PURDUE UNIVERSITY | Department of Computer Science

CS 44800: Introduction To Relational Database Systems

Views and Triggers

Prof. Chris Clifton

30 November 2021

PURDUE UNIVERSITY
Department of Computer Science

View

- Expression that describes a table without creating it
  - Outcome is a named entity that looks and acts like a table
- Suggestions for how to think of this:

VIEW
CREATE ~~TABLE~~ average AS
SELECT assignment, avg(score)
FROM grades
GROUP BY assignment

Student	Assignment	Score
Clifton	2	17
Clifton	3	22
…	…	…

Assignment	Avg(score)
2	17.3
3	24.1
…	…

2

PURDUE UNIVERSITY
Department of Computer Science

Theory behind views

- Every relational query returns a relation
  - Possibly a single row, single column relation
- Query result could be stored in a table
  - Use in future queries
- View: Do this "on the fly"
  - Generate the result *every time the view is used*

PURDUE UNIVERSITY
Department of Computer Science

Using Views

- Access control: Limit who sees data
  - SQL GRANT controls what users can access/modify a table
  - Also works for views (doesn't give access to underlying table)
- Different logical views of the same data
  - Schema migration
- "short cuts"

View Definition

- A view is defined using the **create view** statement which has the form

  create view *v* **as** < query expression >

  where <query expression> is any legal SQL expression. The view name is represented by *v.*
- Once a view is defined, the view name can be used to refer to the virtual relation that the view generates.
- View definition is not the same as creating a new relation by evaluating the query expression
  - Rather, a view definition causes the saving of an expression; the expression is substituted into queries using the view.

View Definition and Use

- A view of instructors without their salary

  create view ***faculty*** **as**
  select *ID*, *name*, *dept_name*
  from *instructor*
- Find all instructors in the Biology department

  select *name*
  from ***faculty***
  where *dept_name* = 'Biology'
- Create a view of department salary totals

  create view ***departments_total_salary(dept_name, total_salary)*** **as**
  select *dept_name*, **sum** (*salary*)
  from *instructor*
  group by *dept_name*;

Views Defined Using Other Views

- One view may be used in the expression defining another view
- A view relation v_1 is said to ***depend directly*** on a view relation v_2 if v_2 is used in the expression defining v_1
- A view relation v_1 is said to ***depend on*** view relation v_2 if either v_1 depends directly to v_2 or there is a path of dependencies from v_1 to v_2
- A view relation v is said to be ***recursive*** if it depends on itself.

Views Defined Using Other Views

- **create view** ***physics_fall_2017*** **as**
  select *course.course_id*, *sec_id*, *building*, *room_number*
  from *course*, *section*
  where *course.course_id = section.course_id*
  and *course.dept_name* = 'Physics'
  and *section.semester* = 'Fall'
  and *section.year* = '2017';

- **create view** ***physics_fall_2017***_*watson* **as**
  select *course_id*, *room_number*
  from ***physics_fall_2017***
  where *building*= 'Watson';

View Expansion

- Expand the view :

```
create view physics_fall_2017_watson as
  select course_id, room_number
  from physics_fall_2017
  where building= 'Watson'
```

- To:

```
create view physics_fall_2017_watson as
    select course_id, room_number
    from (select course.course_id, building, room_number
        from course, section
        where course.course_id = section.course_id
            and course.dept_name = 'Physics'
            and section.semester = 'Fall'
            and section.year = '2017')
     where building= 'Watson';
```


View Expansion (Cont.)

- A way to define the meaning of views defined in terms of other views.
- Let view v_1 be defined by an expression e_1 that may itself contain uses of view relations.
- View expansion of an expression repeats the following replacement step:

  repeat
  Find any view relation v_i in e_1
  Replace the view relation v_i by the expression defining v_i
  until no more view relations are present in e_1

- As long as the view definitions are not recursive, this loop will terminate

PURDUE UNIVERSITY
Department of Computer Science

View Limitations

- Performance
  - Materialized views
- Update
  - Insert
  - Modify
  - Delete
- Solutions to come
  - Triggers

Update of a View

- Add a new tuple to *faculty* view which we defined earlier

  insert into *faculty*

  values ('30765', 'Green', 'Music');
- This insertion must be represented by the insertion into the *instructor* relation
  - Must have a value for salary.
- Two approaches
  - Reject the insert
  - Inset the tuple

    ('30765', 'Green', 'Music', null)

    into the *instructor* relation

Some Updates Cannot be Translated Uniquely

- **create view** *instructor_info* **as**
  select *ID*, *name*, *building*
  from *instructor*, *department*
  where *instructor.dept_name= department.dept_name*;
- **insert into** *instructor_info*
  values ('69987', 'White', 'Taylor');
- Issues
  - Which department, if multiple departments in Taylor?
  - What if no department is in Taylor?

And Some Not at All

- **create view** *history_instructors* **as**
  select *
  from *instructor*
  where *dept_name*= 'History';
- What happens if we insert
  ('25566', 'Brown', 'Biology', 100000)
  into *history_instructors?*

View Updates in SQL

- Most SQL implementations allow updates only on simple views
  - The **from** clause has only one database relation.
  - The **select** clause contains only attribute names of the relation, and does not have any expressions, aggregates, or **distinct** specification.
  - Any attribute not listed in the **select** clause can be set to null
  - The query does not have a **group** by or **having** clause.

Materialized Views

- Certain database systems allow view relations to be physically stored.
  - Physical copy created when the view is defined.
  - Such views are called **Materialized view**:
- If relations used in the query are updated, the materialized view result becomes out of date
  - Need to **maintain** the view, by updating the view whenever the underlying relations are updated.

PURDUE UNIVERSITY
Department of Computer Science

Materialized Views and Query Processing

- Materialized views can speed query processing
  - Allows data that doesn't match good design standards, e.g., not normalized, but matches common queries
- *Logically* data follows design
  - But physical copy that doesn't
- Some work in automating creating of materialized views to support queries

17

PURDUE UNIVERSITY | Department of Computer Science

Triggers

Prof. Chris Clifton
30 November 2021

PURDUE UNIVERSITY
Department of Computer Science

Triggers

- Sometimes we want to take actions when a condition occurs in the database
  - Low balance in an account: Send warning
  - Update to a view that the DBMS can't figure out, but we know how to do
- One option: Program into every transaction
  - And get it right every time
- Option two: Triggers

19

PURDUE UNIVERSITY
Department of Computer Science

Triggers

- Idea: Execute code on an event

```
CREATE TRIGGER low_balance_warning
AFTER UPDATE OF balance ON accounts
FOR EACH ROW
WHEN ( new.balance < 100 )
    BEGIN
    <action to be taken>
    END
```

- *Note: Syntax and capabilities vary considerably between systems*

20

Triggering Events and Actions in SQL

- Triggering event can be **insert**, **delete** or **update**
- Triggers on update can be restricted to specific attributes
  - For example, **after update of** *takes* **on** *grade*
- Values of attributes before and after an update can be referenced
  - **referencing old row as** : for deletes and updates
  - **referencing new row as** : for inserts and updates
- Triggers can be activated before an event, which can serve as extra constraints. For example, convert blank grades to null.

```
create trigger setnull_trigger before update of takes
referencing new row as nrow
for each row
      when (nrow.grade = ' ')
        begin atomic
          set nrow.grade = null;
 end;
```


Trigger to Maintain credits_earned value

```
create trigger credits_earned after update of takes on (grade)
referencing new row as nrow
referencing old row as orow
for each row
when nrow.grade <> 'F' and nrow.grade is not null
   and (orow.grade = 'F' or orow.grade is null)
begin atomic
    update student
    set tot_cred= tot_cred +
        (select credits
         from course
         where course.course_id= nrow.course_id)
    where student.id = nrow.id;
end;
```

Statement Level Triggers

- Instead of executing a separate action for each affected row, a single action can be executed for all rows affected by a transaction
  - Use **for each statement** instead of **for each row**
  - Use **referencing old table** or **referencing new table** to refer to temporary tables (called ***transition tables***) containing the affected rows
  - Can be more efficient when dealing with SQL statements that update a large number of rows

PURDUE UNIVERSITY
Department of Computer Science

Triggers for View Update

- Given a table employee(name, address, dept, salary)
  - CREATE VIEW employee_directory AS SELECT name, dept FROM employee
- What happens when someone tries to insert an employee in employee_directory?

```
CREATE TRIGGER ViewUpdate
INSTEAD OF INSERT ON employee_directory
FOR EACH ROW
    BEGIN
        INSERT INTO employee VALUES ( :new.name, NULL, :new.dept, NULL )
    END
```

24

When Not To Use Triggers

- Triggers were used earlier for tasks such as
  - Maintaining summary data (e.g., total salary of each department)
  - Replicating databases by recording changes to special relations (called **change** or **delta** relations) and having a separate process that applies the changes over to a replica
- There are better ways of doing these now:
  - Databases today provide built in materialized view facilities to maintain summary data
  - Databases provide built-in support for replication
- Encapsulation facilities can be used instead of triggers in many cases
  - Define methods to update fields
  - Carry out actions as part of the update methods instead of through a trigger

When Not To Use Triggers (Cont.)

- Risk of unintended execution of triggers, for example, when
  - Loading data from a backup copy
  - Replicating updates at a remote site
  - Trigger execution can be disabled before such actions.
- Other risks with triggers:
  - Error leading to failure of critical transactions that set off the trigger
  - Cascading execution