Chapter 4: Intermediate SQL

Database System Concepts, 6th Ed.

Chapter 4: Intermediate SQL

- Views
- Integrity Constraints
- SQL Data Types and Schemas
- Authorization

Views

- In some cases, it is not desirable for all users to see the entire logical model (that is, all the actual relations stored in the database.)
- Consider a person who needs to know an instructors name and department, but not the salary. This person should see a relation described, in SQL, by

select *ID*, *name*, *dept_name*
from *instructor*

Views

- A **view** provides a mechanism to hide certain data from the view of certain users.
- The virtual relation is not precomputed and stored, but instead is computed by executing the query whenever the virtual relation is used.
- Any relation that is not of the conceptual model but is made visible to a user as a “virtual relation” is called a **view**

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

Example Views

- A view of instructors without their salary

```
create view faculty as
   select ID, name, dept_name
   from instructor
```

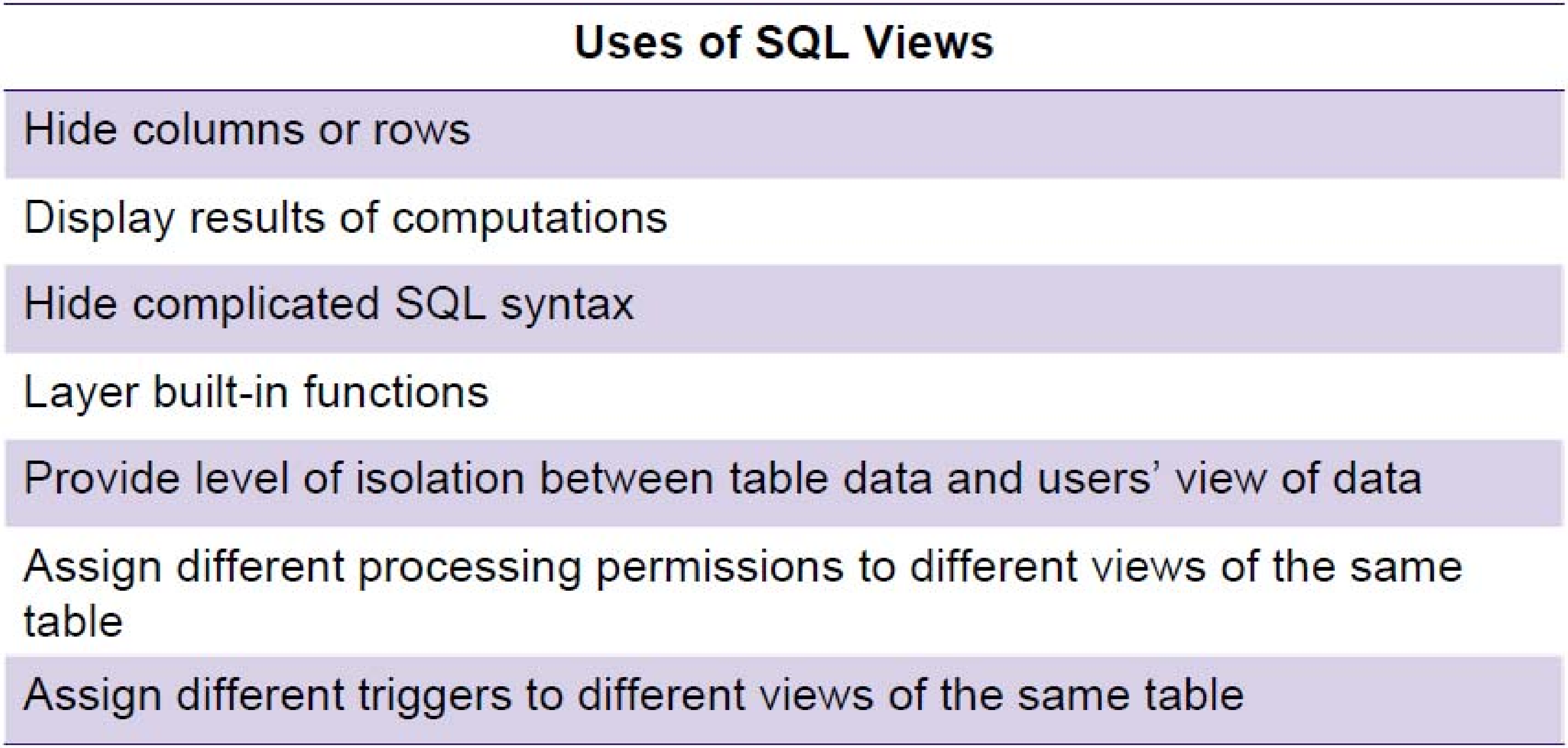
- Find all instructors in the Biology department

```
select name
from faculty
where dept_name = ‘Biology’
```

Views Defined Using Other Views

-
  ```
  create view physics_fall_2009 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 = ’2009’;
  ```

-
  ```
  create view physics_fall_2009_watson as
      select course_id, room_number
      from physics_fall_2009
      where building= ’Watson’;
  ```



Uses of SQL Views
Hide columns or rows
Display results of computations
Hide complicated SQL syntax
Layer built-in functions
Provide level of isolation between table data and users' view of data
Assign different processing permissions to different views of the same table
Assign different triggers to different views of the same table

Kroenke et al. Database Processing: Fundamentals, Design, and Implementation,15/E, Pearson

Integrity Constraints

- Integrity constraints guard against accidental damage to the database, by ensuring that authorized changes to the database do not result in a loss of data consistency.
  - An instructor name cannot be null
  - A checking account must have a balance greater than $10,000.00
  - Every department name in the course relation must have a matching department name in the department relation
  - A salary of a bank employee must be at least $4.00 an hour
  - A customer must have a (non-null) phone number

Integrity Constraints on a Single Relation

- **not null**
- **primary key**
- **unique**
- **check** (P), where P is a predicate

Not Null and Unique Constraints

- **not null**
  - Prohibits the insertion of a null value for the attribute
  - Declare *name* and *budget* to be not null

    name varchar(20) not null
    budget numeric(12,2) not null

- **unique** (A_1, A_2, …, A_m)
  - No two tuples in the relation can be equal on the listed attributes
  - Can have null values

The check clause

- **check** (P) where P is a predicate

Ex: ensure that semester is one of fall, winter, spring or summer:

```
create table section (
    course_id varchar (8),
    sec_id varchar (8),
    semester varchar (6),
    year numeric (4,0),
    building varchar (15),
    room_number varchar (7),
    time slot id varchar (4),
    primary key (course_id, sec_id, semester, year),
    check (semester in ('Fall', 'Winter', 'Spring', 'Summer'))
);
```

Referential Integrity

- Ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.
  - Example: If “Biology” is a department name appearing in one of the tuples in the *instructor* relation, then there exists a tuple in the *department* relation for “Biology”.
- Let A be a set of attributes. Let R and S be two relations that contain attributes A and where A is the primary key of S. A is said to be a **foreign key** of R if for any values of A appearing in R these values also appear in S.

Cascading Actions in Referential Integrity

-
  ```
  create table course (
      course_id   char(5) primary key,
      title       varchar(20),
      dept_name varchar(20) references department
  )
  ```

-
  ```
  create table course (
      …
      dept_name varchar(20),
      foreign key (dept_name) references department
                  on delete cascade
                  on update cascade,
      . . .
  )
  ```

Built-in Data Types in SQL

- **date:** Dates, containing a (4 digit) year, month and date
  - Example: **date** ‘2005-7-27’
- **time:** Time of day, in hours, minutes and seconds.
  - Example: **time** ‘09:00:30’ **time** ‘09:00:30.75’
- **timestamp**: date plus time of day
  - Example: **timestamp** ‘2005-7-27 09:00:30.75’
- **interval:** period of time
  - Example: interval ‘1’ day
  - Subtracting a date/time/timestamp value from another gives an interval value
  - Interval values can be added to date/time/timestamp values

Index Creation

- **create table** *student*
  (*ID* **varchar** (5),
  name **varchar** (20) **not null**,
  dept_name **varchar** (20),
  tot_cred **numeric** (3,0) **default** 0,
  primary key (*ID*))
- **create index** *studentID_index* **on** *student*(*ID*)
- Indices are data structures used to speed up access to records with specified values for index attributes
  - e.g. **select ***
    from *student*
    where *ID* = ‘12345’

  can be executed by using the index to find the required record, without looking at all records of *student*

User-Defined Types

- **create type** construct in SQL creates user-defined type

  create type *Dollars* **from numeric (12,2) not null**

create table *employee*

(ID **varchar(5**),

name **varchar(20**) **not null**,

salary *Dollars*,

primary key (ID));

Domains

- **create domain** construct in SQL-92 creates user-defined domain types

  create domain *person_name* **char**(20) **not null**

- Types and domains are similar. Domains can have constraints, such as **not null**, specified on them. Domains can also have default values.
- **create domain** *degree_level* **varchar**(10)
  constraint *degree_level_test*
  check (**value in** ('Bachelors', 'Masters', 'Doctorate'));
- Check support in specific DBMS (not supported in MS SQL Server)

Large-Object Types

- Large objects (photos, videos, CAD files, etc.) are stored as a *large object*:
  - **blob**: binary large object -- object is a large collection of uninterpreted binary data (whose interpretation is left to an application outside of the database system)
  - **clob**: character large object -- object is a large collection of character data
  - When a query returns a large object, a pointer is returned rather than the large object itself.
  - Check support in specific DBMS

Authorization

Forms of authorization on parts of the database (**privilege**):

- **Read** - allows reading, but not modification of data.
- **Insert** - allows insertion of new data, but not modification of existing data.
- **Update** - allows modification, but not deletion of data.
- **Delete** - allows deletion of data.

Forms of authorization to modify the database schema

- **Index** - allows creation and deletion of indices.
- **Resources** - allows creation of new relations.
- **Alteration** - allows addition or deletion of attributes in a relation.
- **Drop** - allows deletion of relations.

Authorization Specification in SQL

- The **grant** statement is used to confer authorization

  grant <privilege list>

  on <relation name or view name> **to** <user list>

- <user list> is:
  - a user-id
  - **public**, which allows all valid users the privilege granted
  - A role (more on this later)
- Granting a privilege on a view does not imply granting any privileges on the underlying relations.
- The grantor of the privilege must already hold the privilege on the specified item (or be the database administrator).

Privileges in SQL

- **select:** allows read access to relation,or the ability to query using the view
  - Example: grant users U_1, U_2, and U_3 **select** authorization on the *instructor* relation:

    grant select on *instructor* **to** U_1, U_2, U_3

- **insert**: the ability to insert tuples
- **update**: the ability to update using the SQL update statement
- **delete**: the ability to delete tuples.
- **all privileges**: used as a short form for all the allowable privileges

Revoking Authorization in SQL

- The **revoke** statement is used to revoke authorization.

  revoke <privilege list>

  on <relation name or view name> **from** <user list>

- Example:

  revoke select on *branch* **from** U_1, U_2, U_3

- <privilege-list> may be **all** to revoke all privileges the revokee may hold.
- If <revokee-list> includes **public,** all users lose the privilege except those granted it explicitly.

Roles

- **create role** instructor;
- **grant** *instructor* **to Amit;**
- Privileges can be granted to roles:
  - **grant select on** *takes* **to** *instructor*
- Roles can be granted to users, as well as to other roles
  - **create role** *teaching_assistant*
  - **grant** *teaching_assistant* **to** *instructor*;
    - *Instructor* inherits all privileges of *teaching_assistant*
- Chain of roles
  - **create role** *dean*;
  - **grant** *instructor* **to** *dean*;
  - **grant** *dean* **to** Satoshi;

Quiz

- 30 minutes, in class from 3:30 PM, Jan 31
- Don’t be late, otherwise you may not be allowed to attend
- Bring Photo ID
- Chapters 1, 3, and 4 (content covered in classes, slides and exercises)
  - Introduction to database (ch 1)
  - SQL (ch3 and ch4)
  - Paper-based, no textbook/slides/cheat sheet
- Possible formats: MCQ, T/F, Fill blank, Short questions