CS306: Introduction to IT Security

Fall 2018

Lecture 12: Cloud & Database Security

Instructor: **Nikos Triandopoulos**

November 27, 2018

STEVENS
INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY®

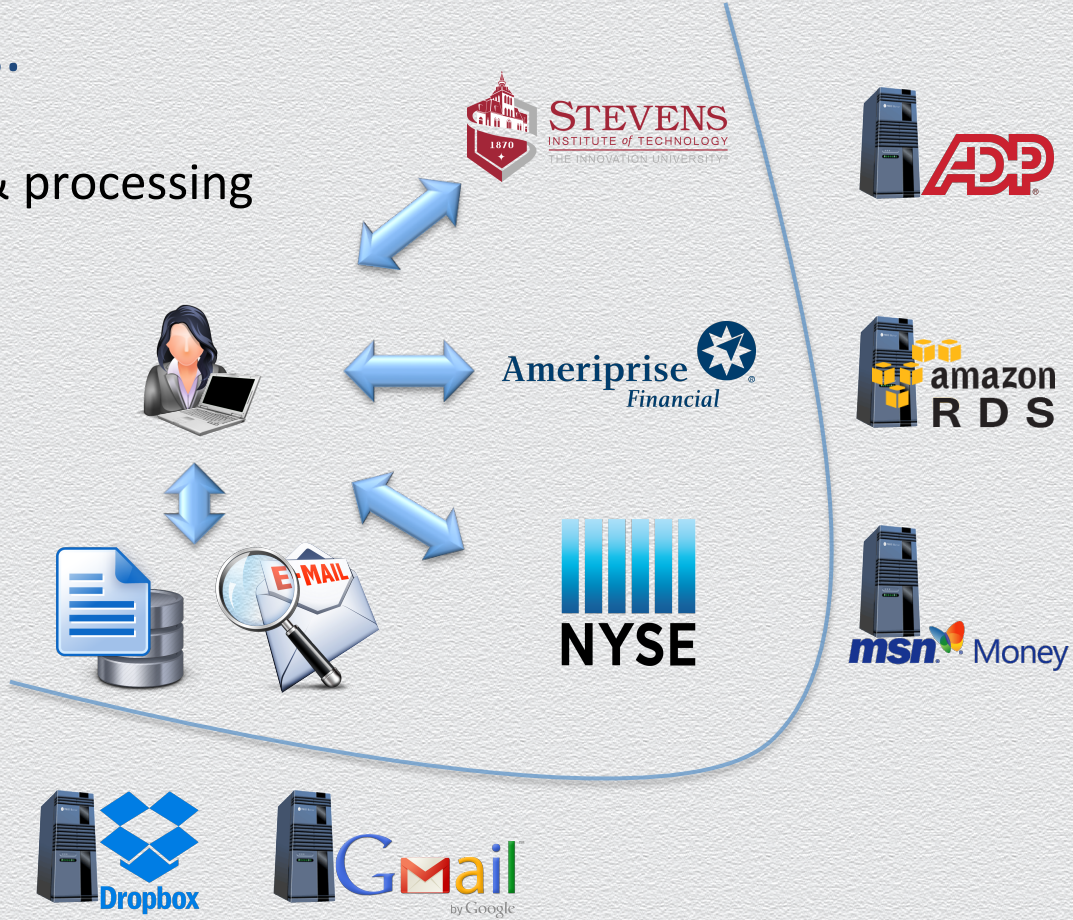
Cloud security

Tax return preparation…

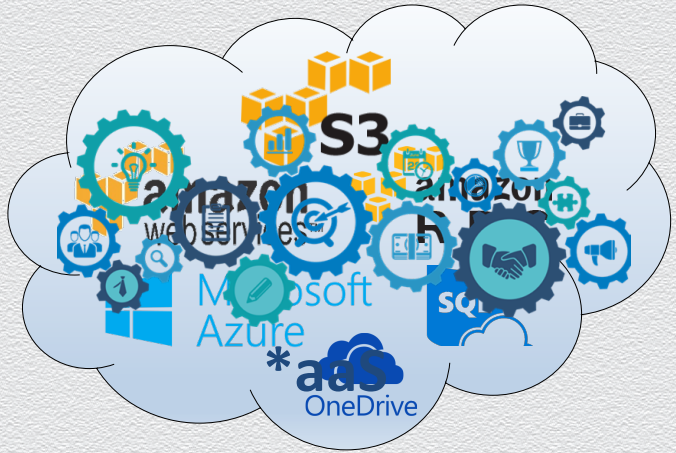
Involves information collection & processing

- calculate financial data
- manage personal data
- submit – done!

… by many different unknown machines!



The era of outsourcing



Cloud-based services

- storage, computation, databases, analytics, …

Transformative multi-platform technology

- businesses, organizations or individuals

Internet protocols

social networks

big-data analytics

sharing economy

FinTech

What is cloud computing?

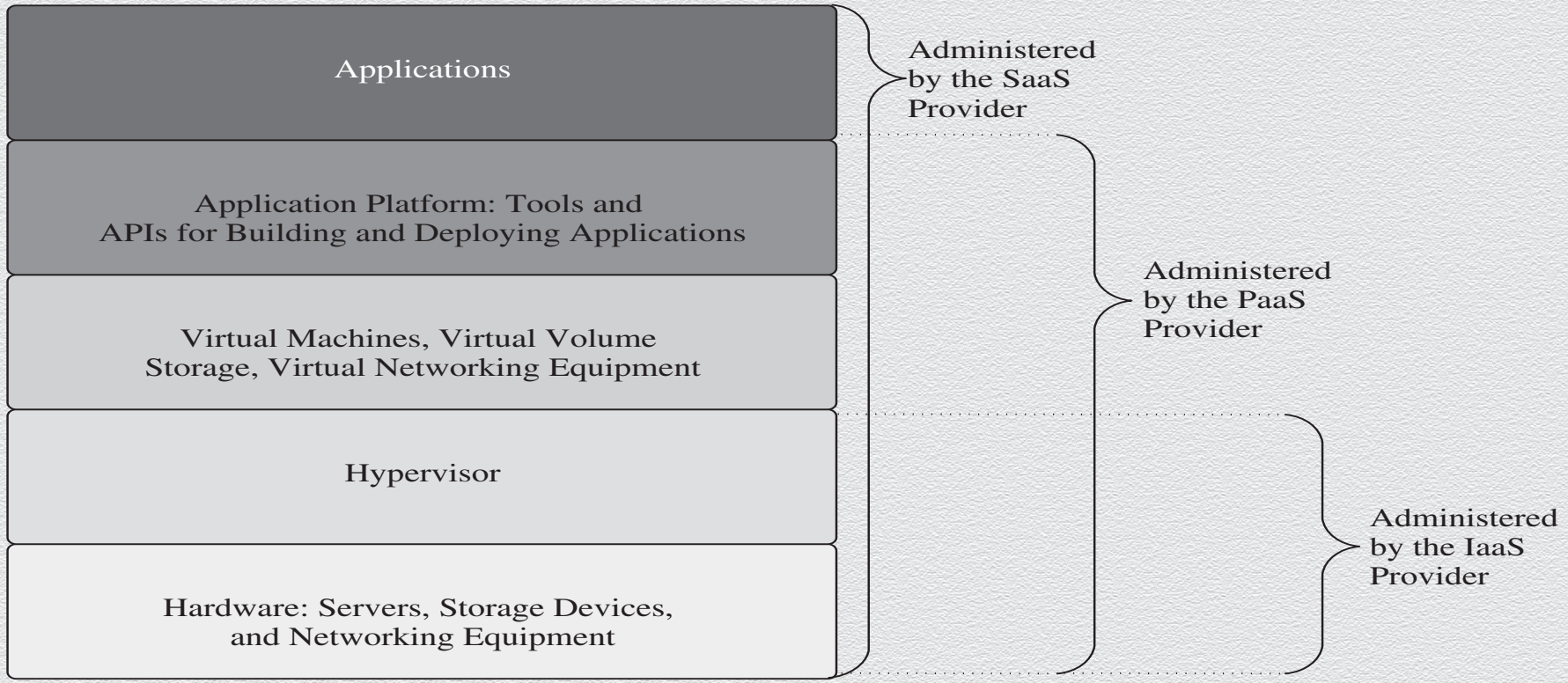
- On-demand self-service
  - Add or subtract resources as necessary
- Broad network access
  - Mobile, desktop, mainframe
- Resource pooling
  - Multiple tenants share resources that can be reassigned dynamically according to need and invisibly to the tenants
- Rapid elasticity
  - Services can quickly and automatically scale up or down to meet customer need
- Measure service
  - Like water, gas, or telephone service, usage can be monitored for billing

Cloud-computing service models

On-demand self-service computing

- Software as a service (SaaS)
  - the cloud provider gives the customer access to applications running in the cloud
- Platform as a service (PaaS)
  - the customer has his or her own applications,
    but the cloud provides the languages and tools for creating and running them
- Infrastructure as a service (IaaS)
  - the cloud provider offers processing, storage, networks, and other computing resources that enable customers to run any kind of software

Service models



Deployment models

- Private cloud
  - Infrastructure that is operated exclusively by and for the organization that owns it
- Community cloud
  - Shared by several organizations with common needs, interests, or goals
- Public cloud
  - Owned by a cloud service provider and offered to the general public
- Hybrid cloud
  - Composed of two or more types of clouds, connected by technology that enables data and applications to balance loads among those clouds

Cloud provider assessment

- Security issues to consider
  - Authentication, authorization, and access control options
  - Encryption options
  - Audit logging capabilities
  - Incident response capabilities
  - Reliability and uptime

Security benefits of cloud services

- Geographic diversity
  - many cloud providers run data centers in disparate geographic locations and mirror data across locations, providing protection from natural and other local disasters
- Platform & infrastructure diversity
  - different platforms and infrastructures mean different bugs and vulnerabilities, which makes a single attack or error less likely to bring a system down
  - using cloud services as part of a larger system can be a good way to diversify your technology stack
  - e.g., Honeywords, virtualization, etc.

Cloud-based security functions

Some security functions may be best handled by cloud service providers

- Email filtering
  - since email is already hopping through a variety of SMTP servers, adding a cloud-based email filter is as simple as adding another hop
- DDoS protection
  - cloud-based DDoS protection services update your DNS records to insert their servers as proxies in front of yours
  - they maintain sufficient bandwidth to handle the flood of attack traffic
- Network monitoring
  - cloud-based solutions can help customers deal with steep hardware requirements and can provide monitoring and incident response expertise

Switching cloud providers

- Switching cloud providers is expensive and difficult but sometimes becomes necessary and urgent
- It is best to have backup options in place in case a migration away from a cloud provider is necessary, but many cloud providers make that practically impossible
- SaaS providers are generally hardest to migrate away from, followed by PaaS, then IaaS

Database security

Database (DB)

Organized collection of structured data

- high-level data representation
  - relationships among data elements
  - semantics and logical interpretation
- set of rules for fine-grained data management
  - data retrieval and analysis
  - selective & user-specific data access

cf. unstructured/"flat"

- low-level representation
  - e.g., file
- coarse-grained
  - e.g., name, location
  - e.g., size, format

Database management system (DBMS)

System through which users interact with a database

- provides **data-management** functions
- **data definition**
  - creation, modification and removal of data relationships and organization specs
- **update**
  - insertion, modification, and deletion of the actual data
- **retrieval**
  - derivation and presentation of information in forms directly usable by apps
- **administration**
  - definition and enforcement of rules related to reliable data management
    - e.g., user registration, performance monitoring, concurrency control, data recovery

Relational databases

Predominant model for databases

- **collection** of records and **relations** among them
- **record/tuple**
  - one related group of data elements (representing specific entities)
  - e.g., a student, department, customer or product record
- **attributes/fields/elements**
  - elementary data items (related to entities)
  - e.g., name, ID, major, GPA, address, city, school, …
- **relations**
  - "inter-connections" of interest among records (e.g., faculty of same department)

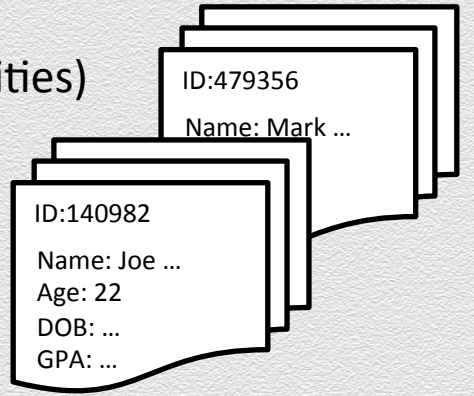


Table representation of relational DBs

Data is organized in tables

- entity-type tables
  - rows are individual records
  - columns are attributes of an entity
- relation-type table
  - rows are "inter-connected" records
  - columns are relevant attributes

a record or row

Table: CS-306 students

First_Name	Last_Name	ID	…
John	Myers	123459	
Maria	Palm	222235	
Alex	Klein	211123	
…..	….	….	

Table: CS-579 students

First_Name	Last_Name	ID	…
John	Myers	123459	
Olga	Johnson	227800	
Alex	Klein	211123	
…..	….	….	

Table: CS-579 & CS-306 students

First_Name	Last_Name	ID	…
John	Myers	123459	
Alex	Klein	211123	

Table representation of relational DBs

Data is organized in tables

- entity-type tables
  - rows are individual records
  - columns are attributes of an entity

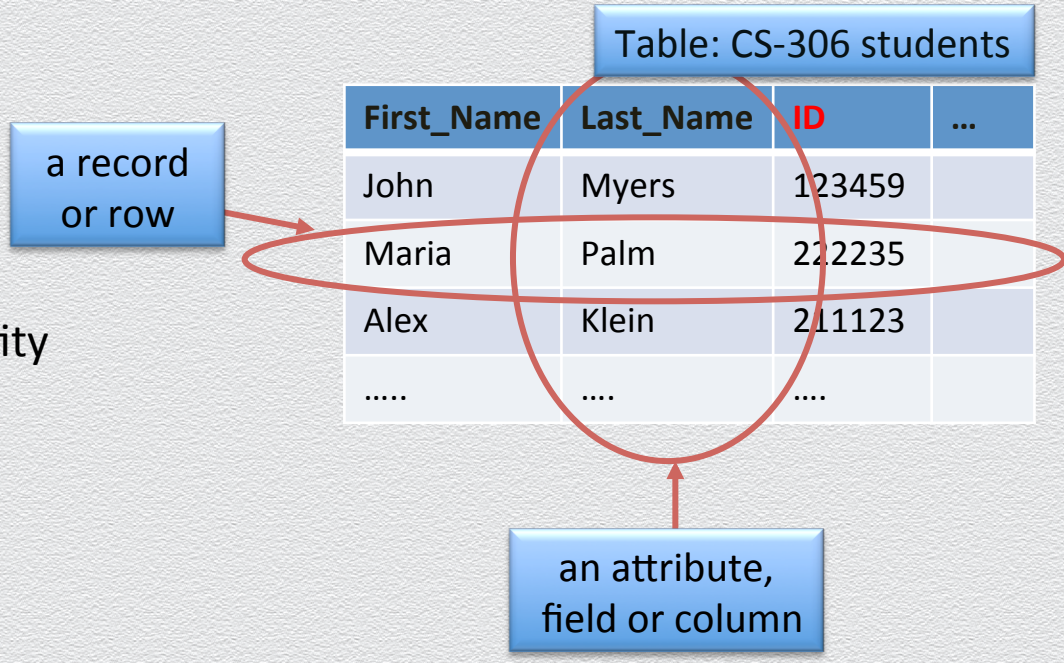


Table representation of relational DBs

Data is organized in tables

- entity-type tables
- relation-type table
  - rows are “inter-connected” records
  - columns are relevant attributes

Table: CS-306 students

First_Name	Last_Name	ID	…
John	Myers	123459	
Maria	Palm	222235	
Alex	Klein	211123	
…..	….	….	

Table: CS-579 students

First_Name	Last_Name	ID	…
John	Myers	123459	
Olga	Johnson	227800	
Alex	Klein	211123	
…..	….	….	

Table: CS-579 & CS-306 students

First_Name	Last_Name	ID	…
John	Myers	123459	
Alex	Klein	211123	
…..	….	….	

A entity-type table example

Table: Home_Address

Name	First	Address	City	State	Zip	Airport
ADAMS	Charles	212 Market St.	Columbus	OH	43210	CMH
ADAMS	Edward	212 Market St.	Columbus	OH	43210	CMH
BENCHLY	Zeke	501 Union St.	Chicago	IL	60603	ORD
CARTER	Marlene	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Beth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Ben	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Lisabeth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Mary	411 Elm St.	Columbus	OH	43210	CMH

More technically…

A **relational database** is a database perceived as a collection of **tables**

- a relation R is a subset of $D_1 \times \cdots \times D_n$
  - $D_1, \ldots, D_n$ are the domains on n attributes
  - elements in the relation are n-tuples $(v_1, \ldots, v_n)$ with $v_i \in D_i$
  - the value of the i-th attribute has to be an element from D_i
  - a special null value indicates that a field does not contain any value

Types of relations

- **Base** (or real) relations
  - named, autonomous relations comprising entity-type tables
  - exist in their own right and have ‘their own’ stored data
- **Views**
  - named, derived relations, defined in terms of other named relations
  - they **do not store** data of their own
- **Snapshots**
  - named, derived relations, defined by other named relations
  - **store** data of their own
- **Query results**
  - may or may not have a name; no persistent existence in the database per se

Database keys

Tuples in a relation must be uniquely identifiable

- **primary keys (PKs)**
  - subset of attributes uniquely identifying records (tuples)
- every relation R must have a primary key K that is
  - **unique**: at any time, no tuples of R have the same value for K
  - **minimal**: no component of K can be omitted without destroying uniqueness
- **foreign keys**
  - a primary key of one relation that is an attribute in some other

Schema of relational DBs

- schema
  - logical structure of a database
- subschema
  - portion of a database
  - e.g., a given user has access to

Table: Cyber Security students

First_Name	Last_Name	ID
…..	….	….

Table: CS-306 students

First_Name	Last_Name	ID	…
…..	….	….	

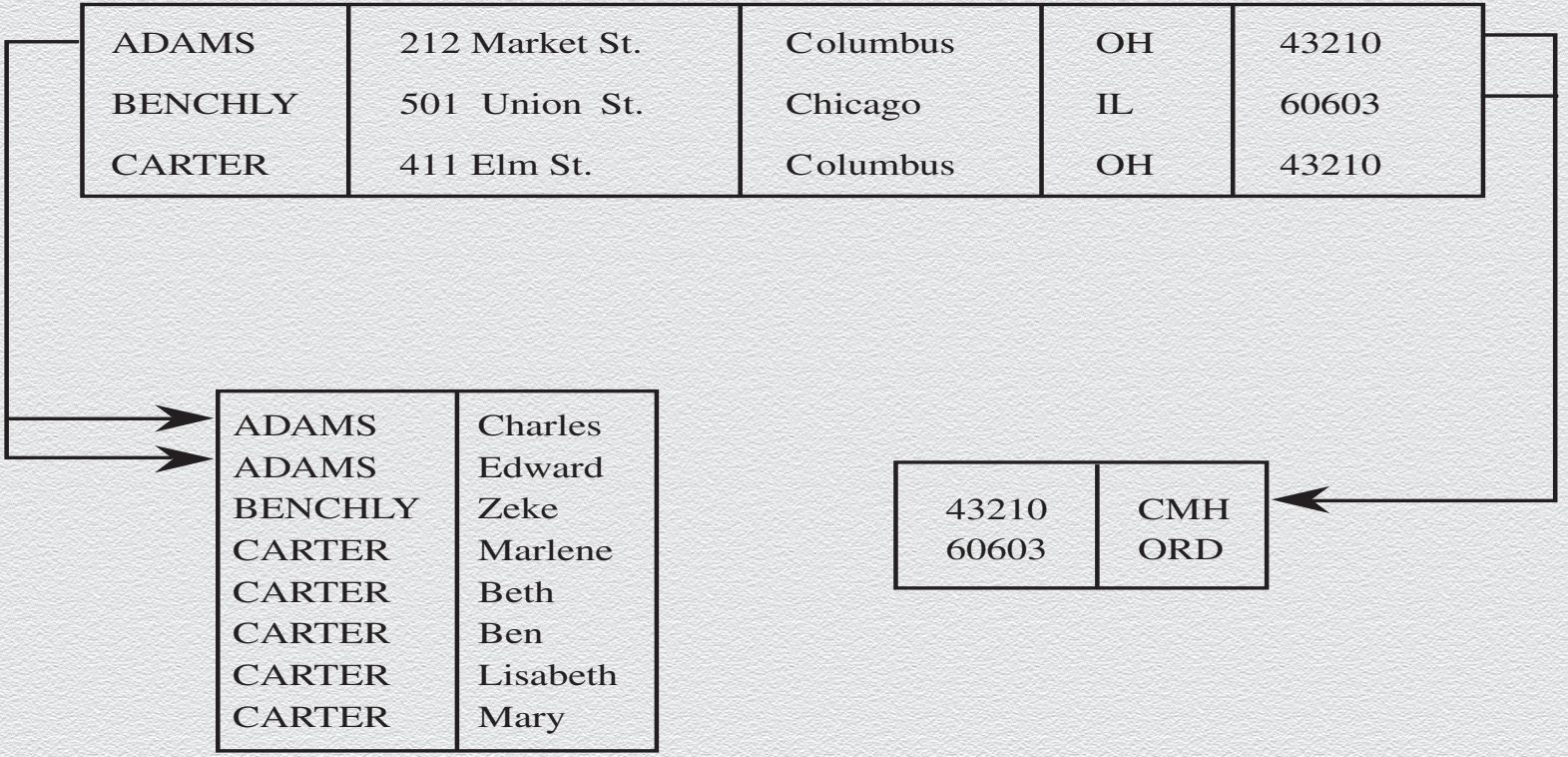
Table: CS-579 students

First_Name	Last_Name	ID	…
…..	….	….	

Table: CS579 & CS-306 students

First_Name	Last_Name	ID	Average Grade	…
…..	….	….		

A database example



Database queries

Commands for accessing databases

- how information in a relational DBs can be retrieved and updated
  - specify how to retrieve, modify, add, or delete fields or records
  - specify how to derive information from database contents

The most common database query language is SQL

- Structured Query Language (SQL)
  - very widely used in practice: successful, solid technology
    - runs in banks, hospitals, governments, businesses, …
    - offered in cloud platforms (e.g., Azure SQL, AWS RDB)

SQL – general features

Rich set of operations

- data manipulation, retrieval, presentation
- nested queries, operators, pattern matching

Main operations

- SELECT: retrieves data from a relation
- UPDATE: update fields in a relation
- DELETE: deletes tuples from a relation
- INSERT: adds tuples to a relation

Example SQL Query

- SELECT *
  FROM HOME_ADDRESS
  ZIP='43210'

Name	First	Address	City	State	Zip	Airport
ADAMS	Charles	212 Market St.	Columbus	OH	43210	CMH
ADAMS	Edward	212 Market St.	Columbus	OH	43210	CMH
CARTER	Marlene	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Beth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Ben	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Lisabeth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Mary	411 Elm St.	Columbus	OH	43210	CMH

Table: Home_Address

Name	First	Address	City	State	Zip	Airport
ADAMS	Charles	212 Market St.	Columbus	OH	43210	CMH
ADAMS	Edward	212 Market St.	Columbus	OH	43210	CMH
BENCHLY	Zeke	501 Union St.	Chicago	IL	60603	ORD
CARTER	Marlene	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Beth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Ben	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Lisabeth	411 Elm St.	Columbus	OH	43210	CMH
CARTER	Mary	411 Elm St.	Columbus	OH	43210	CMH

SELECT operation

```
SELECT [FROM WHERE]
```

- projections, range restrictions, aggregation, etc.
- JOIN sub-query related to set operations

Table: CS-579 students

First_Name	Last_Name	ID	Age	…
John	Myers	123459	20	
Olga	Johnson	227800	21	
Alex	Klein	211123	22	
…..	….	….		

Table: CS-306 students

First_Name	Last_Name	ID	Final_Grade	…
John	Myers	123459	A+	
Maria	Palm	222235	A+	
Alex	Klein	211123	A-	
…..	….	….		

SQL syntax example 1

```
SELECT  First_Name
FROM CS-306
WHERE Final_Grade = A+
```

Table: CS-306 students

First_Name	Last_Name	ID	Final_Grade	…
John	Myers	123459	A+	
Maria	Palm	222235	A+	
Alex	Klein	211123	A-	
…..	….	….		

- SELECT statement
  - used to select data FROM one or more tables in a database
- result-set is stored in a result table
- WHERE clause is used to filter records in terms of attribute contents

SQL syntax example 2

```
SELECT  Last_Name
FROM CS-579
WHERE age=21
ORDER BY First_Name ASC
LIMIT 3
```

Table: CS-579 students

First_Name	Last_Name	ID	Age	…
John	Myers	123459	20	
Olga	Johnson	227800	21	
Alex	Klein	211123	22	
…..	….	….		

- ORDER BY
  - used to order data following one or more fields (columns)
- LIMIT
  - allows to retrieve just a certain numbers of records (rows)

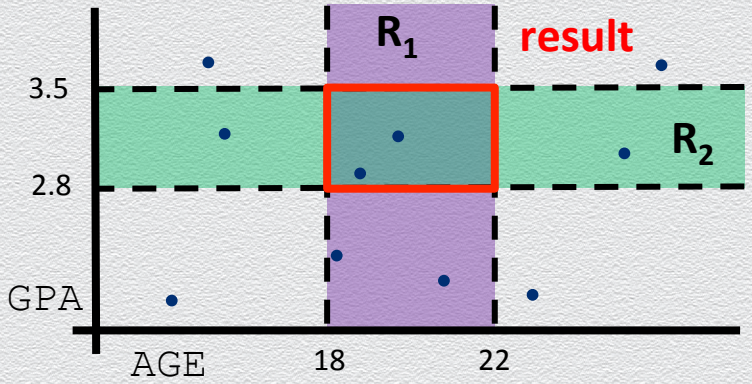
SQL syntax example 3

```
SELECT * FROM STUDENT
   WHERE 18  < AGE < 22
   AND 2.8 < GPA < 3.5
```

Table: CS-579 students

First_Name	Last_Name	ID	Age	GPA	…
John	Myers	123459	20	3.5	
Olga	Johnson	227800	21	4.0	
Alex	Klein	211123	22	2.9	
…..	….	….			

- range searching



intersection of partial results

Database security

- DBs store **data** and provide **information** to their users
- DB security
  - ensure users update or retrieve information in **a reliable and controlled manner**
- **CIA – confidentiality, integrity, availability**
  - protect sensitive data **& disallow unauthorized leakage of information**
  - ensure data integrity **& guarantee correctness/consistency of authorized operations**
  - allow DB access **& ensure authorized access at all times**

Confidentiality & integrity requirements

- **Physical / logical / element integrity**
  - e.g., ensure reliability (i.e., running for long times without interruptions)
  - e.g., protect database as a whole against catastrophic failures
  - e.g., updates do not change the DB schema
  - e.g., elementary data are inserted with correct / accurate values by authorized data “owners”
- **Data / privacy protection**
  - e.g., protect against unauthorized **direct or indirect** disclosure of information
  - e.g., protect against server breaches

Additional DB security requirements

- **Auditability**
  - e.g., DB accessed are recorded and can be traced any time in the future
- **Access control**
  - e.g., different users get different DB views and can update only their “own” data
- **User authentication**
  - e.g., positively identify users (both for auditability and access control)

Database security in the man-machine scale…

Difference to operating-system security

- DB security controls **access to information** more than access to data

Integrity rules

- **entity integrity rule**
  - no PK component of a base relation is allowed to accept nulls
- **referential integrity rule**
  - the database must not contain unmatched foreign key values
- **application specific integrity rules**
  - field checks: correct data entry
  - scope checks: queries over statistical DBs of large support
  - consistency checks: guarantee users get the same DB view

Concurrency via locked query-update cycles

Controls for DB consistency (when multiple users access DB concurrently)

- solves the “double-booking” or “full-flight” problems
- due to concurrent reads & writes
  - e.g., two distinct agencies reserve at the same time the same airplane seat which appears to be empty for a given flight
  - e.g., an agency cancels a previous reservations but another agency cannot reserve it as the flight still appears to be full

Solutions

- treat a (seat availability) query and (seat reservation) update as one **single atomic operation**
- use **locks** to block read (seat availability) requests while a write (seat cancelation) operation is still processed

Consistency via two-phase updates

Control for DB consistency (when failures result in partial data updates)

- solves the “inconsistent inventory” problem

Phase 1: **Intent**

- DBMS does everything it can to **prepare** for the update
  - collects records, opens files, locks out users, makes calculations
  - but it makes **no changes** to the database
- DBMS **commits** by writing a commit flag to the database

Phase 2: **Write**

- DBMS **completes** all update operations and **removes** the commit flag

If either phase fails, it is repeated without causing any harm to the DBMS!

Other DB security mechanisms for integrity

- Error detection and correction codes to protect data integrity
- For recovery purposes, a database can maintain a change log, allowing it to repeat changes as necessary when recovering from failure
- Databases use locks and atomic operations to maintain consistency
  - writes are treated as atomic operations
  - records are locked during write so they cannot be read in a partially updated state

SQL security model for access control

Discretionary access control using privileges and views, based on:

- **users**: authenticated during logon
- **actions**: include SELECT, UPDATE, DELETE, and INSERT
- **objects**: tables, views, columns (attributes) of tables and views

Users invoke actions on objects permitted or denied by DBMS

- when an object is created, it is assigned an owner
- initially only the owner has access to the object
- other users have to be issued with a **privilege**
  - (grantor, grantee, object, action, grantable)

Sensitive data

- Inherently sensitive
  - passwords, locations of weapons
- From a sensitive source
  - confidential informant
- Declared sensitive
  - classified document, name of an anonymous donor
- Part of a sensitive attribute or record
  - salary attribute in an employment database
- Sensitive in relation to previously disclosed information
  - an encrypted file combined with the decryption key to open it

Types of disclosures

- Exact data
  - e.g., finding the exact value of a field
- Bounds
  - e.g., finding a range in which a field value is contained
- Negative result
  - e.g., finding whether one has been convicted 0 times
- Existence
  - e.g., finding whether a person is in a black list
- Probable value
  - e.g., knowing that half of the students have outstanding loans

Means of disclosure

- Direct inference
  - e.g., through a SQL query
- Inference by arithmetic
  - e.g., via computation of sums, counts, means, medians, etc.
  - e.g., via tracker attacks, e.g., count(a & b & c) = count(a) – count(a & ~(b & c))
  - e.g., by solving a linear system
- Aggregation
  - e.g., data mining
  - e.g., by correlating with data from other users, other sources, or prior knowledge
- Hidden data attributes/meta-data
  - e.g., file tags, geo-tags, device tracking / fingerprinting

Disclosure-prevention techniques

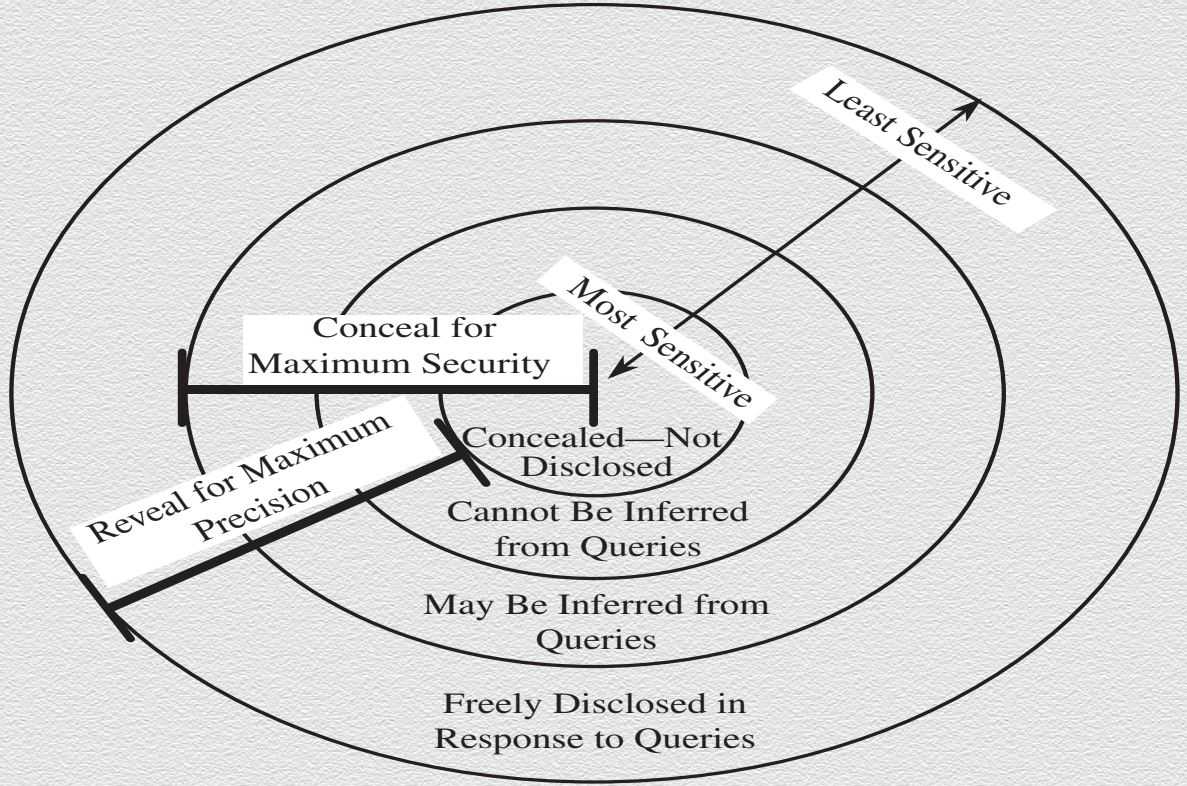
- Suppress obviously sensitive information
  - e.g., never return the SSN number of a customer or the disease of a patient
- Keep track of what each user knows based on past queries, e.g.,
  - use audit logs for the entire query history of a user or a group of users
  - compare new queries against possibly leaked information given past query history
- Disguise the data
  - e.g., perturb data by adding some “zero-mean” random noise
  - e.g., use of differential privacy techniques
- Cryptographically protect database
  - e.g., use of “structured-preserving” encryption

Suppression techniques

- Limited response suppression
  - eliminate certain low-frequency elements from being displayed
- Combined results
  - use ranges, rounding, sums, averages
- Random samples and blocking small sample sizes
- Random data perturbation
  - randomly add/subtract a small error value to/from actual values
- Swapping
  - randomly swap values for individual records while keeping statistical results the same

Security vs. precision

Precise, complete & consistent responses to queries against sensitive information make it more likely that the sensitive information will be disclosed



Cryptographic means

Encrypting data records protects against leakage due to server breaches

- but it reduces utility/usability to zero…

Solution concept: “Compute over encrypted data”

- Multi-party computation
  - parties compute (reliably) only a specific result and nothing not implied by this!
- Fully-homomorphic encryption
  - encryption schemes that allow to compute any function over ciphertext data!
- Structure/Order-preserving encryption
  - encryption schemes that preserver a property over plaintext data (e.g., order)

Take-home messages

Data & privacy protection

- way beyond data record/field suppression (of simple data contents)
  - e.g., keeping data from being dumped out of DB is insufficient to prevent disclosure
- all possible ways of maliciously deducing DB contents must be considered
  - e.g., by taking into account the possible ranges of data fields
  - e.g., by understanding what a priori information potential attackers may possess
- existing disclosure-prevention techniques induce inconvenient trade-offs
  - e.g., between utility and privacy (loss of precision/completeness makes DB unusable)
  - e.g., computing over encrypted data is still impractical

Data mining

- Data mining uses statistics, machine learning, mathematical models, pattern recognition, and other techniques to discover patterns and relations on large datasets
- The size and value of the datasets present an important security and privacy challenge, as the consequences of disclosure are naturally high

Data mining challenges

- Correcting mistakes in data
- Preserving privacy
- Granular access control
- Secure data storage
- Transaction logs
- Real-time security monitoring

SQL injection (or SQLI) attack

- many web applications take user input from a form
- often a user's input is used literally in the construction of a SQL query submitted to a database
  - e.g.,

```
SELECT user FROM table WHERE name = 'user_input';
```

- an SQL injection attack involves placing SQL statements in the user input

Login authentication query

- Standard query to authenticate users
  - select * from users where user='$usern' AND pwd='$password'
- Classic SQL injection attacks
  - Server side code sets variables $username and $passwd from user input to web form
  - Variables passed to SQL query
  - select * from users where user='$username' AND pwd='$passwd'
- Special strings can be entered by attacker
  - select * from users where user='M' OR '1=1' AND pwd='M' OR '1=1'
- Result: access obtained without password
- Solution: Careful with single quote characters
  - filter them out!