It’s time for another installment of....

Friday Nights With Databases!
Announcements

- HW’s and quizzes:
  - HW #6 (!) underway, deadline next Tuesday (5 pm)
  - Midterm #2, on All Things Query, is next Friday, 3/3
  - Please take the EEE AsterixDB platform survey

- Next week’s meetings
  - Back to discussions & quizzes as usual

- Today’s lecture plan:
  - Finish query section of course
  - Start on indexing & beyond (last section of course)

To Briefly Re-View:

- Uses of views
  - Logical data independence
  - Simplified view of data
  - Unit of authorization

- Views can
  - Rename/permute columns
  - Change units/representations of columns
  - Select/project/join/etc. tables

★ Virtual tables, defined by (SQL) queries
A Simple Example (Re-View)

CREATE VIEW YoungSailorsView (yid, yname, yage, yrating) AS
SELECT sid, sname, age, rating
FROM Sailors
WHERE age < 18;

SELECT * FROM YoungSailorsView;

SELECT yname
FROM YoungSailorsView
WHERE yrating > 5;

Another Example (Re-View)

CREATE VIEW ActiveSailors (sid, sname, rating) AS
SELECT S.sid, S.sname, S.rating
FROM Sailors S WHERE EXISTS
(SELECT * FROM Reserves R WHERE R.sid = S.sid);

SELECT * FROM ActiveSailors;

UPDATE ActiveSailors
SET rating = 11
WHERE sid = 22;
So What About Views & Updates?

Ex:

CREATE VIEW SailsBoats AS
SELECT DISTINCT S.*, B.*
FROM Sailors S, Boats B, Reserves R
WHERE S.sid = R.sid and R.bid = B.bid;

Q: What if we now say...

UPDATE SailsBoats
SET rating = 12
WHERE sid = 22 AND bid = 101;

This view is not updatable since there is no update to the real (stored) tables that would have (just) the asked-for effect – see next slide!!

... Views & Updates (Cont’d.)?

UPDATE SailsBoats SET rating = 12
WHERE sid = 22 AND bid = 101;
... Views & Updates? (Cont’d.)

- A legal update $U$ to view $V$ must be translatable into an equivalent update $U'$ on the underlying table(s) $T$, i.e.:

  
  ![Diagram](Diagram.png)

- If this isn’t possible, the system will reject the update
- Systems differ in how well they do this and err on the conservative side (i.e., declining more view updates)

SQL Access Control

- Based on the concept of access rights or privileges for objects (tables, views, stored procedures, ...) and mechanisms for giving users privileges (and revoking privileges).

- Creator of a database object automatically gets all privileges on it.
  - DBMS keeps track of who subsequently gains and loses privileges, and ensures that only requests from users who have the necessary privileges (at the time the request is issued) are allowed to execute.
GRANT Command

GRANT privileges ON object TO users [WITH GRANT OPTION]

- The following privileges can be specified:
  - SELECT: Can read all columns (including those added later via ALTER TABLE command).
  - INSERT(col-name): Can insert tuples with non-null or non-default values in this column.
    - INSERT means same right with respect to all columns.
  - DELETE: Can delete tuples.
  - REFERENCES (col-name): Can define foreign keys (in other tables) that refer to this column.
- If a user has a privilege with the GRANT OPTION, can pass privilege on to other users (with or without passing on the GRANT OPTION).
- Only the owner can execute CREATE, ALTER, or DROP.

GRANT and REVOKE of Privileges

- GRANT INSERT, SELECT ON Sailors TO Horatio
  - Horatio can query Sailors or insert tuples into it.
- GRANT DELETE ON Sailors TO Yuppy WITH GRANT OPTION
  - Yuppy can delete tuples and can authorize others to do so.
- GRANT UPDATE (rating) ON Sailors TO Dustin
  - Dustin can update (only) the rating field of Sailors tuples.
- GRANT SELECT ON ActiveSailors TO Guppy, Yuppy
  - This does NOT allow the ‘uppies to query Sailors directly!
- REVOKE: When a privilege is revoked from X, it is also revoked from all users who got it solely from X.
**GRANT/REVOKE on Views**

- Great combination to enforce restrictions on data visibility for various users/groups
- If view creator loses the SELECT privilege on an underlying table, the view is dropped!
- If view creator loses a privilege held with the grant option on an underlying table, (s)he loses it on the view as well – and so do users who were granted the privilege on the view!

**Views & Security**

- Views can be used to present just the necessary information (or a summary) while hiding some details of the underlying relation(s):
  - Given *ActiveSailors*, but not Sailors or Reserves, we can find sailors who have a reservation, but not the *bid's* of boats that have been reserved.
- Creator of view has a privilege on the view if (s)he has the privilege on all underlying tables.
- Used together with GRANT/REVOKE commands, views are a very powerful access control tool.
- Stored procedures can be utilized similarly!
SQL Summary (I)

- SQL was a big factor in the early acceptance of the relational model; users found it more natural than earlier, procedural query languages.
- SQL is relationally complete; it has significantly more expressive power than the relational algebra.
- Queries that can be expressed in rel. alg. can often be expressed more naturally in SQL. (Ex: max 😊)
- Many alternative ways to write a query; optimizer will look for the most efficient evaluation plan.
  - In practice, expert users are aware of how queries are optimized and evaluated. (Optimizers are imperfect.)

SQL Summary (II)

- NULL for unknown field values brings many complications (as well as a SQL specification divergence for Oracle w.r.t. VARCHAR data).
- Allows specification of rich integrity constraints (real RDBMSs implement just some of SQL IC spec).
- Triggers can respond to changes in the database (and make up the difference when the set of available integrity features falls short).
- Stored procedures (and CALL) are also available.
- Views and authorization are both useful features, and can be especially powerful in combination. (!)
That’s it for SQL!

- ANY LINGERING QUESTIONS...?

Disks and Files

- DBMSs store data on secondary storage.
- This has major implications for DBMS design!
  - **READ**: xfer data from disk (or flash) to memory (RAM).
  - **WRITE**: xfer data from RAM to disk (or flash).
  - Both are high-cost operations, relative to in-memory operations, so must be considered carefully!
Storage Hierarchy & Latency (Jim Gray): How Far Away is the Data?

- **10^9** Tape/Optical Robot, 2,000 years
- **10^6** Disk, 2 years
- Memory, 1.5 hr
- On Board Cache, 10 min
- On Chip Cache, 10 min
- Registers, 1 min

Why Not Store Data in Main Memory??

- **Costs too much.** Dell was once asking $65 for 500GB of disk, $600 for 256GB of SSD, and $57 for 4GB of RAM → $0.13, $2.34, $14.25 per GB
- **Main memory is volatile.** We want data to be saved in between runs. (Obviously...!!)
- Your typical (basic) storage hierarchy:
  - Main memory (RAM) for currently used data
  - (Flash often sits here now as well...)
  - Disk for the main database (secondary storage)
  - Tapes for archiving the data (tertiary storage)