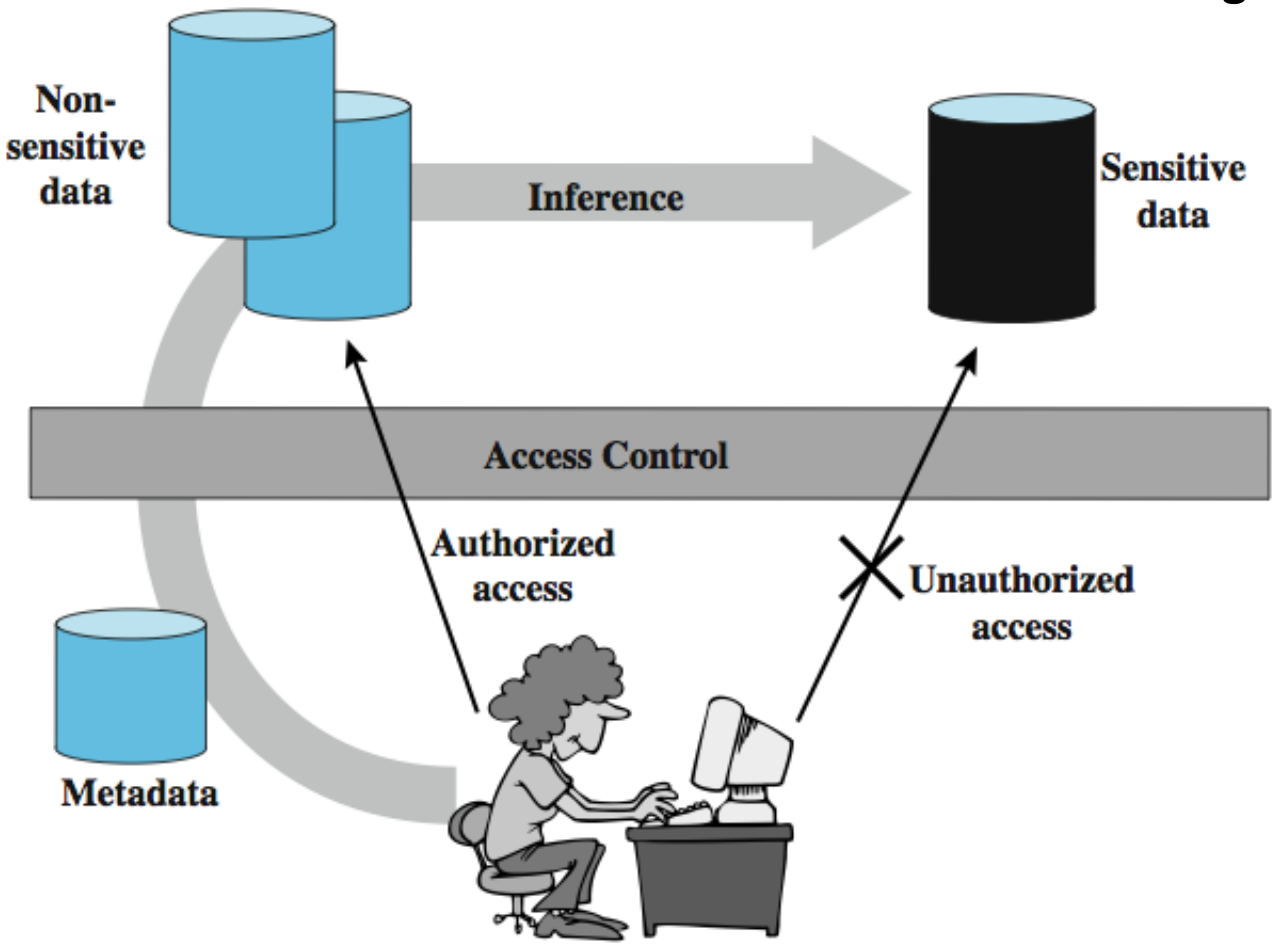
- **Role-based access control work well for DBMS**
  - **eases admin burden, improves security**

- **Categories of database users:**
  - **application owner**
  - **end user**
  - **Administrator**

- **DB RBAC must manage roles and their users**

Inference

- The process of performing authorized queries and deducing unauthorized information from the legitimate responses received
- A combination of data items can be used to infer data of a higher sensitivity



Inference Example

Name	Position	Salary ($)	Department	Dept. Manager
Andy	senior	43,000	strip	Cathy
Calvin	junior	35,000	strip	Cathy
Cathy	senior	48,000	strip	Cathy
Dennis	junior	38,000	panel	Herman
Herman	senior	55,000	panel	Herman
Ziggy	senior	67,000	panel	Herman

(a) Employee table

Position	Salary ($)
senior	43, 000
junior	35,000
senior	48,000

Name	Department
Andy	strip
Calvin	strip
Cathy	strip

(b) Two views

Name	Position	Salary ($)	Department
Andy	senior	43,000	strip
Calvin	junior	35,000	strip
Cathy	senior	48,000	strip

(c) Table derived from combining query answers

Inference example

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

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

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

Inference Countermeasures

- **Inference detection at database design**
  - **alter database structure or access controls**
  - **Splitting a table into multiple tables**
  - **More fine grain access control roles**

- **Inference detection at query time**
  - **by monitoring and altering or rejecting queries**

Statistical Databases

- Provides data of a statistical nature
  - e.g. counts, averages
- Two types:
  - pure statistical database
  - ordinary database with statistical access
    - some users have normal access, others statistical
- Access control objective to allow statistical use without revealing individual entries
- Security problem is one of inference

Example

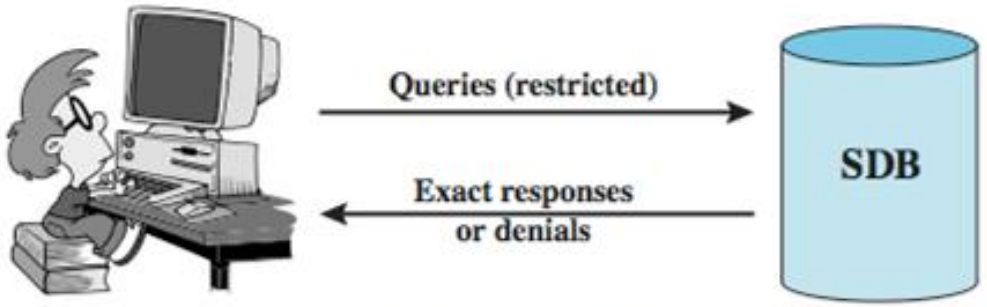
- Count (EE.female) = 1
- Sum (EE.Female, GP) = 2.7

- Salary range for system analyst with BS (50K, 60K)
- Salary range for system analyst with MS (65K, 70K)

- What if the change in payroll is 70K or 140K?

Protecting Against Inference

Query restriction:



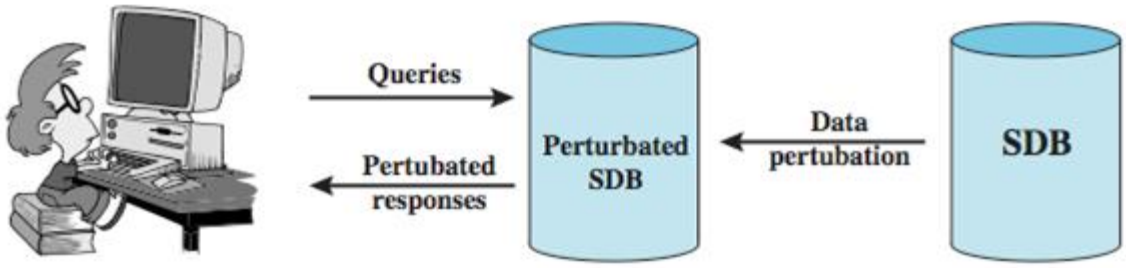
(a) Query set restriction

- Rejects a query that can lead to a compromise.
- The answers provided are accurate.
- The simplest form of query restriction is query size restriction. For a database of size N (number of rows, or records), a query q(C) is permitted only if the number of records that match C satisfies: k <= |X(C)| where k is a fixed integer greater than 1.
- In practice, queries of the form q(All) are allowed, enabling users to easily access statistics calculated on the entire database.

Perturbation

- **Add noise to statistics generated from data**
  - **will result in differences in statistics**

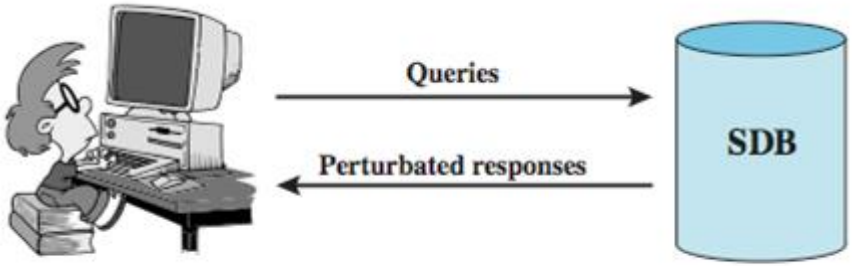
- **Data perturbation techniques**
  - **data swapping**



(b) Data perturbation

Perturbation

- **Output perturbation techniques**
  - **Random-sample query**
    - **Calculate the statistics on a properly selected sampled query subset**
  - **Statistic adjustment**
    - **Adjusting the answer up or down by a given amount in some systematic fashion**



(c) Output perturbation

  - **Must minimize loss of accuracy in results**

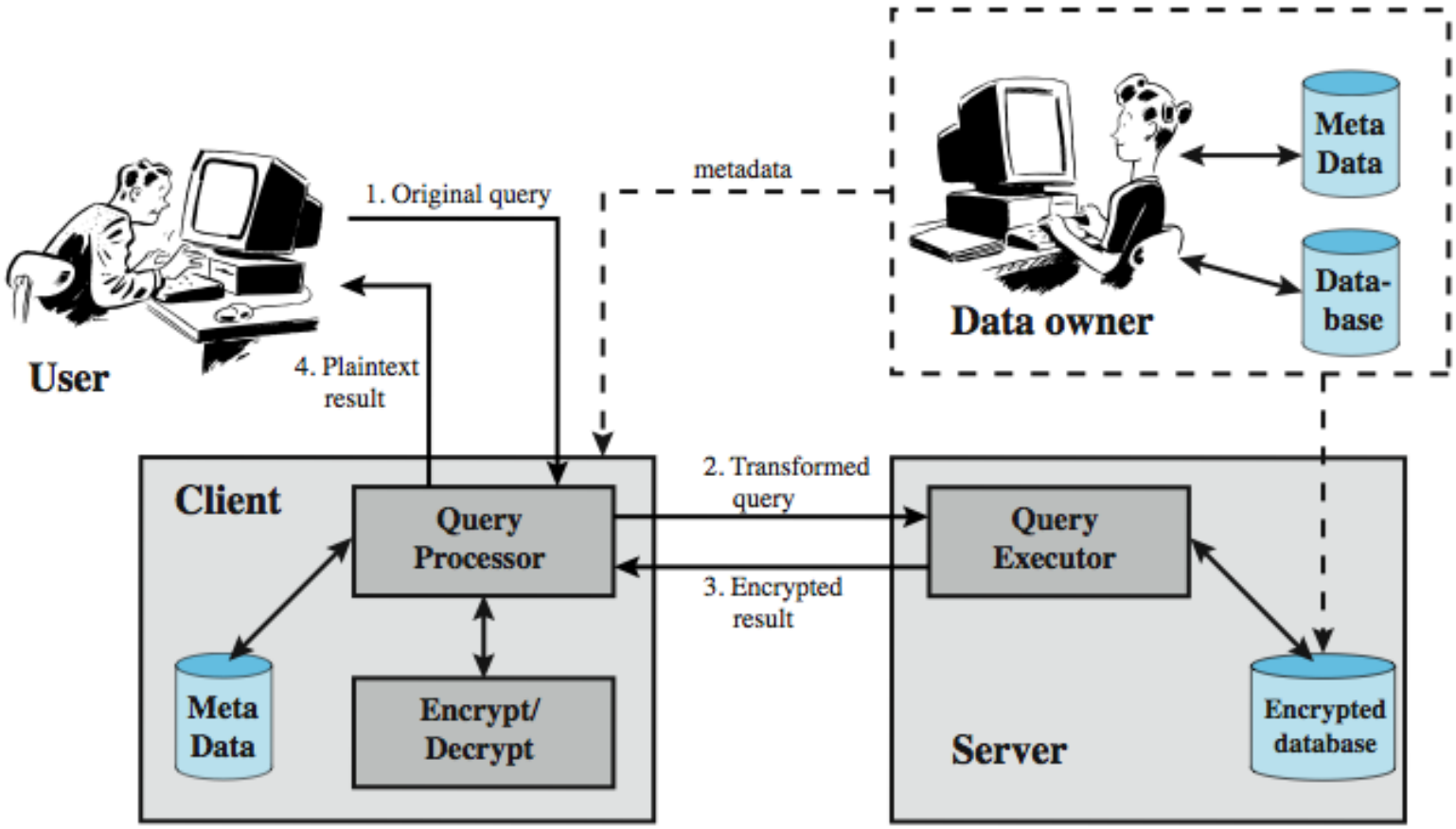
Perturbation

- Add noise to statistics generated from data
  - will result in differences in statistics
- Data perturbation techniques
  - data swapping
- Output perturbation techniques
  - random-sample query
  - statistic adjustment
- Must minimize loss of accuracy in results

Database Encryption

- **Databases typically a valuable info resource**
  - **protected by multiple layers of security: firewalls, authentication, O/S access control systems, DB access control systems, and database encryption**
- **Can encrypt**
  - **entire database - very inflexible and inefficient**
  - **individual fields - simple but inflexible**
  - **records (rows) or columns (attributes) - best**
    - **also need attribute indexes to help data retrieval**
- **Varying trade-offs**

Database Encryption



Database Encryption

- **Databases typically a valuable info resource**
  - **protected by multiple layers of security: firewalls, authentication, O/S access control systems, DB access control systems, and database encryption**
- **Can encrypt**
  - **entire database - very inflexible and inefficient**
  - **individual fields - simple but inflexible**
  - **records (rows) or columns (attributes) - best**
    - **also need attribute indexes to help data retrieval**
- **Varying trade-offs**

Secured operations

- **Suppose that each individual item in the database is encrypted separately.**
- **The encrypted database is stored at the server, but the server does not have the key, so that the data is secure at the server. The client system does have a copy of the encryption key.**
- **A user at the client can retrieve a record from the database with the following sequence:**
  1. **The user issues an SQL query for fields from one or more records with a specific value of the primary key.**
  2. **The query processor at the client encrypts the primary key, modifies the SQL query accordingly, and transmits the query to the server.**
  3. **The server processes the query using the encrypted value of the primary key and returns the appropriate record or records.**
  4. **The query processor decrypts the data and returns the results.**

Example

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

Assume that the encryption key *k* is used and that 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

Issues

- **This method is certainly straightforward but 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.

Record level encryption and indexing

- **Each record (row) of a table in the database is encrypted as a block which is treated as a contiguous block.**
- **To assist in data retrieval, attribute indexes are associated with each table. For some or all of the attributes an index value is created.**
- **For each row in the original database, there is one row in the encrypted database.**
- **For any attribute, the range of attribute values is divided into a set of non-overlapping partitions that encompass all possible values, and an index value is assigned to each partition.**

Example

- **Suppose that 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]; and 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.**

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)
11001100110011011…	1	10	3	7	4
01110001110010010…	5	7	2	7	8
11000100100011101…	2	5	1	9	5
00110100111111101…	5	5	2	4	9