Database Security and Privacy
Security Objectives

Secrecy
- Prevent/detect/deter improper Disclosure of information
- Prevent/detect/deter improper modification of information

Integrity
- Prevent/detect/deter improper Denial of access to services

Availability
- Prevent/detect/deter improper Denial of access to services
Databases

- Collection of
  - interrelated data and
  - set of programs to access the data
- **Convenient** and **efficient** processing of data
- Database Application Software
Database Security

- **Protect Sensitive Data from**
  - Unauthorized disclosure
  - Unauthorized modification
  - Denial of service attacks

- **Security Controls**
  - Security Policy
  - Access control models
  - **Integrity protection**
  - Privacy problems
  - **Fault tolerance and recovery**
  - **Auditing** and intrusion detection
Protection of Data Confidentiality

- **Access control** – which data users can access
- **Information flow control** – what users can do with the accessed data
- **Data Mining**
Access Control

- Ensures that all direct accesses to object are authorized
- Protects against accidental and malicious threats by regulating the read, write and execution of data and programs
Access Control

Requires:

- Proper user identification
- Information specifying the access rights is protected form modification
Access Control

- Access control components:
  - Access control policy: specifies the authorized accesses of a system
  - Access control mechanism: implements and enforces the policy
HOW TO SPECIFY ACCESS CONTROL?
Access Control

- **Subject**: active entity that requests access to an object
  - e.g., user or program
- **Object**: passive entity accessed by a subject
  - e.g., record, relation, file
- **Access right** (privileges): how a subject is allowed to access an object
  - e.g., subject $s$ can read object $o$
Protection Object

- Database
- Relation
- Record
- Attribute
- Element

Advantages vs. disadvantages of supporting different granularity levels
**Relation-Level Granularity**

Confidential relation

<table>
<thead>
<tr>
<th>Person-name</th>
<th>Company-name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>BB&amp;C</td>
<td>$43,982</td>
</tr>
<tr>
<td>Dell</td>
<td>Bell</td>
<td>$97,900</td>
</tr>
<tr>
<td>Black</td>
<td>BB&amp;C</td>
<td>$35,652</td>
</tr>
</tbody>
</table>
## Tuple-level Granularity

### Works

<table>
<thead>
<tr>
<th>Person-name</th>
<th>Company-name</th>
<th>Salary</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>BB&amp;C</td>
<td>$43,982</td>
<td>Public</td>
</tr>
<tr>
<td>Dell</td>
<td>Bell</td>
<td>$97,900</td>
<td>Conf.</td>
</tr>
<tr>
<td>Black</td>
<td>BB&amp;C</td>
<td>$35,652</td>
<td>Public</td>
</tr>
</tbody>
</table>
# Attribute-Level Granularity

## Works

<table>
<thead>
<tr>
<th>Person-name</th>
<th>Company-name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>BB&amp;C</td>
<td>$43,982</td>
</tr>
<tr>
<td>Dell</td>
<td>Bell</td>
<td>$97,900</td>
</tr>
<tr>
<td>Black</td>
<td>BB&amp;C</td>
<td>$35,652</td>
</tr>
</tbody>
</table>
## Cell-Level Granularity

### Works

<table>
<thead>
<tr>
<th>Person-name</th>
<th>Company-name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smith</td>
<td>BB&amp;C</td>
<td>$43,982</td>
</tr>
<tr>
<td>Dell</td>
<td>Bell</td>
<td>$97,900</td>
</tr>
<tr>
<td>Black</td>
<td>BB&amp;C</td>
<td>$35,652</td>
</tr>
</tbody>
</table>
Access Control Policies

- Discretionary Access Control (DAC)
- Mandatory Access Control (MAC)
- Role-Based Access Control (RBAC)
Discretionary Access Control (DAC)

- For each subject access right to the objects are defined
  - (subject, object, +/- access mode)
  - (Black, Employee-relation, read)
- User based
- Grant and Revoke
- Problems:
  - Propagation of access rights
  - Revocation of propagated access rights
19 Database Security - Sunnie Chung

DAC by Grant and Revoke

grant select on Employee to Black with grant option

Brown (owner)

GRANT UPDATE(Salary) ON Employee TO White

Brown revokes grant given to Black

Black

Brown does not want Red to access the Employee relation

Red

GRANT SELECT ON Employee TO Red

White
Implementation

Access Control List (column)

<table>
<thead>
<tr>
<th>Subject</th>
<th>Access</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joe</td>
<td>Read</td>
<td>File 1</td>
</tr>
<tr>
<td>Joe</td>
<td>Write</td>
<td>File 1</td>
</tr>
<tr>
<td>Joe</td>
<td>Own</td>
<td>File 1</td>
</tr>
<tr>
<td>Joe</td>
<td>Read</td>
<td>File 2</td>
</tr>
<tr>
<td>Sam</td>
<td>Read</td>
<td>File 2</td>
</tr>
<tr>
<td>Sam</td>
<td>Write</td>
<td>File 2</td>
</tr>
<tr>
<td>Sam</td>
<td>Own</td>
<td>File 2</td>
</tr>
</tbody>
</table>

Capability List (row)

Joe: File 1/Read, File 1/Write, File 1/Own, File 2/Read
Sam: File 2/Read, File 2/Write, File 2/Own

Access Control Triples
Access Control Mechanisms

- Security through Views
- Stored Procedures
- Grant and Revoke
- Query modification
Views

Students (sid, name, address, gpa)
Completed ( sid, course, grade)

- A view is a query stored in the database
  - Think of it as a table definition for future use
- Example view definition:
  
  CREATE VIEW GS\textit{tudents} AS
  
  SELECT *
  
  FROM Students
  
  WHERE gpa >= 2.5
- Views can be used like \textit{base tables}, in any query or in any other view. Like a Macro. Different from Insert INTO.
Views for Security

- This is the student table without the gpa field to hide gpa to any user in DB.

CREATE VIEW Sstudents AS
SELECT sid, name, address
FROM students
Security Through Views

- Assign rights to access predefined views

  ```sql
  CREATE VIEW Outstanding-Student
  AS SELECT NAME, COURSE, GRADE
  FROM Student
  WHERE GRADE > B
  ```

Problem:
Difficult to maintain updates.
**Stored Procedures**

- Assign rights to execute compiled programs
- `GRANT RUN ON <program> TO <user>`

**Problem:**
Programs may access resources for which the user who runs the program does not have permission.
SQL GRANT is a command used to provide access or privileges on the database objects to the users.

The Syntax for the GRANT command is:

```
GRANT privilege_name
ON object_name
TO {user_name | PUBLIC | role_name}
[WITH GRANT OPTION];
```
GRANT

privilege_name is the access right or privilege granted to the user. Some of the access rights are ALL, CREATE TABLE, DROP TABLE, EXECUTE, and SELECT, INSERT/DELETE/UPDATE.

object_name is the name of a database object like TABLE, VIEW, STORED PROC and SEQUENCE.

user_name is the name of the user to whom an access right is being granted.

PUBLIC is used to grant access rights to all users.

ROLES are a set of privileges grouped together.

WITH GRANT OPTION - allows a user to grant access rights to other users.

Example:
GRANT Select ON Employee TO PUBLIC WITH GRANT OPTION;
GRANT SELECT ON Person.Address TO [AdventureWorks2012\RosaQdM];
The `REVOKE` command removes user access rights or privileges to the database objects.

The Syntax:

```
REVOKE privilege_name
ON object_name
FROM {user_name | PUBLIC | role_name};
```

Example:

```
REVOKE Select ON Employee FROM Public;
```
Grant and Revoke

GRANT <privilege> ON <relation>  
To <user>  
[WITH GRANT OPTION]

- GRANT SELECT * ON Student TO Matthews
- GRANT SELECT *, UPDATE(GRADE) ON Student TO FARKAS
- GRANT SELECT(NAME) ON Student TO Brown

GRANT command applies to base relations as well as views
Grant and Revoke

REVOKE <privileges> [ON <relation>]
FROM <user>

- REVOKE SELECT* ON Student FROM Blue
- REVOKE UPDATE ON Student FROM Black
- REVOKE SELECT(NAME) ON Student FROM Brown
Non-cascading Revoke

A revokes D’s privileges
Cascading Revoke

A revokes D’s privileges
Positive and Negative Authorization

Problem:
Contradictory authorizations
- GRANT <privilege> ON X TO <user>
- DENY <privilege> ON X TO <user>
Negative Authorization

What should happen with the privilege given by D To F?
Query Modification

- GRANT SELECT(NAME) ON Student TO Blue WHERE COURSE="CSCE 590"

- Blue’s query:
  SELECT *
  FROM Student

- Modified query:
  SELECT NAME
  FROM Student
  WHERE COURSE="CSCE 590"
DAC Overview

Advantages:
- Intuitive
- Easy to implement

Disadvantages:
- Inherent vulnerability (look TH example)
- Maintenance of ACL or Capability lists
- Maintenance of Grant/Revoke
- Limited power of negative authorization
Mandatory Access Control (MAC)

- **Security label**
  - Top-Secret, Secret, Public

- **Objects**: security classification
  - File 1 is Secret, File 2 is Public

- **Subjects**: security clearances
  - Brown is cleared to Secret, Black is cleared to Public

- **Dominance (≥)**
  - Top-Secret ≥ Secret ≥ Public
MAC

- **Access rights**: defined by comparing the security classification of the requested objects with the security clearance of the subject
- If access control rules are satisfied, access is permitted
- Otherwise access is rejected
- **Granularity** of access rights!
MAC – Bell-LaPadula (BLP) Model

- **Single security property**: a subject $S$ is allowed a read access to an object $O$ only if $\text{label}(S)$ dominates $\text{label}(O)$
- **Star-property**: a subject $S$ is allowed a write access to an object $O$ only if $\text{label}(O)$ dominates $\text{label}(S)$

No direct flow of information from high security objects to low security objects!
Multilevel Security

- Multilevel security → users at different security level, see different versions of the database

- **Problem:** different versions need to be kept consistent and coherent *without downward signaling channel* (covert channel)
Multilevel Relation

- Schema $R(A_1,C_1,\ldots,A_n,C_n,T_c)$
  - $R$: relation name
  - $A_i$: attribute name
  - $C_i$: security classes
  - $T_c$: Tuple security classes

- Instantiation of relation: sets of tuples of the form $<a_1,c_1,\ldots,a_n,c_n,t_c>$
  - $a_i$: attribute value
  - $c_i$: attribute classification label
  - $t_c$: tuple classification label
## Multilevel Relation Example

<table>
<thead>
<tr>
<th>SSN</th>
<th>λ(SSN)</th>
<th>Course</th>
<th>λ(Course)</th>
<th>Grade</th>
<th>λ(Grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>111-22-3333</td>
<td>S</td>
<td>CSCE 786</td>
<td>S</td>
<td>A</td>
<td>TS</td>
</tr>
<tr>
<td>444-55-6666</td>
<td>S</td>
<td>CSCE 567</td>
<td>S</td>
<td>C</td>
<td>TS</td>
</tr>
</tbody>
</table>

Top-secret user sees all data

Secret user sees **Secret-View**:

<table>
<thead>
<tr>
<th>SSN</th>
<th>λ(SSN)</th>
<th>Course</th>
<th>λ(Course)</th>
<th>Grade</th>
<th>λ(Grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>111-22-3333</td>
<td>S</td>
<td>CSCE 786</td>
<td>S</td>
<td>null</td>
<td>S</td>
</tr>
<tr>
<td>444-55-6666</td>
<td>S</td>
<td>CSCE 567</td>
<td>S</td>
<td>null</td>
<td>S</td>
</tr>
</tbody>
</table>
Polyinstantiation

Secret user sees Secret-View:

<table>
<thead>
<tr>
<th>SSN</th>
<th>λ(SSN)</th>
<th>Course</th>
<th>λ(Course)</th>
<th>Grade</th>
<th>λ(Grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>111-22-3333</td>
<td>S</td>
<td>CSCE 786</td>
<td>S</td>
<td>null</td>
<td>S</td>
</tr>
<tr>
<td>444-55-6666</td>
<td>S</td>
<td>CSCE 567</td>
<td>S</td>
<td>null</td>
<td>S</td>
</tr>
</tbody>
</table>

- SSN is primary key
- Secret user wants to update Grade for 111-22-3333 from null (i.e., missing value) to F
  - *Allow update*: inconsistent database, at TS level two different tuples exist with the same primary key (see next slide)
  - *Not allow update*: downward signaling channel, update is because of the existence of a TS value
### Polyinstantiation

#### Top-Secret View:

<table>
<thead>
<tr>
<th>SSN</th>
<th>λ(SSN)</th>
<th>Course</th>
<th>λ(Course)</th>
<th>Grade</th>
<th>λ(Grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>111-22-3333</td>
<td>S</td>
<td>CSCE 786</td>
<td>S</td>
<td>A</td>
<td>TS</td>
</tr>
<tr>
<td>111-22-3333</td>
<td>S</td>
<td>CSCE 786</td>
<td>S</td>
<td>F</td>
<td>S</td>
</tr>
<tr>
<td>444-55-6666</td>
<td>S</td>
<td>CSCE 567</td>
<td>S</td>
<td>C</td>
<td>TS</td>
</tr>
</tbody>
</table>