Advanced SQL

http://goo.gl/kM2gy8

CS5200 DBMS
Bruce Chhay

Advanced SQL

L1. Aggregation, Sorting, Limiting Results

SELECT Statement

```
SELECT select_expr
FROM table_references
WHERE where_condition
GROUP BY col_expr
HAVING where_condition
ORDER BY col_expr
LIMIT limit_expr
```

http://dev.mysql.com/doc/refman/5.7/en/select.html

GROUP BY Clause

- Syntax: GROUP BY col_expr
- col_expr:
  - Expresses an aggregation on a list of column names.
  - (Should not reference an alias defined in SELECT clause.)
  - Can be integer representing column position (starting at 1).
  - Can use a WITH ROLLUP modifier for summary rows.
    - Example: http://dev.mysql.com/doc/refman/5.7/en/group-by-modifiers.html
- Allows an aggregation function to be performed in the SELECT clause.
  - List of functions: https://dev.mysql.com/doc/refman/5.7/en/aggregate-functions.html

http://dev.mysql.com/doc/refman/5.7/en/group-by-handling.html
http://dev.mysql.com/doc/refman/5.7/en/group-by-functions-and-modifiers.html

GROUP BY Clause

- Expresses an aggregation on a list of column names.
- Allows an aggregation function in the SELECT clause.

```
SELECT FirstName
FROM BlogUsers
GROUP BY FirstName
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FirstName
Jae
Tony
Dan
James

GROUP BY Clause

- Expresses an aggregation on a list of column names.
- Allows an aggregation function in the SELECT clause.

```
SELECT FirstName
FROM BlogUsers
GROUP BY FirstName
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FirstName
Jae
Tony
Dan
James

GROUP BY Clause

- Expresses an aggregation on a list of column names.
- Allows an aggregation function in the SELECT clause.

```
SELECT FirstName
FROM BlogUsers
GROUP BY FirstName
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FirstName
Jae
Tony
Dan
James

GROUP BY Clause

- Expresses an aggregation on a list of column names.
- Allows an aggregation function in the SELECT clause.

```
SELECT FirstName, LastName
FROM BlogUsers
GROUP BY FirstName, LastName
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FirstName	LastName
Jae	Yoon
Tony	Davidson
Dan	Kwan
James	Marks

GROUP BY Clause

- Expresses an aggregation on a list of column names.
- Allows an aggregation function in the SELECT clause.

```
SELECT FirstName, COUNT(*) AS CNT
FROM BlogUsers
GROUP BY FirstName
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FirstName	CNT
Jae	2
Tony	1
Dan	1
James	1

GROUP BY Clause

- Expresses an aggregation on a list of column names.
- Allows an aggregation function in the SELECT clause.

```
SELECT FirstName, MAX(DoB) AS MaxDob
FROM BlogUsers
GROUP BY FirstName
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FirstName	MaxDob
Jae	2005-01-01
Tony	1996-01-01
Dan	1994-01-01
James	1990-01-01

HAVING Clause

- Syntax: HAVING where_expr
- where_expr:
  - Must reference an aggregate field in GROUP BY clause or is an aggregate function (note that WHERE clause cannot contain aggregate functions).
  - Like a WHERE clause, can have operators and can evaluate to true/false.
- Filters aggregate results
  (HAVING filters on group of rows;
  WHERE filters individual rows).

HAVING Clause

- Reference aggregate field in GROUP BY or is aggregate function.
- Filters aggregate results.

```
SELECT FirstName, COUNT(*) AS CNT
FROM BlogUsers
GROUP BY FirstName
HAVING COUNT(*) > 1
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FirstName	CNT
Jae	2
Tony	1
Dan	1
James	1

FirstName	CNT
Jae	2

HAVING Clause

- Reference aggregate field in GROUP BY or is aggregate function.
- Filters aggregate results.

```
SELECT FirstName, COUNT(*) AS CNT
FROM BlogUsers
GROUP BY FirstName
HAVING COUNT(*) > 1
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FirstName	CNT
Jae	2
Tony	1
Dan	1
James	1

FirstName	CNT
Jae	2

HAVING Clause

- Reference aggregate field in GROUP BY or is aggregate function.
- Filters aggregate results.

```
SELECT FirstName, COUNT(*) AS CNT
FROM BlogUsers
GROUP BY FirstName
HAVING COUNT(*) > 1
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

FirstName	CNT
Jae	2
Tony	1
Dan	1
James	1

FirstName	CNT
Jae	2

ORDER BY Clause

- Syntax: ORDER BY col_expr
- col_expr:
  - Must reference an aggregate field in GROUP BY clause or is an aggregate function (like HAVING and SELECT clauses).
  - Can use a ASC|DESC modifier to sort ascending or descending order per field. (ASC is default.)
- Sort results.

ORDER BY Clause

- Sort results.

```
SELECT *
FROM BlogUsers
ORDER BY FirstName
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

UserName	FirstName	LastName	DoB
dan	Dan	Kwan	1994-01-01
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
james	James	Marks	1990-01-01
tony	Tony	Davidson	1996-01-01

ORDER BY Clause

- Sort results.

```
SELECT *
FROM BlogUsers
ORDER BY FirstName DESC
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

UserName	FirstName	LastName	DoB
tony	Tony	Davidson	1996-01-01
james	James	Marks	1990-01-01
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
dan	Dan	Kwan	1994-01-01

ORDER BY Clause

- Sort results.

```
SELECT *
FROM BlogUsers
ORDER BY FirstName, DoB
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

UserName	FirstName	LastName	DoB
dan	Dan	Kwan	1994-01-01
jo	Jae	Yoon	1980-01-01
jy	Jae	Yoon	2005-01-01
james	James	Marks	1990-01-01
tony	Tony	Davidson	1996-01-01

LIMIT Clause

- Syntax: LIMIT limit_expr
- limit_expr:
  - As an integer, is the max number of rows to return.
  - Can have an optional offset.
- Examples:
  - LIMIT 5: return at most 5 records.
  - LIMIT 3 OFFSET 5: return at most records 6-8.

LIMIT Clause

- Constrain number of rows returned.

```
SELECT *
FROM BlogUsers
LIMIT 1
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01

LIMIT Clause

- Constrain number of rows returned.

```
SELECT *
FROM BlogUsers
LIMIT 2 OFFSET 1
```

UserName	FirstName	LastName	DoB
jy	Jae	Yoon	2005-01-01
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01
dan	Dan	Kwan	1994-01-01
james	James	Marks	1990-01-01

UserName	FirstName	LastName	DoB
jo	Jae	Yoon	1980-01-01
tony	Tony	Davidson	1996-01-01

Standard SQL vs MySQL

- Standard: No expressions allowed in GROUP BY (i.e. functions).
  - MySQL allows expressions in GROUP BY.
  - Standard SQL: use a nested query.
  - Advice: favor MySQL for straight-forward expressions, like functions.

```
SELECT SUBSTRING(FirstName, 1, 1) AS FirstChar, COUNT(*) AS CNT
FROM BlogUsers
GROUP BY SUBSTRING(FirstName, 1, 1)
```

```
SELECT A.FirstChar, COUNT(*) AS CNT
FROM (
   SELECT SUBSTRING(FirstName, 1, 1) AS FirstChar
   FROM BlogUsers) AS A
GROUP BY A.FirstChar
```

Standard SQL vs MySQL

- Standard: Aliases in SELECT clause cannot be referenced in GROUP BY or HAVING clauses.
  - MySQL allows GROUP BY/HAVING to reference aliases defined in SELECT.
  - Standard SQL: use the column/expression and/or use a nested query.
  - Advice: favor MySQL for convenience.

```
SELECT FirstName AS First, COUNT(*) AS CNT
FROM BlogUsers
GROUP BY First
HAVING CNT > 10
```

```
SELECT FirstName AS First, COUNT(*) AS CNT
FROM BlogUsers
GROUP BY FirstName
HAVING COUNT(*) > 10
```

Standard SQL vs MySQL

- Standard: SELECT list, HAVING condition, ORDER BY list must refer either to an aggregation function or a column in GROUP BY list.
  - MySQL allows a dependent column if the primary key is in the GROUP BY list (MySQL evaluates the PK constraint/FD).
  - Standard SQL: reference column name in GROUP BY and/or use nested query.
  - Advice: favor Standard SQL for intuitiveness
    (reflects query execution order and not easy to recall PK constraints).

```
SELECT BlogUsers.FirstName, COUNT(*) AS CNT
FROM BlogUsers INNER JOIN BlogPosts ON BlogUsers.UserName = BlogPosts.UserName
GROUP BY BlogUsers.UserName
```

```
SELECT BlogUsers.FirstName, COUNT(*) AS CNT
FROM BlogUsers INNER JOIN BlogPosts ON BlogUsers.UserName = BlogPosts.UserName
GROUP BY BlogUsers.UserName, BlogUsers.FirstName
```

Query Evaluation

1. Evaluate FROM clause to build table reference, including join operations.
2. Evaluate WHERE clause to filter single records.
3. Evaluate GROUP BY clause for aggregation.
4. Evaluate HAVING clause to filter aggregation.
5. Evaluate SELECT to filter/rename columns.
6. Evaluate ORDER BY to sort records.
7. Evaluate LIMIT to constrain the window of records.

5 SELECT
1 FROM
2 WHERE
3 GROUP BY
4 HAVING
6 ORDER BY
7 LIMIT

Back to INSERT/UPDATE/DELETE

- INSERT/UPDATE/DELETE support SELECT statements.
- Insert results of a SELECT statement.

```
INSERT INTO tbl_name(col_name,...)
  SELECT … FROM … WHERE …
```

- Update records with a join.

```
UPDATE tbl_A,tbl_B
  SET tbl_A.col_name=val WHERE tbl_A.pk=tbl_B.fk
```

- Delete records from multiple tables.

```
DELETE tbl_A,tbl_B
  FROM tbl_A INNER JOIN tbl_B WHERE tbl_A.pk=tbl_B.fk
```

http://dev.mysql.com/doc/refman/5.7/en/insert-select.html
http://dev.mysql.com/doc/refman/5.7/en/update.html
http://dev.mysql.com/doc/refman/5.7/en/delete.html

Back to INSERT/UPDATE/DELETE

- INSERT/UPDATE/DELETE support SELECT statements.
- Insert results of a SELECT statement.

```
INSERT INTO tbl_name(col_name,...)
 SELECT … FROM … WHERE …
```

- Update records with a join.

```
UPDATE tbl_A INNER JOIN tbl_B on tbl_A.pk=tbl_B.fk
 SET tbl_A.col_name=val WHERE tbl_A.pk = 1
```

- Delete records from multiple tables.

```
DELETE tbl_A,tbl_B
 FROM tbl_A INNER JOIN tbl_B WHERE tbl_A.pk=tbl_B.fk
```

http://dev.mysql.com/doc/refman/5.7/en/insert-select.html
http://dev.mysql.com/doc/refman/5.7/en/update.html
http://dev.mysql.com/doc/refman/5.7/en/delete.html

Back to INSERT/UPDATE/DELETE

- INSERT/UPDATE/DELETE support SELECT statements.
- Insert results of a SELECT statement.

```
INSERT INTO tbl_name(col_name,...)
 SELECT … FROM … WHERE …
```

- Update records with a join.

```
UPDATE tbl_A,tbl_B
 SET tbl_A.col_name=val WHERE tbl_A.pk=tbl_B.fk
```

- Delete records from multiple tables.

```
DELETE tbl_A,tbl_B
 FROM tbl_A INNER JOIN tbl_B WHERE tbl_A.pk=tbl_B.fk
```

http://dev.mysql.com/doc/refman/5.7/en/insert-select.html
http://dev.mysql.com/doc/refman/5.7/en/update.html
http://dev.mysql.com/doc/refman/5.7/en/delete.html

Advanced SQL

L2. Exercises

Exercises

1. Are there any unpublished posts that have comments? This checks for unexpected data.
2. Most recent Created timestamp of comments for each UserName.
3. Count of each UserNames' comments.
4. UserNames who commented more than once.
5. Count of comments for each post in descending order, and include the post id and title.
6. UserName with most unpublished posts, and include status level.
7. The most reshared post, and include post id and title.
8. Number of comments for each UserName per day that they commented.
9. Average number of comments per day for each UserName.
10. What is the comment counts for all days in February (even for days without comments)? Need a "date" table for left outer join.
11. UserNames with more comments than posts (hint: include all users).
12. Compare a user's number of reshares to published posts (hint: include all users). Who is more likely to read than write (reshares>posts), and vice versa (reshares<posts)?

Examples

Blog Application queries: http://goo.gl/QAMWOH

Creating tables: http://goo.gl/86a11H

Inserting data: http://goo.gl/m4Y7rh

Advanced SQL

L3. Functions and Procedures

CREATE FUNCTION

CREATE FUNCTION name(params)

RETURNS type

routine_body

Functions must specify a return type. Type must be a valid data type, such as DATE, CHAR, etc. MS SQL allows TABLE.
http://dev.mysql.com/doc/refman/5.7/en/create-procedure.html
http://dev.mysql.com/doc/refman/5.7/en/stored-programs-defining.html

CREATE PROCEDURE

CREATE PROCEDURE name(params)

routine_body

http://dev.mysql.com/doc/refman/5.7/en/create-procedure.html
http://dev.mysql.com/doc/refman/5.7/en/stored-programs-defining.html

Compound-Statement Syntax

routine_body:

- Usually wrapped in BEGIN … END block, which allows a list of statements terminated by ; (since MySQL uses ; as the default delimiter, you need to redefine the delimiter temporarily to pass in the entire routine_body).
- Use DECLARE for local variables.
- Use SET to assign variables. var is routine-specific and reset on every time the routine is called. @var for a sessions-specific variable (can be called by the client after routine completes, is freed when session ends, is not shareable across clients). @@var for global variable (reset when the server restarts).
- Can have flow-control statements, such as CASE, IF, LOOP (plus labeling), WHILE, etc.
- Cursors to iterate each record of a SELECT statement.
- More...

https://dev.mysql.com/doc/refman/5.7/en/sql-compound-statements.html

Procedures vs Functions

- Procedures can have input and output params. Functions only have input params.
- Procedures can have DDL (i.e. can create/drop tables). Functions cannot.
- Procedures can call other procedures and functions. Functions can only call other functions.
- Functions must specify return type.
- MySQL: only procedures can return table results (i.e. SELECT statement). MS SQL: functions can return table results (called a table-valued function, or table-valued user-defined function).
- Functions can be used in a SELECT statement, but procedures cannot. (Execute a procedure with CALL procedure statement.)

https://docs.microsoft.com/en-us/sql/t-sql/statements/create-procedure-transact-sql
https://docs.microsoft.com/en-us/sql/t-sql/statements/create-function-transact-sql
https://docs.microsoft.com/en-us/sql/relational-databases/user-defined-functions/create-user-defined-functions-database-engine

Examples

Function: http://goo.gl/VgQPWw

Procedure: http://goo.gl/Mt0eeH