MySQL - RDBMS

Agenda

- Views
- Security
- Transactions
- Row locking
- Indexes

Views

- Projection of data -- SELECT.
- CREATE VIEW viewname AS SELECT ...
- MySQL views are non-materialized.
- When query is executed on view, data is read from target table.
- Simple view: DQL + DML
- Complex view: DQL

```
SELECT USER(), DATABASE();
-- | sunbeam@localhost | classwork  |

SHOW FULL TABLES;

SHOW CREATE VIEW v_richemp;

SELECT empno, ename, sal FROM v_richemp;

INSERT INTO v_richemp(empno, ename, sal) VALUES(1000, 'JAMES', 2600);

SELECT empno, ename, sal FROM v_richemp;

INSERT INTO v_richemp(empno, ename, sal) VALUES(1001, 'HARRY', 2200);

SELECT empno, ename, sal FROM v_richemp;

SELECT empno, ename, sal FROM emp;

EXPLAIN FORMAT=JSON
SELECT empno, ename, sal FROM v_richemp;

INSERT INTO v_richemp(empno, ename, sal) VALUES(1002, 'MERRY', 2000);

ALTER VIEW v_richemp AS
SELECT * FROM emp WHERE sal > 2500
WITH CHECK OPTION;

INSERT INTO v_richemp(empno, ename, sal) VALUES(1003, 'ADAM', 1500);
-- error: CHECK OPTION failed -- sal <= 2500.
```

```
INSERT INTO v_richemp(empno, ename, sal) VALUES(1004, 'EVE', 2800);
```

```
CREATE VIEW v_richemp2 AS
SELECT empno, ename, sal, comm FROM v_richemp;

SELECT * FROM v_richemp2;

SHOW CREATE VIEW v_richemp2;

DROP VIEW v_richemp;

SELECT * FROM v_richemp2;
-- error: invalid table/view.

DROP VIEW v_richemp2;
```

```
SELECT e.empno, e.ename, d.deptno, d.dname FROM emp e INNER JOIN dept d ON
e.deptno = d.deptno;

CREATE VIEW v_empdept AS
SELECT e.empno, e.ename, d.deptno, d.dname FROM emp e INNER JOIN dept d ON
e.deptno = d.deptno;

SELECT * FROM v_empdept;

-- display all emps in ACCOUNTING dept.
SELECT e.empno, e.ename, d.deptno, d.dname FROM emp e INNER JOIN dept d ON
e.deptno = d.deptno
WHERE d.dname = 'ACCOUNTING';

SELECT * FROM v_empdept
WHERE dname = 'ACCOUNTING';

EXPLAIN FORMAT=JSON
SELECT e.empno, e.ename, d.deptno, d.dname FROM emp e INNER JOIN dept d ON
e.deptno = d.deptno
WHERE d.dname = 'ACCOUNTING';

EXPLAIN FORMAT=JSON
SELECT * FROM v_empdept
WHERE dname = 'ACCOUNTING';
```

- Assign: sales database
  - create view on all three tables joined together with all columns.
  - solve join assignments on that view.

JSON = Java Script Object Notation

- Data format to represent the data
- Another data format example is XML, CSV, ...
- JSON --> C struct like -- key-value pairs

```
"table": {
    "table_name": "d",
    "access_type": "ALL",
    "rows_examined_per_scan": 4,
    "rows_produced_per_join": 1,
}
```

- { ... } --> object
- [...] --> array
- "..." --> string
- 123.34, true, null --> number, bool

DCL

```
sunbeam> SHOW DATABASES;
-- classwork, sales, hr

root> SHOW GRANTS FOR sunbeam@localhost;

root> REVOKE ALL PRIVILEGES ON hr.* FROM sunbeam@localhost;

sunbeam> SHOW DATABASES;
-- classwork, sales
```

```
root> CREATE USER 'mgr'@'%' IDENTIFIED BY 'mgr';

root> CREATE USER 'teamlead'@'localhost' IDENTIFIED BY 'teamlead';

root> CREATE USER 'dev1'@'localhost' IDENTIFIED BY 'dev1';

root> CREATE USER 'dev2'@'localhost' IDENTIFIED BY 'dev2';

root> SELECT user, host FROM mysql.user;

root> SHOW GRANTS FOR 'mgr'@'%';
-- USAGE -- no permissions

mgr> SHOW DATABASES;

root> GRANT ALL ON classwork.* TO 'mgr'@'%' WITH GRANT OPTION;

mgr> SHOW DATABASES;
mgr> GRANT INSERT, UPDATE, DELETE, SELECT ON classwork.emp TO
'teamlead'@'localhost';

mgr> GRANT INSERT, UPDATE, DELETE, SELECT ON classwork.dept TO
'teamlead'@'localhost';

mgr> GRANT INSERT, SELECT ON classwork.books TO 'teamlead'@'localhost';

root> SHOW GRANTS FOR 'teamlead'@'localhost';
```

```
teamlead> SHOW DATABASES;

teamlead> USE classwork;

teamlead> SHOW TABLES;

teamlead> DELETE FROM books;
-- error: Access denied

teamlead> INSERT INTO books VALUES (1, 'Atlas Shrugged', 'Ayn Rand', 'Novell',
727.29);

teamlead> SELECT * FROM books;
```

```
mgr> USE classwork;

mgr> CREATE VIEW v_empsummary AS
SELECT job, SUM(sal) salsum, AVG(sal) salavg FROM emp GROUP BY job;

mgr> GRANT SELECT ON classwork.v_empsummary TO 'dev1'@'localhost';

mgr> CREATE VIEW v_deptsummary AS
SELECT d.dname, SUM(e.sal) salsum, AVG(e.sal) salavg FROM emp e RIGHT JOIN dept d
ON e.deptno = d.deptno GROUP BY d.dname;

mgr> GRANT SELECT ON classwork.v_deptsummary TO 'dev1'@'localhost';

dev1> SHOW DATABASES;

dev1> USE classwork;

dev1> SHOW FULL TABLES;

dev1> SELECT * FROM v_deptsummary;

dev1> DESCRIBE v_deptsummary;
```

```
dev1> SHOW CREATE VIEW v_deptsummary;
-- error: Access denied.
```

```
root> SHOW GRANTS FOR 'teamlead'@'localhost';

root> REVOKE SELECT ON classwork.books FROM 'teamlead'@'localhost';

root> SHOW GRANTS FOR 'teamlead'@'localhost';

teamlead> SELECT * FROM books;
-- error: Access denied.
```

Transaction

```
-- sunbeam@localhost, classwork

CREATE TABLE accounts(id INT PRIMARY KEY, type CHAR(20),balance DECIMAL(9,2));

INSERT INTO accounts VALUES
(1, 'Saving', 20000.00),
(2, 'Current', 60000.00),
(3, 'Saving', 5000.00),
(4, 'Saving', 3000.00),
(5, 'Current', 50000.00),
(6, 'Saving', 10000.00);

SELECT * FROM accounts;

-- transfer Rs. 2000 from acc 1 to acc 3.
UPDATE accounts SET balance = balance - 2000
WHERE id = 1;

UPDATE accounts SET balance = balance + 2000
WHERE id = 3;

SELECT * FROM accounts;
```

- In MySQL, by default each DML operation is executed as a transaction with the single query and it is auto-committed.
- In MySQL, transaction is explicitly started using START TRANSACTION command. It is finalized using COMMIT command or disarded using ROLLBACK command.
- TCL commands
  - START TRANSACTION
  - COMMIT
  - ROLLBACK

```sql
-- transfer Rs. 5000 from acc 6 to acc 4.
START TRANSACTION;

UPDATE accounts SET balance = balance - 5000
WHERE id = 6;

SELECT * FROM accounts;

UPDATE accounts SET balance = balance + 5000
WHERE id = 4;

SELECT * FROM accounts;

COMMIT;
```

```sql
-- transfer Rs. 2000 from acc 6 to acc 4.
START TRANSACTION;

UPDATE accounts SET balance = balance - 2000
WHERE id = 6;

SELECT * FROM accounts;

UPDATE accounts SET balance = balance + 2000
WHERE id = 4;

SELECT * FROM accounts;

ROLLBACK;

SELECT * FROM accounts;
```

```sql
START TRANSACTION;

DELETE FROM accounts;

SELECT * FROM accounts;

ROLLBACK;

SELECT * FROM accounts;
```

```sql
DELETE FROM accounts;
-- delete without transaction
-- single dml query tx --> auto-committed.
-- START TX ++ DELETE ++ COMMIT.
-- all rows are deleted permanently.

ROLLBACK;
-- useless -- no current tx.

SELECT * FROM accounts;
```

- Transaction

```
START TRANSACTION;

dml1;
dml2;
dml3;

COMMIT;
-- changes of dml 1,2,3 are permanent.
-- transaction is completed.
```

```
START TRANSACTION;

dml1;
dml2;
dml3;

ROLLBACK;
-- changes of dml 1,2,3 are discarded.
-- transaction is completed.
```

- In MySQL, all DML operations are auto-commited (for each query).
- However MySQL can be configured to start transaction automatically after commit/rollback of previous transaction.

```
SELECT @@autocommit;

SET @@autocommit=0;

SELECT @@autocommit;

SELECT * FROM books;

DELETE FROM books;

ROLLBACK;

SELECT * FROM books;
```

```sql
SELECT * FROM dept;

DELETE FROM dept;

SELECT * FROM dept;

ROLLBACK;

SELECT * FROM dept;

SELECT * FROM accounts;

INSERT INTO accounts VALUES
(1, 'Saving', 20000.00),
(2, 'Current', 60000.00),
(3, 'Saving', 5000.00),
(4, 'Saving', 3000.00),
(5, 'Current', 50000.00),
(6, 'Saving', 10000.00);

SELECT * FROM accounts;

COMMIT;

SET @@autocommit=1;

SELECT @@autocommit;
```

- Transaction is set of DML queries executed as a single unit.
- Transaction is limited to same RDBMS server.

```sql
START TRANSACTION;

SELECT * FROM emp;

DELETE FROM emp WHERE empno < 1100;
-- changed in tx -- not permanent

SELECT * FROM emp;

TRUNCATE TABLE books;
-- ddl command -- current tx is auto committed.
-- DELETE FROM emp ... changes are permanent

SELECT * FROM books;

ROLLBACK;
-- has no effect ... tx is alreay commited.

SELECT * FROM emp;
```

```
START TRANSACTION;

SELECT * FROM dept;

DELETE FROM dept;

SELECT * FROM dept;

EXIT;
-- auto rollback current tx.
```

```
SELECT * FROM dept;
```