CMSC 508
Database Theory

Advanced SQL (I)

Dr. Alberto Cano
Assistant Professor
Department of Computer Science

Chapter 4 from Database System Concepts, 6th Ed. by Silberschatz, Korth, Sudarshan, 2011
Chapter 5 from Database Management Systems, 3rd Ed. by Ramakrishnan, Gehrke, 2003

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

    select *employee_id*, *last_name*, *department_id*
    from *employees;*

  - A view provides a mechanism to hide certain data (columns) from the view of certain users
  - Any relation that is not of the conceptual model but is made visible to a user as a "virtual relation" is called a view

- Views
  - A view is defined using the **create view** statement

    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

- Views examples

```
create view employeesData (employee_id, last_name, department_name)
as
select employee_id, last_name, department_name
from employees, departments
where employees.department_id = departments.department_id;

create view employeesLocation (employee_id, last_name, city)
as
select employee_id, last_name, city
from employees join departments
on employees.department_id = departments.department_id
join locations
on departments.location_id = locations.location_id;
```

- Views using other views
  - One view may be used in the expression defining another view
  - A view v_1 is said to **depend directly** on a view v_2 if v_2 is used in the expression defining v_1
  - A view v_1 is said to **depend on** view 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 v is said to be **recursive** if it depends on itself

- Views using other views

```
create view employeesDataLocation (last_name, department_name, city)
as
select employeesData.last_name, department_name, city
from employeesData join employeesLocation
on employeesData.employee_id = employeesLocation.employee_id;
```

- Views filtering

```
create view highSalaryEmployees
(last_name, department_name, salary) as
select e.last_name, d.department_name, e.salary
from employees e join departments d
on e.department_id = d.department_id
where e.salary >
      (select AVG(salary) from employees c
      where e.department_id = c.department_id);
```

- Updatable and Insertable Views
  - Some views are updatable and references to them can be used to specify tables to be updated in data change statements. That is, you can use them in statements such as **UPDATE**, **DELETE**, or **INSERT** to update the contents of the underlying table
  - For a view to be updatable, there must be a **one-to-one relationship** between the rows in the view and the rows in the underlying table.

- Updatable and Insertable Views

```
create view regionsView
as select region_id, region_name from regions;

insert into regionsView values (99,'Antarctica');

update regionsView
set region_name = 'Potato Land' where region_id = 99;

delete from regionsView where region_id = 99;

drop view regionsView;
```

- Updatable and Insertable Views
  - Specifically, a view is **not** updatable if it contains any of:
    - Aggregate functions
    - DISTINCT, GROUP BY, HAVING, UNION
    - Subquery in the select list
    - Reference to nonupdatable view in the FROM clause
    - Subquery in the WHERE clause that refers to a table in the FROM clause
    - Refers only to literal values
    - Multiple references to any column of a base table

- Updatable and Insertable Views
  - Example: view is **not** updatable

create view *employeesDataLocation* (*last_name, department_name, city*) **as**
select *employeesData.last_name, department_name, city*
from *employeesData* **join** *employeesLocation*
on *employeesData.employee_id = employeesLocation.employee_id*;

insert into *employeesDataLocation*
values ('John', 'Marketing', 'Seattle');

SQL Error: ORA-01776: cannot modify more than one base table through a join view

What's the employeeID, deparmentID, salary, etc?

- Materialized views
  - Materializing a view: create a physical table containing all the tuples in the result of the query defining the 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
  - Oracle uses materialized views (also known as snapshots) to replicate data to non-master sites in a replication environment and to cache expensive queries in a data warehouse environment
  - Trade-off performance vs extra storage usage depending on application and frequency

- Create view exercise

Write a query to show the department id, department name, the average salary of all employees working for that department, and the number of employees of the department. Show only the departments having less than 10 employees and having a department average salary > 5,000. Implement the average salaries as a view containing department_id and the average. Implement the count of the employees as a view containing department_id and employee count.

- Authorization

  Forms of authorization on contents of relations:

  - **Read** - allows reading, but not modification of data
  - **Insert** - allows insertion of new data, but not modification
  - **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
  - 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
  - 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)

- SQL privileges
  - A user privilege is a right to execute a particular type of statement, or a right to access another user's object
  - **select:** allows read access to relation or 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 privileges
  - The **revoke** statement is used to revoke authorization

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

    revoke select on *instructor* **from** U_1, U_2, U_3
  - <privilege-list> may be **all** to revoke all
  - If <revokee-list> includes **public,** all users lose the privilege except those granted explicitly
  - If the same privilege was granted twice to the same user by different grantees, the user may retain the privilege after the revocation
  - All privileges that depend on the privilege being revoked are also revoked

- Roles
  - Roles are created by users (usually administrators) and are used to group together privileges or other roles

    create role staff; (Need privileges to create roles, YOU can't)
  - Privileges can be granted to roles:

    grant select on *employeesData* **to** *staff*;
  - Roles can be granted to users, as well as to other roles

    create role *department_director*

    grant … to *department_director*

    grant *department_director* **to** *kcios*;
    - *User kcios* inherits all privileges of *department_director*

CMSC 508
Database Theory

Advanced SQL (I)

Dr. Alberto Cano
Assistant Professor
Department of Computer Science

Chapter 4 from Database System Concepts, 6th Ed. by Silberschatz, Korth, Sudarshan, 2011
Chapter 5 from Database Management Systems, 3rd Ed. by Ramakrishnan, Gehrke, 2003