Some Reminders for a Seamless Online Class…

- Please turn on your video
- Mute yourself (press and hold spacebar when you'd like to talk)
- Don't do anything you wouldn't do in an in-person class
- I will occasionally check the chat for messages if you'd like to share there instead
- Please say your name before you speak

Recap

- Data-savviness is the future!
- "Classical" relational databases
    - Notion of a DBMS
    - The relational data model and algebra: bags and sets
    - SQL Queries, Modifications, DDL
    - Database Design
    - Views, constraints, triggers, and indexes
    - Query processing & optimization
    - Transactions
- Non-classical data systems
    - Data preparation:
        - Semi-structured data and document stores
        - Unstructured data and search engines
    - Data Exploration:
        - Cell-structured data and spreadsheets
        - Dataframes and dataframe systems
        - OLAP, summarization, and visual analytics
    - Batch Analytics:
        - Compression and column stores
        - Parallel data processing and map-reduce
        - Streaming, sketching, approximation
    - Special Topics:
        - Graph processing systems
        - **Security and Privacy**

Today's Lecture

- Let's start by talking about database security
  - Access control
  - Authentication
  - SQL injection attacks

Access Control

- Relational databases support the ability to give users certain privileges to do certain types of activities on tables or columns within tables
- DBMS keeps track of which users can do what
- The privileges so granted can be revoked as well

- These privileges can be granted to specific *roles* instead of specific users
  - Roles can be, for example, sales_employee, data scientist, manager, …
  - Syntax for roles or individual users is similar

Access Control Syntax

- GRANT privileges ON object TO users [WITH GRANT OPTION]

- Object: table or view
- Privileges:
    - SELECT
        - The right to read all columns of the object
    - INSERT/UPDATE [(column name)]
        - The right to insert/update rows for the named column names
        - Can omit column name if right is for all columns
    - DELETE
        - The right to delete rows from the object
    - REFERENCES [(column name)]
        - The right to define foreign keys (in other tables) that refer to the specified column of object, or to all columns

Access Control Syntax

- GRANT privileges ON object TO users [WITH GRANT OPTION]

- Object: table or view
- Privileges: SELECT/ INSERT / UPDATE / DELETE / REFERENCES
- GRANT OPTION allows the user to pass the privileges onto other users
- Only the original creator of the object (table or view) has the option of doing CREATE, ALTER, or DROP on the object
    - The creator of a view has automatic SELECT privileges on the view: this is because they had to have SELECT privileges on the underlying tables/views to be able to even define the view
    - And so they have grant option only if they had grant option on the underlying tables/views that the new view was defined
    - Similarly, if the view is updatable, and the user holds INSERT, DELETE, UPDATE on the underlying table the user similarly has same privileges on the view

Let's take an example…

- Sailors (sid, sname, rating, age)
- Boats (bid, bname, color)
- Reserves (sid, bid, day)

- CREATE VIEW ActiveSailors (name, age, day) AS SELECT S.sname, S.age, R.day FROM Sailors AS S, Reserves AS R WHERE S.sid = R.sid AND S.Rating > 6

- A user who can access ActiveSailors but not Sailors or Reserves knows the names of sailors who have reservations, but not the bids of boats reserved

Let's take an example…

- Sailors (sid, sname, rating, age)
- Boats (bid, bname, color)
- Reserves (sid, bid, day)

- Say Tarique created Boats, Sailors, Reserves
- Examples of GRANT commands issued by Tarique:
  - GRANT INSERT, DELETE ON Reserves TO Janice WITH GRANT OPTION
  - GRANT SELECT ON Reserves TO Amy
  - GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION
  - GRANT UPDATE (rating) ON Sailors TO Carlos
  - GRANT REFERENCES (bid) ON Boats TO Bob
- Amy tries to declare the view ActiveSailors via the command:
  - CREATE VIEW ActiveSailors (name, age, day) AS SELECT S.sname, S.age, R.day FROM Sailors AS S, Reserves AS R WHERE S.sid = R.sid AND S.Rating > 6
  - Q: Can Amy do this?
  - Yes. She has the SELECT privileges on underlying relations Sailors and Reserves
  - Q: Can she now give SELECT privileges on ActiveSailors to Bob via:
    - GRANT SELECT ON ActiveSailors TO Bob
  - No. She doesn't have GRANT OPTION on Reserves, and therefore not on ActiveSailors

Let's take an example…

- Sailors (sid, sname, rating, age)
- Boats (bid, bname, color)
- Reserves (sid, bid, day)

- Say Tarique created Boats, Sailors, Reserves
- Examples of GRANT commands issued by Tarique:
  - GRANT INSERT, DELETE ON Reserves TO Janice WITH GRANT OPTION
  - GRANT SELECT ON Reserves TO Amy
  - GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION
  - GRANT UPDATE (rating) ON Sailors TO Carlos
  - GRANT REFERENCES (bid) ON Boats TO Bob
- Amy declares the view ActiveSailors via the command:
  - CREATE VIEW ActiveSailors (name, age, day) AS SELECT S.sname, S.age, R.day FROM Sailors AS S, Reserves AS R WHERE S.sid = R.sid AND S.Rating > 6
- Next Amy declares the view YoungSailors via:
  - CREATE VIEW YoungSailors (sid, age, rating) AS SELECT * FROM Sailors WHERE age<18
- She can then give privileges on the view to others:
  - GRANT SELECT ON YoungSailors TO Ben, Martha
  - Ben and Martha can execute queries on YoungSailors but not on Sailors directly
- Carlos can run the following command:
  - UPDATE Sailors SET rating = 8
  - But cannot run UPDATE Sailors SET rating = rating - 1, since this involves reading it

Revoking Privileges

- Syntax:
  - REVOKE [GRANT OPTION FOR] privileges ON object FROM users {RESTRICT | CASCADE}
  - CASCADE:
    - Withdraw the privileges not just from the specified users, but also all other users who hold these privileges thanks *solely* to the specified users
    - So those users would have go get their privileges "another way"
    - RESTRICT only does so for the specified users

Example of Revocations

- Focusing on Sailors
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Amy)
- REVOKE SELECT ON Sailors FROM Amy CASCADE (Tarique)

- Q: What will happen?
- Both Amy and Bin will lose their privileges on Sailors

Example of Revocations

- Focusing on Sailors; new sequence
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Amy)
- REVOKE SELECT ON Sailors FROM Amy CASCADE (Tarique)

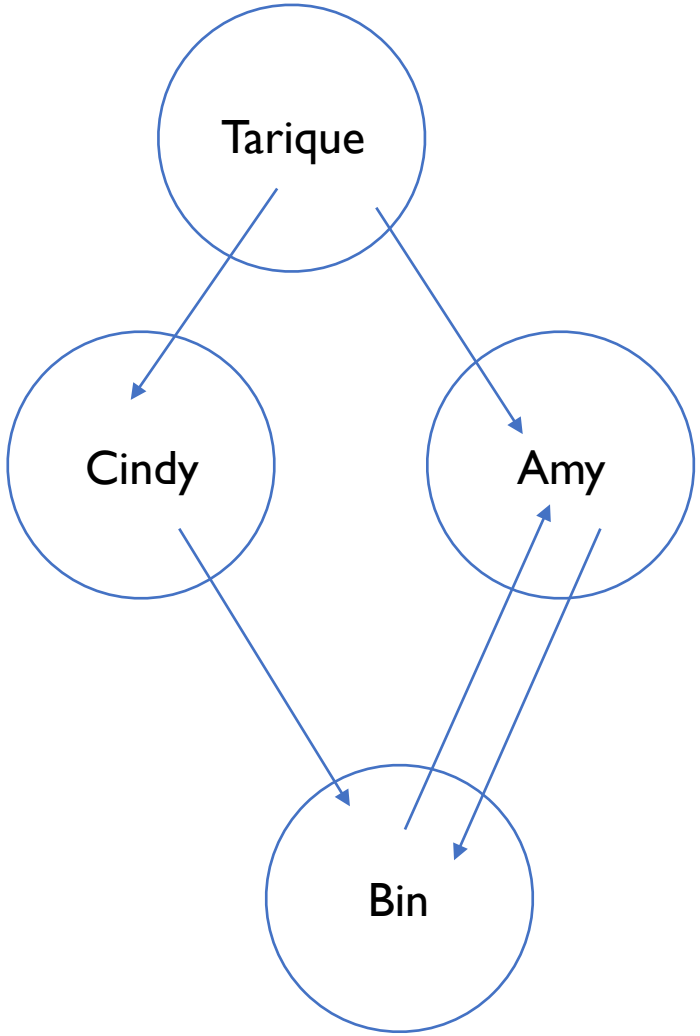
- Q: What will happen?
- Only Amy will lose her privileges

Even more complicated example

- Focusing on Sailors; new sequence
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Amy)
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Bin)
- GRANT SELECT ON Sailors TO Cindy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Cindy)
- REVOKE SELECT ON Sailors FROM Amy CASCADE (Tarique)
- Q: What will happen?
- No real changes: everyone continues to hold the same privileges
- Q: What will happen if Tarique removes the privileges from Cindy as well?
- Everyone loses privileges

Even more complicated example

- Focusing on Sailors; new sequence
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Amy)
- GRANT SELECT ON Sailors TO Amy WITH GRANT OPTION (Bin)
- GRANT SELECT ON Sailors TO Cindy WITH GRANT OPTION (Tarique)
- GRANT SELECT ON Sailors TO Bin WITH GRANT OPTION (Cindy)
- REVOKE SELECT ON Sailors FROM Amy CASCADE (Tarique)
- Q: What will happen?
- No real changes: everyone continues to hold the same privileges
- Q: What will happen if Tarique removes the privileges from Cindy as well?
- Everyone loses privileges



OK, so now what

- We can handle access control via granting and revoking privileges
- Amy may be accessing the database via an internet application. How do we ensure that Amy is not deceived by a scammy website?
- Likewise, how do we ensure that Amy is actually Amy and not Bin?

- Enter authentication. A key ingredient of authentication is encryption.
- We'll cover encryption very briefly…

Encryption/Decryption

- Encryption takes a message and an encryption key, and encrypts the message:
  - encrypt: (message, key) —> encrypted_message
- Decryption takes the encrypted message and a decryption key, and decrypts the message:
  - decrypt: (encrypted_message, key) —> message
- Two types of encryption/decryption:
  - Symmetric: encryption and decryption keys are the same and hidden
    - These schemes are often cheaper
    - AES (Advanced Encryption Standard), DES (Data E. S.) are examples
  - Asymmetric: the keys are different
    - Popular example: public-key encryption
    - These schemes are often more expensive

Public-Key Encryption: Key Ideas

- Each user holds two types of keys:
  - A private key and a public key each: k1 and k2
  - (it doesn't matter which one is which)
  - G(F (message, k1), k2) = message
  - But determining inverses of F or G are impossible
- So how does this work: if Alice wants to send a message m to Bob, then she can simply encrypt the message with Bob's public key k1, knowing that only Bob has the private key k2
  - So, she'll apply F(message, k1) = packet
  - Bob will receive the packet and then apply G(packet, k2) to get back the message
- How can Bob be sure the message is from Alice?
  - Alice herself has a public key k1' and a private key k2'
  - ….