CS 327E Lecture 11

Shirley Cohen

October 31, 2016

Happy Halloween!
Announcements

• Guest lecture next class
• Heads-up on Lab 3
• Only 4 more quizzes (including today’s)
Homework for Today

• Chapter 14 from the Learning SQL book
• Exercises at the end of Chapter 14
Question 1

What is a database view?

A. A mechanism for caching database files
B. A mechanism for querying database tables
C. A mechanism for doing bulk imports and exports
D. A web interface for running SQL queries
E. None of the above
Question 2

Creating a view is giving a name to a ___ statement:

A. INSERT
B. UPDATE
C. DELETE
D. SELECT
E. CREATE TABLE
Question 3

What is **NOT** a motivation for views?

A. Aggregation: to appear as though data is aggregated
B. Complexity: making multiple tables appear to be a simple table
C. Security: to avoid having to reveal individual data rows
D. Space saving: to reduce the storage of database tables
Question 4

mysql> desc Customer;
+--------------------------+----------+--------+--------------------------------+--------------+---------|
| Field        | Type             | Null | Key | Default | Extra          |
+--------------------------+----------+--------+--------------------------------+--------------+---------|
| cust_id      | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| fed_id       | varchar(12)      | NO   |     | NULL    |                |
| cust_type_cd | enum('I','B')    | NO   |     | NULL    |                |
+--------------------------+----------+--------+--------------------------------+--------------+---------|

Which of these views hides the `fed_id` field from the `Customer` table?

A. CREATE VIEW Customer_VW (cust_id, cust_type_cd) AS
   SELECT cust_id, cust_type_cd
   FROM Customer;

B. CREATE VIEW Customer_VW AS
   SELECT cust_id, cust_type_cd
   FROM Customer;

C. CREATE VIEW Customer_VW (cust_id, cust_type_cd) AS
   SELECT c.cust_id, c.cust_type_cd
   FROM Customer c;

D. CREATE VIEW Customer_VW (cust_num, cust_type) AS
   SELECT cust_id, cust_type_cd
   FROM Customer;

E. All of the above
Is it possible to update the data through a view?

A. No, views are only designed to simplify a \texttt{SELECT} statement
B. No, views are statically-generated tables and do not update
C. Yes, with several restrictions on clauses and functions
D. Yes, all views are updatable and insertable
Views

- Views are like procedures in SQL
- They are defined by a SQL query
- They return a table of results from the SQL query

Example view:

Employee (ssn, first_name, last_name, role, title, salary)

```
CREATE VIEW SeniorStaff AS
  SELECT ssn, first_name, last_name, role, title, salary
  FROM Employee
  WHERE title LIKE 'Senior%'
  ORDER BY salary
```

SeniorStaff(ssn, first_name, last_name, title, salary) = virtual table

We can now use the SeniorStaff view as if it were a table
Concept Question 1

What fields and/or records do the following views hide?

**Employee** \((\text{ssn, first\_name, last\_name, role, title, salary})\)

CREATE VIEW **All\_Employee\_View** AS

SELECT first\_name, last\_name, role, title
FROM Employee
ORDER BY last\_name, first\_name

CREATE VIEW **Manager\_Employee\_View** AS

SELECT ssn, first\_name, last\_name, role, title, salary
FROM Employee
WHERE role <> 'Executive'
ORDER BY last\_name, first\_name

A. SSN and salary details for all employees
B. Salary details for executives
C. All employee records
D. Executive employee records
E. A and D
Demo

See code samples in Github
Query Modification

Orders(order_id, item_id, customer_id, quantity, store)
Items(id, item_name, price)

CREATE VIEW CustomerSales AS
    SELECT o.customer_id, i.price
    FROM Orders o, Items i
    WHERE o.item_id = i.id

CustomerSales(customer_id, price) = virtual table

Query using the view:

SELECT c.customer_id, c.price, o.store
FROM CustomerSales c, Orders o
WHERE c.customer_id = o.customer_id
AND c.price > 100

Question: How will this query be computed?
Query Modification

Using the view:

```sql
SELECT c.customer_id, c.price, o.store
FROM CustomerSales c, Orders o
WHERE c.customer_id = o.customer_id
AND c.price > 100
```

Modified query (at runtime):

```sql
SELECT c.customer_id, c.price, o.store
FROM (SELECT x.customer_id, y.price,
    FROM Orders x, Items y
    WHERE x.item_id = y.id) c, Orders o
WHERE c.customer_id = o.customer_id
AND c.price > 100
```
Query Modification

Rewritten query (at runtime):

```sql
SELECT c.customer_id, c.price, o.store
FROM (SELECT x.customer_id, y.price,
    FROM Orders x, Items y
    WHERE x.item_id = y.id) c, Orders o
WHERE c.customer_id = o.customer_id
AND c.price > 100
```

Flattened query (at runtime):

```sql
SELECT o.customer_id, i.price, o.store
FROM Orders o, Items i
WHERE o.item_id = i.id
AND i.price > 100
```
Concept Question 2

Orders(order_id, item_id, customer_id, quantity, store)
Items(id, item_name, price)

CREATE VIEW CustomerSales AS
  SELECT o.customer_id, o.store, i.price
  FROM Orders o, Items i
  WHERE o.item_id = i.id

CustomerSales(customer_id, store, price) = virtual table

Query using the View:

SELECT customer_id
FROM CustomerSales
WHERE store = 'Texas Union'

Question: Which base table(s) will be used to answer the above query?

A. Only Orders  B. Only Items  C. Orders and Items  D. Orders or Items
Types of Views

- **Virtual views:**
  - computed only on-demand
  - always up-to-date

- **Materialized views:**
  - pre-computed offline
  - requires extra storage
  - may be out-of-date with the base tables
Applications of Views

- **Security**
  - controlled access to fields and records

- **Logical Data Independence**

- **Query Optimizations**
  - vertical partitioning
  - horizontal partitioning
  - materialized views
Vertical Partitioning

Student(eid, first_name, middle_initial, last_name)
Photo(eid, photo, date_taken)

CREATE VIEW StudentsView AS
SELECT s.eid, s.first_name, s.middle_initial, s.last_name, p.photo, p.date_taken
FROM Student s, Photo p
WHERE s.eid = p.eid

Query using the View:
SELECT eid, first_name, middle_initial
FROM StudentsView
WHERE last_name = 'Chen'

Concept Question 2: Which base table(s) will be used to answer this query?

A. Student  B. Photo  C. Student and Photo  D. Student or Photo
Horizontal Partitioning

Student(eid, first_name, middle_initial, last_name)
Photo_2015(eid, photo, date_taken)
Photo_2016(eid, photo, date_taken)

CREATE VIEW StudentPhotosView AS
    SELECT eid, photo, date_taken
    FROM    Photo_2015
    UNION ALL
    SELECT eid, photo, date_taken
    FROM    Photo_2016

Query using the View:

SELECT s.eid, s.first_name, s.middle_initial, s.last_name,
    p.photo, p.date_taken
FROM    Student s, StudentPhotosView p
WHERE s.eid = p.eid
AND     p.date_taken >= '2016-01-01'

Concept Question 3: Which base table(s) will be used to answer this query?

A. Student  B. Photo_2015 and Photo_2016
C. Student and Photo_2015  D. Student and Photo_2016  E. All base tables