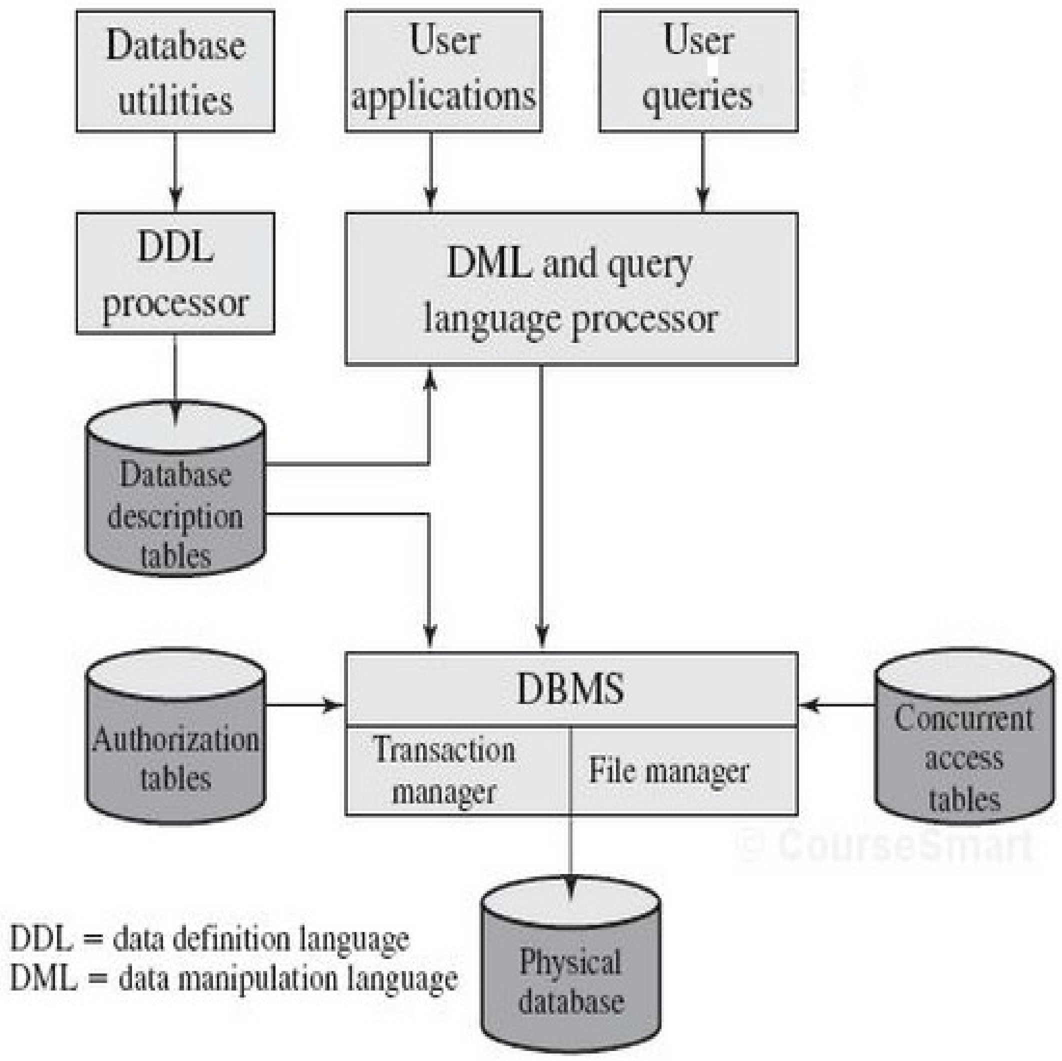
Module 8: Database Security

Dr. Natarajan Meghanathan
Associate Professor of Computer Science,
Jackson State University, Jackson, MS
E-mail: natarajan.meghanathan@jsums.edu

Database Management Systems

- A database is a structured collection of data; contains (captures) relationships between data items and groups of data items.
- A Database Management System (DBMS) is a suite of programs for constructing and maintaining the database (DDL); and for offering ad hoc query facilities to multiple end users (SQL) and applications (DML).
- A query language provides a uniform interface to the database for users and applications.



Source: Figure 5.1: W. Stallings: Computer Security: Principles and Practice, 2nd edition

Database Security

- The authorization tables ensure the user has permission to execute the query language statement on the database.

- The concurrent access table prevents conflicts when simultaneous, conflicting commands are executed.

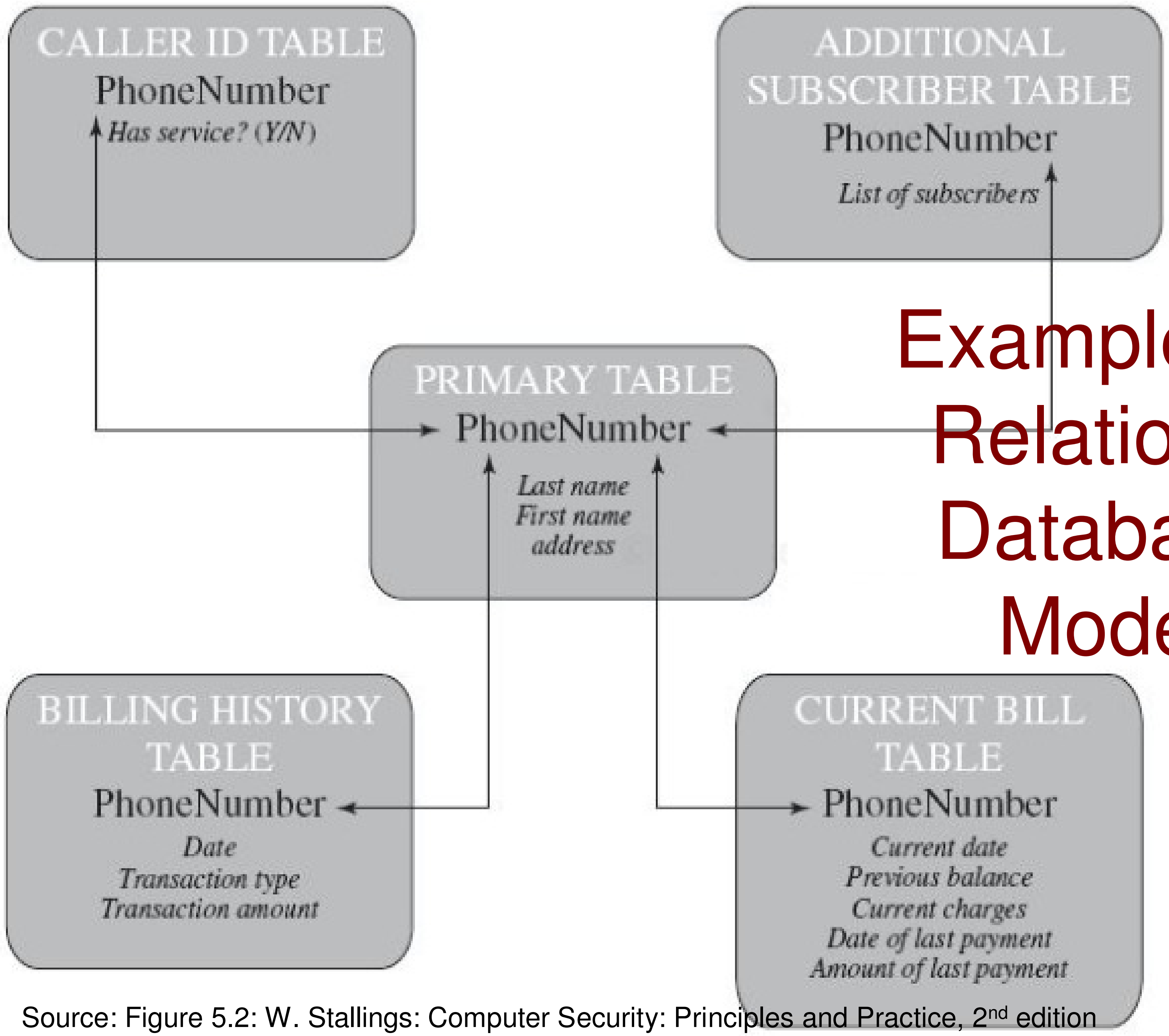
- Operating System security mechanisms typically control read and write access to entire files.
  - They could not be used to limit access to specific records or fields in that file.

- A DBMS allows more detailed access control to be specified over the data records as well as over a wide range of commands, such as to select, insert, update, or delete specified items in the database.

Relational Database

- Basic building block: Table of data, consisting of rows and columns.
  - Each column holds a particular type of data (ref. as attribute)
  - Each row contains a specific value for each column (tuple/ record)
- Need to store the table with a unique identifier (for each row) so that the table can be indexed (searched).
- The drawback of using a single table is that some of the column positions for a given row may be blank (not used).
  - Also, any time a new service or new type of information is incorporated in the database, more columns must be added and the database and accompanying software must be redesigned and rebuilt.
- The relational database structure enables the creation of multiple tables tied together by a unique identifier (primary key/foreign key) that is present in all the tables.
- The query language allows the user to request selected items of data from all records that fit a given set of criteria. The software then figures out how to extract the requested data from one or more tables.

Example for Relational Database Model

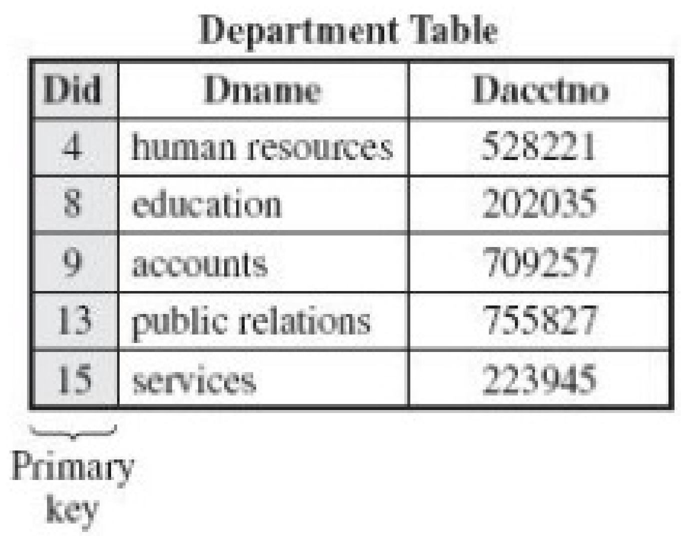


Source: Figure 5.2: W. Stallings: Computer Security: Principles and Practice, 2nd edition

Primary Key, Foreign Key and View

- To create a relationship between two tables, the attributes that define the primary key in one table must appear as attributes in another table, where they are referred to as a foreign key.
- Whereas the value of a primary key must be unique for each row of its table, a foreign key value can appear multiple times in a table, so that there is a one-to-many relationship.
- A “**View**” is a virtual table:
  - Is the result of a query that returns selected rows and columns from one or more tables.
  - It is possible to construct a view from a single table.
  - Views are often used for security purposes. A view can provide restricted access to a relational database so that a user or application only has access to certain rows or columns.

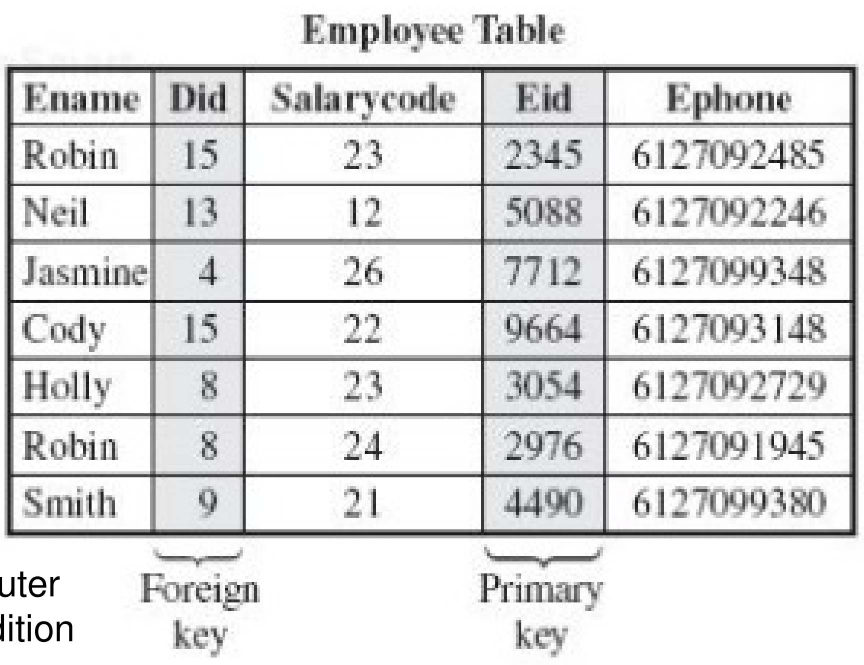
Relational Database Example



Department Table

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

Primary key

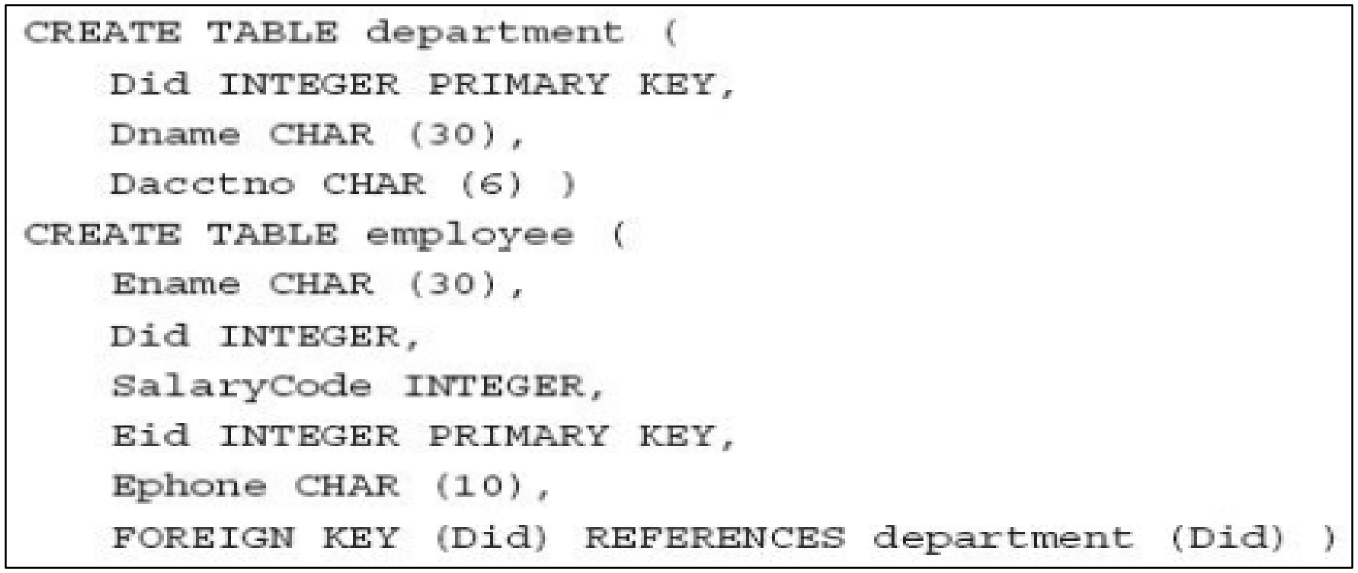


Employee Table

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

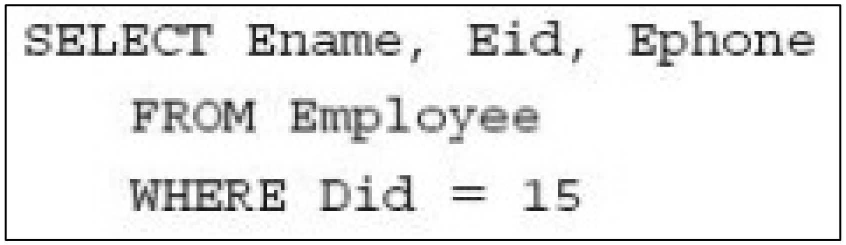
Source: Figure 5.3: W. Stallings: Computer Security: Principles and Practice, 2nd edition



```
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) )
```

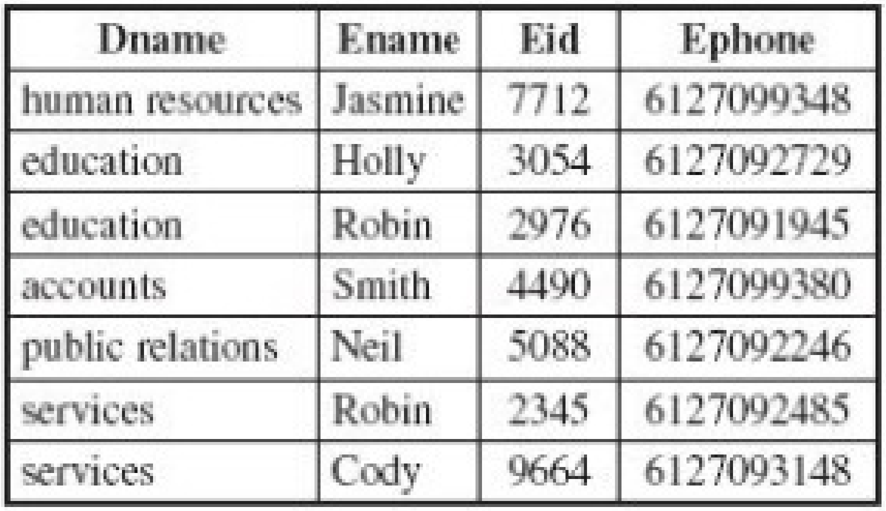
Structured Query Language (SQL)

- SQL is used to define schema, manipulate, and query data in a relational database.

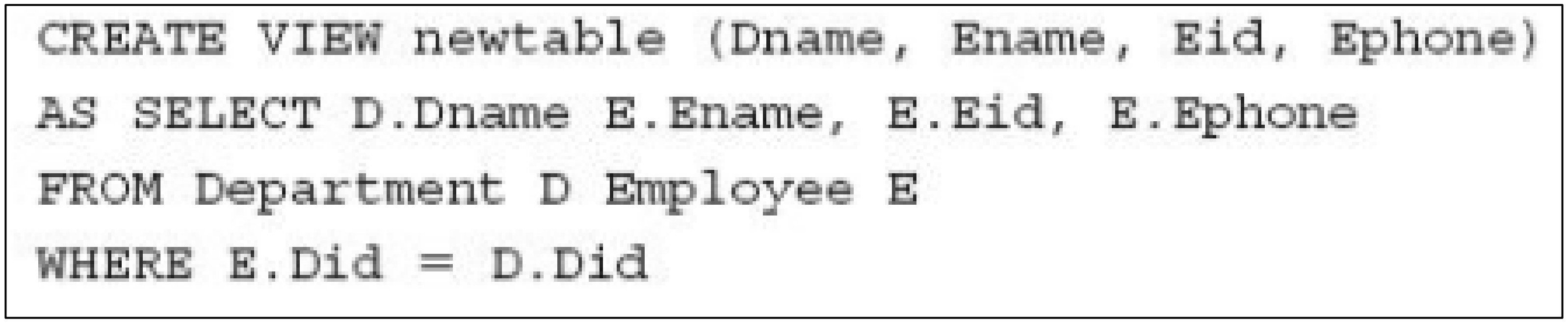


```
SELECT Ename, Eid, Ephone
    FROM Employee
    WHERE Did = 15
```

A View on the two tables



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



```
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
```

Database Access Control

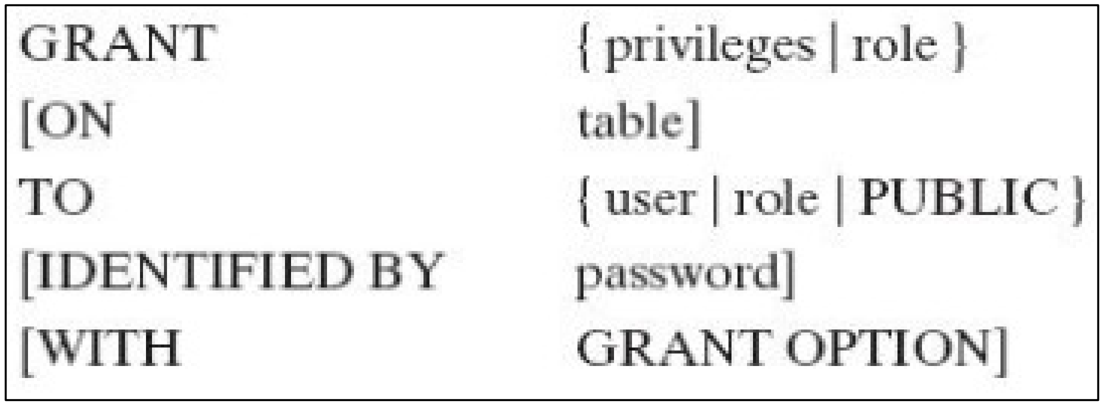
- Assumption: The underlying computer system (housing the database) has authenticated the user to access the system as well as granted access to the database.
- A Database Access Control System provides a specific capability that controls access to portions of the database.
- A DBMS can support a range of administrative policies:
  - Centralized admin: A small number of privileged users may grant and revoke access rights
  - Ownership-based admin: The owner
- Access rights for a DBMS:
  - Create, Insert, Delete, Update, Read, and Write
- Access rights can be at different levels of granularity
  - Entire database, individual tables, selected rows or columns within a table.
- Access rights can be determined based on the contents of a table entry.
  - E.g., In a personnel database, a department manager may only be allowed to view salary info for employees in his/her department.

Common Access Rights (Privileges)

- Select:
  - Grantee may read entire database; individual tables; or specific columns in a table
- Insert
  - Grantee may insert rows in a table; or insert rows with values for specific columns in a table.
- Update:
  - Semantics is similar to INSERT
- Delete:
  - Grantee may delete rows from a table
- References:
  - Grantee is allowed to define foreign keys in another table that refer to the specified columns.

SQL-based Access Definition

- SQL provides two commands for managing access rights: GRANT and REVOKE.
- GRANT command
  - Used to grant one or more access rights or can be used to assign a user to a role.
  - For access rights, the command can optionally specify that it applies only to a specified table.
  - The TO clause specifies the user or role to which the rights are granted.
  - A PUBLIC value indicates that any user has the specified access rights.
  - The optional IDENTIFIED BY clause specifies a password that must be used to revoke the access rights of this GRANT command.
  - The GRANT OPTION indicates that the grantee can grant this access right to other users, with or without the grant option.

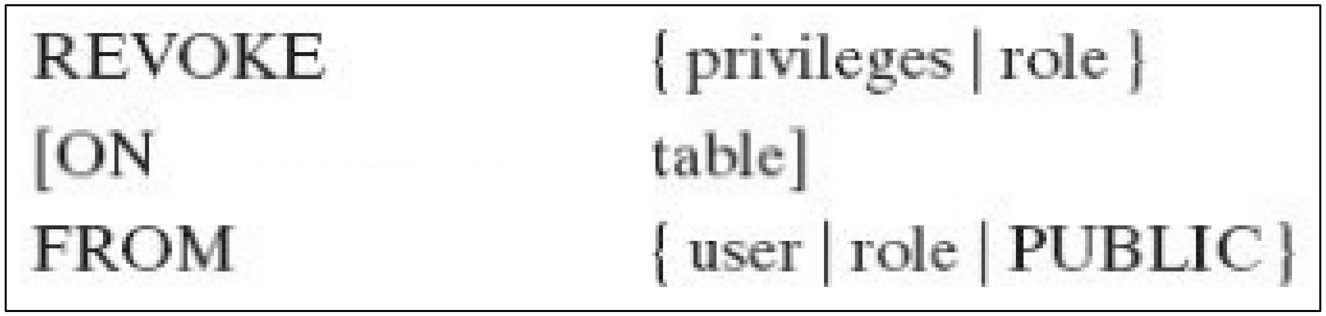


```
GRANT               { privileges | role }
[ON                 table]
TO                  { user | role | PUBLIC }
[IDENTIFIED BY      password]
[WITH               GRANT OPTION]
```

Example:
GRANT SELECT ON ANY TABLE TO kenny

This statement permits user ‘kenny’ to query any table in the database.

REVOKE command



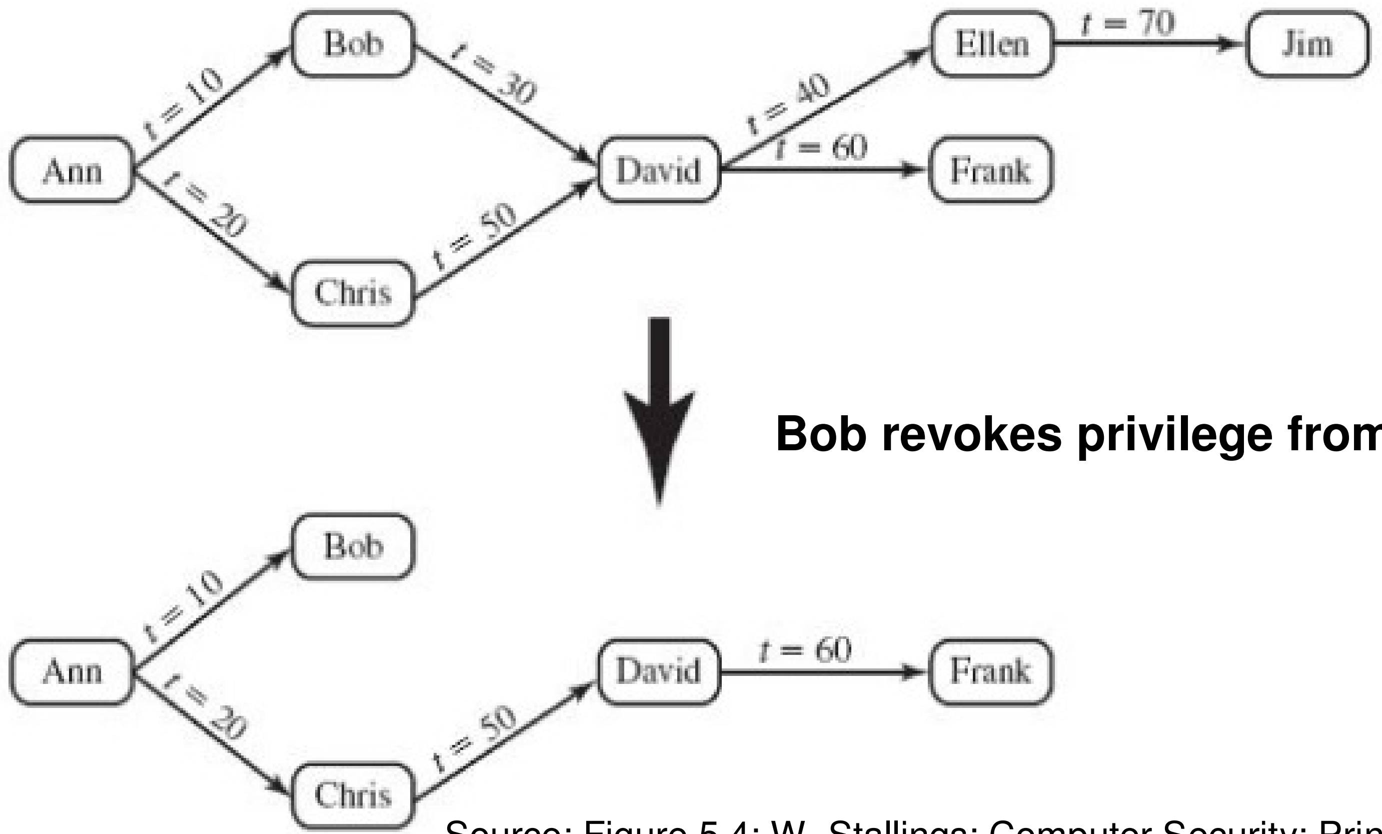
```
REVOKE      { privileges | role }
[ON         table]
FROM        { user | role | PUBLIC }
```

Example:
REVOKE SELECT ON ANY TABLE FROM kenny

This statement revokes the access rights of the preceding example.

Cascading Authorizations

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



Source: Figure 5.4: W. Stallings: Computer Security: Principles and Practice, 2nd ed.

Inference

- Inference is the process of performing authorized queries and deducing unauthorized information from the legitimate responses received.
- The inference problem arises when a 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 attackers may use non-sensitive data and metadata (knowledge about correlations or dependencies among data items)
- The information transfer path by which unauthorized data is obtained is referred to as an inference channel.
- Two techniques to derive additional information:
  - Analyzing functional dependencies between attributes within a table or across tables
  - Merging views with the same constraints

Inference Example

Employee Table

Name	**Position**	**Salary ($)**	**Department**	**Location**
Andrew	Programmer Analyst	$80,000	Software	Jackson, MS
Robert	Quality Control	$65,000	Software	Memphis, TN
Sheela	Software Developer	$90,000	Software	Atlanta, GA
Victor	Systems Engineer	$75,000	Systems	San Jose, CA
Mary	Systems Administrator	$95,000	Systems	Seattle, WA
Ryan	Network Engineer	$87,000	Systems	Boston, MA

View V1

Name	**Position**
Andrew	Programmer Analyst
Robert	Quality Control
Sheela	Software Developer

View V2

Salary ($)	**Location**
$80,000	Jackson, MS
$65,000	Memphis, TN
$90,000	Atlanta, GA

```
CREATE View V1 AS
SELECT Name, Position
FROM Employee
WHERE Department = “Software”
```

```
CREATE View V2 AS
SELECT Salary, Location
FROM Employee
WHERE Department = “Software”
```

Inference Example (continued…)

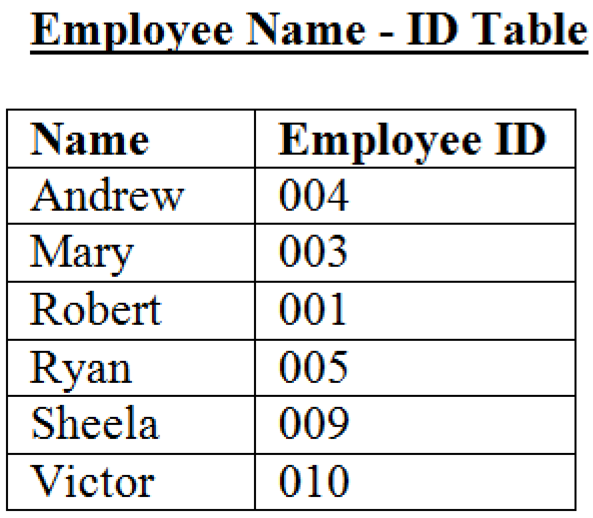
- A user who knows the structure of the Employee table and that who knows that the View tables maintain the same row order as that of the Employee table can then merge the two views and construct a table from which s/he can infer the salary of each employee in the Software department.

Name	Position	Salary ($)	Location
Andrew	Programmer Analyst	$80,000	Jackson, MS
Robert	Quality Control	$65,000	Memphis, TN
Sheela	Software Developer	$90,000	Atlanta, GA

- **Interference Countermeasures**
  - At Design time: Alter the database structure. For example: split the table into multiple tables with some common attribute(s) among them and change the access control regime.
  - At Query time: Monitor and alert or reject the query; An inference detection algorithm is needed and is more difficult: on-going research.

Inference Countermeasures (implementation)

- Split the Employee Table into three tables: Employee Name Table, Salary Table and Employee Name-ID Table
- Set the regular user for permissions to access only the Employee Name and Salary Tables.
- Set the administrator to be the only one to have access to the Employee Name - ID Table.
- Store each table (sorted) according to a particular attribute



Employee Name - ID Table

Name	Employee ID
Andrew	004
Mary	003
Robert	001
Ryan	005
Sheela	009
Victor	010

Inference Countermeasures (implementation… continued…)

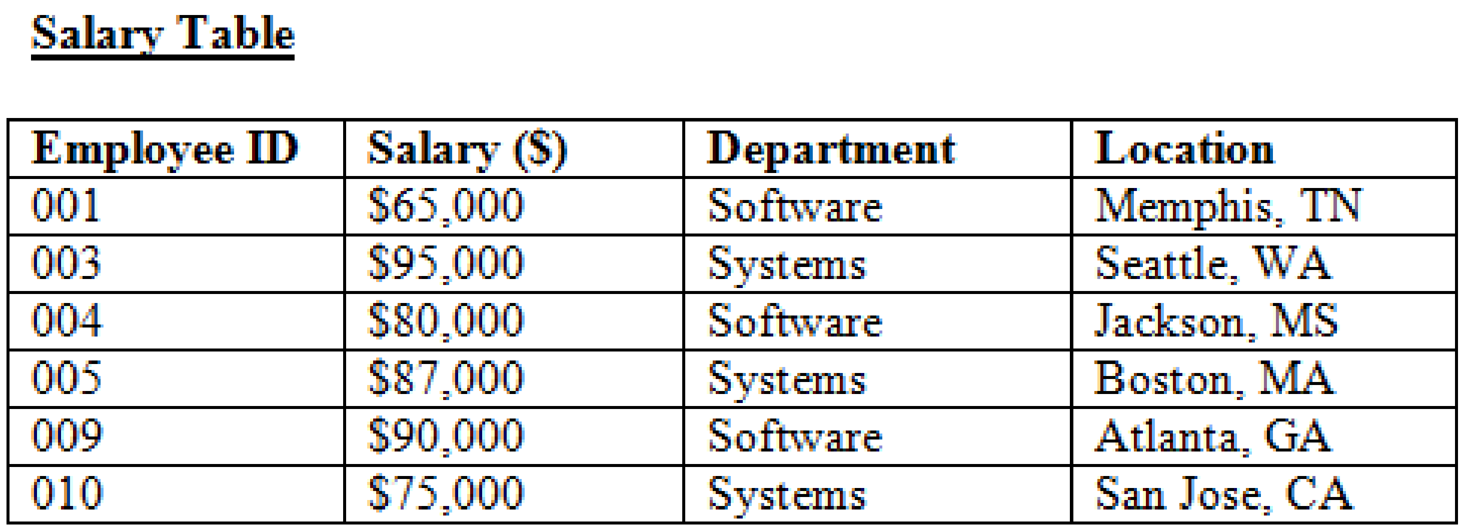
Employee Name Table

Name	Position	Department	Location
Andrew	Programmer Analyst	Software	Jackson, MS
Robert	Quality Control	Software	Memphis, TN
Sheela	Software Developer	Software	Atlanta, GA
Victor	Systems Engineer	Systems	San Jose, CA
Mary	Systems Administrator	Systems	Seattle, WA
Ryan	Network Engineer	Systems	Boston, MA

```
Create View V3 as
SELECT Name, Position
From Employee
where Department = 'Software'
```

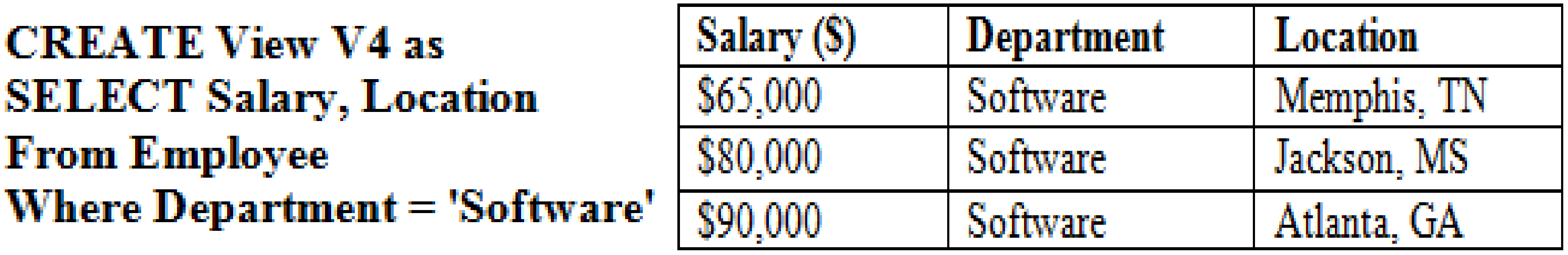
Name	Position
Andrew	Programmer Analyst
Robert	Quality Control
Sheela	Software Developer

Inference Countermeasures (implementation… continued…)



Salary Table

Employee ID	Salary ($)	Department	Location
001	$65,000	Software	Memphis, TN
003	$95,000	Systems	Seattle, WA
004	$80,000	Software	Jackson, MS
005	$87,000	Systems	Boston, MA
009	$90,000	Software	Atlanta, GA
010	$75,000	Systems	San Jose, CA



CREATE View V4 as
SELECT Salary, Location
From Employee
Where Department = 'Software'

Salary ($)	Department	Location
$65,000	Software	Memphis, TN
$80,000	Software	Jackson, MS
$90,000	Software	Atlanta, GA

Can Inference work?

Inferred Information

Name	Position	Salary ($)	Department	Location
Andrew	Programmer Analyst	$65,000	Software	Memphis, TN
Robert	Quality Control	$80,000	Software	Jackson, MS
Sheela	Software Developer	$90,000	Software	Atlanta, GA

Original Employee Table

Name	Position	Salary ($)	Department	Location
Andrew	Programmer Analyst	$80,000	Software	Jackson, MS
Robert	Quality Control	$65,000	Software	Memphis, TN
Sheela	Software Developer	$90,000	Software	Atlanta, GA
Victor	Systems Engineer	$75,000	Systems	San Jose, CA
Mary	Systems Administrator	$95,000	Systems	Seattle, WA
Ryan	Network Engineer	$87,000	Systems	Boston, MA

Statistical Database (SDB)

- SDB provides data of a statistical nature such as counts and averages
- Two types:
  - Pure statistical database:
    - Only stores statistical data (like census database)
  - Ordinary database with statistical access:
    - Contains individual entries
    - Access using DAC, MAC and RBAC models
    - Permit statistical queries based on the underlying raw data.
- The access control objective of an SDB is to provide users with the aggregate information without compromising the confidentiality of any individual entity present in the database.
  - The security problem is “inference” through one or a series of statistical queries.

SDB: Characteristic Formula

- Statistics are derived from a database by means of a logical Boolean formula (referred as Characteristic formula) over the values of attributes.

- A Characteristic formula uses the operators OR, AND, and NOT (+, *, ~), written here in the increasing order of priority.
  - E.g., (*Sex* = Male) * ((Major = CS) + (Major = EE)) specifies all male students majoring in either CS or EE
  - For numerical attributes, relational operators may be used. E.g., (GP > 3.7) specifies all students whose grade point average is above 3.7.

- For simplicity, we may omit the attribute names if they are clear from context. E.g., Male * (CS + EE)

SDB Example

Database with Statistical Access with 13 Students

Name	Sex	Major	Class	SAT	GPA
Allen	Female	CS	1980	600	3.4
Baker	Female	EE	1980	520	2.5
Cook	Male	EE	1978	630	3.5
Davis	Female	CS	1978	800	4.0
Evans	Male	Bio	1979	500	2.2
Frank	Male	EE	1981	580	3.0
Good	Male	CS	1978	700	3.8
Hall	Female	IT	1979	580	2.8
Iles	Male	CS	1981	600	3.2
Jones	Female	Bio	1979	750	3.8
Kline	Female	IT	1981	500	2.5
Lane	Male	EE	1978	600	3.0
Moore	Male	CS	1979	650	3.5

Attribute A_j	Possible Values	$\|A_j\|$
Sex	Male, Female	2
Major	Bio, CS, EE, IT,...	30
Class	1978, 1979, 1980, 1981	4
SAT	310, 320, 330, ..., 790, 800	50
GPA	0.0, 0.1, 0.2, ..., 3.9, 4.0	41

Source: Table 5.3:
W. Stallings: Computer Security: Principles and Practice, 2nd ed.

SDB: Characteristic Formula

- The **query set** of characteristic formula *C*, denoted as X(C), is the set of records matching that characteristic.
- For example, for C = Male * EE, X(C) = 3, matching the records of Cook, Frank and Lane (all Male and EE majors).
- A statistical query is a query that produces a value calculated over a query set.
  - Examples: **count** (Female * CS) = 2; **sum** (Female * CS, SAT) = 1400
- **Inference:** Assume a questioner knows that Baker is a female EE student; but, he does not know that she is the only one. The following sequence of two queries will reveal Baker's GPA.
  - **count** (EE * Female) = 1
  - **Sum** (EE * Female, GPA) = 2.5

Solutions for SDB Inference
Query Restriction
Perturbation

Query Restriction Techniques

- **<u>Idea:</u>**
  - If a query can lead to a compromise,
    - Reject the query
  - Else
    - Return accurate answers.
- <u>Technique # 1: Query Size Restriction</u>
- For any fixed integer $\boldsymbol{k > 1}$, a query q(C) on a Characteristic formula C is allowed if and only if the returned set of records X(C) satisfies:
  - $\boldsymbol{k \leq |X(C)| \leq N - k}$.
- This way, queries that can lead to revealing data based on an individual record can be denied.
- **Why is there a need for the Upper Bound?**
- **<u>Proof</u>**
- The upper bound is needed to avoid someone from inferring using a query of type q(*All*) that would return statistics based on the entire database.
  - Instead of directly querying using C to collect statistics on an individual record, one could collect statistics by computing

    q(*All*) – q(~C)

Query Size Restriction

- **Proof for Need of Upper Bound (continued…)**
  - Assume, we have only the lower bound (say, $k = 2$) for a database of $N > 2$ users,
  - We have a query q(C) that would return only one record. Hence, q(~C) would return N-1 records.
  - $| X(All) | = N \geq 2$ (checking for lower bound)
  - $| X(\sim C) | = N-1 \geq 2$. (checking for lower bound).
  - Hence, both queries q(All) and q(~C) would be evaluated and their results returned to the user.
  - However, $| X(All) | - | X(\sim C) | = 1 = | X(C) |$.
  - Hence results of q(*All*) – q(~C) can be considered the statistics for the individual record that will returned if q(C) is directly evaluated.
  - Hence, we need the upper bound (N-*k*)*;* in the above case, $| X(\sim C) | = N - 1 \nleq N - 2$ (check for upper bound fails).

- Note that considering the potential benefits in collecting statistics on the entire database, we do allow q(*All*).

Tracker : Query Size Restriction

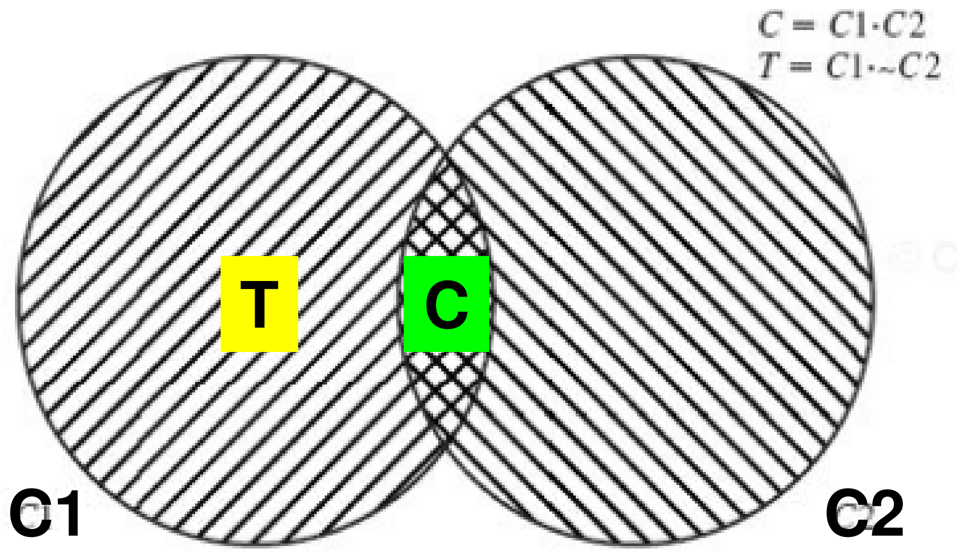
- The questioner can divide his/her knowledge of an individual into parts, such that queries can be made on the parts without violating the query size restriction.
- The combination of parts is called a *tracker*, because it can be used to track down the characteristics of an individual.
- Let there be a characteristic formula C *D that corresponds to zero or one record, so that the query **count** (C*D) is not permitted.
- But, suppose that the formula C can be decomposed into two parts C = C1 * C2, such that the query sets for both C1 and T = (C1 * ~C2) satisfy the query size restriction.

If it is not known whether or not individual *I* is uniquely identified by C, then we can find **count (C)** as shown below and check if it is equal to 1.

Count (C) = Count (C1) – Count (T)

One can then run the query as

q(C*D) = q(C1*D) – q((C1*~C2) * D)



Tracker: Example

- Consider the table below. Suppose that we want to know whether Evans scored 600 or above in SAT.
- Assume that we know that Evans is a Bio major of class 1979.
- If we directly launch the query based on the formula C = Male * Bio * 1979 on the database table, the count (C) = 1 and the results of the query will not be returned.

Name	Sex	Major	Class	SAT	GPA
Allen	Female	CS	1980	600	3.4
Baker	Female	EE	1980	520	2.5
Cook	Male	EE	1978	630	3.5
Davis	Female	CS	1978	800	4.0
Evans	Male	Bio	1979	500	2.2
Frank	Male	EE	1981	580	3.0
Good	Male	CS	1978	700	3.8
Hall	Female	IT	1979	580	2.8
Iles	Male	CS	1981	600	3.2
Jones	Female	Bio	1979	750	3.8
Kline	Female	IT	1981	500	2.5
Lane	Male	EE	1978	600	3.0
Moore	Male	CS	1979	650	3.5

Tracker: Example (continued…)

- Consider the table below. Suppose that we want to know whether Evans scored 600 or above in SAT.
- Assume that we know that Evans is a Bio major of class 1979.
- If we directly launch the query based on the formula C = Male * Bio * 1979 on the database table, the count (C) = 1 and the results of the query will not be returned.
- Suppose, we break C = C1 * C2; where C1 = Male; and C2 = Bio * 1979.
- ➔ T = C1 * ~C2 = Male * ~(Bio * 1979).
- Count(C) = Count(C1) – Count(T)
- Count(C1) = Count(Male) = 7
- Count(T) = Count(Male * ~ (Bio * 1979)) = 6
- Hence, Count(C) = Count(Male * Bio * 1979) = 7 – 6 = 1
- Thus, one can now be confirmed that Evans is the only Male, Bio major of class 1979.

Tracker: Example (continued…)

- Let D = SAT ≥ 600
- Since Count(C) = Count(C1) – Count(C1*~C2) = 1
- We can execute the query q(C*D) to find:
  - Count(C*D) = Count(C1*D) – Count(C1*~C2*D)
  - C1 = Male
  - C2 = Bio * 1979
  - Count (C1*D) = Count (Male * SAT ≥ 600) = 5
  - Count (C1*~C2*D) = Count (Male* ~(Bio*1979) * SAT ≥ 600) = 5
  - Hence, Count(C*D) = 5 – 5 = 0. Thus, we can conclude that Evans **did not score 600** or above in SAT.

Query Set Overlap Control

- Note that the idea behind tracker is to make use of queries wherein there is a considerable overlap in the query sets.

- In the previous example, we had
- Count (Male * Bio * 1979) = Count (Male) – Count (Male * ~ (Bio * 1979))
- That is, we found the number of males and subtracted from it the number of males who are ***not*** Bio majors of class 1979.

- There was considerable overlap between the above two characteristic formulae, resulting in the difference in their counts to be 1, leading to the inference that Evans is the only Bio major of class 1979.

Query Set Overlap Control

- The idea behind Query Set Overlap control is to return the results of a query for a user only if the number of common records (i.e., extent of overlap) with the results of the previous query of the same user is within a threshold *k*.
- A query q(C) is permitted only if the number of records that match C satisfies $| \mathbf{X(C)} \cap \mathbf{X(D)} | \leq \boldsymbol{r}$ for all q(D) that have been answered for this user, and where *r* is a fixed integer greater than 0.
- Problems with Query Set Overlap Control
  - Ineffective to prevent the cooperation of several users to compromise and infer from the database
  - Statistics for both a set and its subset (e.g., all patients and all patients undergoing a given treatment) cannot be released, thus limiting the usefulness of the database
  - For each user, a user profile has to be kept up to date.

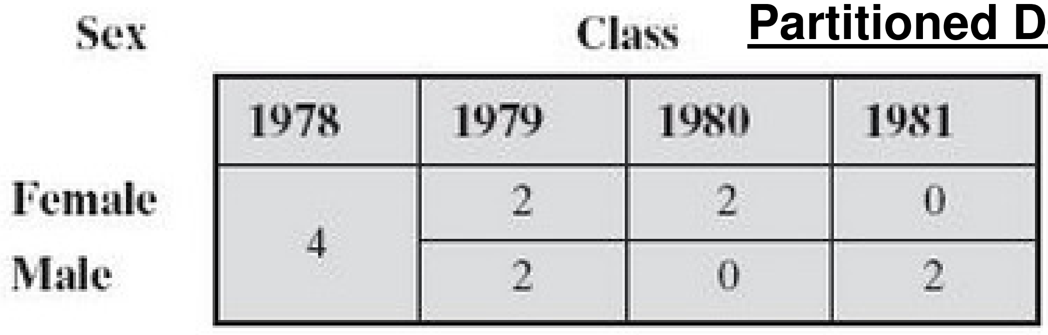
Partitioning

- Partitioning can be viewed as “Query Set Overlap Control at its logical extreme:” **Not allowing overlapping queries at all.**
- The idea is to cluster the records in a database into a number of mutually exclusive groups.
  - The user may only query the statistical properties of each group as a whole.
  - The user may not select a subset of a group.
- **Restrictions:**
  - A group of a single record is forbidden (because a user can find out details of the record): More design effort is needed.
  - It is possible for a user to gain information about a record by collecting statistics about the record before and after insertion or deletion.
    - Due to this restriction, **insertion or deletion of records can occur only in pairs**; and **each group has to have 0 or an even number of records**.

Side Effects of Partitioning

Certain records (in the original database) have to be even omitted to maintain 0 or even number of records for each group.

Name	Sex	Major	Class	SAT	GPA
Allen	Female	CS	1980	600	3.4
Baker	Female	EE	1980	520	2.5
Cook	Male	EE	1978	630	3.5
Davis	Female	CS	1978	800	4.0
Evans	Male	Bio	1979	500	2.2
Frank	Male	EE	1981	580	3.0
Good	Male	CS	1978	700	3.8
Hall	Female	IT	1979	580	2.8
Iles	Male	CS	1981	600	3.2
Jones	Female	Bio	1979	750	3.8
Kline	Female	IT	1981	500	2.5
Lane	Male	EE	1978	600	3.0
Moore	Male	CS	1979	650	3.5



Sex \ Class	1978	1979	1980	1981
Female	4	2	2	0
Male		2	0	2

Partitioned Database

Note that the record for **Kline** has to be omitted; because she is the only female in the class of 1981.

Source: Table 5.5: W. Stallings: Computer Security: Principles and Practice, 2nd ed.

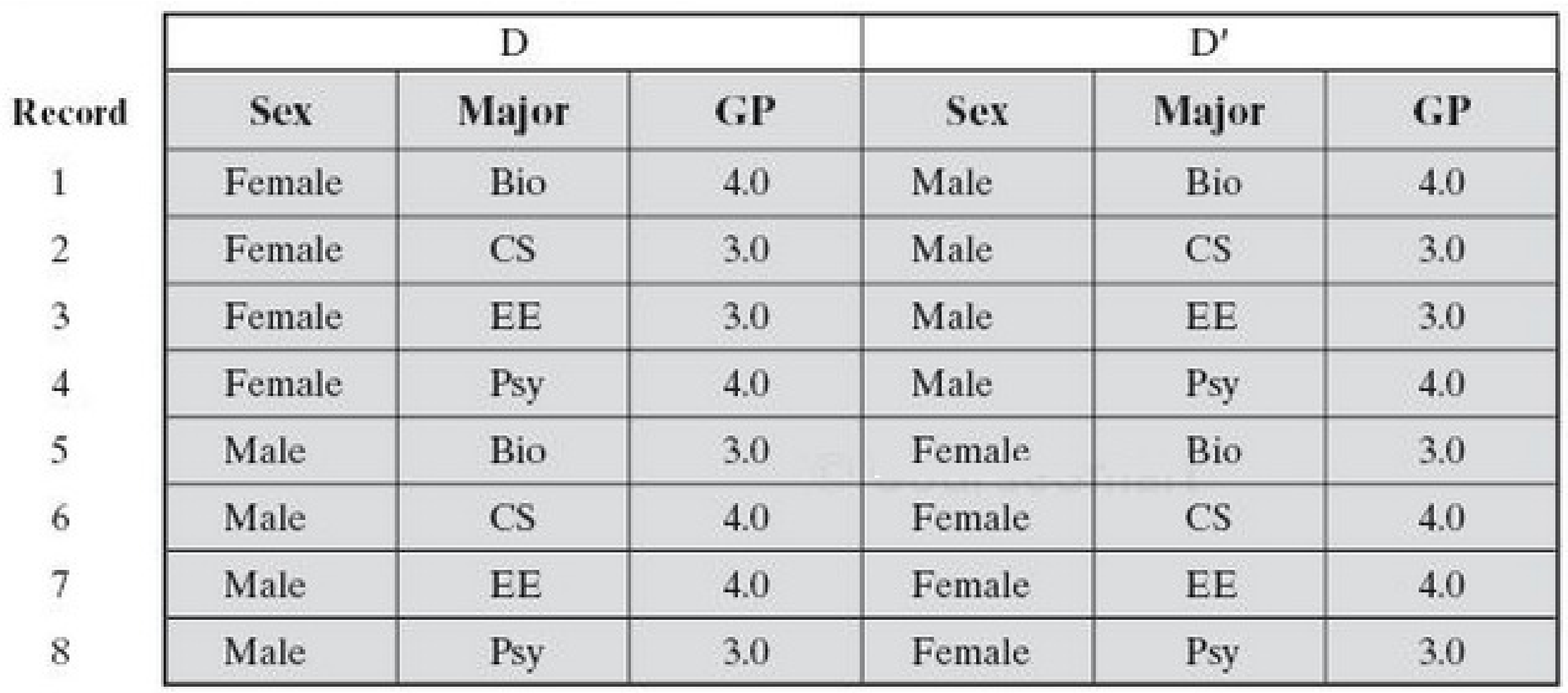
Query Denial: Information Leakage

- With query restriction techniques (that either deny or return an exact answer), the denial of a query may provide sufficient clues that an attacker can deduce underlying information.
- <u>Example:</u> Assume a database has real-valued entries and that a query is denied only if it would enable the requestor to deduce a value.
  - Suppose the requestor poses the query for sum(x1, x2, x3) and the response is 15.
  - Then, the requestor queries max(x1, x2, x3) and the query is denied. What can the requestor infer from this?
    - If max(x1, x2, x3) < 5, then the sum(x1, x2, x3) has to be < 15.
    - If max(x1, x2, x3) > 5, then revealing the maximum value would not lead to the inference of any individual value.
    - Hence, the query is denied only if max(x1, x2, x3) = 5. This implies that x1 = x2 = x3 = 5.
- **Solution to prevent information leakage (more conservative):**
  - In the above example, the max query was denied only when x1 = x2 = x3 = 5.
  - The idea behind the solution is to **deny queries based on the history of the queries posed by the user** (i.e., a max query following a sum query or vice-versa) and not based on the actual values in the database.

Perturbation

- The idea is to add noise to the statistics generated from the original data.
- **Data Perturbation:** The data in the SDB can be modified (perturbed) so as to produce statistics that cannot be used to infer values for individual records
- **Output Perturbation:** When a statistical query is made, the system can generate statistics that are modified from those that the original database would provide.
- The goal is to minimize the differences (between the perturbed results and ordinary results) and to provide users with consistent/usable results; but nevertheless, the questioner of the query should not be able to infer any sensitive information.
  - The main challenge is to determine the average size of the error to be used.

Data Perturbation: Data Swapping



	D			D'		
Record	Sex	Major	GP	Sex	Major	GP
1	Female	Bio	4.0	Male	Bio	4.0
2	Female	CS	3.0	Male	CS	3.0
3	Female	EE	3.0	Male	EE	3.0
4	Female	Psy	4.0	Male	Psy	4.0
5	Male	Bio	3.0	Female	Bio	3.0
6	Male	CS	4.0	Female	CS	4.0
7	Male	EE	4.0	Female	EE	4.0
8	Male	Psy	3.0	Female	Psy	3.0

The transformed statistics D’ has the same statistics as that of D for one or two attributes. However, three-attribute statistics are not preserved.
Example: Count (EE * Male * 4.0) = 1 in D and it is 0 in D’. (statistics not preserved)
Average GPA of Biology majors (statistics is preserved).

Data Perturbation: Modify Data based on Underlying Probability Distribution of Attribute Values

- For each confidential or sensitive attribute, determine the probability distribution function that best matches the data and estimate the parameters of the distribution function.
- Generate a sample series of data from the estimated density function for each sensitive attribute.
- Substitute the generated data of the confidential attribute for the original data in the same rank order.
  - That is the smallest value of the new sample should replace the smallest value in the original data, and so on.

Output Perturbation

- Random-Sample Query
  - A user issues a query q(C) to return a query set (results of a query) X(C).
  - The system replaces X(C) with a sampled query set, which is a properly selected subset of X(C).
  - The system calculates the requested statistic on the sampled query set and returns the value.
- Another approach:
  - Calculate the statistic on the requested query set and then adjust the answer up or down by a given amount in some systematic or randomized fashion.

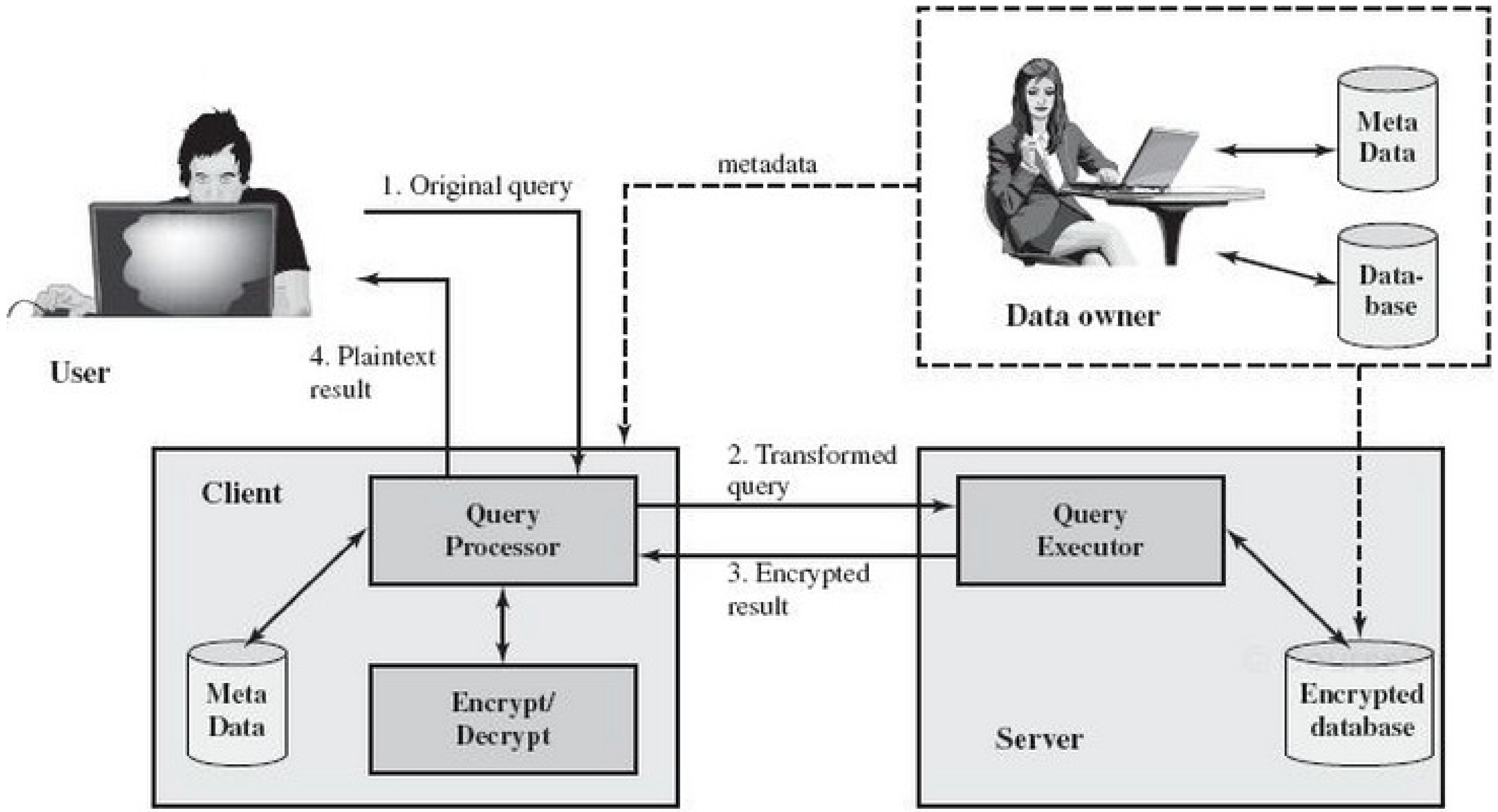
Limitations of Perturbation Techniques

- It is difficult to add sufficient perturbation to hide data without badly distorting the results.
- As the size of the database grows, the effectiveness of the perturbation techniques increases.
- For a database of *n* records, if the number of queries is linear to the size of the database, then the amount of perturbation needed is of the order of sqrt(n).

Database Encryption

- Encryption can be applied to the entire database, at the record level (encrypt selected records), at the attribute level (encrypt selected attributes), or at the level of the individual field.
- Two disadvantages to database encryption:
  - Key management: Distributing keys to authorized users to access selected parts of the database.
  - Inflexibility: Searching an encrypted database becomes difficult.
- To be more flexible, it must be possible to work with the database in its encrypted form.

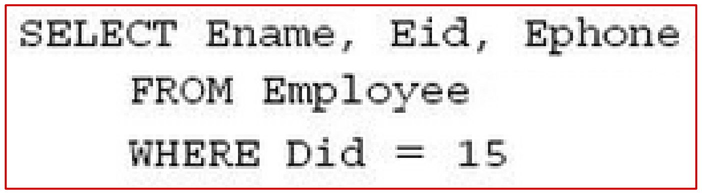
A Database Encryption Scheme



Source: Figure 5.10: W. Stallings: Computer Security: Principles and Practice, 2nd ed.

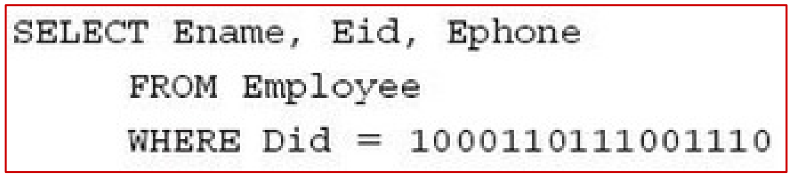
A Simple DB Encryption Scheme

- Steps:
  - The user issues a query for fields from one or more records with a specific value of the primary key.
  - The query processor (stores the query); encrypts the primary key and sends it to the server.
  - The server obtains encrypted records from the Data Owner using the encrypted primary key as the index.
  - The retrieved records (in their encrypted form) are sent to the query processor, which decrypts them using the secret key shared between the query processor and the data owner.
  - The values for the decrypted fields of interest are sent to the user, who is unaware of the database encryption.



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

Unencrypted query



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

Query with the value for the primary key encrypted

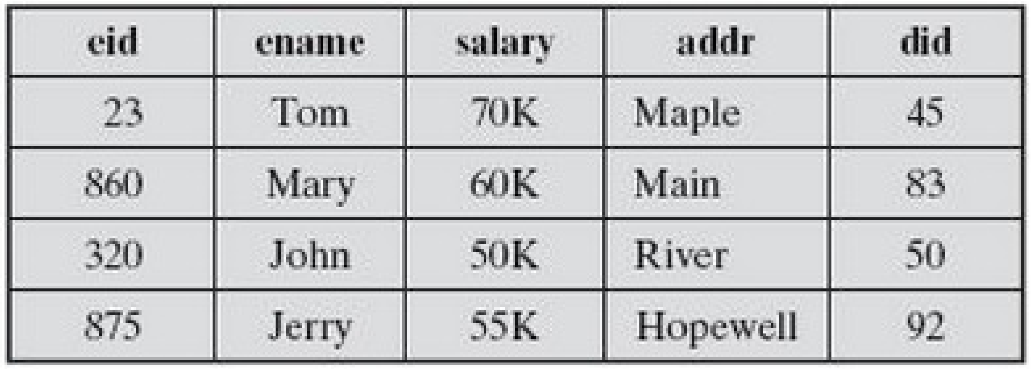
Secret key for encryption is stored as part of metadata.

Cannot search queries based on ranges of values for attributes (Salary > 500)

Binary Encryption Scheme

- Treat each record as a contiguous block of bits (when the attribute values are concatenated together).
- For some or all of the attributes, an index value is created.
- For each row Bi of the unencrypted database, the mapping is: $\boxed{(x_{i1}, x_{i2}, \ldots, x_{iM}) \rightarrow [\mathrm{E}(k, B_i), I_{i1}, I_{i2}, \ldots, I_{iM}]}$
- For the numeric attributes, introduce range of values and associate them with indexes. For example: [1-200] – 1; [201-400] – 2; etc. The range of the first alphabet in text fields (a-c: 1; d-f: 2, etc) could be assigned unique index values. These are the metadata stored at the query processor and the data owner (not stored at the server).
- The encrypted database stores the entire records in their encrypted form, as well as the index values of their attribute columns.

Binary Encryption Scheme



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

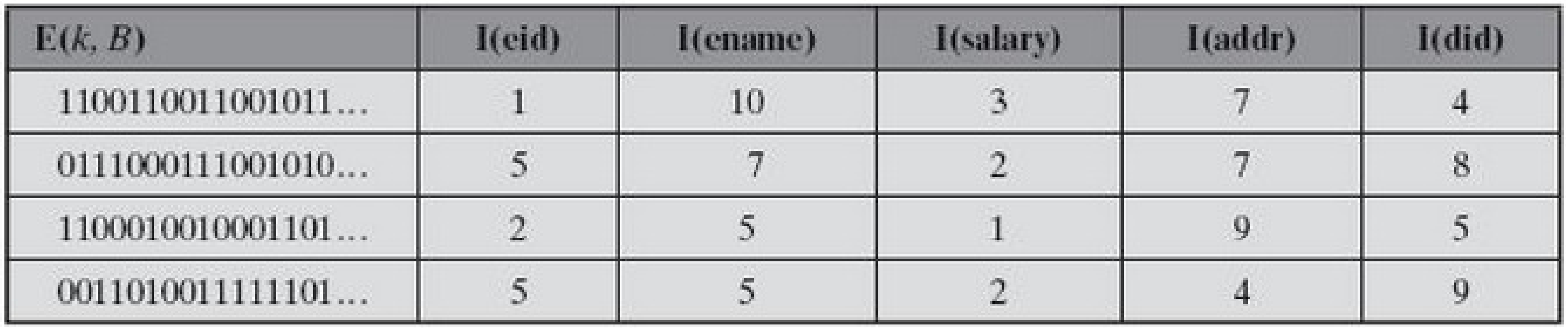
Employee Table

Suppose a user queries for all records with eid < 300.

The query processor request 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.

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

Source: Table 5.7: W. Stallings: Computer Security: Principles and Practice, 2nd ed.