Training Centre Database

BY PAUL FLEMING

28/10/2019

Student ID : 880513

`

Contents

Page 3-4 Create Table Statements

Page 5-7 Inserting Into Table Statements

Page 7-7 Create Indexes

Page 8-10 Query Functionality

Page 11–11 Create View Statements

Page 11–13 Create Procedure Statements

Page 13 – 14 Create Trigger Statements

|  |
| --- |
| CREATE TABLE STATEMENTS |

/\*Drop Tables\*/

DROP TRIGGER IF EXISTS trg\_audit\_trail;

DROP TABLE IF EXISTS audit;

DROP PROCEDURE IF EXISTS assign\_schedule;

DROP VIEW IF EXISTS future\_session;

DROP TABLE IF EXISTS take;

DROP TABLE IF EXISTS delegate;

DROP TABLE IF EXISTS `session`;

DROP TABLE IF EXISTS module;

DROP TABLE IF EXISTS course;

CREATE TABLE course(

`code` CHAR(4) NOT NULL,

`name` VARCHAR(30) NOT NULL,

credits TINYINT NOT NULL,

CONSTRAINT chk\_course CHECK (credits IN (50,75,100)),

CONSTRAINT pk\_course PRIMARY KEY (`code`));

CREATE TABLE module(

`code` CHAR(2) NOT NULL,

`name` VARCHAR(30) NOT NULL,

cost DECIMAL(8,2) DEFAULT 0.0 NOT NULL,

credits TINYINT NOT NULL,

course\_code CHAR(3) NOT NULL,

CONSTRAINT chk\_module CHECK (credits IN (25,50)),

CONSTRAINT pk\_module PRIMARY KEY (`code`),

CONSTRAINT fk\_module FOREIGN KEY (course\_code) References course (`code`) ON UPDATE CASCADE ON DELETE CASCADE);

CREATE TABLE `session`(

`code` CHAR(2) NOT NULL,

`date` DATE NOT NULL,

room VARCHAR(30) NULL,

CONSTRAINT pk\_session PRIMARY KEY (`code`,`date`),

CONSTRAINT fk\_session FOREIGN KEY (`code`) references module (`code`) ON UPDATE CASCADE ON DELETE CASCADE);

CREATE TABLE delegate(

`no` INT NOT NULL,

`name` VARCHAR(30) NOT NULL,

phone VARCHAR(30) NULL,

CONSTRAINT uni\_delegate UNIQUE (phone),

CONSTRAINT pk\_delegate PRIMARY KEY (`no`));

CREATE TABLE take(

`no` INT NOT NULL,

`code` CHAR(2) NOT NULL,

grade TINYINT NULL,

CONSTRAINT pk\_take PRIMARY KEY (`no`, `code`),

CONSTRAINT fk\_take1 FOREIGN KEY (`no`) REFERENCES delegate (`no`) ON UPDATE CASCADE ON DELETE CASCADE,

CONSTRAINT fk\_take2 FOREIGN KEY (`code`) REFERENCES module (`code`) ON UPDATE CASCADE ON DELETE CASCADE);

|  |
| --- |
| INSERT TABLE STATEMENTS |

/\* Course Table \*/

INSERT INTO course (`code`, `name`, credits)

VALUES ('WSD', 'Web Systems Development', 75);

INSERT INTO course (`code`, `name`, credits)

VALUES ('DDM', 'Database Design & Management', 100);

INSERT INTO course (`code`, `name`, credits)

VALUES ('NSF', 'Network Security & Forensics', 75);

/\* Module Table \*/

INSERT INTO Module (`code`, `name`, cost, credits, course\_code)

VALUES ('A2', 'ASP.NET', 250, 25, 'WSD');

INSERT INTO module (`code`, `name`, cost, credits, course\_code)

VALUES ('A3', 'PHP', 250, 25, 'WSD');

INSERT INTO module (`code`, `name`, cost, credits, course\_code)

VALUES ('A4', 'JavaFX', 350, 25, 'WSD');

INSERT INTO module (`code`, `name`, cost, credits, course\_code)

VALUES ('B2', 'Oracle', 750, 50, 'DDM');

INSERT INTO module (`code`, `name`, cost, credits, course\_code)

VALUES ('B3', 'SQLS', 750, 50, 'DDM');

INSERT INTO module (`code`, `name`, cost, credits, course\_code)

VALUES ('C2', 'Law', 250, 25, 'NSF');

INSERT INTO module (`code`, `name`, cost, credits, course\_code)

VALUES ('C3', 'Forensics', 350, 25, 'NSF');

INSERT INTO module (`code`, `name`, cost, credits, course\_code)

VALUES ('C4', 'Networks', 250, 25, 'NSF');

/\* Session Table \*/

INSERT INTO `session` (`code`, `date`, room)

VALUES ('A2', '2019.06.05', '305');

INSERT INTO `session` (`code`, `date`, room)

VALUES ('A3', '2019.06.06', '307');

INSERT INTO `session` (`code`, `date`, room)

VALUES ('A4', '2019.06.07', '305');

INSERT INTO `session` (`code`, `date`, room)

VALUES ('B2', '2019.08.22', '208');

INSERT INTO `session` (`code`, `date`, room)

VALUES ('B3', '2019.08.23', '208');

INSERT INTO `session` (`code`,`date`, room)

VALUES ('A2', '2020.05.01', '303');

INSERT INTO `session` (`code`,`date`, room)

VALUES ('A3', '2020.05.02', '305');

INSERT INTO `session` (`code`, `date`, room)

VALUES ('A4', '2020.05.03', '303');

INSERT INTO `session` (`code`,`date`, room)

VALUES ('B2', '2020.07.10', NULL);

INSERT INTO `session` (`code`, `date`, room)

VALUES ('B3', '2020.07.11', NULL);

/\* Delegate Table \*/

INSERT INTO delegate (`no`, `name`, phone)

VALUES ('2001', 'Mike', NULL);

INSERT INTO delegate (`no`, `name`, phone)

VALUES ('2002', 'Andy', NULL);

INSERT INTO delegate (`no`, `name`, phone)

VALUES ('2003', 'Sarah', NULL);

INSERT INTO delegate (`no`, `name`, phone)

VALUES ('2004', 'Karen', NULL);

INSERT INTO delegate (`no`, `name`, phone)

VALUES ('2005', 'Lucy', NULL);

INSERT INTO delegate (`no`, `name`, phone)

VALUES ('2006', 'Steve', NULL);

INSERT INTO delegate (`no`, `name`, phone)

VALUES ('2007', 'Jenny', NULL);

INSERT INTO delegate (`no`, `name`, phone)

VALUES ('2008','Tom', NULL);

/\* Take Table \*/

INSERT INTO take (`no`, `code`, grade)

VALUES ('2003', 'A2', 68);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2003', 'A3', 72);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2003', 'A4', 53);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2005', 'A2', 48);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2005', 'A3', 52);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2002', 'A2', 20);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2002','A3',30);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2002','A4',50);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2008','B2',90);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2007','B2',73);

INSERT INTO take (`no`, `code`, grade)

VALUES ('2007','B3',63);

|  |
| --- |
| CREATING INDEXES |

/\* Creating an Index on the Table called session\*/

CREATE UNIQUE INDEX takeIndex ON take (`no`,`code`);

/\* Creating an Index on the Table called take\*/

CREATE INDEX sessionIndex ON `session` (`code`);

/\* Creating an Index on the Table called module\*/

CREATE INDEX moduleIndex ON module (course\_code);

|  |
| --- |
| QUERY FUNCTIONALITY |

/\* Question 1: Fetch Every modules code, name and credits \*/

SELECT `code`, `name`, credits

FROM module;

/\* Question 2: Fetch Every Delegates no and name in descending order by name\*/

SELECT `no`, `name`

FROM delegate

ORDER BY `name` DESC;

/\* Question 3: Fetch the Course's code, name and credits where the name contains the string "Network". \*/

SELECT `code`, `name`, credits

FROM course

WHERE `name` LIKE 'Network%';

/\* Question 4 : Calculate the highest grade in any module \*/

SELECT MAX(grade) As 'highest grade'

FROM take;

/\* Question 5 : Modify the query from Question 4 to now fetch only the delegate no\*/

SELECT `no`

FROM take

WHERE grade = (SELECT MAX(grade) FROM take);

/\* Question 6: Modify the query from Question 5 to also fetch the delegate name \*/

SELECT `no`, `name`

FROM delegate

WHERE `no` = (SELECT `no` FROM take WHERE grade = (SELECT MAX(grade) FROM take));

/\* Question 7: Fetch the session's code and date for sessions which are running in the next year and for which no room has been allocated. \*/

SELECT `code`, `date`, room

FROM session

WHERE `date` >= CURRENT\_DATE()AND `date` <= DATE\_ADD(CURRENT\_DATE(), INTERVAL 1 YEAR) AND (room IS NULL);

/\* Question 8: Fetch the delegate's no and name along with the modules code and name for delegates who have taken a module but have a failing grade. \*/

SELECT D.`no`, D.`name`, M.`code`, M.`name`, T.grade

FROM delegate D INNER JOIN take T

ON D.`no` = T.`no`

INNER JOIN module M

ON T.`code` = M.`code`

WHERE T.grade < 40;

/\* Question 9: Solve the problem from question 6 using JOINS where possible \*/

SELECT D.`no`, D.`name`, T.grade -- pasted on assignment complete

FROM delegate D INNER JOIN take T

ON D.`no` = T.`no`

WHERE grade = ((SELECT MAX(grade) FROM take));

/\* Question 10: Calculate and display every delegate's no and name along with their attained credits versus the course's code, name and credits \*/

SELECT D.`no`, D.`name`, C.`code`, C.`name`, C.credits, SUM(M.credits) As 'attained credits'

FROM delegate D INNER JOIN take T

ON D.`no` = T.`no`

INNER JOIN module M

ON T.`code` = M.`code`

INNER JOIN course C

ON M.course\_code = C.`code`

WHERE T.grade >= 40

GROUP BY D.`no`, D.`name`, C.`code`, C.`name`, C.credits;

/\* Question 11 Modify the query from question 10 to only show a delegate when they have attained the course's credits \*/

SELECT D.`no`, D.`name`, C.`code`, C.`name`, C.credits, SUM(M.credits) As 'attained credits'

FROM delegate D INNER JOIN take T

ON D.`no` = T.`no`

INNER JOIN module M

ON T.`code` = M.`code`

INNER JOIN course C

ON M.course\_code = C.`code`

WHERE T.grade >= 40

GROUP BY D.`no`, D.`name`, C.`code`, C.`name`, C.credits

HAVING SUM(M.credits) >= C.credits;

|  |
| --- |
| CREATE VIEW STATEMENTS |

/\* CREATE VIEW \*/

/\* Produce a view that returns the sessions in the future and rejects any attempt to insert or update sessions in the past and write a simple statement to test reject \*/

CREATE VIEW future\_session -- pasted on assignment

AS

SELECT `code`, `date`, room

FROM `session`

WHERE `date` > CURDATE()

WITH CHECK OPTION;

INSERT INTO future\_session(`code`, `date`, room)

VALUES ('A2', '2016,08.01', '202'); /\*Rejects date in the past\*/

|  |
| --- |
| CREATE PROCEDURE STATEMENTS |

/\* Create a Procedure \*/

DELIMITER $$

CREATE PROCEDURE assign\_schedule (IN assign\_code CHAR(3), IN assign\_date DATE)

BEGIN

DECLARE complete BOOLEAN DEFAULT FALSE;

DECLARE new\_code CHAR(2);

DECLARE module\_cursor CURSOR FOR SELECT `code` FROM module WHERE course\_code = assign\_code;

DECLARE CONTINUE HANDLER FOR NOT FOUND

SET complete = TRUE;

IF NOT EXISTS (SELECT `code` FROM course WHERE `code` = assign\_code) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Course does not exist';

END IF;

IF assign\_date < (DATE\_ADD(CURDATE(), INTERVAL 1 MONTH)) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'Date is not valid, the start date must be at least a month in the future';

END IF;

OPEN module\_cursor;

`loop` : LOOP

FETCH NEXT FROM module\_cursor INTO new\_code;

IF complete THEN

LEAVE `loop`;

ELSE

IF WEEKDAY(assign\_date) = 5 THEN

SET assign\_Date = DATE\_ADD(assign\_date, INTERVAL 2 day); /\* IF SATURDAY MOVE TO MONDAY \*/

ELSEIF WEEKDAY(assign\_date) = 6 THEN

SET assign\_date = DATE\_ADD(assign\_date, INTERVAL 1 day); /\* IF SUNDAY MOVE TO MONDAY \*/

END IF;

INSERT INTO `session`(`code`, `date`, room)

VALUES (new\_code, assign\_date, NULL);

END IF;

IF(assign\_date) = (assign\_date) THEN

SET assign\_date = DATE\_ADD(assign\_date, INTERVAL 1 day);

END IF;

END LOOP;

CLOSE module\_cursor;

END $$

DELIMITER ;

CALL assign\_schedule('WSA','2019-12-05'); /\* Course does not exist \*/

CALL assign\_schedule('NSF','2019-10-24'); /\* Date is invalid \*/

Call assign\_schedule ('NSF', '2022-01-08'); /\* Check Saturday & Skip Date to Monday \*/

Call assign\_schedule ('NSF', '2022.01.16'); /\* Check Sunday & Skip Date to Monday \*/

SELECT \*

FROM `session`;

|  |
| --- |
| Create Trigger Statements |

/\*Create Trigger\*/

CREATE TABLE audit (

audit\_no INT NOT NULL AUTO\_INCREMENT,

`no` INT NOT NULL,

`code` CHAR(3) NOT NULL,

old\_grade TINYINT NULL,

new\_grade TINYINT NULL,

user\_name VARCHAR(30) NOT NULL,

system\_user VARCHAR(30) NOT NULL,

date\_time DATETIME NOT NULL,

CONSTRAINT pk\_audit PRIMARY KEY (audit\_no));

DELIMITER $$

CREATE TRIGGER trg\_audit\_trail

AFTER UPDATE ON take

FOR EACH ROW

BEGIN

IF !(OLD.grade <=> NEW.grade) THEN

INSERT INTO audit

(`no`, `code`, old\_grade, new\_grade, user\_name, system\_user, date\_time)

VALUES (OLD.`no`, OLD.`code`, OLD.grade, NEW.grade, CURRENT\_USER(), SYSTEM\_USER(), SYSDATE() );

END IF;

END $$

DELIMITER ;

UPDATE take SET grade = 30 WHERE `no` = 2008 AND `code` = 'B2';

SELECT \*

FROM audit;