View, Index and Explain

Adaptation of the course of Ms. CHHUN SOPHEA

Creating View

Syntax

```
CREATE
    [OR REPLACE]
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Ex:

```
CREATE VIEW current_dept_emp AS
        SELECT * FROM dept_emp
        WHERE to_date = '9999-01-01';
CREATE VIEW current_finance_emp AS
        SELECT * FROM departments natural join dept_emp natural
            join employees
        WHERE to_date = '9999-01-01' and dept_name = 'finance' ;
```

Alter View

Syntax

```
ALTER
    [ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]
    [DEFINER = { user | CURRENT_USER }]
    [SQL SECURITY { DEFINER | INVOKER }]
    VIEW view_name [(column_list)]
    AS select_statement
    [WITH [CASCADED | LOCAL] CHECK OPTION]
```

Ex:

```
ALTER VIEW current_dept_emp AS
        SELECT emp_no as eid, dept_no as did, from_date, to_date
        FROM dept_emp
        WHERE to_date = '9999-01-01';
```

Dropping View

- Syntax

DROP VIEW view_name

Ex:

DROP VIEW COM_COMPLETE;

Show View

- Syntax

```
SHOW FULL TABLES IN database_name
  WHERE TABLE_TYPE LIKE 'VIEW';
```

Ex:

```
SHOW FULL TABLES IN employees WHERE
  TABLE_TYPE LIKE 'VIEW';
```

Show Definition of View

- Syntax

SHOW CREATE VIEW view_name;

Ex:

SHOW CREATE VIEW current_dept_emp;

Creating of Indexes (1/4)

- Syntax

CREATE [UNIQUE|FULLTEXT|SPATIAL]
INDEX *index_name* ON *tb_name*
(*index_col_name*,...);

Ex:

Student(stuId, name, address);

>Create index indName on student(name);

>show keys from tb_name;

>Show index from tb_name;

>Desc tb_name;

Creating of Indexes (2/4)

- Indexes can be created during creation of table

Ex:

Student(stuId, name, address);

```
Create table student(
   stuId int auto_increment primary key,
   name varchar(40) not null,
   address varchar(200) not null,
   index (name)
);
```

Creating of Indexes (3/4)

- Alter statement is also used to create index

ALTER TABLE tbName ADD INDEX [index_name] (index_col_name,...)

Ex: alter table student add index (name);

Creating of Indexes (4/4)

Note: Most MySQL indexes (PRIMARY KEY, UNIQUE, INDEX, and FULLTEXT) are stored in B-trees.

Indexes on spatial data types use R-trees, and that MEMORY tables also support hash indexes.

Dropping Index

- Syntax

ALTER TABLE tbname drop index key-name;

Ex:

Alter table student drop index name;

Show Index

- Syntax

SHOW INDEX FROM table_name;

Ex:

SHOW INDEX FROM departments;

Explain statement (1/11)

- How to determine if MySQL is going to perform a full table scan?
  - Use explain statement
- EXPLAIN statement can be used
  - either as a synonym for DESCRIBE
  - Or as a way to obtain information about how MySQL executes a SELECT statement

Syntax: EXPLAIN *tb_name* is synonymous with DESCRIBE *tb_name* or SHOW COLUMNS FROM *tb_name*

Creating of Indexes (1/4)

- Syntax

CREATE [UNIQUE|FULLTEXT|SPATIAL]
INDEX *index_name* ON *tb_name*
(*index_col_name*,...);

Ex:

Student(stuId, name, address);

>Create index indName on student(name);

>show keys from tb_name;

>Show index from tb_name;

>Desc tb_name;

Explain statement (2/11)

- When we precede a SELECT statement with the keyword EXPLAIN, MySQL displays information from the optimizer about the query execution plan.
  - Syntax: Explain select statement
- EXPLAIN EXTENDED can be used to provide additional information.

Ex:

explain select * from student;

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	student	ALL	*NULL*	*NULL*	*NULL*	*NULL*	6	

Explain statement (3/11)

Output of Explain statement

- id: The SELECT identifier. This is the sequential number of the SELECT within the query.
- select_type: the type of SELECT, which can be any of those shown in the following.
  - Simple: basic select without union or subqueries

  Ex: select * from customer where lastName="smith";

Explain statement (4/11)

- select_type
  - Primary key: outermost or highest level select
  - Derived: if a query involves a derived table (including a view).
  - Union: second or later select in union

Ex:

```
select cm.customer_id
from customer_master cm
Where cm.date_joined_project > “2002-01-15”
Union
Select cm1.customer_id from custombер_master cm1
Where cm1.sex=‘M’;
```

Explain statement (4/11)

- select_type
  - Primary key: outermost or highest level select
  - Derived: if a query involves a derived table (including a view).
  - Union: second or later select in union

Ex:

```
select cm.customer_id
from customer_master cm
Where cm.date_joined_project > “2002-01-15”
Union
Select cm1.customer_id from custombер_master cm1
Where cm1.sex=‘M’;
```

Explain statement (5/11)

- select_type
  - Subquery: when a query includes a subquery the first SELECT in the subquery is identified as SUBQUERY
  - Dependent subquery: if a subquery relies on information from an outer query the first SELECT in the subquery is labeled dependent subquery.

Ex:

```
Select cm.last_name, cm.first_name
From customer_master as cm
Where cm.customer_id IN
(select ca.custombер_id from customer_address as ca
Where country=“canada”);
```

Explain statement (6/11)

SIMPLE	Simple **SELECT** (not using **UNION** or subqueries)
PRIMARY	Outermost **SELECT**
UNION	Second or later **SELECT** statement in a **UNION**
DEPENDENT UNION	Second or later **SELECT** statement in a **UNION**, dependent on outer query
UNION RESULT	Result of a **UNION**.
SUBQUERY	First **SELECT** in subquery
DEPENDENT SUBQUERY	First **SELECT** in subquery, dependent on outer query
DERIVED	Derived table **SELECT** (subquery in **FROM** clause)
UNCACHEABLE SUBQUERY	A subquery for which the result cannot be cached and must be re-evaluated for each row of the outer query
UNCACHEABLE UNION	The second or later select in a **UNION** that belongs to an uncacheable subquery (see **UNCACHEABLE SUBQUERY**)

- table: The table to which the row of output refers.

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	SIMPLE	student	ALL	*NULL*	*NULL*	*NULL*	*NULL*	6	

Means, output are queried from table student

Explain statement (7/11)

- type: the join type, the different join types are listed here, ordered from the best type to the worst.
  - system: the table has only one row, this is a special case
  - const: the table has almost one matching row. It is used when we compare all parts of PK or unique index to constant value.

  Ex:

  Explain select * from tb_name where pk=1;

  Explain select * from student where studentId=1 and studentId=2;

Explain statement (8/11)

- Type
  - fulltext: the join is performed using a FULLTEXT index.
  - all: a full table scan is done for each combination of rows from the previous tables. This is normally not good if the table is the first table not marked const, and usually *very* bad in all other cases.

  We can avoid ALL by adding indexes that allow row retrieval from the table based on constant values or column values from earlier tables.

  Ex: select last_name from customer where last_name like “%john%”;

Explain statement (9/11)

- Type:
  - index: this join type is the same as ALL, except that only the index tree is scanned. This usually is faster than ALL because the index file usually is smaller than the data file.
  - Etc.
- possible_keys: indicates which index MySQL can use to find the rows in table.
  - If this column is NULL, there are no relevant indexes.
  - In this case, we may be able to improve the performance of our query by examining the WHERE clause to check whether it refers to some column or columns that would be suitable for indexing.
  - If so, create an appropriate index and check the query with EXPLAIN again.

Explain statement (10/11)

- key: the key column indicates the key (index) that MySQL actually decided to use.
  - If MySQL decides to use one of the possible_keys indexes to look up rows, that index is listed as the key value.
- key_len: indicates the length of the key that MySQL decided to use.
  - The length is NULL if the key column says NULL.
  - Note that the value of key_len enables us to determine how many parts of a multiple-part key MySQL actually uses.

Explain statement (11/11)

- ref: any columns used with the key to retrieve results
- rows: estimate number of rows returned
- extra: contains additional information about how MySQL resolves the query

Check more at:

http://dev.mysql.com/doc/refman/5.1/en/using-explain.html

Explain extra

`Using index`	The result is created straight from the index
`Using where`	Not all rows are used in the result
`Distinct`	Only a single row is read per row combination
`Not exists`	A **`LEFT JOIN`** missing rows optimization is used
`Using filesort`	An extra row sorting step is done
`Using temporary`	A temporary table is used
`Range checked for each record`	The read type is optimized individually for each combination of rows from the previous tables