

SQL Cheat Sheet: Views, Stored Procedures and Transactions



Views

Topic	Syntax	Description	Example
Create View	<code>CREATE VIEW view_name AS SELECT column1, column2, ... FROM table_name WHERE condition;</code>	A <code>CREATE VIEW</code> is an alternative way of representing data that exists in one or more tables.	<code>CREATE VIEW EMP_SALARY AS SELECT EMP_ID, F_NAME, L_NAME, B_DATE, SEX, SALARY FROM EMPLOYEES;</code>
Update a View	<code>CREATE OR REPLACE VIEW view_name AS SELECT column1, column2, ... FROM table_name WHERE condition;</code>	The <code>CREATE OR REPLACE</code> VIEW command updates a view.	<code>CREATE OR REPLACE VIEW EMP_SALARY AS SELECT EMP_ID, F_NAME, L_NAME, B_DATE, SEX, JOB_TITLE, MIN_SALARY, MAX_SALARY FROM EMPLOYEES, JOBS WHERE EMPLOYEES.JOB_ID = JOBS.JOB_ID;</code>
Drop a View	<code>DROP VIEW view_name;</code>	Use the <code>DROP VIEW</code> statement to remove a view from the database.	<code>DROP VIEW EMP_SALARY;</code>

Stored Procedures in IBM Db2 using SQL

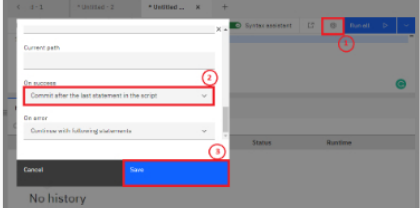
Stored Procedures	<pre>--#SET TERMINATOR @ CREATE PROCEDURE PROCEDURE_NAME LANGUAGE BEGIN END @</pre>	<p>A <code>stored procedure</code> is a prepared SQL code that you can save, so the code can be reused over and over again.</p> <p>The default terminator for a stored procedure is semicolon(;). To set a different terminator we use <code>SET TERMINATOR</code> clause followed by the terminator such as '@' .</p>	<pre>--#SET TERMINATOR @ CREATE PROCEDURE RETRIEVE_ALL LANGUAGE SQL READS SQL DATA DYNAMIC RESULT SETS 1 BEGIN DECLARE C1 CURSOR WITH RETURN FOR SELECT * FROM PETSAL; OPEN C1; END @</pre>
-------------------	---	--	--

Stored Procedures in MySQL using phpMyAdmin

Stored Procedures	<pre>DELIMITER // CREATE PROCEDURE PROCEDURE_NAME BEGIN END // DELIMITER ;</pre>	<p>A <code>stored procedure</code> is a prepared SQL code that you can save, so the code can be reused over and over again.</p> <p>The default terminator for a stored procedure is semicolon (;). To set a different terminator we use <code>DELIMITER</code> clause followed by the terminator such as \$\$ or //.</p>	<pre>DELIMITER // CREATE PROCEDURE RETRIEVE_ALL() BEGIN SELECT * FROM PETSAL; END // DELIMITER ;</pre>
-------------------	--	--	--

Transactions with Db2

Commit command	<pre>COMMIT;</pre>	<p>A <code>COMMIT command</code> is used to persist the changes in the database.</p> <p>The default terminator for a COMMIT command is semicolon (;).</p>	<pre>CREATE TABLE employee(ID INT, Name VARCHAR(20), City VARCHAR(20), Salary INT, Age INT); INSERT INTO employee(ID, Name, City, Salary, Age) VALUES(1, 'Priyanka pal', 'Nasik', 36000, 21), (2, 'Riya chowdary', 'Bangalore', 82000, 29); SELECT *FROM employee; COMMIT;</pre>
			<p>As auto-commit is enabled by default, all transactions will be committed. We need to disable this option to see how rollback works</p>

Rollback command	<pre>ROLLBACK;</pre>	<p>A ROLLBACK command is used to rollback the transactions which are not saved in the database. The default terminator for a ROLLBACK command is semicolon (;).</p>	<p>For db2, we have to disable auto-commit manually. Click the gear icon located on the right side of the SQL Assistant window. Next, select the "On Success" drop-down and choose "commit after the last statement in the script" Remember to save your changes!</p>  <pre>INSERT INTO employee VALUES (3, 'Swetha Tiwari', 'Kanpur', 38000, 38); SELECT *FROM employee; ROLLBACK; SELECT *FROM employee;</pre>
------------------	----------------------	--	--

Transactions with MySQL

Commit command	<pre>COMMIT;</pre>	<p>A COMMIT command is used to persist the changes in the database. The default terminator for a COMMIT command is semicolon (;).</p>	<pre>CREATE TABLE employee(ID INT, Name VARCHAR(20), City VARCHAR(20), Salary INT, Age INT); START TRANSACTION; INSERT INTO employee(ID, Name, City, Salary, Age) VALUES(1, 'Priyanka pal', 'Nasik', 36000, 21), (2, 'Riya chowdary', 'Bangalore', 82000, 29); SELECT *FROM employee; COMMIT;</pre>
Rollback command	<pre>ROLLBACK;</pre>	<p>A ROLLBACK command is used to rollback the transactions which are not saved in the database. The default terminator for a ROLLBACK command is semicolon (;).</p>	<p>As auto-commit is enabled by default, all transactions will be committed. We need to disable this option to see how rollback works. For MySQL use the command "SET autocommit = 0;"</p> <pre>INSERT INTO employee VALUES (3, 'Swetha Tiwari', 'Kanpur', 38000, 38); SELECT *FROM employee; ROLLBACK; SELECT *FROM employee;</pre>

Db2 Transactions using Stored Procedure

Commit command	<pre>--SET TERMINATOR @ CREATE PROCEDURE PROCEDURE_NAME BEGIN COMMIT; END @ END</pre>	<p>A COMMIT command is used to persist the changes in the database. The default terminator for a COMMIT command is semicolon (;).</p>	<pre>--SET TERMINATOR @ CREATE PROCEDURE TRANSACTION_ROSE LANGUAGE SQL MODIFIES SQL DATA ----- DECLARE SQLCODE INTEGER DEFAULT 0; DECLARE retcode INTEGER DEFAULT 0; DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET retcode = SQLCODE; UPDATE BankAccounts SET Balance = Balance-200 WHERE AccountName = 'Rose'; UPDATE BankAccounts SET Balance = Balance-300 WHERE AccountName = 'Rose'; IF retcode < 0 THEN ROLLBACK WORK; ELSE COMMIT WORK; END IF; IF retcode < 0 THEN</pre>
----------------	--	--	---

	<pre>@ </pre>		<pre>IF retcode < 0 THEN ROLLBACK WORK; ELSE COMMIT WORK; END IF; END @ </pre>
Rollback command	<pre>--#SET TERMINATOR @ CREATE PROCEDURE PROCEDURE_NAME BEGIN ROLLBACK; COMMIT; END @ </pre>	<p>A ROLLBACK command is used to rollback the transactions which are not saved in the database. The default terminator for a ROLLBACK command is semicolon (;).</p>	<pre>--#SET TERMINATOR @ CREATE PROCEDURE TRANSACTION_ROSE LANGUAGE SQL MODIFIES SQL DATA BEGIN DECLARE SQLCODE INTEGER DEFAULT 0; DECLARE retcode INTEGER DEFAULT 0; DECLARE CONTINUE HANDLER FOR SQLEXCEPTION SET retcode = SQLCODE; UPDATE BankAccounts SET Balance = Balance-200 WHERE AccountName = 'Rose'; UPDATE BankAccounts SET Balance = Balance-300 WHERE AccountName = 'Rose'; IF retcode < 0 THEN ROLLBACK WORK; ELSE COMMIT WORK; END IF; END @ </pre>

MySQL Transactions using Stored Procedure

Commit command	<pre>DELIMITER // CREATE PROCEDURE PROCEDURE_NAME BEGIN COMMIT; END // DELIMITER ; </pre>	<p>A COMMIT command is used to persist the changes in the database. The default terminator for a COMMIT command is semicolon (;).</p>	<pre>DELIMITER // CREATE PROCEDURE TRANSACTION_ROSE() BEGIN DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN ROLLBACK; RESIGNAL; END; START TRANSACTION; UPDATE BankAccounts SET Balance = Balance-200 WHERE AccountName = 'Rose'; UPDATE BankAccounts SET Balance = Balance-300 WHERE AccountName = 'Rose'; COMMIT; END // DELIMITER ; </pre>
	<pre>DELIMITER // CREATE PROCEDURE PROCEDURE_NAME </pre>		<pre>DELIMITER // CREATE PROCEDURE TRANSACTION_ROSE() BEGIN DECLARE EXIT HANDLER FOR SQLEXCEPTION BEGIN ROLLBACK; RESIGNAL; END; </pre>

Rollback command	<pre>BEGIN ROLLBACK; COMMIT; END // DELIMITER ; }</pre>	<p>A <code>ROLLBACK command</code> is used to rollback the transactions which are not saved in the database. The default terminator for a ROLLBACK command is semicolon (;).</p>	<pre>START TRANSACTION; UPDATE BankAccounts SET Balance = Balance-200 WHERE AccountName = 'Rose'; UPDATE BankAccounts SET Balance = Balance-300 WHERE AccountName = 'Rose'; COMMIT; END // } DELIMITER ;</pre>
------------------	---	--	---

Author(s)

D.M Naidu

Changelog