Database Security

CISC 6640 PRIVACY AND SECURITY IN BIG DATA

Instructor:
Md Zakirul Alam Bhuiyan
Assistant Professor
Department of Computer and Information Sciences
Fordham University

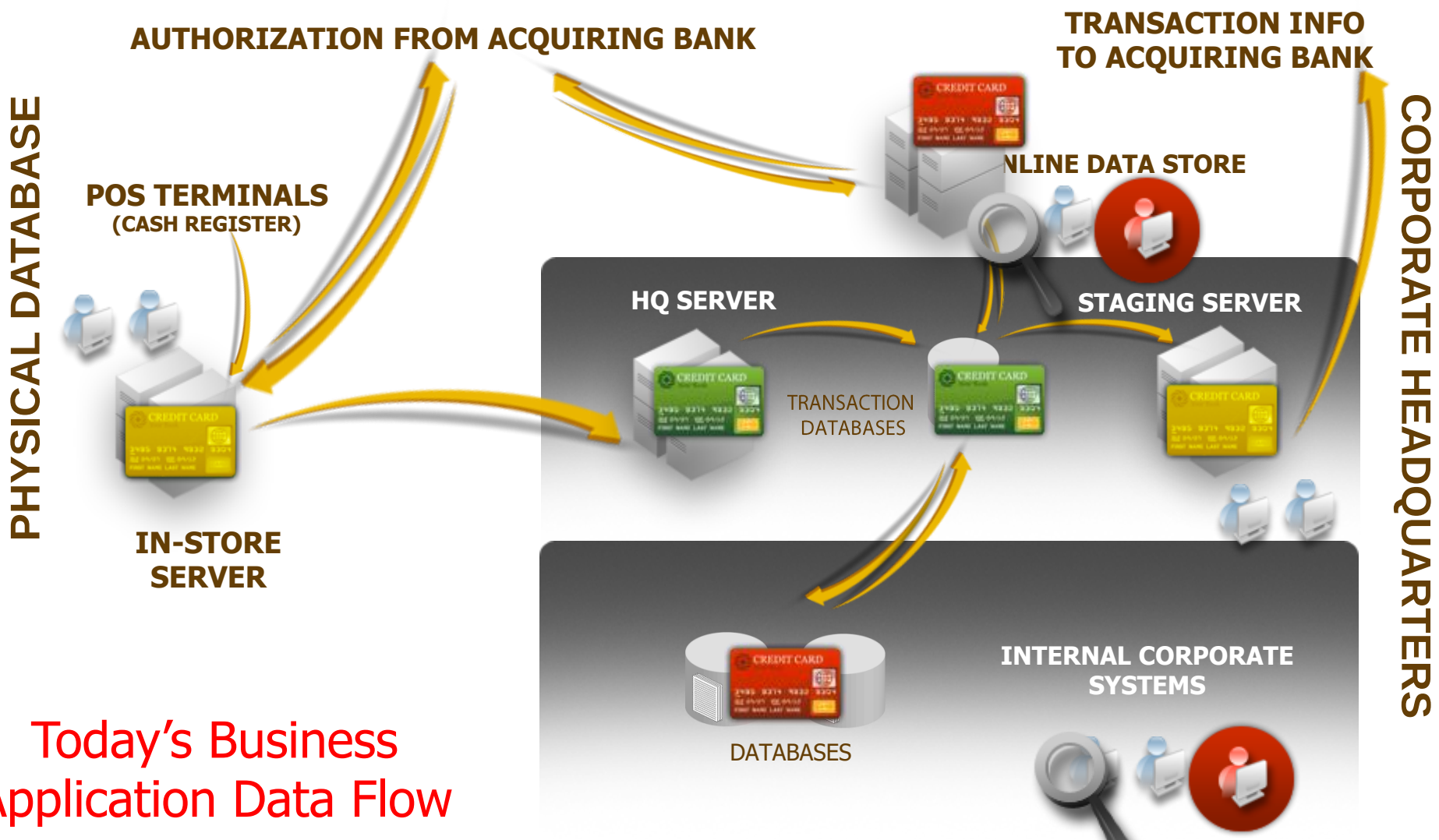
Review Quiz

- Give a few examples to describe privacy vs. security
- What are Big Data privacy concerns?
- What are methods to protect privacy concerns?
- What are uses of crypto for Big Data privacy?
- Tell a few Big Data security and privacy challenges.

We Are Going to Learn

- Database Security: Definition
- Relational Databases
  - Database security models
- No SQL Databases
- Object Based vs. Object Oriented
- Overview of Database Vulnerabilities
  - Common DBMS vulnerabilities
- Overview of Database topics/issues (indexing, inference, aggregation, polyinstantiation)
  - Security issues of inference and aggregation
- Hashing and Encryption
- Database access controls (DAC, MAC, RBAC, Clark-Wilson)
- Information flow between databases/servers & applications

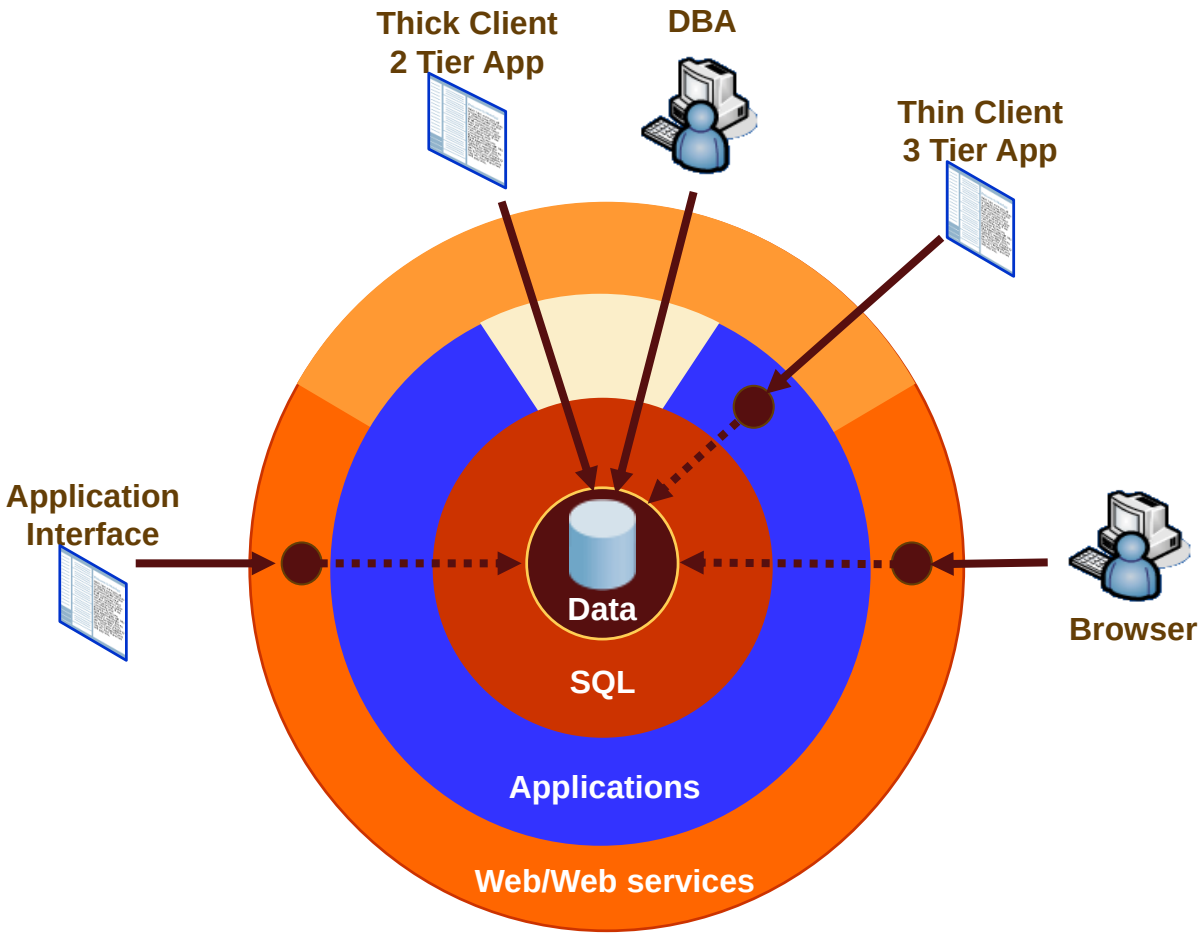
Database Security



Today's Business Application Data Flow

Database Security

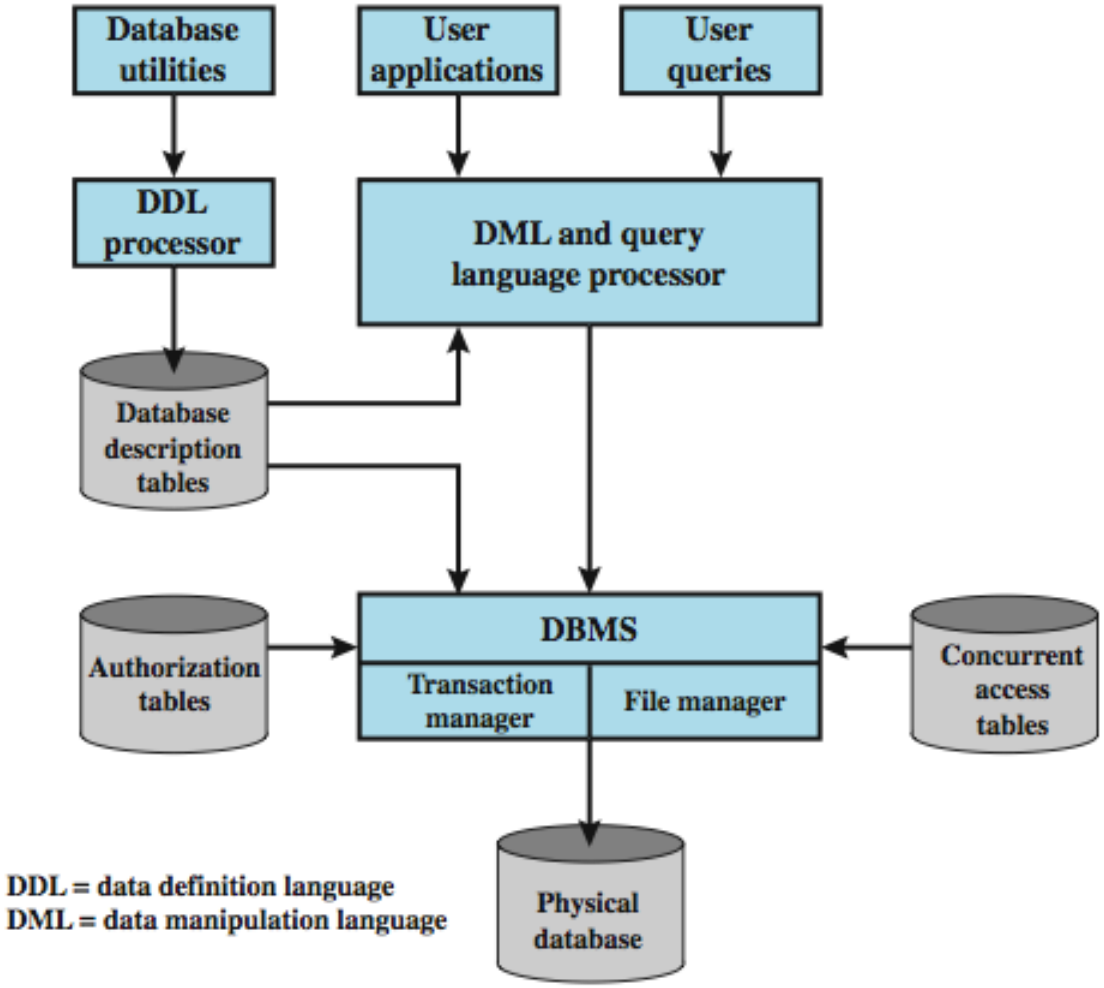
Business Application Data Security Challenge



Database systems are often very complex, combining the core database with a collection of applications…It is not sufficient to protect the database alone, all the associated applications need to be secured.

--SANS Top 20 Internet Security Risks of 2007

Database Security



Database Security

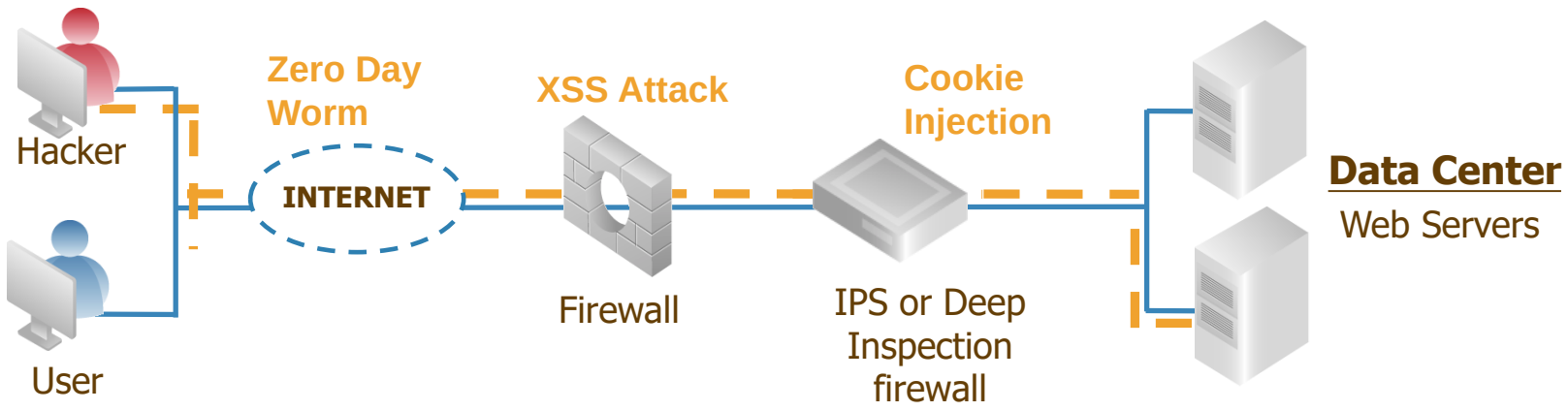
- **Why Should You Care?**

85% of organizations have experienced a data breach

Theft, Abuse, Misuse & Leakage Happen Even in Leading Organizations

Database Security

- Traditional firewalls only detect network attacks
  - Only inspect IP address, port/service number
- IPS/IDS signatures only detect known threats
  - No application or data understanding
  - No user/session tracking
  - High rate of false positives/negatives
  - No protection of SSL traffic



Database Security: Definition

- **Database Security is**

  the mechanism that protect the database against intentional or accidental threats.

- **We consider database security in relation to the following situations:**
  - **Theft and Fraud**
  - **Loss of confidentiality**

Database Security: Types

- Types of Security
  - Legal and ethical issues
  - Policy issues
  - System-related issues
  - The need to identify multiple security levels

Database Security: Issues

- Threats to Databases
  - **Loss of integrity**
    - Users should not be able to modify things they are not supposed to.
      - – E.g., Only instructors can assign grades.
  - **Loss of availability**
    - Users should be able to see and modify things they are allowed to.
  - **Loss of confidentiality**
    - Users should not be able to see things they are not supposed to.
      - E.g., A student can't see other students' grades.

Database Security: Countermeasures

- To protect databases against these types of threats, four kinds of countermeasures can be implemented
  - Database access control
  - Inference control
  - Information flow control
  - Encryption

We Are Going to Learn

- Database Security
- **Database access controls (DAC, MAC, RBAC, Clark-Wilson)**
- Relational Databases
- No SQL Databases
- Object Based vs. Object Oriented
- Overview of Database Vulnerabilities
- Overview of Database topics/issues (indexing, inference, aggregation, polyinstantiation)
- Hashing and Encryption

Database Access Controls

- The security mechanism of a DBMS must include provisions for restricting access to the database as a whole
  - A function, called database access control to control which (active) subject have access to a which (passive) object with some specific access operation.
    - Objects: files, directories, etc

Database Access Controls

- It is handled by creating user accounts and passwords to control login process by the DBMS.
  - Given a subject and object pair (s, o)
    - A requested operation r , from s to o , returns a true value if requested is permitted

Database Access Controls

- DAC
  - Discretionary Access Control
- MAC
  - Mandatory Access Control
- RBAC
  - Rule-Based Access Control
  - Role-Based Access Control
- Clark-Wilson
  - Integrity Policy based Access Control

Database Access Controls: DAC

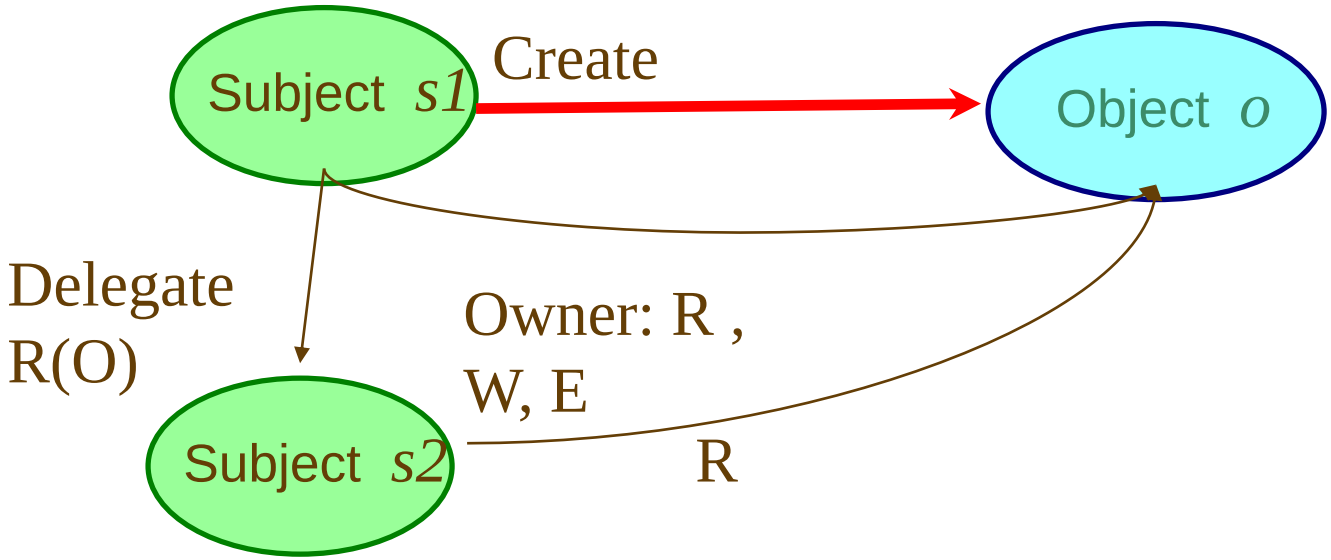
- Discretionary access control (DAC)
  - Restricts access to objects based on the identity of subjects and/or groups to which they belong.
  - The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject
    - Granting and revocation of privileges
  - DAC is flexible in terms of policy specification

Database Access Controls: DAC

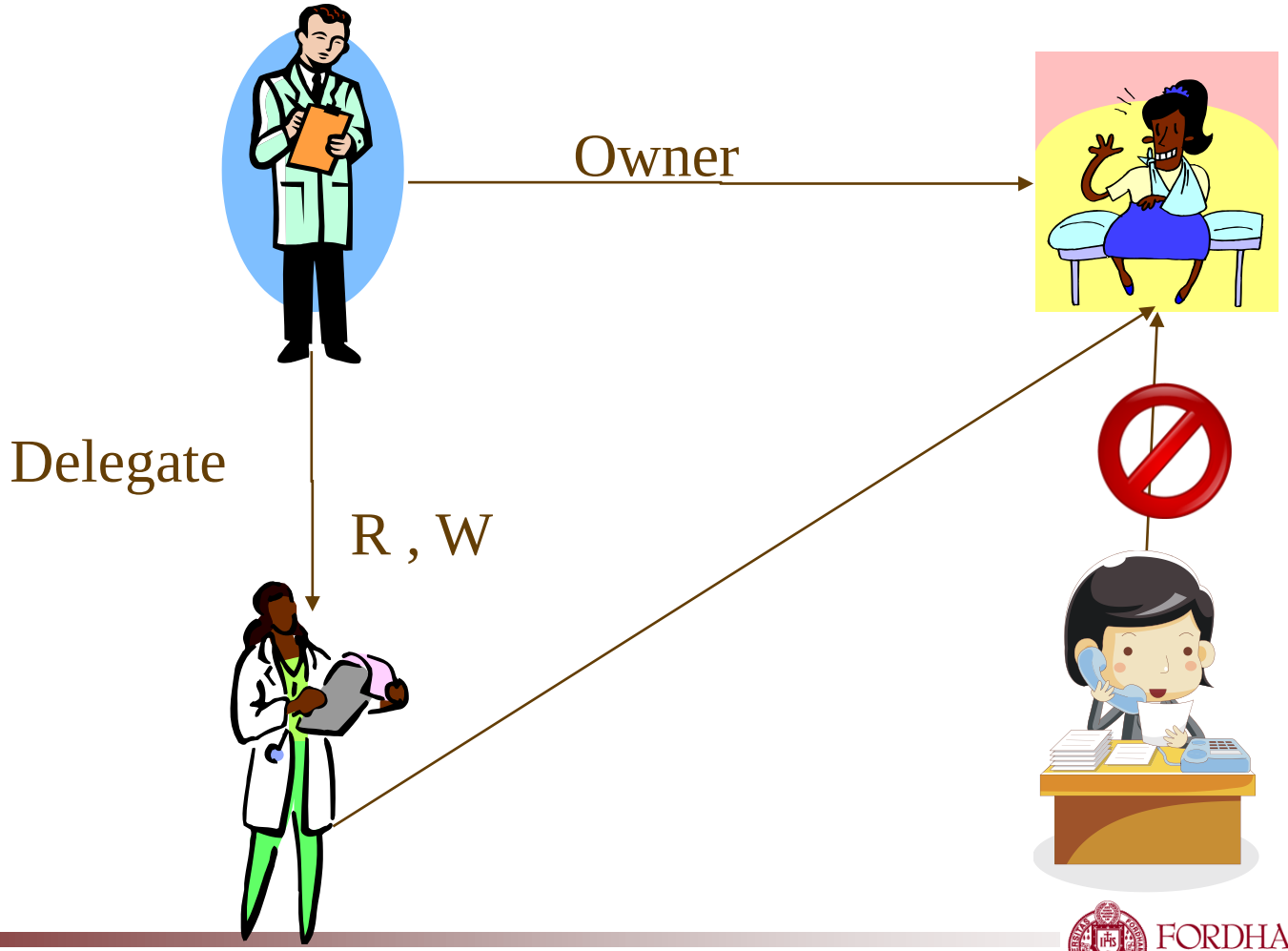
- Discretionary access control (DAC)
  - **Discretionary security models provide access control on an individual basis:**
    - Access Control matrix is a fundamental and widely used Disc Access control Model for enforcing security policies
    - A security policy is a statement that specifies what privileges and limitations a certain subject has on an object
      - Ex: subject s can access object x if it has not accessed object y.

Database Access Controls: DAC

- The owner of an object may delegate the permission of the object to another user.

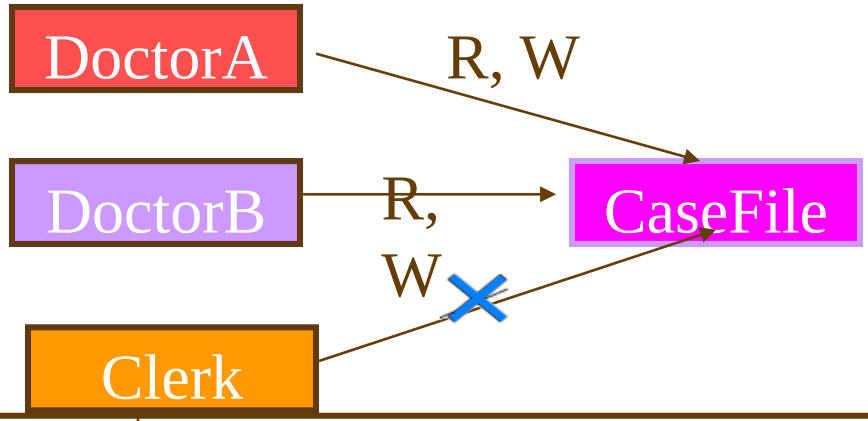


Database Access Controls: DAC



Database Access Controls: DAC

Database security models



Object	Permission
CaseFile1	DoctorA:: R, W
CaseFile2	DoctorA:: R, W
CaseFile1	DoctorA: DoctorB:: R, W
CaseFile3	DoctorB:: R, W

Access Control List

Database Access Controls: DAC

Database security models

- Access Control Matrix (ACM)
  - Describes protection state precisely
  - Matrix describing rights of subjects
  - State transitions change elements of matrix

Sub \ Obj	File a	File B	File c	File d
User A	Owner	Read / write	Execute	Owner
User B	Copy read	Owner		
User C		Read	Owner	Append

(a) Resource ACM

Sub \ Obj	Process A	Process B	Process C
Process A		Send/ unblock	Send / unblock
Process B	Receive		Block
Process C	Receive	Block	

(b) Process ACM

Database security models

objects (entities)

subjects

	o_1	…	o_m	s_1	…	s_n
s_1						
s_2						
…						
s_n						

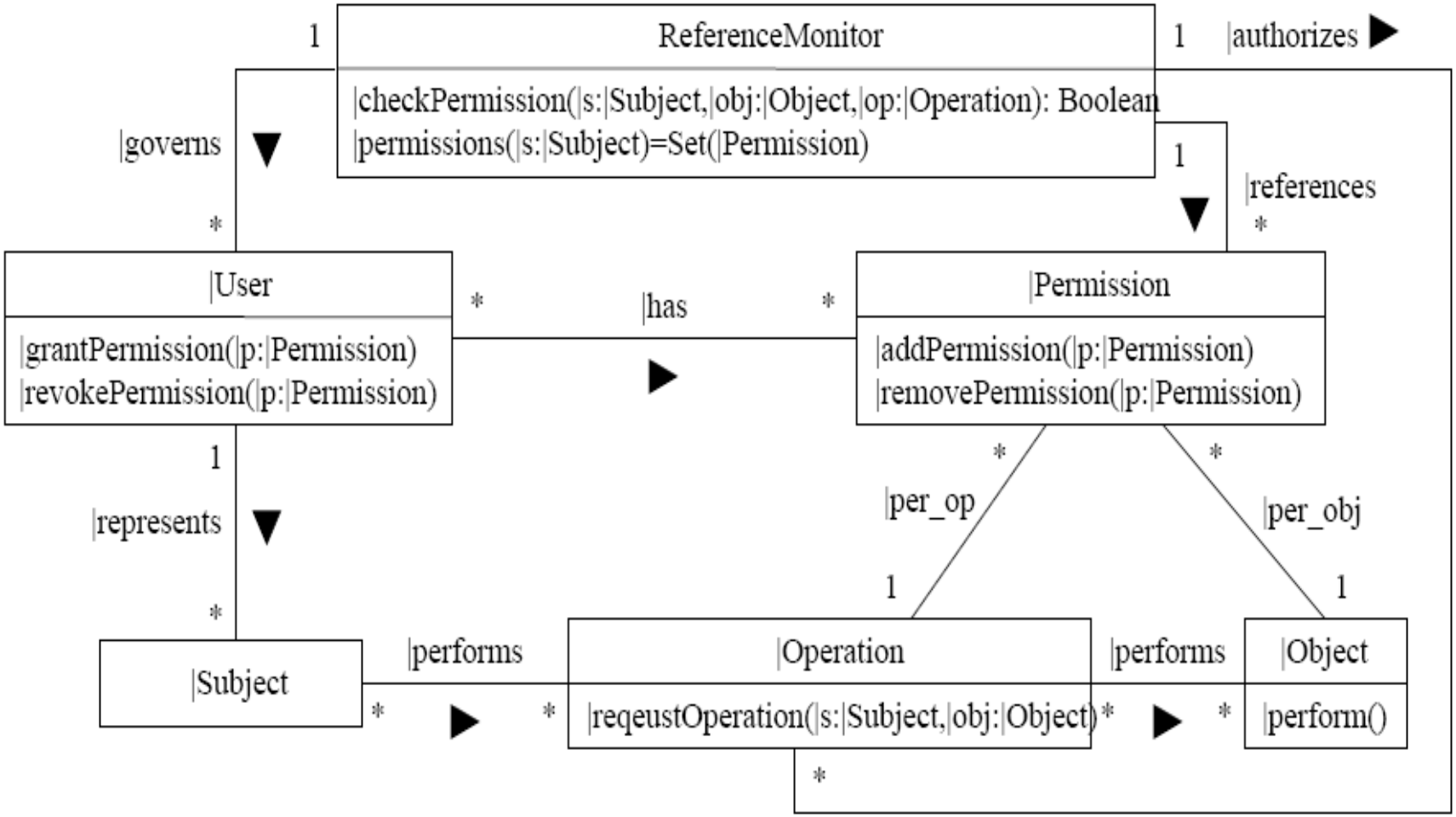
- Subjects $S = \{ s_1,\ldots,s_n \}$
- Objects $O = \{ o_1,\ldots,o_m \}$
- Rights $R = \{ r_1,\ldots,r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ means subject s_i has rights $r_x, \ldots, r_y$ over object o_j

	File 1	File 2	File 3	Program 1
Ann	own read write	read write		execute
Bob	read		read write	
Carl		read		execute read

Database Access Controls: DAC

Database security models

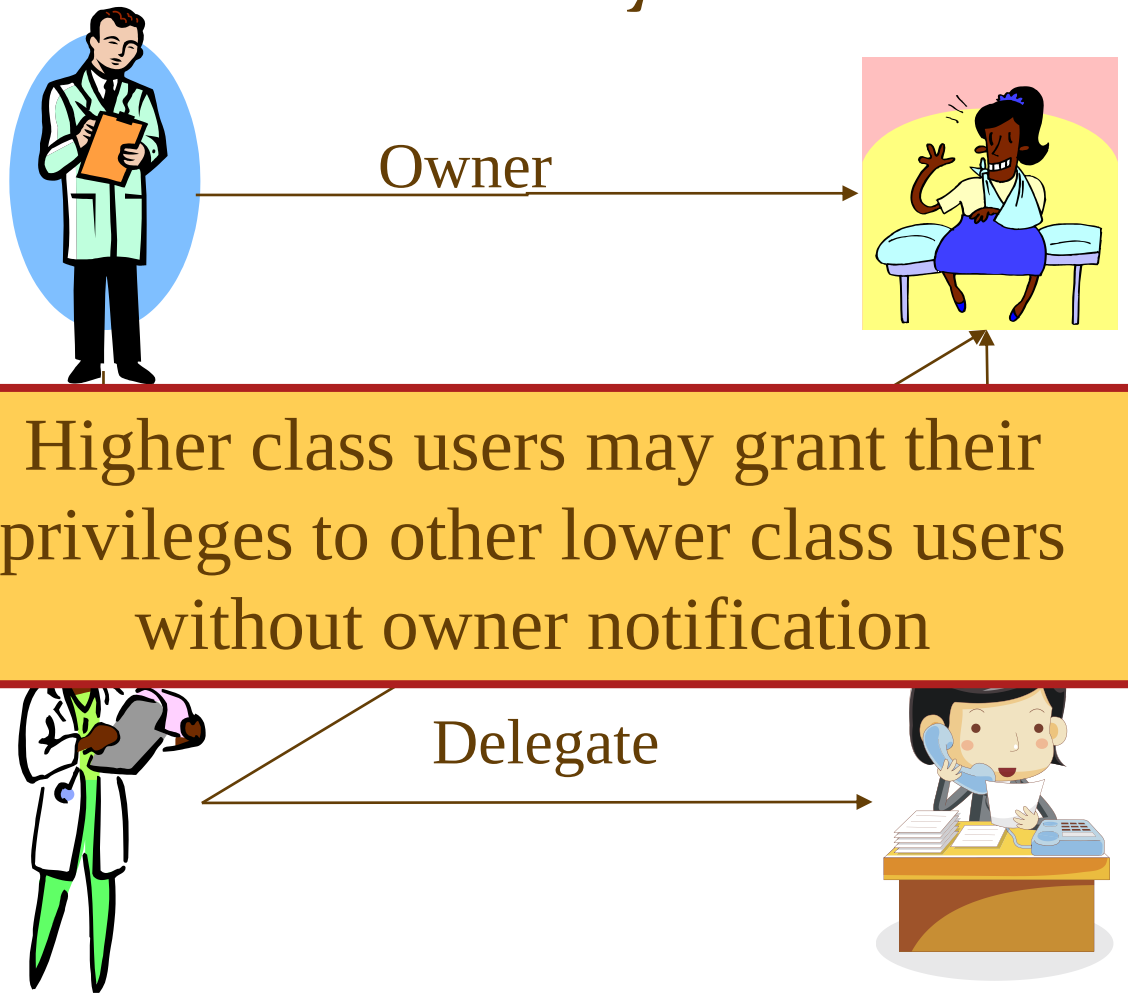
DAC Design Pattern Structure



Database Access Controls: MAC

- **Mandatory Access Control (MAC)**
  - **Refers to a type of security strategies that restricts the ability individual resource owners have to grant or deny access to resource objects in a file system.**
  - **Refers to a type of access controls by which the OS constrains the ability of a subject or initiator to access or generally perform some sort of operation on an object or target.**
    - Level of subjects (e.g., users) and objects (e.g., data).
    - Access to an object is granted only if the security levels of the subject and the object satisfy certain constraints.

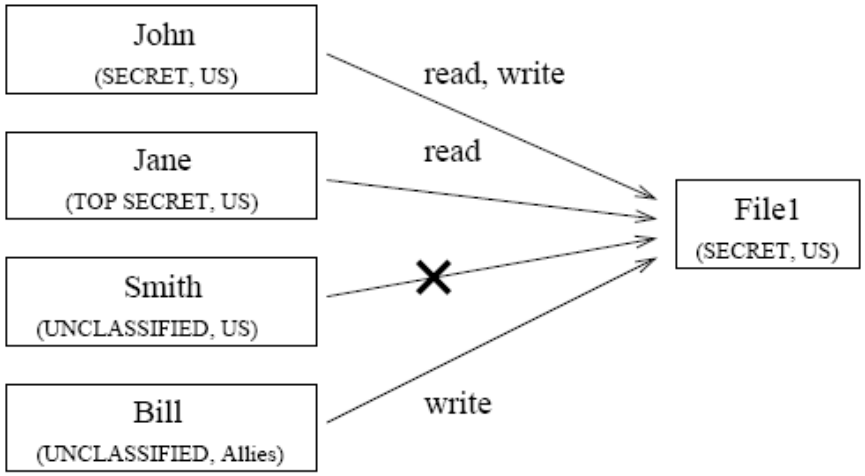
- **Known as multilevel security model**



Classification	Category
UNCLASSIFIED	U.S.
CONFIDENTIAL	U.S.
SECRET	U.S.
TOP SECRET	U.S.
UNCLASSIFIED	Allies
CONFIDENTIAL	Allies
SECRET	Allies
TOP SECRET	Allies

User	Classification	Category
John	SECRET	US
Jane	TOP SECRET	US
Smith	UNCLASSIFIED	Allies
Bill	UNCLASSIFIED	US

File	Classification	Category
File1	SECRET	US



Database Access Controls: RBAC

- **RBAC**
  - **Role-Based Access Control**
    - Also known as *Non discretionary Access Control*
    - Access is based on a user's job function within the organization to which the computer system belongs.
      - E.g., Limited login hours, limited BitTorrent traffic
  - **Rule-Based Access Control**
    - Access is allowed or denied to resource objects based on a set of rules defined by a system administrator.
      - E.g., SecurityAdmin, DatabaseAdmin, EmailAdmin

Database Access Controls: Clark-Wilson

- Integrity policy based access control
- Where integrity is enforced across both the OS and the application.
  - In the commercial environment, the goal is to prevent fraud and errors – no user, even if authorized, should be able to modify data in an invalid way
    - So the model focus on integrity enforcement and authorization mechanisms to prevent illegal modification. The seminal work is Clark-Wilson's integrity model.

Database Access Controls: Clark-Wilson

- For our commercial security requirements, the Clark-Wilson model results in a conceptual security model, defined by the identification of,
  - Data items for which security enforcement is crucial (CDIs);
  - Transformation procedures (TPs) that can access data;
  - User roles, in terms of authorization to use particular TPs.

Prevent Access Control

- Don't let users shoot themselves in the foot
- Main driver for early features
- Not security per-se, but a critical first step
- Doesn't require strong authentication

Stop Malicious Users

- Early features were necessary, but not sufficient
- Security has to get real
- Hadoop runs arbitrary code
- Implicit trust doesn't prevent the insider threat

We Are Going to Learn

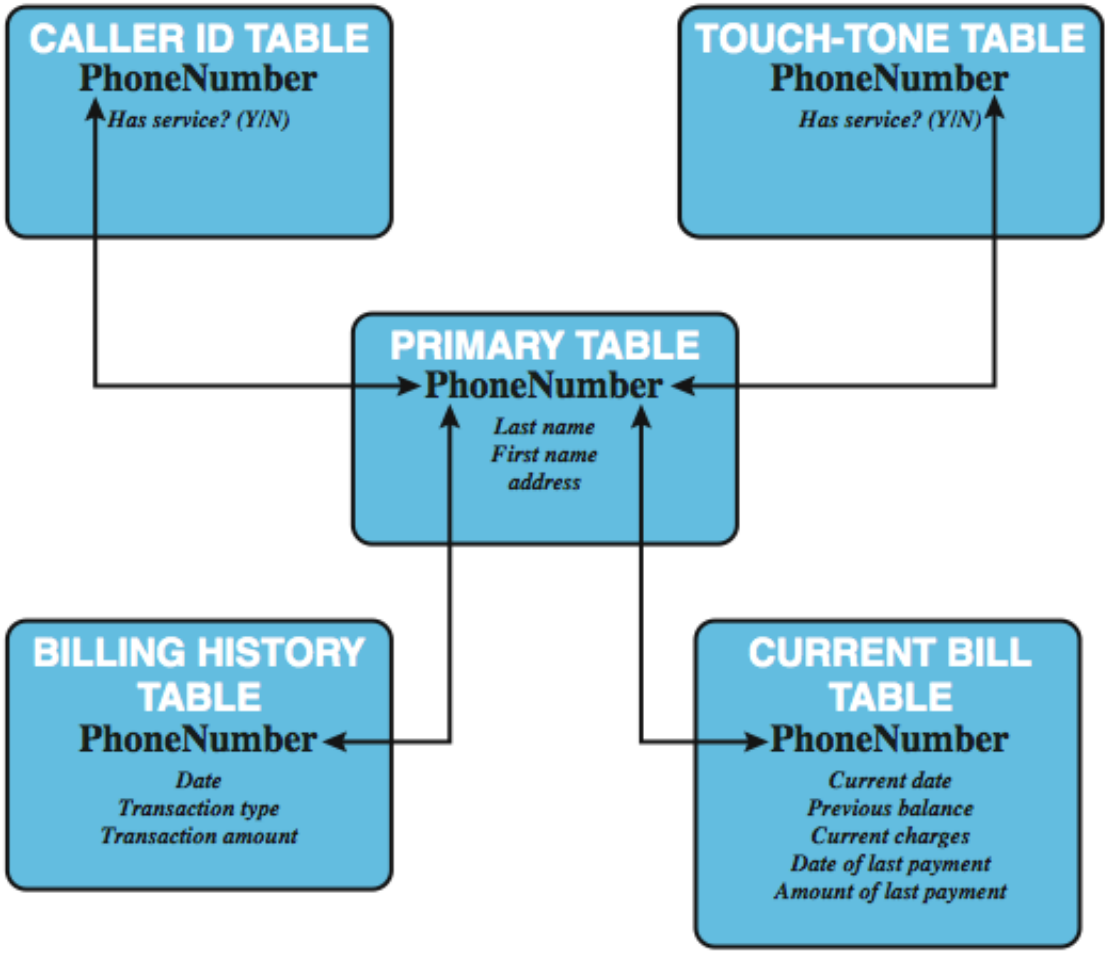
- Database Security
- Database access controls (DAC, MAC, RBAC, Clark-Wilson)
- **Relational Databases**
- No SQL Databases
- Object Based vs. Object Oriented
- Overview of Database Vulnerabilities
- Overview of Database topics/issues (indexing, inference, aggregation, polyinstantiation)
- Hashing and Encryption

Relational Databases

- Constructed from tables of data
  - Each column holds a particular type of data
  - Each row contains a specific value
  - Ideally has one column where all values are unique, forming an identifier/key for that row
- Has multiple tables linked by identifiers
- Use a query language to access data items meeting specified criteria

Relational Databases

- Relational Database Example



Relational Databases

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

(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

Relational Databases

- Relational Database Example
  - 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 Databases

- Structure Query Language (SQL)
  - That is 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
```

Relational Databases: Security

- 92% of Web applications have vulnerabilities
  - 93% of vulnerable sites are still vulnerable after code fixes!!
  - SQL Injection – still majority cause of data leakage
    - Ponemon estimates breaches cost on average $202 per compromised record

Relational Databases: Security

- Access Control
  - The relation level (or table level)
  - At this level, the DBA can control the privilege to access each individual relation or view in the database.

Relational Databases: Security

- The privileges at the account level apply to the capabilities provided to the account itself and can include
  - the CREATE SCHEMA or CREATE TABLE privilege, to create a schema or base relation;
  - the CREATE VIEW privilege;
  - the ALTER privilege, to apply schema changes such adding or removing attributes from relations;
  - the DROP privilege, to delete relations or views;
  - the MODIFY privilege, to insert, delete, or update tuples;
  - and the SELECT privilege, to retrieve information from the database by using a SELECT query.

Relational Databases: Security

- **The second level of privileges applies to the relation level**
  - **This includes base relations and virtual (view) relations.**
  - **The granting and revoking of privileges generally follow an authorization model for discretionary privileges known as the access matrix model where**
    - The rows of a matrix M represents subjects (users, accounts, programs)
    - The columns represent objects (relations, records, columns, views, operations).
    - Each position M(i,j) in the matrix represents the types of privileges (read, write, update) that subject i holds on object j.

Database security models

objects (entities)

subjects

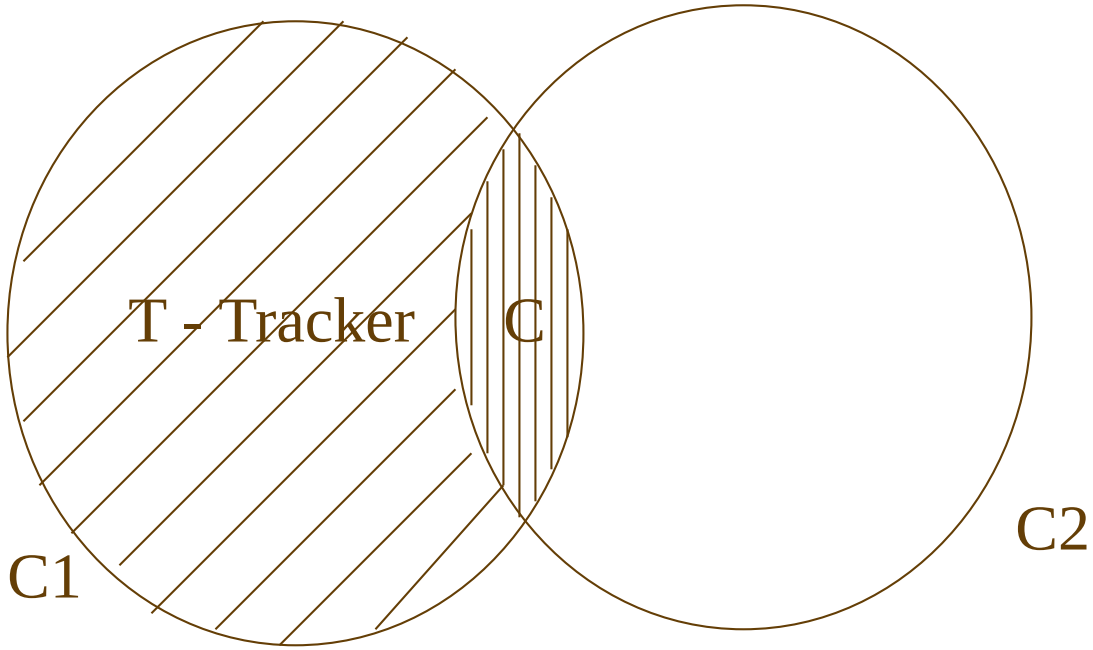
	o_1	…	o_m	s_1	…	s_n
s_1						
s_2						
…						
s_n						

- Subjects $S = \{ s_1,\ldots,s_n \}$
- Objects $O = \{ o_1,\ldots,o_m \}$
- Rights $R = \{ r_1,\ldots,r_k \}$
- Entries $A[s_i, o_j] \subseteq R$
- $A[s_i, o_j] = \{ r_x, \ldots, r_y \}$ means subject s_i has rights $r_x, \ldots, r_y$ over object o_j

	File 1	File 2	File 3	Program 1
Ann	own read write	read write		execute
Bob	read		read write	
Carl		read		execute read

Tracker Attack 1

Query q(C) is disallowed



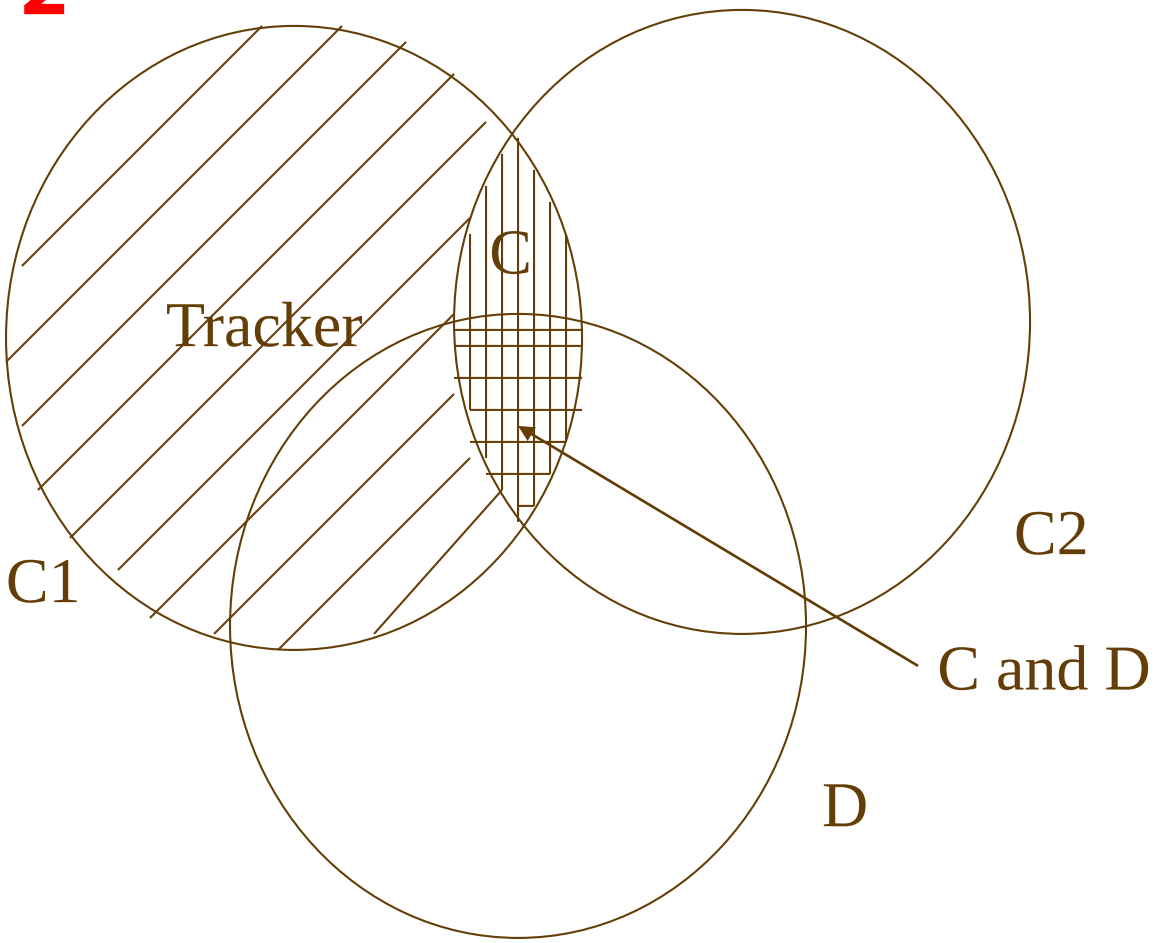
C = C1 and C2
T = C1 and ~C2

Attacker runs instead 2 queries: q(C1) and q(T)
where q(C) = q(C1) – q(T)
=> infers q(C) from q(C1) and q(T)

Tracker Attack 2

Query q(C and D)
is disallowed

C = C1 and C2
T = C1 and ~C2



Attacker runs instead 2 queries: q(T or C and D) and q(T)
where q(C and D) = q(T or C and D) – q(T)
=> infers q(C and D) from q(T or C and D) and q(T)

Insertion/Deletion Attack

- Observing changes over time
- $q_1 = q(C)$
- Insert(i)
- $q_2 = q(C)$
- $q(i) = q_2$ „-" q_1
  - where „-" means compensation for insertion that permist to infer

Relational Databases: Security

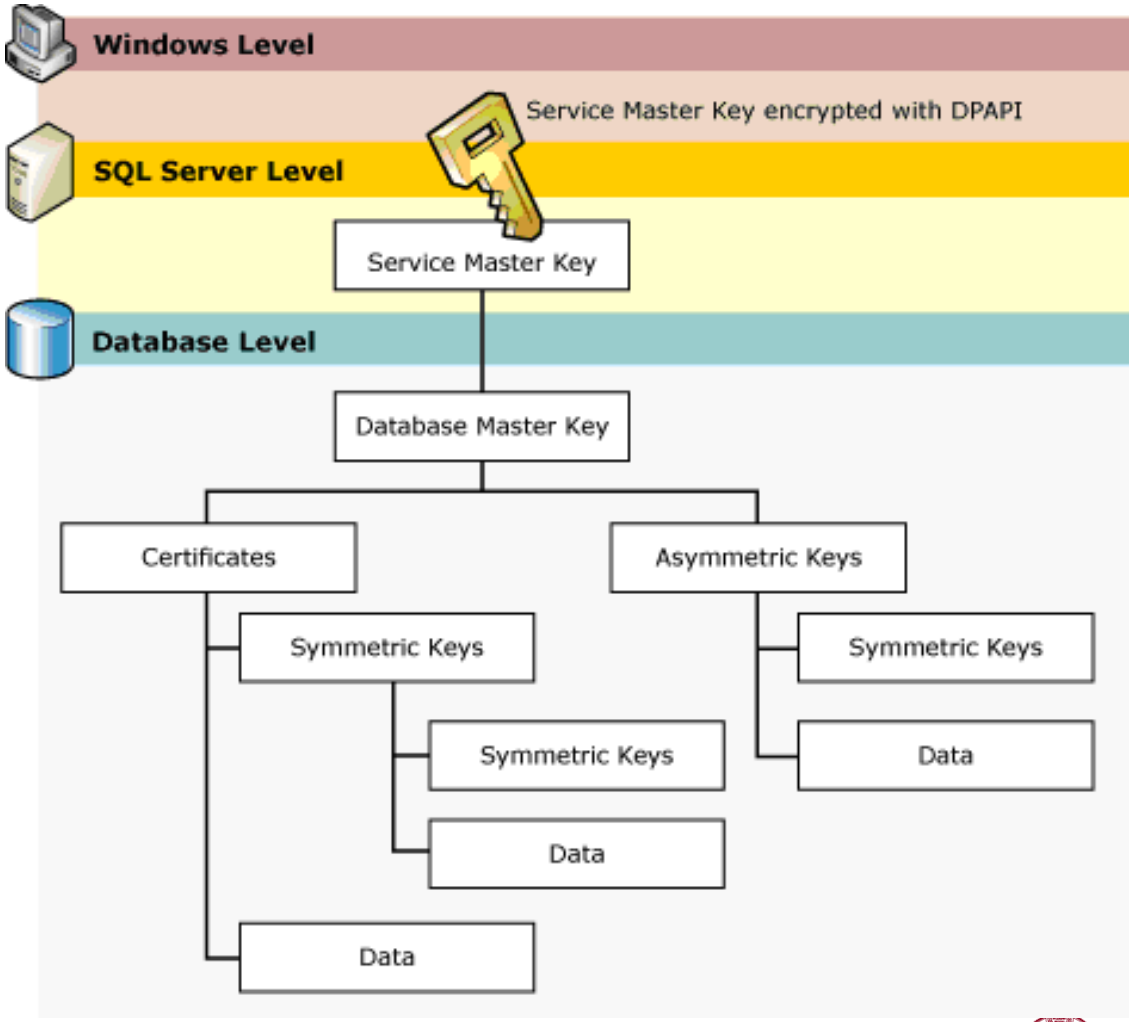
- In SQL the following types of privileges can be granted on each individual relation R:
  - SELECT (retrieval or read) privilege on R
  - MODIFY privileges on R

Relational Databases: Security

- **Specifying Privileges Using Views**
  - **The mechanism of views is an important discretionary authorization mechanism in its own right.**
- **Revoking Privileges**
  - **In some cases it is desirable to grant a privilege to a user temporarily.**
    - Suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 can issue:

      REVOKE SELECT ON EMPLOYEE FROM A3;

Relational Databases: Security



We Are Going to Learn

- Database Security
- Database access controls (DAC, MAC, RBAC, Clark-Wilson)
- Relational Databases
- **No SQL Databases**
- Object Based vs. Object Oriented
- Overview of Database Vulnerabilities
- Overview of Database topics/issues (indexing, inference, aggregation, polyinstantiation)
- Hashing and Encryption

NoSQL Databases

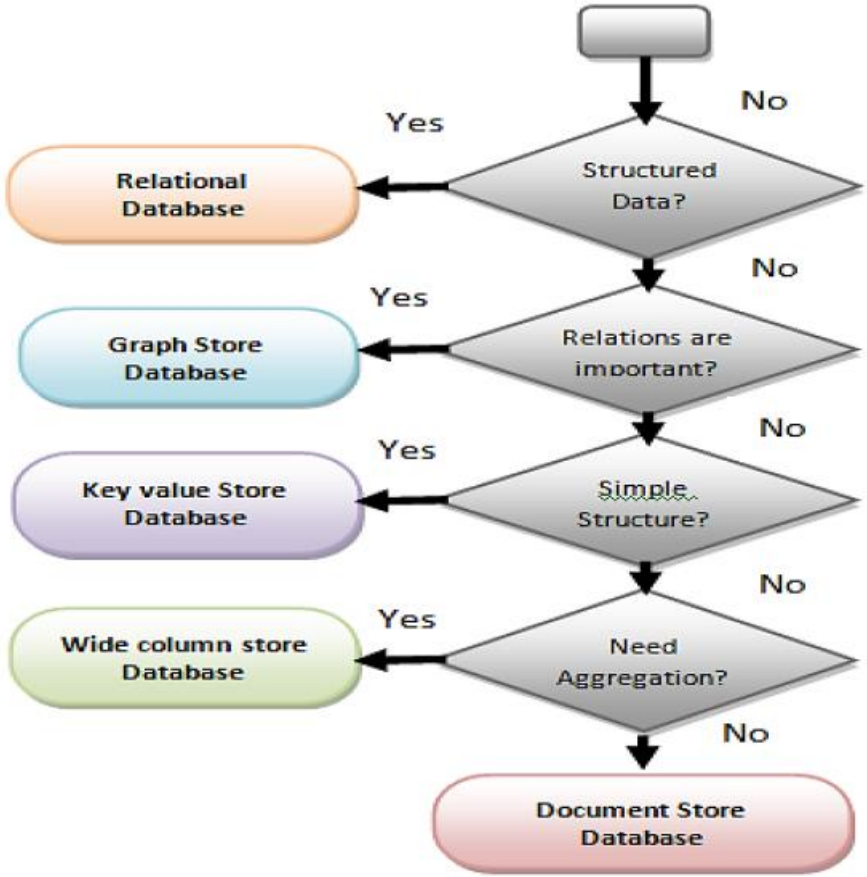
- Designed for solving the Big Data issue
- "non SQL", "non relational" or "not only SQL
- Non-relational database management systems
  - **It provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases**
    - Generally don't use tables
    - Highly scalable & very quick
    - Eventual consistency

NoSQL Databases: Current User

Company Name	NoSQL Name	NoSQL Storage Type
Adobe	HBase	Column
Amazon	Dynamo | SimpleDB	Key-Value | Document
BestBuy	Riak	Key-Value
eBay	Cassandra | MongoDB	Column | Document
Facebook	Cassandra | Neo4j	Column | Graph
Google	BigTable	Column
LinkedIn	Voldemort	Key-Value
LotsOfWords	CouchDB	Document
MongoHQ	MongoDB	Document
Mozilla	HBase | Riak	Column | Key-Value
Netflix	SimpleDB | HBase | Cassandra	Document | Column | Column
Twitter	Cassandra	Column

NoSQL Databases (vs. SQL Databases)

- It depends on the application requirements
  - Size of data
  - Complexity
  - Format of data



NoSQL Databases: Security

- **Most of NoSQL databases don't provide any feature of embedding security in the database itself.**
  - **Developers need to impose security in the middleware.**

- **Security may be difficult**
  - Owing to the unstructured (dynamic) nature of the data stored in these databases
  - Distributed environment
  - Cost of security in contrast to prformance
  - No strong consistency

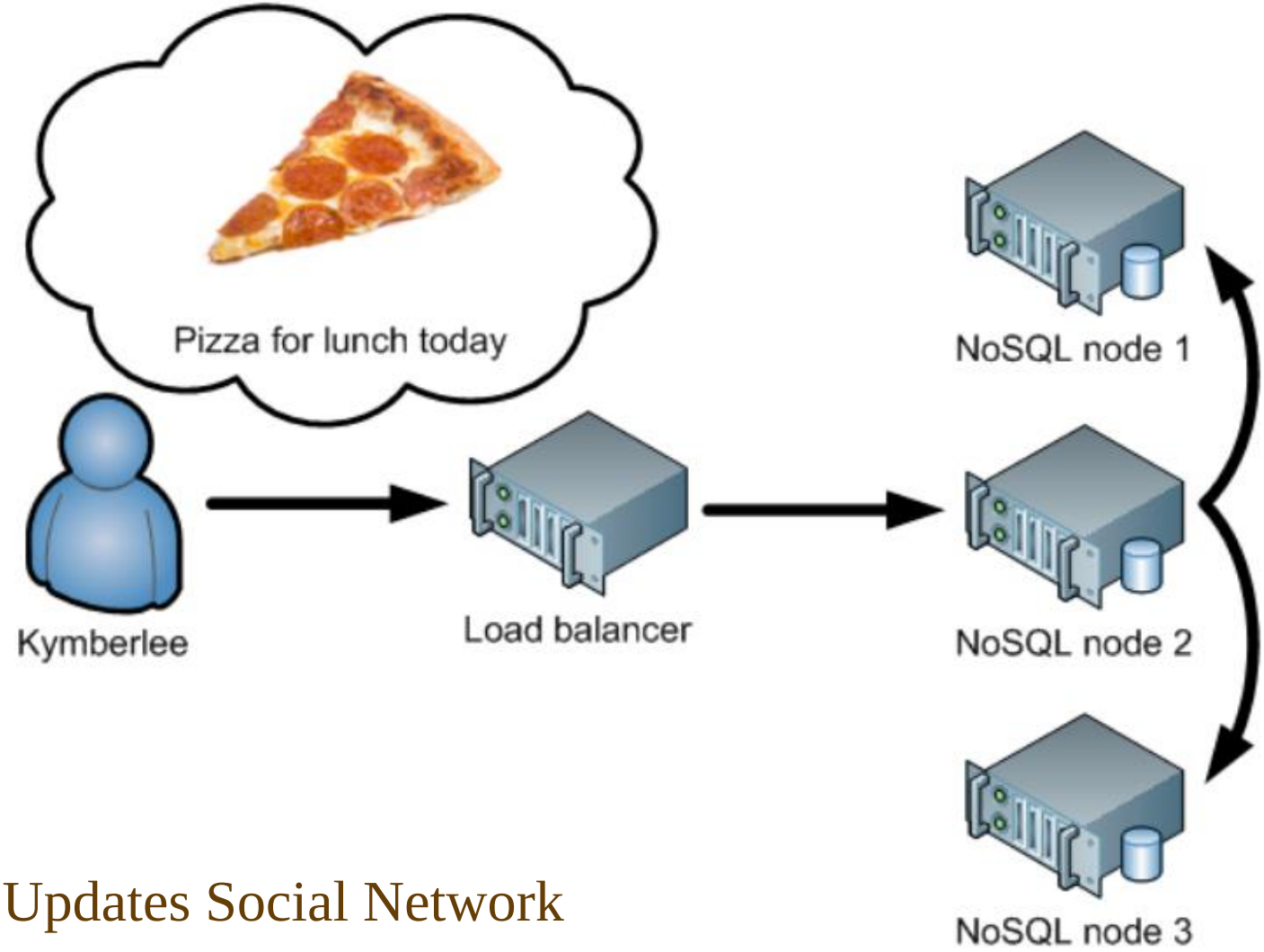
NoSQL Databases: Security

- Security Involves
  - Controlling threats posed by distributed environments
  - Authorization and access control
  - Safeguarding integrity
  - Protection of data at rest
  - User data privacy

NoSQL Databases: Security

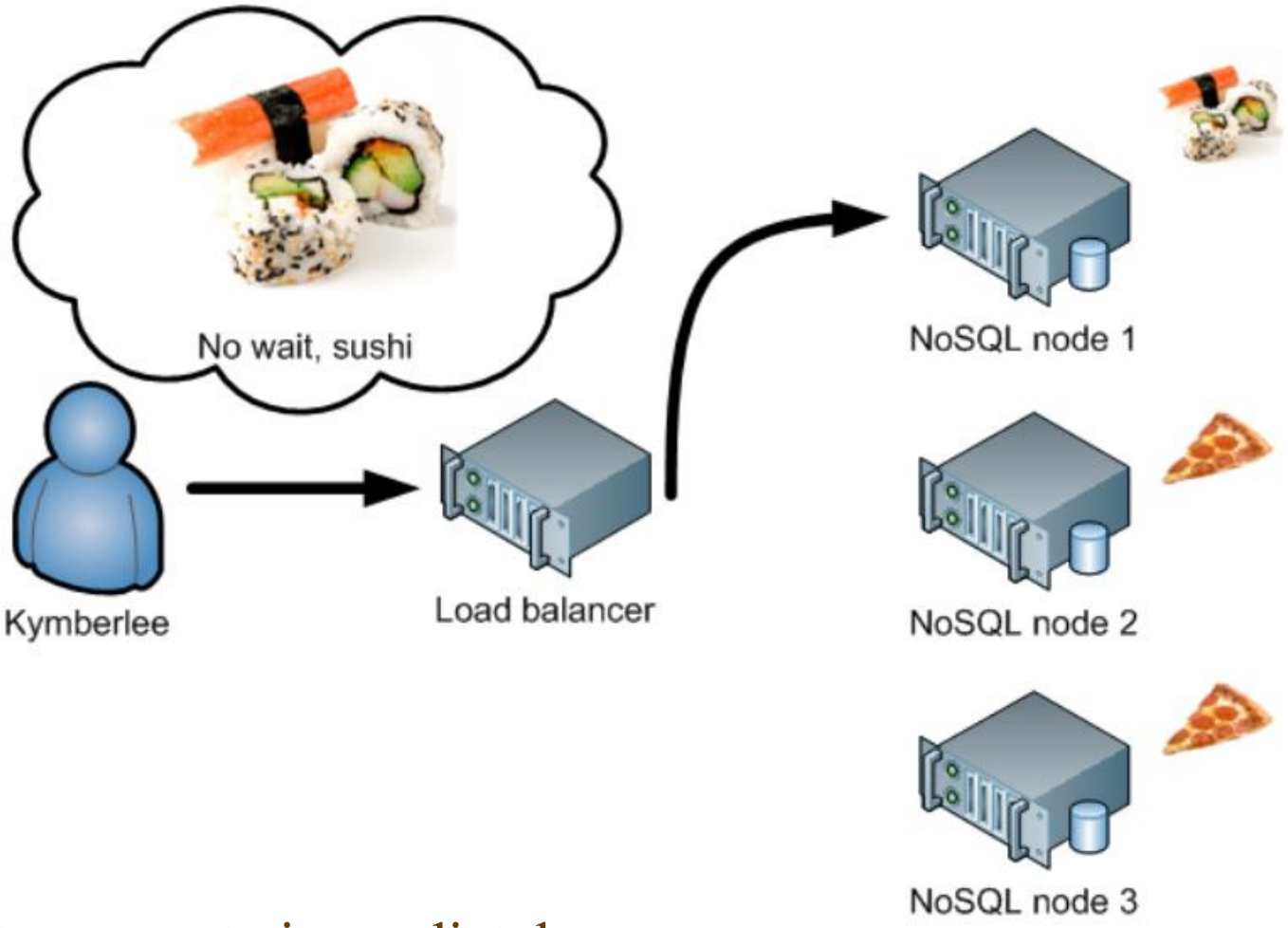
- **Security Involves**
  - **Authentication (Users and Clients)**
    - Cassandra & MongoDB: By default, no support
  - **Audit And Logging**
    - Poor logging and log analysis methods
  - **Protection of Data at Motion**
    - Inter-Node Communications: by default, is not encrypted.
    - SSL can be configured.
  - **API Security**
    - APIs can be subjected to several attacks such as Code injection, buffer over flows, command injection as they access the NoSQL databases.

NoSQL Databases



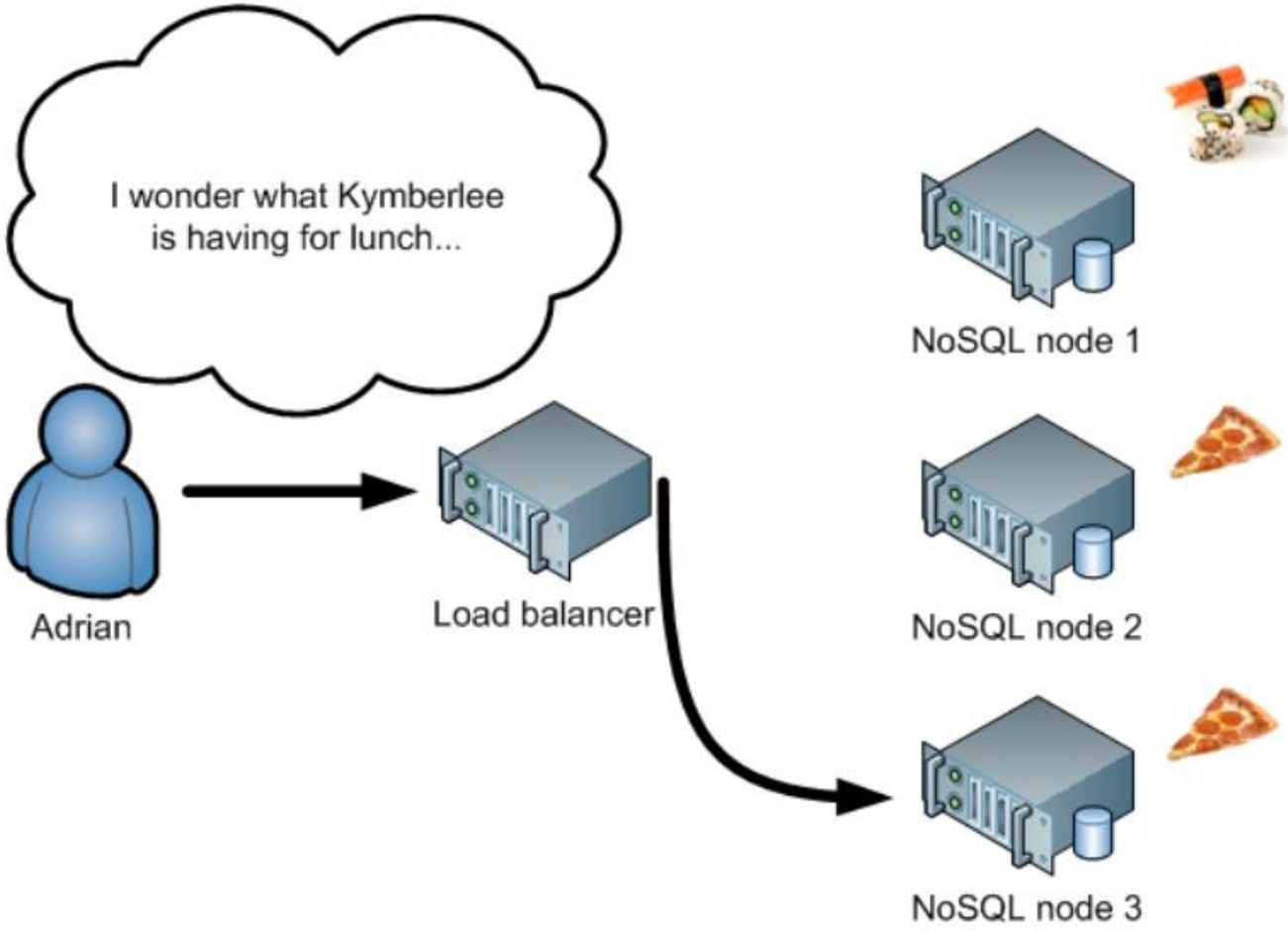
User Updates Social Network

NoSQL Databases: Security



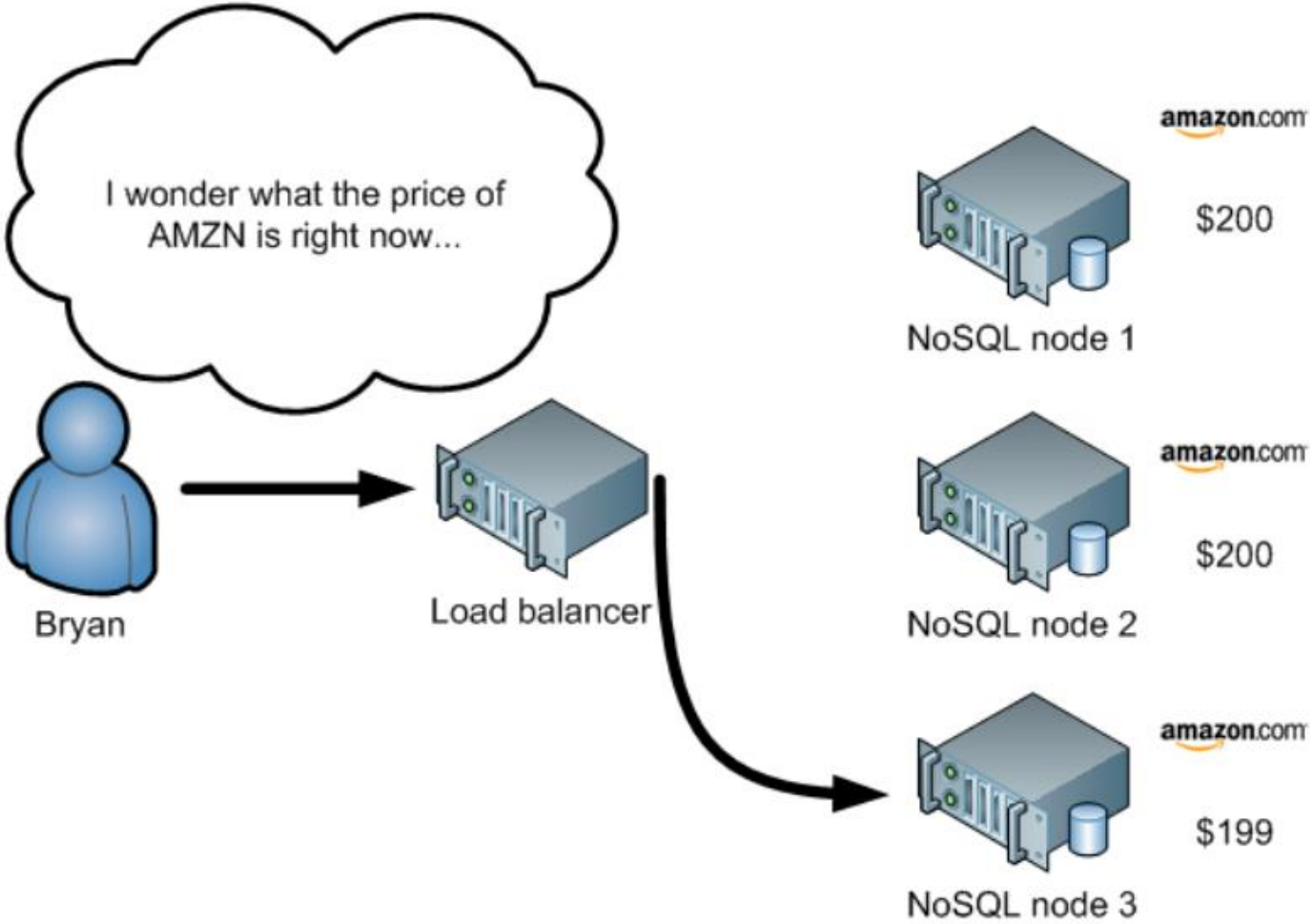
Writes don't propagate immediately

NoSQL Databases: Security



Reading Stale Data

NoSQL Databases: Security



A more serious example

NoSQL Databases: Security

- NoSQL Security Vulnerabilities
  - Little to no Authentication
  - NoSQL Injection

NoSQL Databases: Security

- MongoDB Example
  - **http://example.com/login.php?username=admin&passwd=mysuperpassword**

```
$collection->find(array(
"username" => $_GET['username'],
"passwd" => $_GET['passwd']
));
```

```
$collection->find(array(
"username"=>'admin',
"passwd" => 'mysuperpassword'
));
```

NoSQL Databases: Security

- MongoDB Example
  - **http://example.com/login.php?username=admin&passwd=mysuperpassword**

```
$collection->find(array(
"username" => $_GET['username'],
"passwd" => $_GET['passwd']
));
```

```
$collection -> find (array
("username" => "admin" ,"passwd" => array
( "$ne" => 1 )) );
```

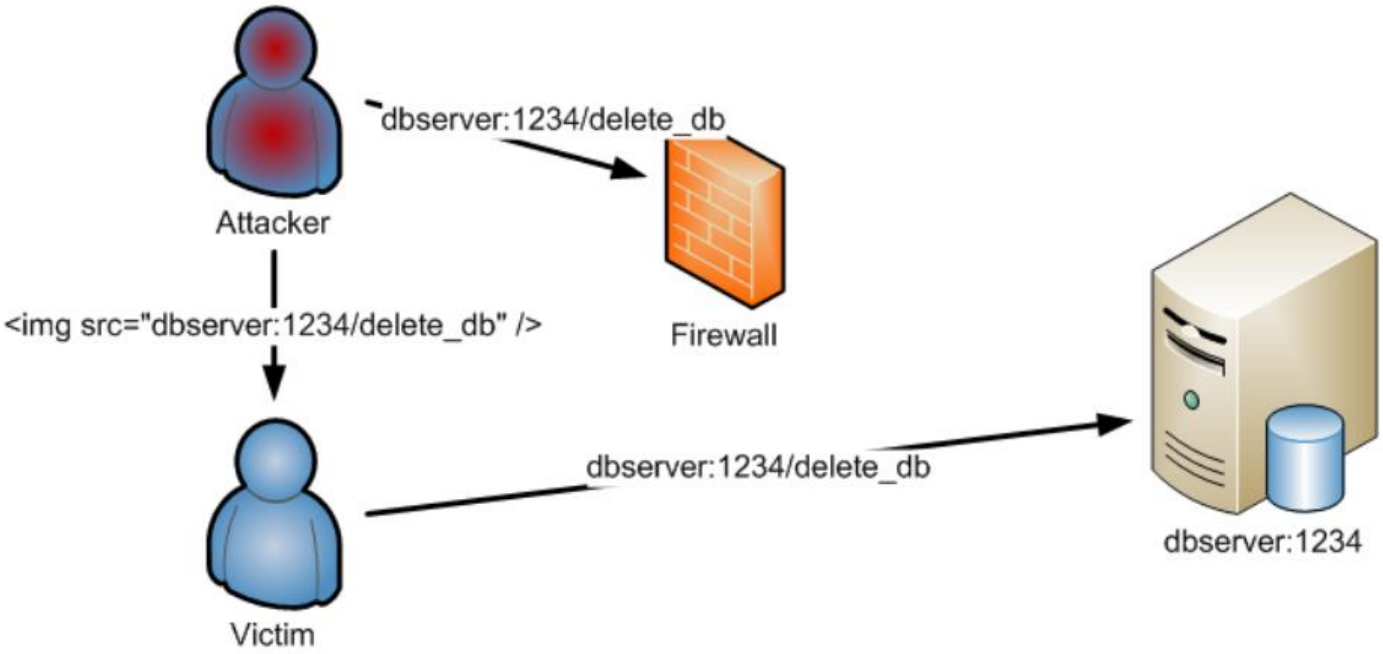
NoSQL Databases: Security

- Server Side Javascript Injection

```
var http = require('http'); #Get HTTP Lib
http.createServer(function (request, response) { #Create HTTP Server Function
    if (request.method === 'POST') { #Await Post
        var data = ''; #Create holder for data
        request.addListener('data', function (chunk) { #Add a Listener for data
            data += chunk; #Add Data to the variable
        });
        request.addListener('end', function () { #Add listener for end of request
            var stockQuery = eval("(" + data + ")"); #Build our Stock Query
            getStock(stockQuery.symbol); #Check Stock
        });
```

NoSQL Databases: Security

- CSFR can be used to bypass firewalls



Cross-Site Request Forgery

NoSQL Databases: Security

- POST is all an Attacker needs
  - Inserting Data

Inserting Script Data

- Execute any REST command from inside the firewall

NoSQL Databases: Security

- Securing NoSQL
  - **Sanitize Inputs**
    - Don't trust users (or other systems!)
  - **Be in control of your query building**

We Are Going to Learn

- Database Security
- Database access controls (DAC, MAC, RBAC, Clark-Wilson)
- Relational Databases
- No SQL Databases
- **Object Based vs. Object Oriented**
- Overview of Database Vulnerabilities
- Overview of Database topics/issues (indexing, inference, aggregation, polyinstantiation)
- Hashing and Encryption

Object based vs Object Oriented Database

- Object based Database Security
  - Object-based security applies to individual instances of entities and is provided by using access rights.
  - An access right is granted to a user for a particular entity instance.

Object based vs Object Oriented Database

- Object Oriented Database Security
  - **Object and object identifier**
    - Objects have data attributes and operations on the object.
    - Class
      - Abstraction mechanism for objects with the same structure.
    - Complex objects
      - The value of an attribute can itself be an object.

Object based vs Object Oriented Database

- Object Oriented Database Security
  - Database systems has a unique id for each object
  - Simply having a reference to an object implies the right to use it.
  - Security is effected by preventing objects from obtaining references to other objects to which they should not have access.

Object based vs Object Oriented Database

- Security in Object-oriented DBMS
  - Most OODBMS do not provide (discretionary) security controls that are comparable to those of relational DBMS. (exceptions: Orion and Iris)
  - Through "multiple interfaces" one can enforce some kind of view mechanism without compromising performance.
  - Message filters are used
  - Strong authorization base
  - Rules for deriving implicit authorizations

We Are Going to Learn

- Database Security
- Database access controls (DAC, MAC, RBAC, Clark-Wilson)
- Relational Databases
- No SQL Databases
- Object Based vs. Object Oriented
- **Overview of Database Vulnerabilities**
- Overview of Database topics/issues (indexing, inference, aggregation, polyinstantiation)
- Hashing and Encryption

Overview of Database Vulnerabilities

- The top ten most common database security vulnerabilities
  1. Deployment Failures
  2. Broken databases
  3. Data leaks
  4. Stolen database backups
  5. The abuse of database features
  6. A lack of segregation
  7. Hopscotch
  8. SQL injections
  9. Sub-standard key management
  10. Database inconsistencies

Overview of Database Vulnerabilities

- More…
  - Default, blank, and weak username/password
  - Extensive user and group privileges
  - Unnecessarily enabled database features
  - Broken configuration management
  - Buffer overflows
  - Denial-of-service attack
  - Unencrypted sensitive data at rest and in motion

We Are Going to Learn

- Database Security
- Database access controls (DAC, MAC, RBAC, Clark-Wilson)
- Relational Databases
- No SQL Databases
- Object Based vs. Object Oriented
- Overview of Database Vulnerabilities
- **Overview of Database topics/issues (indexing, inference, aggregation, polyinstantiation)**
- Hashing and Encryption

More Database topics/issues

- Indexing
- Inference
- Aggregation
- Polyinstantiation

More Database topics/issues: Indexing

- Extract specific information from data and access data through it
- Indexing based on
  - Primary key: single attribute, no duplicates
  - Secondary keys: one or more attributes

More Database topics/issues: Indexing

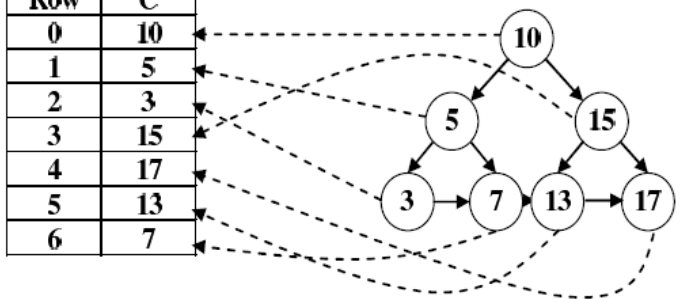
Secure Database Indexing

- **A secure index is a data structure that allows a querier with a ``trapdoor'' for a word**

a) Table T before Encryption

Row	C
0	10
1	5
2	3
3	15
4	17
5	13
6	7

b) Index before Encryption



c) Encryption of Table T in [4]

Row	C
0	$E_k(10 \oplus \mu(T,0,C))$
1	$E_k(5 \oplus \mu(T,1,C))$
2	$E_k(3 \oplus \mu(T,2,C))$
3	$E_k(15 \oplus \mu(T,3,C))$
4	$E_k(17 \oplus \mu(T,4,C))$
5	$E_k(13 \oplus \mu(T,5,C))$
6	$E_k(7 \oplus \mu(T,6,C))$

d) Encryption of the Index in [4]

Row	Struc.	Data
0	1,2	$E_k(10 \| 0)$
1	3,4	$E_k(5 \| 1)$
2	5,6	$E_k(15 \| 3)$
3	6	$E_k(3 \| 2)$
4	5	$E_k(7 \| 6)$
5	4	$E_k(13 \| 5)$
6	-	$E_k(17 \| 4)$

More Database topics/issues: Inference

- **Statistical Inference Theory**
  - **Given unlimited number of statistics and correct statistical answers, all statistical databases can be compromised**

    [Ullman]

  - **Fortunately:**
    - Number of statistics can be limited by statistical DB controls
    - Statistical DB can give approximate rather than 'correct' statistical answers

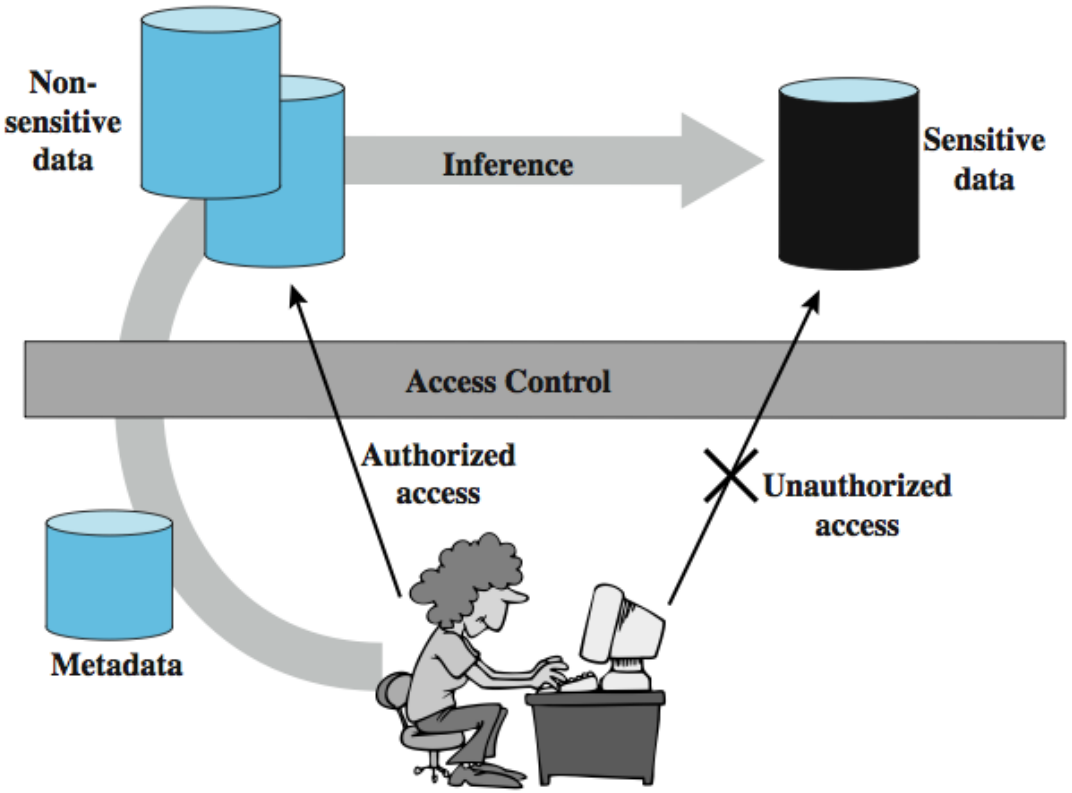
More Database topics/issues: Inference

- Inference via Queries Based on Sensitive Data
  - Sensitive info (salary) used in selection condition, but not returned to the user
  - Returns only Name to user
  - "Infers" (quite mechanically – no intelligence needed) salary for everybody making between $25,000 and $110,000
- Inference via DB Constraints
  - Integrity constraints
  - DB dependencies
  - Key integrity

- Protection: apply query of database views at different security levels

More Database topics/issues: Inference

- Inference

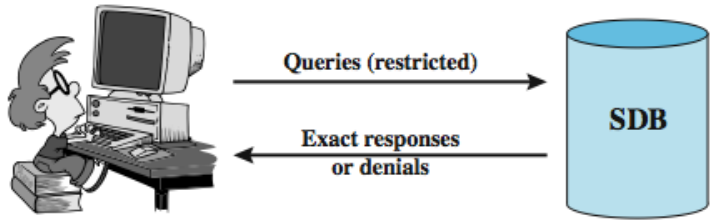


More Database topics/issues: Inference

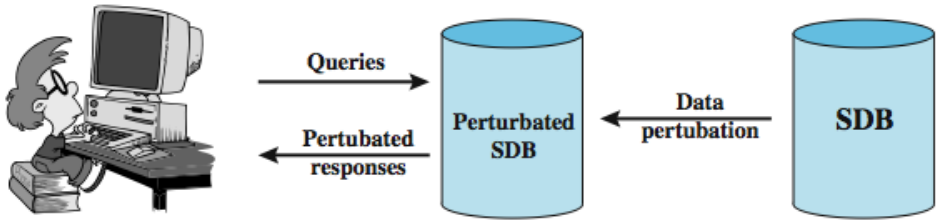
- Inference Countermeasures
  - **Inference detection at database design**
    - Alter database structure or access controls
  - **Inference detection at query time**
    - By monitoring and altering or rejecting queries
  - **Need some inference detection algorithm**
    - A difficult problem
    - Much current research on this aspect of security

More Database topics/issues: Inference

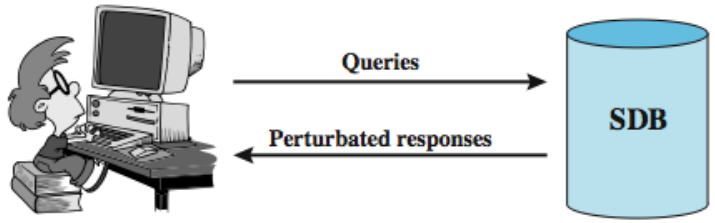
- Protecting Against Inference



(a) Query set restriction



(b) Data perturbation



(c) Output perturbation

More Database topics/issues: Aggregation

- Aggregation
  - Max, Min
  - Count, Sum
  - Average
  - Median

More Database topics/issues: Aggregation

- Network attacks
  - Eavesdropping
  - DoS
  - Replay
  - Artificial data insertion (Stealthy Attack)
  - Intruder Nodes
- Physical Attacks
  - Tampering
  - Physical compromise of nodes

More Database topics/issues: Aggregation

- Secure Database Aggregation
  - **Security needed to transfer data reliably from the sensor to the base station.**
  - **With aggregation intermediate nodes require access to the data for the aggregation.**
    - This introduces a need to determine if the data received from aggregators is reliable.
  - **Cannot bootstrap all keys to device as applications require a dynamic structure.**

More Database topics/issues: Polyinstantiation

- **Polyinstantiation**
  - **Polyinstantiation is a database technique that allows the database to contain multiple instances of the same data but with different classifications**
    - For example, it allows a relation to contain multiple rows with the same primary key; the multiple instances are distinguished by their security levels.
  - A single object may have attributes with different values at different security levels.
  - OODBMS do not support such objects.

More Database topics/issues: Polyinstantiation

- Polyinstantiation

Name	C_{Name}	Dept#	$C_{Dept\#}$	Salary	$C_{Dept\#}$	TC
Bob	Low	Dept1	Low	100K	Low	Low
Ann	**High**	**Dept2**	**High**	**200K**	**High**	**High**
Sam	Low	Dept1	Low	150K	High	High

INSERT INTO Employee
VALUES (Ann, Dept1, 100k)

We Are Going to Learn

- Database Security
- Database access controls (DAC, MAC, RBAC, Clark-Wilson)
- Relational Databases
- No SQL Databases
- Object Based vs. Object Oriented
- Overview of Database Vulnerabilities
- Overview of Database topics/issues (indexing, inference, aggregation, polyinstantiation)
- **Hashing and Encryption**

Hashing and Encryption

Database Hashing

- **Protect sensitive data such as passwords; however it is also used to improve the efficiency of database reference**
- **Problem arises:**
  - When using hashing for password management in the context of database encryption is the fact that a malicious user could potentially use an Input to Hash table for the specific hashing algorithm that the system uses.
    - Salting: a solution for this issue is to 'salt' the hash.
    - Salting is the process of encrypting more than just the password in a database. The more information that is added to a string that is to be hashed, the more difficult it becomes to collate rainbow tables.

Hashing and Encryption

- Database Hashing
  - **A method to protect sensitive data such as passwords; however it is also used to improve the efficiency of database reference**
  - **Problem arises:**
    - Some database incorporate a "pepper" in addition to salts in their hashing systems.
      - A pepper is a value that is added to a hashed password that has been salted.
      - This pepper is often unique to one website or service, and the same pepper is usually added to all passwords saved in a database.
      - In theory the inclusion of peppers in password hashing systems has the potential to decrease the risk of database (Input : Hash) tables

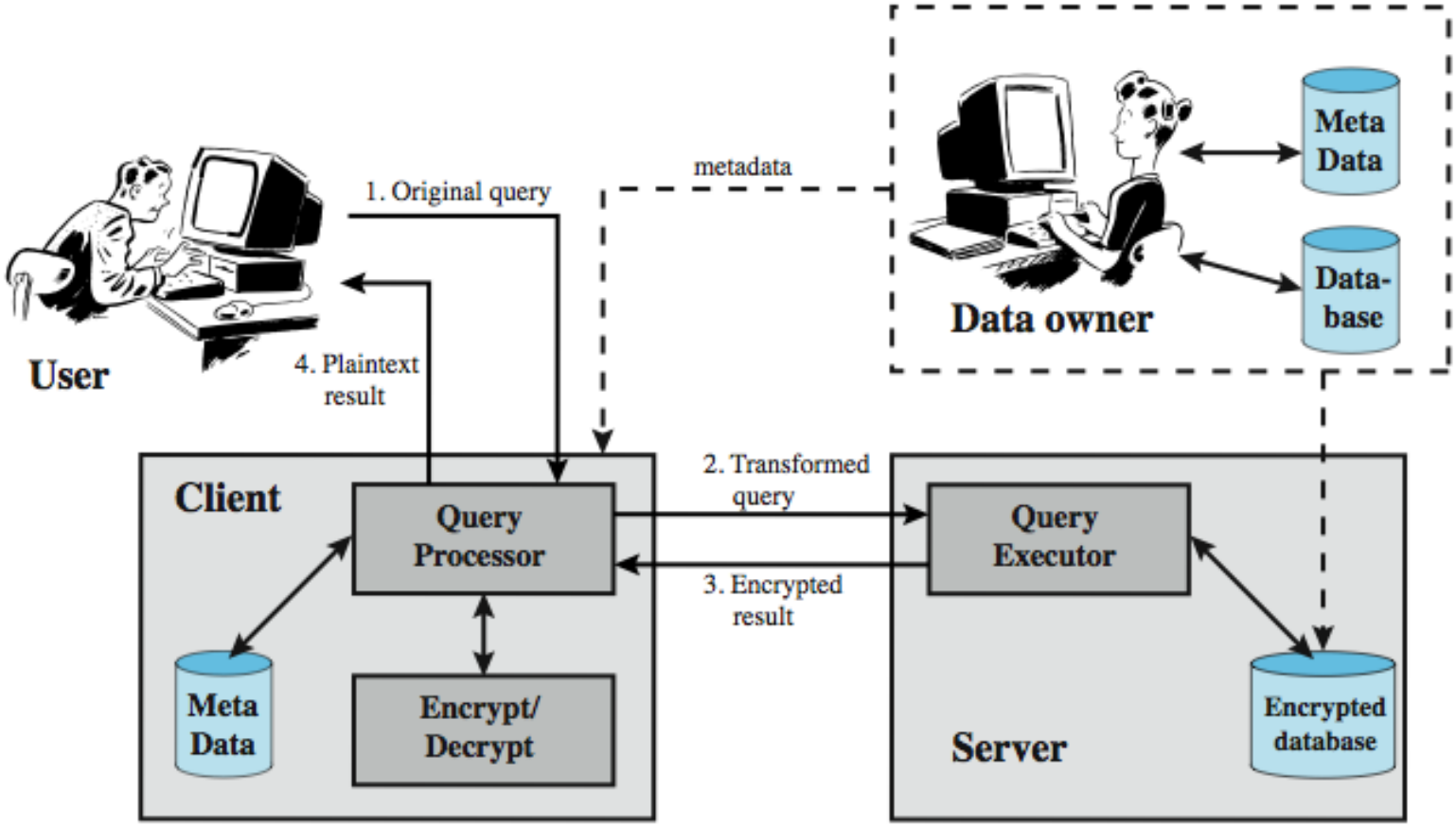
Hashing and Encryption

Database Encryption

- **Databases typical a valuable info resource**
  - Protected by multiple layers of security: firewalls, authentication, O/S access control systems, DB access control systems, and database encryption
- **Database encryption**
  - Protect data from compromise and abuse.
  - 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

Hashing and Encryption

- Database Encryption



Next Class

- Topics
  - Project Proposal
- Assignment 1
- Review Quiz