

IT 4823

Information Security Administration

Database Security – Cloud Security



Notice: This session is
being recorded.

What is a Database?

Database:

- An organized,
- self-defining collection of
- logically-related data.

Metadata:

The descriptive data that makes the database self-defining. The metadata is stored in the database.

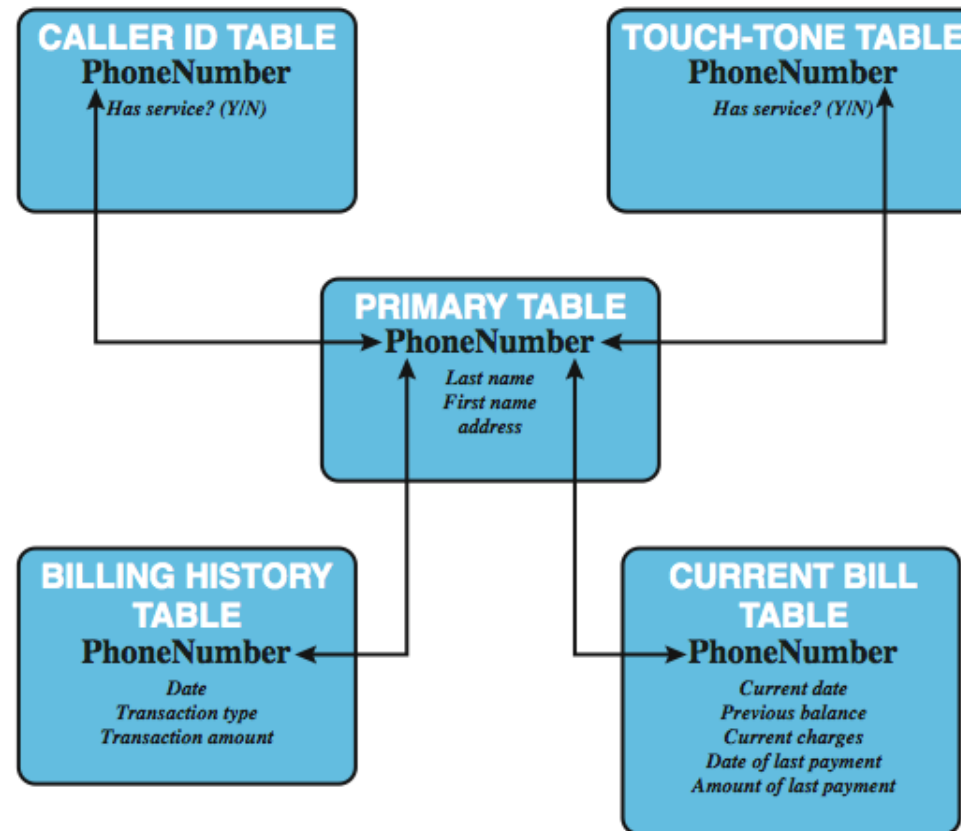
About the Relational Model

- A collection of tables, called “relations”
- With columns defined by the database designer called “attributes” or sometimes “fields”
- A subset of the columns in a relation is also a relation.
- An arbitrary number of rows sometimes called “entity instances” (“records” in the textbook)
- Each row has a unique “primary key”

Relational Database Terminology

- relation / table / file
- tuple / row / record
- attribute / column / field
- primary key
 - uniquely identifies a row
- foreign key
 - links one table to attributes in another
- view / virtual table

Relational Database Example



Relational Database Elements

Department Table			Employee Table				
Did	Dname	Dacctno	Ename	Did	SalaryCode	Eid	Ephone
4	human resources	528221	Robin	15	23	2345	6127092485
8	education	202035	Neil	13	12	5088	6127092246
9	accounts	709257	Jasmine	4	26	7712	6127099348
13	public relations	755827	Cody	15	22	9664	6127093148
15	services	223945	Holly	8	23	3054	6127092729
primary key			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

Constraints

- The relational data model uses data to describe connections among tables.
- The model includes the idea of constraints:
 - Range and domain of attributes
 - Constraints on the relationships among tables
- The database management system itself enforces the pre-defined constraints.

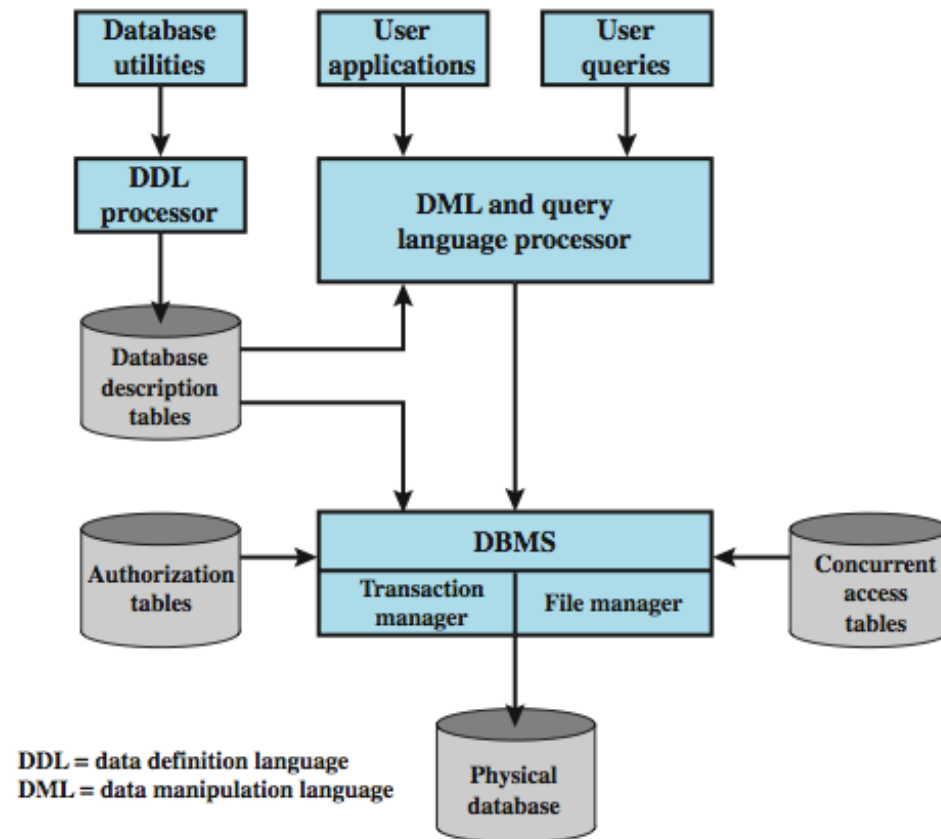
Advantages of Databases

- Shared access
- Minimal redundancy
 - From sharing
 - From normalization
- Consistent view (data consistency)
- Data integrity
 - Access mediated by DBMS
 - Constraints
- Controlled access

Database Access Control

- DBMS provide access control for database
- assume have authenticated user
- 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

Database Management System



The ACID Property

Remember *ACID*

Atomic	All or nothing
Consistent	Always obeys constraints
Isolated	Transactions are serialized
Durable	Transactions are not lost

Example Problem: The Joint Checking Account

Ted and Alice have a joint account;
The starting balance is \$200
Ted is withdrawing \$50 at a branch bank
Alice is depositing \$1,000 at a different branch

Ted's teller queries account, asks for ID
Alice's teller queries account, adds money
Alice's teller updates account
Ted's teller updates account

What is the balance? What should it be?

Solution: Transactions

Transactions are ACID:

```
BEGIN TRANSACTION;  
SELECT ...; //The account balance  
balance -= 50.00;  
UPDATE...; // The balance  
COMMIT TRANSACTION;
```

Atomic
Isolated

The DBMS causes transactions to *appear* to have occurred one after another: serially.

A Transaction that Fails

Suppose the account didn't have \$50...

```
if ($balance < 0 ){  
    ROLLBACK TRANSACTION;  
}
```

(instead of “commit”)

Concurrency Control

Original Scenario:

Ted's teller queries account, asks for ID
Alice's teller queries account, adds money
Alice's teller updates account
Ted's teller updates account

Serializing Transactions:

Ted's teller queries account, asks for ID
Ted's teller updates account

Alice's teller queries account, adds money
Alice's teller updates account

Alice has to wait. (Pessimistic locking)

Optimistic Locking

Assumes there will be no interference:

Ted's teller queries account, asks for ID

Alice's teller queries account, adds money

Alice's teller updates account

Ted's teller updates account – FAILS

(because the balance has changed.)

The DBMS automatically restarts the Ted transaction. This time it will succeed.

If Alice had been withdrawing instead of depositing, it would still work right. (But Ted could not get money if there weren't \$50 left.)

Isolation and Consistency

Database management systems maintain isolation and consistency by locking.

- *Read lock*: Others can read the same data, but no one can write it because the transaction with the read lock could get inconsistent data.
- *Write lock*: No one else can read until the write transaction has completed.

Consistency is Enforced

The database designer describes what is required for consistency. The DBMS enforces those rules.

- *Attribute integrity*: Each field (attribute) contains valid data.
- *Entity integrity*: Rows are unique; no part of primary key is null
- *Referential integrity*: Connections among tables are consistent.

Durability

Journaling provides for durability.

- Every modification to a database is also written to a journal file.
- If the database must be restored from backup, the journal can be “played back” to restore to the point of failure.
- This is called “forward recovery.”
- Implication: the journal file should not be on the same disk as the database!

Security Requirements

- Physical Integrity
- Logical integrity
- Attribute integrity
- Auditability
- User authentication
- Access control
- Availability

Structured Query Language

The relational model includes SQL (sometimes pronounced “sequel”) a powerful data manipulation language.

- Selection: subsetting the rows of a table.
- Projection: subsetting the columns
- Join: form a new relation (table) with columns from two or more other relations.

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) );
```

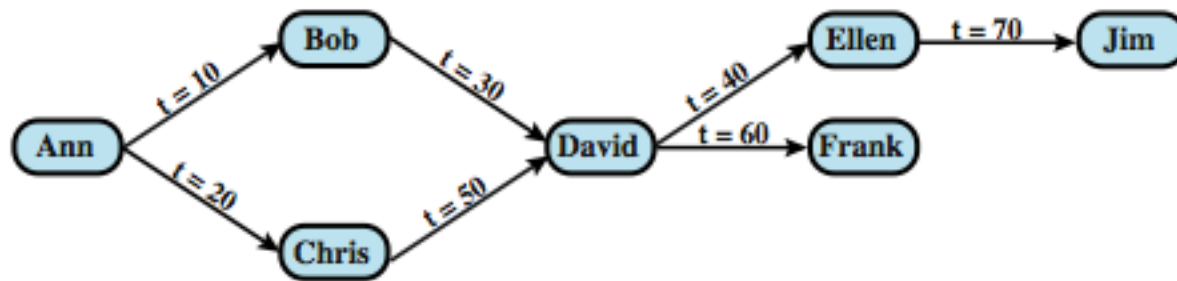
```
CREATE VIEW newtable (Dname, Ename, Eid, Ephone)  
AS SELECT D.Dname E.Ename, E.Eid, E.Ephone  
FROM Department D Employee E  
WHERE E.Did = D.Did;
```

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

SQL Access Controls

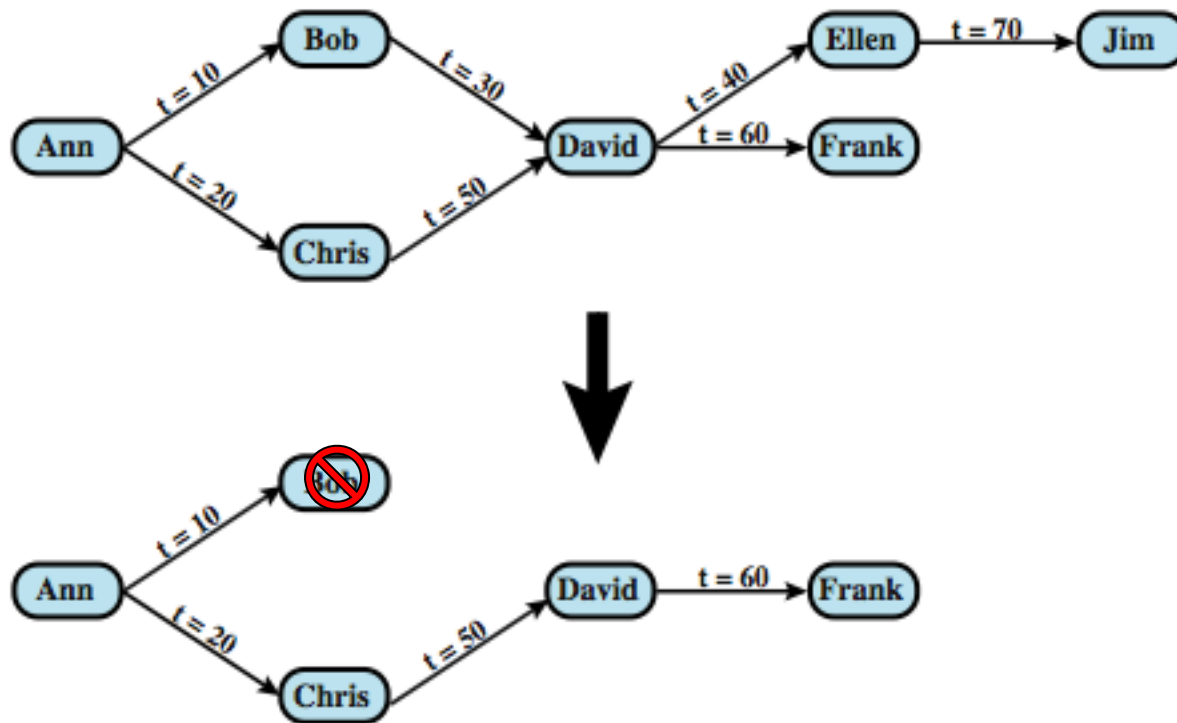
- two commands:
 - **GRANT { privileges | role } [ON table] TO { user | role | PUBLIC } [IDENTIFIED BY password] [WITH GRANT OPTION]**
 - *e.g.* GRANT SELECT ON ANY TABLE TO ricflair
 - **REVOKE { privileges | role } [ON table] FROM { user | role | PUBLIC }**
 - *e.g.* REVOKE SELECT ON ANY TABLE FROM ricflair
- typical access rights are:
 - **SELECT, INSERT, UPDATE, DELETE, REFERENCES**

Cascading Authorizations



What should happen to the other access rights if Bob's access rights are revoked?

Cascading Authorizations



Role-Based Access Control

- role-based access control works 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
 - *cf.* RBAC on Microsoft's SQL Server

Leakage through Inference

- A cost-accounting analyst is allowed to know average salaries and numbers by job class.
- The analyst is not allowed to know individual salaries.
- The analyst asks the DBMS for the average salary for job class “President.”

An 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 (\$)	Name	Department
senior	43,000	Andy	strip
junior	35,000	Calvin	strip
senior	48,000	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 Countermeasures

- Inference detection at database design
 - alter database structure or access controls
- Inference detection at query time
 - by monitoring and altering or rejecting queries
- We need an inference detection algorithm
 - a difficult problem
 - consider the employee-salary example

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
- The security problem is one of inference

Statistical Database Security

- Statistics are derived using a characteristic formula C
 - a logical formula over the values of attributes
 - *e.g.* $(\text{Sex}=\text{Male}) \text{ AND } ((\text{Major}=\text{CS}) \text{ OR } (\text{Major}=\text{EE}))$
- Query set $X(C)$ of characteristic formula C , is the set of records matching C
- A statistical query is a query that produces a value calculated over a query set

Statistical Database Example

(a) Database with statistical access with $N = 13$ students

Name	Sex	Major	Class	SAT	GP
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	Psy	1979	580	2.8
Iles	Male	CS	1981	600	3.2
Jones	Female	Bio	1979	750	3.8
Kline	Female	Psy	1981	500	2.5
Lane	Male	EE	1978	600	3.0
Moore	Male	CS	1979	650	3.5

Find Baker's GPA (not allowed.) We know that Baker is a female EE student, but we don't know whether she is the only one. Do this:

```
count(EE, Female) = 1
sum(EE Female, GP) = 2.5
```


Types of Disclosures

- Exact data
- Bounds (upper and lower)
- Negative result
- Existence
- Probabilities

Problem: Aggregate Data

- A user may have access to aggregate even when access to specific rows is disallowed.
- Aggregate data includes counts, averages, etc.
- Example: find the salary of the president of a company:
 - Count salaries over 100,000: 4
 - Count salaries over 200,000: 1
 - Count salaries over 300,000: 0
 - Count salaries over 250,000: 1
- Salary is bounded by 250,000 and 300,000

A Defense: Perturbation

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

Controls

- Views!
- Suppression: data are not provided
- Concealment: data close to the value are provided.
- Combining results. Example: release financial aid counts in ranges.
- Sampling
- Query analysis (hard)

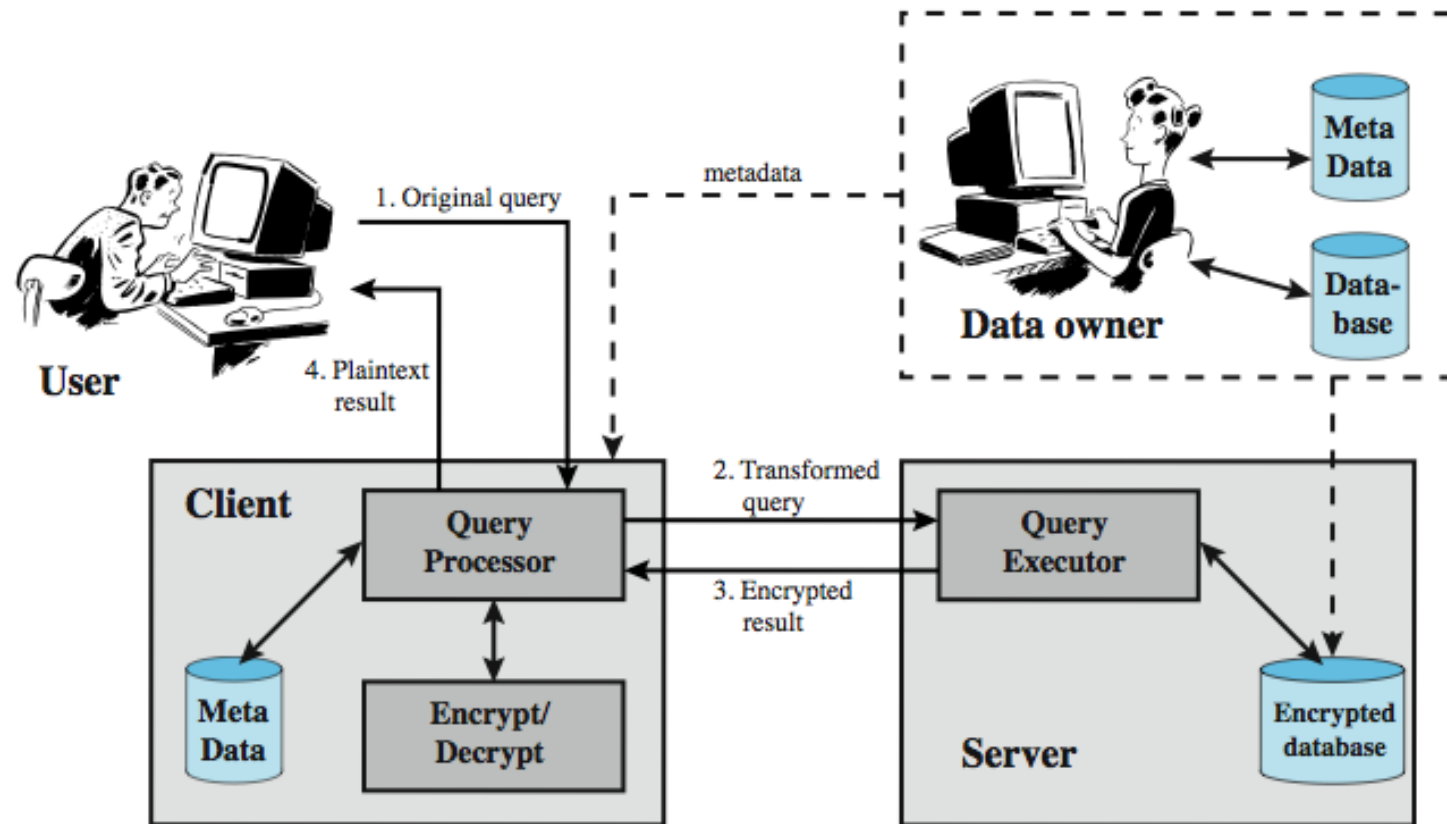
Other Query Restrictions

- Query set overlap control
 - limit overlap between new and previous queries
 - has problems and overheads
- Partitioning
 - cluster records into exclusive groups
 - only allow queries on entire groups
- Query denial and information leakage
 - denials can leak information
 - to counter, we must track queries from user

Database Encryption

- Databases are typically a valuable information resource
 - protected by multiple layers of security: firewalls, authentication, O/S access control systems, DB access control systems, and database encryption
- We can encrypt:
 - entire database - very inflexible and inefficient
 - individual fields - simple but inflexible
 - records (rows) or columns (attributes) – a good compromise, but we also need attribute indexes to help data retrieval
- There are varying trade-offs

Database Encryption



SQL Injection

- Another widely exploited attack on databases
- When input is used in SQL query to database
 - similar to command injection (studied later)
 - SQL meta-characters are the concern
 - must check and validate input for these

```
$name = $_REQUEST['name'];  
$query = "SELECT * FROM suppliers WHERE name = '" . $name . "'";  
$result = mysql_query($query);
```

Bob'; drop table suppliers; --

```
$name = $_REQUEST['name'];  
$query = "SELECT * FROM suppliers WHERE name = '" .  
    mysql_real_escape_string($name) . "'";  
$result = mysql_query($query);
```


SQL Injection Defined

An attack against a service that works by entering SQL commands where data are expected. Faulty programming may allow such SQL commands to be executed by the DBMS.

SQL injection attacks are best prevented by good coding practices because they rely on services that must be available for ordinary operation.

A Bad Example

```
$userid = $_POST['userid'];  
$passwd = $_POST['passwd'];  
$q="SELECT * from logins where ";  
$q .= "userid = '$userid' ";  
$q .= "and passwd = '$passwd'";  
$dbResult = @pg_query($q);  
$loggedIn = pg_num_rows($dbResult);  
// $loggedIn will be 0 or 1  
// 0 = false; non-zero = true
```

That Select Sentence

The programmer's design calls for the constructed SELECT sentence to look like this:

```
SELECT * from logins where  
userid = 'bbrown'  
and passwd = 'xyzzzy';
```

If all is well, this will retrieve Brown's login information, and the "loggedIn" variable will be TRUE. If there's no user bbrown or the password doesn't match, the "loggedIn" variable will be zero, which is FALSE.

Bad Guys Discover Brown's ID

But not his password, because he's careful!

The bad guys try to log on like this:

User ID:

bbrown

Password:

hi' or 1=1; #

The Select Sentence Again

With the data supplied, the SELECT sentence looks like this:

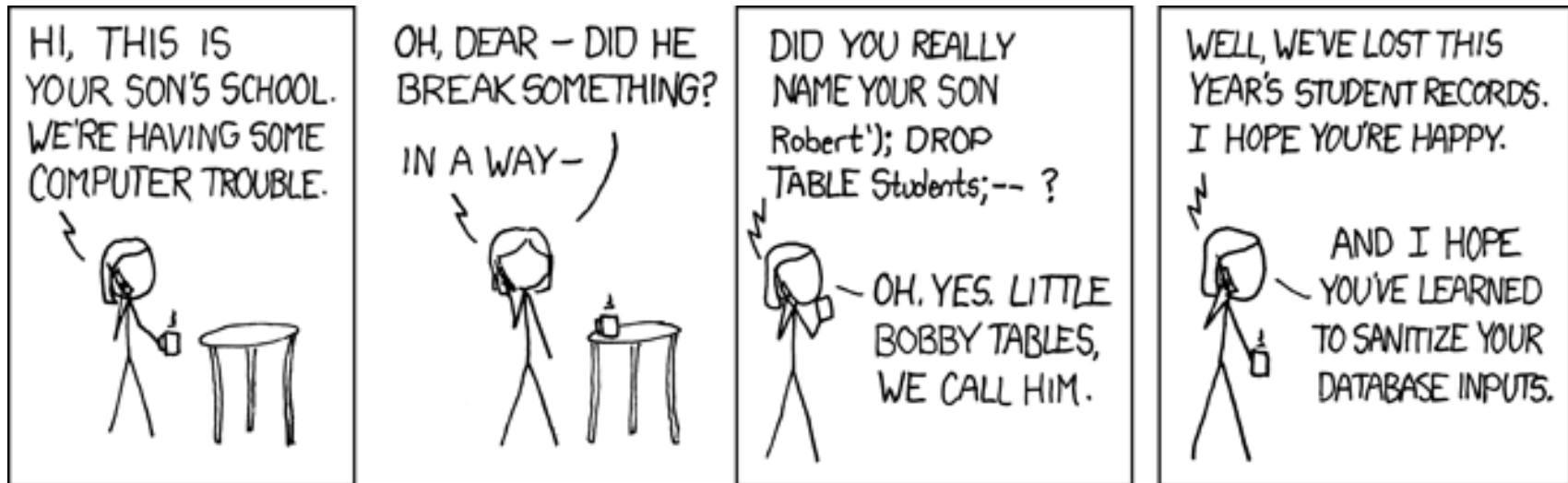
```
SELECT * from logins where  
userid = 'bbrown'  
and passwd = 'hi' or 1=1; #';
```

User ID:

Password:

The # is an SQL comment introducer. The bad guys are logged in as Brown. *Oops.*

Even Non-Malicious Input



<http://xkcd.com/327/>

The Defense

PostgreSQL (and recently, MySQL and MariaDB) allows parameterized queries. The parameters are always treated as variables and never as commands, no matter what they look like.

The Defense

```
$userid = $_POST['userid'];
$passwd = $_POST['passwd'];
$q="SELECT pw from logins where userid = $1;";
$dbResult =
    @pg_query_params($q,array($userid))
    or die("database failure");
$loggedIn = FALSE;
$num = pg_num_rows();
if ($num == 1) {
    $row = pg_fetch_assoc($dbResult);
    $pw = $row['pw'];
    if ($pw === $passwd) $loggedIn = TRUE;
}
```


Parameterized Queries

Also known as:

- Prepared queries
- “bind variables”

(There is a slight technical difference between parameterized queries and prepared queries in that prepared queries have been “pre-compiled” by the SQL processor.)

Cloud Computing

NIST defines cloud computing as follows:

“A model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (*e.g.*, networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models.”

Cloud: Essential Characteristics

Cloud computing provides:

- Resource pooling
- Broad network access
- Rapid elasticity
- Measured service
- On-demand self-service

Cloud: Service Models

- Software as a service
- Platform as a service
- Infrastructure as a service

Cloud: Deployment Models

- Public
- Private
- Hybrid
- Community

Cloud Security Risks

- Abuse and misuse of cloud computing
- Insecure interfaces and APIs
- Malicious insiders
- Shared technology problems
- Data loss or leakage
- Account or service hijacking
- Unknown risk profile

Threat of Data Compromise Increases

- There are risks and challenges unique to cloud computing
- Multi-instance model: each customer gets a unique instance of DBMS, OS, etc.
- Multi-tenant model: one instance serves multiple customers. Provider must establish security.

Questions

