

Advanced SQL (Cont)

Modern Database Management

12th Edition

*Jeff Hoffer, Ramesh Venkataraman,
Heikki Topi*

Objectives

- ▶ Subqueries
- ▶ Introduction to view
- ▶ Benefits and drawbacks of views
- ▶ Stored procedures

Subqueries

- ▶ A subquery can return one value or multiple values. To be precise, the subquery can return:
 - ▶ One single value (one column and one row). This subquery is used anywhere a single value is expected, as in the right side of a comparison expression.
 - ▶ A list of values (one column and multiple rows). This type of subquery is used anywhere a list of values is expected, such as when using the IN clause
 - ▶ A virtual table (multicolumn, multirow set of values). This type of subquery can be used anywhere a table is expected, such as when using the FROM clause.

Select subqueries

```
-- find all tickets that have min/max ticket price  
select * from ticket where ticket_price =  
(select min(ticket_price) from ticket);
```

IN Subqueries

- ▶ Use IN to compare a single attribute to a list of values
- ▶ When the condition values are not known before hand, but they can be derived using a query, you must use an IN subquery

```
select * from employee where emp_num in  
(select distinct emp_num from hours where hour_rate >6);
```

Having subqueries

```
select employee.*, sum(hours_per_attract)
from employee natural join hours
group by employee.emp_num
having sum(hours_per_attract) >
(select max(HOURS_PER_ATTRACT) from hours);
```

Using and Defining Views

- ▶ Views provide users controlled access to tables
- ▶ Base Table-table containing the raw data
- ▶ Dynamic View
 - ▶ A “virtual table” created dynamically upon request by a user
 - ▶ No data actually stored; instead data from base table made available to user
 - ▶ Based on SQL SELECT statement on base tables or other views

SYNTAX TO CREATE A VIEW

CREATE VIEW *viewname* **AS SELECT** *query*

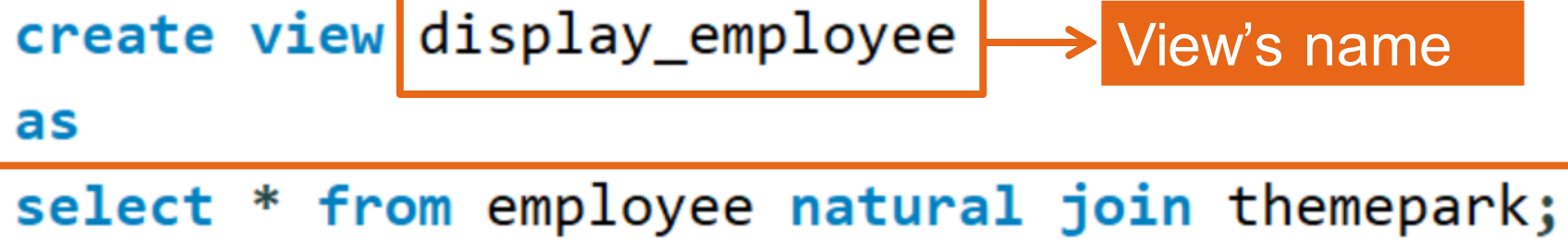
► To display the contents of a view :

SELECT * FROM *viewname*;

Sample CREATE VIEW

To create a view:

```
create view display_employee  
as  
select * from employee natural join themepark;
```



View is based on a SELECT statement.

To call a view:

```
select * from display_employee;
```

Advantages of Views

- ▶ Simplify query commands
- ▶ Assist with data security (but don't rely on views for security, there are more important security measures)
- ▶ Enhance programming productivity
- ▶ Contain most current base table data
- ▶ Use little storage space
- ▶ Provide customized view for user
- ▶ Establish physical data independence

Disadvantages of Views

- ▶ Use processing time each time view is referenced
- ▶ May or may not be directly updateable

Stored Procedure (Stored routine)

- ▶ A stored routine is a set of SQL statements that can be stored in the server.
- ▶ Once this has been done, clients don't need to keep reissuing the individual statements but can refer to the stored routine instead.
- ▶ When to use:
 - ▶ When multiple client applications are written in different languages or work on different platforms but need to perform the same database operations.
 - ▶ When security is paramount: This provides a consistent and secure environment, and routines can ensure that each operation is properly logged

Syntax to create a Stored procedure

DELIMITER //

changes the default delimiter

```
CREATE PROCEDURE procedure_name (IN parameter1 datatype, OUT parameter2 datatype, ...)  
BEGIN  
    -- Procedure logic goes here  
    -- For example, select or update statements, conditional logic, etc.  
END //
```

DELIMITER ;

reset the default delimiter

Stored procedure

Stored procedure with IN parameter

```
DELIMITER &&  
create procedure find_theme_park  
(in a_code varchar(30))  
begin  
    select * from themepark  
    where park_code like a_code;  
end&&  
DELIMITER ;
```

Stored procedure

Stored procedure with OUT parameter

```
## procedure with OUT parameters
delimiter //
create procedure get_average_quantity(out average int)
begin
    select avg(line_qty) into average from sales_line;
end //
delimiter ;
#call the procedure
call get_average_quantity(@average);
# use the result of procedure
select @average, sales_line.* from sales_line
where line_qty < @average;
```

Benefit of stored procedures

- ▶ **Reduce the Network Traffic:**
 - ▶ Multiple SQL Statements are encapsulated in a stored procedure.
 - ▶ When executing it, instead of sending multiple queries, only the name and the parameters of the stored procedure are sent
- ▶ **Easy to maintain:**
 - ▶ The stored procedure are reusable.
 - ▶ If any change is required, you need to make a change in the stored procedure only
- ▶ **Secure:**
 - ▶ The permission can be granted to the user to execute the stored procedure without giving permission to the tables used in the stored procedure.
 - ▶ The stored procedure helps to prevent the database from SQL Injection

Trigger

```
create trigger myTrigger
after insert
on student
for each row
begin
    insert into student_log values (new.id, 'add', now());
end;&&
delimiter ;
```