
Database Security

CS461/ECE422

Information Assurance

Fall 2007

Overview

- Database Model
- Inherent database integrity and availability
- Access control models
- Indirectly deducing information

Reading Material

- Security In Computing, Chapter 6
- Griffiths and Wade, “An Authorization Mechanism for a Relational Database”
 - <http://seclab.cs.uiuc.edu/docs/CS463/DatabaseSecurity.html>

Motivation

- Databases are a common element in today's system architecture
- Hold important information
 - Target of attacks

Relational Model

- Information stored in relations or tables
 - Each row is a tuple of attributes
 - Manipulated by standard SQL language

-

Name	UID	College	GPA	Financial Aid
Alice	1232	Eng	4	0
Bob	3234	Eng	1.2	\$5,000.00
Carol	4565	Bus	3.8	0
Dave	8988	Edu	2.1	0
Ellen	3234	ACES	3.1	\$100.00
Alice	4534	LAS	2.9	\$10,000.00

Combining tables

- Can use Join to create single set of tuples from multiple tables.

Name	UID	Major	UID	Dorm
Alice	1234	ECE	1234	LAR
Bob	2345	NUC	2345	ISR
Carol	3456	BA	3456	FAR
Dave	4567	French	4567	PAR

Name	Dorm	Major
Alice	LAR	ECE
Bob	ISR	NUC
Carol	FAR	BA
Dave	PAR	French

Making Queries

- Can select rows to create subtables

- Select Name, UID, Financial Aid from Students where College = 'Eng'

Name	UID	College	GPA	Financial Aid
Alice	1232	Eng	4	0
Bob	3234	Eng	1.2	\$5,000.00
Carol	4565	Bus	3.8	0
Dave	8988	Edu	2.1	0
Ellen	3234	ACES	3.1	\$100.00
Alice	4534	LAS	2.9	\$10,000.00

Name	UID	Financial Aid
Alice	1232	0
Bob	3234	\$5,000.00

Database Advantages

- Years and years of technology improvements
 - Data integrity and consistency
 - Decent performance in face of integrity and consistency requirements
- Common well understood model
 - Shared access
 - Controlled access

Data Consistency

- Data is consistent if
 - It never changes OR
 - Only one person ever changes things at a time AND
 - The system never crashes during a change OR
 - All related changes can be made at once (atomic)
- Can loosen these restrictions with transactions and two-phase commits

ACID Transactions

- Atomic – All changes in the transaction have occurred or none of them occur
- Consistent – Transaction does not leave database in half-finished state
- Isolation – Other participants do not see transaction changes until transaction is completed
- Durability – Committed changes will survive system failure

Student Moving

- A space opens up in dorm (ISR)
 - Both Alice and Carol want to move there
 - Independently talk with clerks to make the move
- Actions
 - Remove student from old dorm list
 - Add to new dorm list
- Only one space, so only Alice or Carol can make the move
 - Don't want to leave either without a dorm

Student moving

Dorm	Room	Student 1	Student 2
LAR	10	Alice	Eve
FAR	13	Mary	Carol
ISR	123	Nancy	

Two Phase Commit

- First phase
 - Check the commit-flag (lock)
 - Gather information to perform the transaction
 - Store values persistently: shadow values or log
 - Turn on the commit flag (lock)
- Second phase
 - Perform the changes
 - Copy logged or shadow values to real values
 - Repeat until success
 - Turn off the commit flag (lock)

Students Moving

- Alice and Carol check commit-flag
 - Both create shadow versions of their old room row and the new room row
- Alice sets commit-flag first
 - Transaction checks that the target room is still available
 - Makes real copies same as her shadow
 - Clears commit-flag
- Carol fails to get commit flag

Access Control in the SQL Model

- Don't have to have a single owner of all data
 - Can create new table
 - Use “Grant” to give others privileges on table
- Can create views to have finer granularity with access control
- Can delegate privilege granting authority to others

SQL **grant** Syntax

```
grant privilege_list on resource  
  to user_list;
```

- Privileges include **select**, **insert**, *etc.*
- Resource may be a table, a database, a function, *etc.*
- User list may be individual users, or may be a user group

Example Application

- Alice owns a database table of company employees:

name **varchar** (50) ,

ssn **int**,

salary **int**,

email **varchar** (50)

- Some information (ssn, salary) should be confidential, others can be viewed by any employee.

Simple Access Control Rules

- Suppose Bob needs access to the whole table (but doesn't need to make changes):

```
grant select on employee to bob;
```

- Suppose Carol is another employee, who should only access public information:

```
grant select (name, email) on employee to  
  carol;
```

- not implemented in PostgreSQL (see next two slides for how to work around this)
- not implemented for **select** in Oracle
- implemented in MySQL

Creating Views

- Careful with definitions!
 - A subset of the database to which a user has access, or:
 - A virtual table created as a “shortcut” query of other tables
- View syntax:

```
create view view_name as  
  query_definition;
```
- Querying views is nearly identical to querying regular tables

View-Based Access Control

- Alternative method to grant Carol access to name and email columns:

```
create view employee_public as  
  select name,email from employee;  
grant select on employee_public to  
  carol;
```

Row-Level Access Control

- Suppose we also allow employees to view their own ssn, salary:

```
create view employee_Carol as  
  select * from employee  
  where name='Carol';
```

```
grant select on employee_Carol to carol;
```

- And we allow them to update their e-mail addresses:

```
grant update(email) on employee_Carol to  
  carol;
```

– (Or create yet another new view...)

Delegating Policy Authority

```
grant privilege_list on resource to  
user_list with grant option;
```

- Allows other users to grant privileges, including “with grant option” privileges
- “Copy right” from Access Control lecture (slide 21)
- Can grant subset privileges too
 - Alice: **grant select on table1 to bob with grant option**;
 - Bob: **grant select(column1) on table1 to carol with grant option**;

SQL **revoke** Syntax

```
revoke privilege_list on resource  
  from user_list;
```

- What happens when a user is granted access from two different sources, and one is revoked?
- What happens when a “with grant option” privilege is revoked?

Disadvantages to SQL Model

- Too many views to create
 - Tedious for many users, each with their own view
 - View redefinitions that change the view schema require dropping the view, redefining, then reissuing privileges
 - Fine-grained policies each require their own view—and no obvious way to see that the views come from the same table
- Other techniques being developed but not yet widely deployed

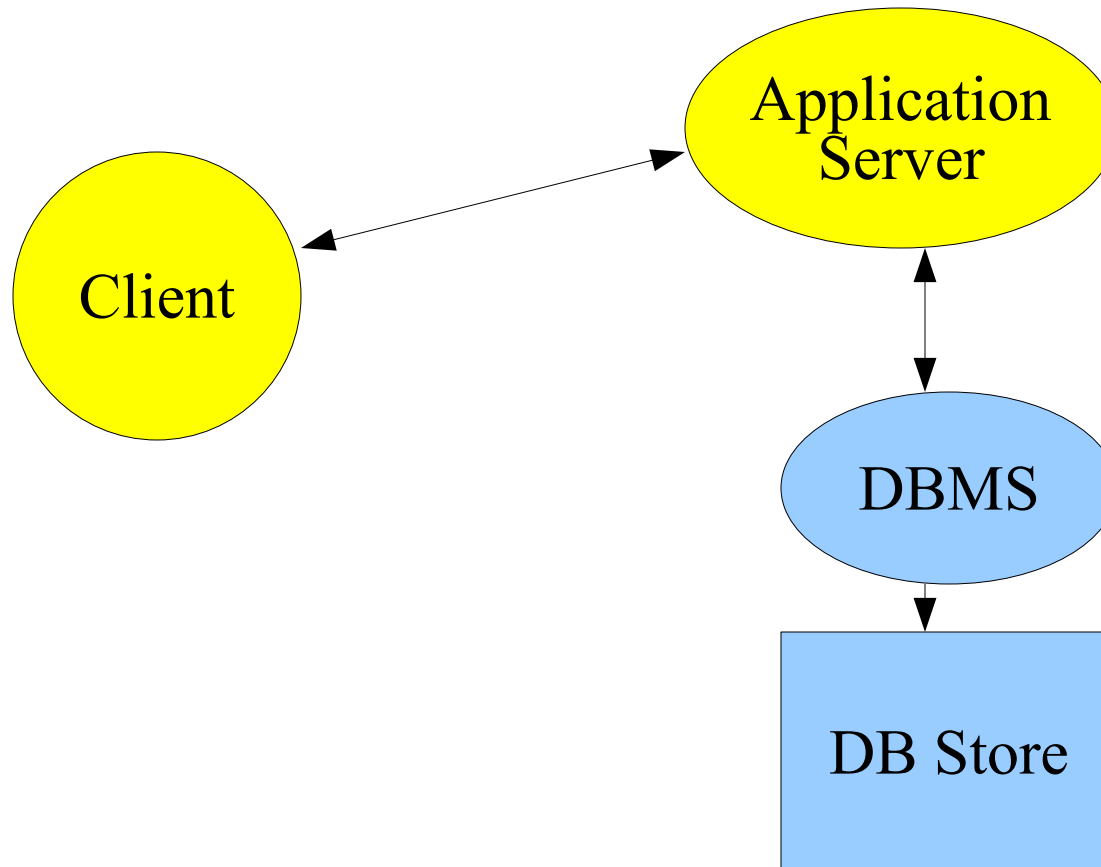
MLS Databases

- Developed during the 80's and 90's with MLS Operating systems
 - Label data and users with security levels and categories
 - Ideally use same labels as OS
- Limited availability today
 - Oracle has one, but labels are not shared with OS
 - Trusted Rubix <http://rubix.com>
 - Integrates with Trusted Solaris 8

MLS Issues

- Many of the same issues of MLS OS
- Granularity?
 - Per table, row, or value
- Need for trusted entities
 - Work outside MLS restrictions for maintenance
- Prevent knowledge of sensitive information in shared name space
 - Polyinstantiation
- Where do you store the labels?

Access Control in System Design



Types of Information Disclosure

- Exact data
 - Show the student's GPA attribute
- Negative result
 - See aggregate count for financial aid is non-zero means someone is getting financial aid
- Bounded results
 - Knowledge of high and low values
- Probable value
 - Know that 100 people in Eng live at ISR and 40 of them receive financial aid

Inference

- Use non-sensitive data to deduce sensitive data
 - Obscure queries
 - Combine statistical results
 - Use outside knowledge

Inference Direct Attack

- One approach: Obscure query
 - Determine who has self reported drug use
 - `list NAME where SEX='M' ^ DRUGS = '1'`
 - `list NAME where`
 - `(SEX='M' ^ DRUGS='1') v`
 - `(SEX!='M' ^ SEX!='F') v`
 - `(DORM = 'BOGUS')`
- Hard for program to determine that additional clauses are bogus
- Could use access control to just not give access to sensitive attributes to inappropriate entities

Indirect Inference Attack

- Work with statistical results
 - Sum, count, mean, median
- If you can get a statistical result over a small sample set, your task is easy
 - No students in the Business College are receiving financial aid, therefore Carol is not receiving financial aid
 - In general, systems will refuse to provide statistics if set is “too small”
 - Less than $k\%$ of n items

Combining results

- Compute the president's salary in three steps
 - Mean of all employee's salaries
 - Mean of all employee's salaries except the president
 - Find the number of employees

Tracker Attack

- System refuses to answer
 - $\text{count}((\text{SEX}=\text{F}) \wedge (\text{RACE}=\text{C}) \wedge (\text{DORM}=\text{ISR}))$
 - $\text{count}(A \wedge B \wedge C)$
 - Result based on only one row
- Break into multiple queries
 - $\text{count}(A) - \text{count}(A \wedge \neg(B \vee C))$
- Can generalize this approach to solve for linear systems of queries

Possible controls

- Concealing
 - Introduce slight random perturbations to the results
 - Trading precision for security
- Suppress multiple results
 - Not just results that are too small, but some additional results
 - Thwart tracker attacks
- Compute results on random subset
- Track history of individuals queries

Key Points

- Database technology has inherently developed good integrity mechanisms
- Access control models are available but not perfect
- Think very carefully about what information you reveal
 - Even seemingly non-sensitive data can be combined to infer sensitive information