|  |  |
| --- | --- |
| **Homework 4** | **Basic SQL** |
| **Due Wed, Oct 7 at 11:30 pm** | **Objectives:** Practice of Basic SQL |

**CS 4347: Database Systems Alex Lundin aml140830**

**6.6.** Repeat Exercise 6.5, but use the AIRLINE database schema of Figure 5.8.

(a) What are the referential integrity constraints that should hold on the schema?

The following referential integrity constraints should hold (we use the notation:

**R.(A1, ..., An) --> S.(B1, ..., Bn)**

R is the referencing relation with a foreign key containing attributes (A1, ..., An)

S is the referenced relation with a primary key containing attributes (B1, …, Bn)

FLIGHT\_LEG.(FLIGHT\_NUMBER) --> FLIGHT.(NUMBER)

FLIGHT\_LEG.(DEPARTURE\_AIRPORT\_CODE) --> AIRPORT.(AIRPORT\_CODE)

FLIGHT\_LEG.(ARRIVAL\_AIRPORT\_CODE) --> AIRPORT.(AIRPORT\_CODE)

LEG\_INSTANCE.(FLIGHT\_NUMBER, LEG\_NUMBER) --> FLIGHT\_LEG.(FLIGHT\_NUMBER, LEG\_NUMBER)

LEG\_INSTANCE.(AIRPLANE\_ID) --> AIRPLANE.(AIRPLANE\_ID)

LEG\_INSTANCE.(DEPARTURE\_AIRPORT\_CODE) --> AIRPORT.(AIRPORT\_CODE)

LEG\_INSTANCE.(ARRIVAL\_AIRPORT\_CODE) --> AIRPORT.(AIRPORT\_CODE)

FARES.(FLIGHT\_NUMBER) --> FLIGHT.(NUMBER)

CAN\_LAND.(AIRPLANE\_TYPE\_NAME) --> AIRPLANE\_TYPE.(TYPE\_NAME)

CAN\_LAND.(AIRPORT\_CODE) --> AIRPORT.(AIRPORT\_CODE)

AIRPLANE.(AIRPLANE\_TYPE) --> AIRPLANE\_TYPE.(TYPE\_NAME)

SEAT\_RESERVATION.(FLIGHT\_NUMBER, LEG\_NUMBER, DATE) -->LEG\_INSTANCE.(FLIGHT\_NUMBER, LEG\_NUMBER, DATE)

(b) Write appropriate SQL DDL statements to define the database.

­­ CREATE TABLE AIRPORT

(Airport\_code INT NOT NULL,

Name VARCHAR(40) NOT NULL,

City VARCHAR(40) NOT NULL,

State VARCHAR(2) NOT NULL,

PRIMARY KEY(Airport\_code));

CREATE TABLE FLIGHT

(Flight\_number INT NOT NULL,

Airline VARCHAR(40) NOT NULL,

Weekdays VARCHAR(40) NOT NULL,

PRIMARY KEY(Flight\_number));

CREATE TABLE FLIGHT\_LEG

(Flight\_number INT NOT NULL,

Leg\_number INT NOT NULL,

Departure\_airport\_code INT NOT NULL,

Scheduled\_departure\_time TIMESTAMP(6) NOT NULL,

Arrival\_airport\_code INT NOT NULL,

Scheduled\_arrival\_time TIMESTAMP(6) NOT NULL,

PRIMARY KEY(Flight\_number, Leg\_number),

FOREIGN KEY(Flight\_number) REFERENCES FLIGHT(Flight\_number),

UNIQUE(Leg\_number));

CREATE TABLE LEG\_INSTANCE

(Flight\_number INT NOT NULL,

Leg\_number INT NOT NULL,

Date1 DATE NOT NULL,

Number\_of\_avaiable\_seats INT NOT NULL,

Airplane\_id INT NOT NULL,

Departure\_airport\_code INT NOT NULL,

Departure\_time TIMESTAMP(6) NOT NULL,

Arrival\_airport\_code INT NOT NULL,

Arrival\_time TIMESTAMP(6) NOT NULL,

PRIMARY KEY(Flight\_number, Leg\_number, Date1),

FOREIGN KEY(Flight\_number) REFERENCES FLIGHT(Flight\_number),

FOREIGN KEY(Leg\_number) REFERENCES FLIGHT\_LEG(Leg\_number));

CREATE TABLE FARE

(Flight\_number INT NOT NULL,

Fare\_code INT NOT NULL,

Amount DECIMAL(10,2) NOT NULL,

Restrictions VARCHAR(40) NOT NULL,

PRIMARY KEY(Flight\_number,Fare\_code),

FOREIGN KEY(Flight\_number) REFERENCES FLIGHT(Flight\_number),

UNIQUE(Fare\_code));

CREATE TABLE AIRPLANE\_TYPE

(Airplane\_type\_name VARCHAR(40) NOT NULL,

Max\_seats INT NOT NULL,

Company VARCHAR(40) NOT NULL,

PRIMARY KEY(Airplane\_type\_name));

CREATE TABLE CAN\_LAND

(Airport\_code INT NOT NULL,

Airplane\_type\_name VARCHAR(40) NOT NULL,

PRIMARY KEY(Airport\_code,Airplane\_type\_name),

FOREIGN KEY(Airport\_code) REFERENCES AIRPORT(Airport\_code),

FOREIGN KEY(Airplane\_type\_name) REFERENCES AIRPLANE\_TYPE(Airplane\_type\_name));

CREATE TABLE AIRPLANE

(Airplane\_id INT NOT NULL,

Total\_number\_of\_seats INT NOT NULL,

Airplane\_type VARCHAR(40) NOT NULL,

PRIMARY KEY(Airplane\_id));

CREATE TABLE SEAT\_RESERVATION

(Flight\_number INT NOT NULL,

Leg\_number INT NOT NULL,

Date1 DATE NOT NULL,

Seat\_number INT NOT NULL,

Customer\_name VARCHAR(40) NOT NULL,

Customer\_phone VARCHAR(40) NOT NULL,

PRIMARY KEY(Flight\_number,Leg\_number, Date1, Seat\_number),

FOREIGN KEY(Flight\_number) REFERENCES FLIGHT(Flight\_number),

FOREIGN KEY(Leg\_number) REFERENCES FLIGHT\_LEG(Leg\_number),

UNIQUE(Seat\_number));

**6.8.** Write appropriate SQL DDL statements for declaring the LIBRARY relational database schema of Figure 6.6. Specify the keys and referential triggered actions.

CREATE TABLE PUBLISHER

(Name VARCHAR(20) NOT NULL,

Address VARCHAR(40) NOT NULL,

Phone CHAR(12),

PRIMARY KEY (Name)

--ON DELETE CASCADE

--ON UPDATE CASCADE

);

CREATE TABLE BOOK

(BookId CHAR(20) NOT NULL,

Title VARCHAR(30) NOT NULL,

PublisherName VARCHAR(20),

PRIMARY KEY (BookId),

--ON DELETE CASCADE

--ON UPDATE CASCADE

FOREIGN KEY (PublisherName) REFERENCES PUBLISHER (Name)

ON DELETE CASCADE

--ON UPDATE CASCADE

);

CREATE TABLE BOOK\_AUTHORS

(BookId CHAR(20) NOT NULL,

AuthorName VARCHAR(30) NOT NULL,

PRIMARY KEY (BookId, AuthorName),

--ON DELETE CASCADE

--ON UPDATE CASCADE

FOREIGN KEY (BookId) REFERENCES BOOK (BookId)

ON DELETE CASCADE

--ON UPDATE CASCADE

);

CREATE TABLE BOOK\_COPIES

(BookId CHAR(20) NOT NULL,

BranchId INTEGER NOT NULL,

No\_Of\_Copies INTEGER NOT NULL,

PRIMARY KEY (BookId, BranchId),

--ON DELETE CASCADE

--ON UPDATE CASCADE

FOREIGN KEY (BookId) REFERENCES BOOK (BookId),

FOREIGN KEY (BranchId) REFERENCES LIBRARY\_BRANCH (BranchId)

ON DELETE CASCADE

--ON UPDATE CASCADE

);

CREATE TABLE BORROWER

(CardNo INTEGER NOT NULL,

Name VARCHAR(30) NOT NULL,

Address VARCHAR(40) NOT NULL,

Phone CHAR(12),

PRIMARY KEY (CardNo)

--ON DELETE CASCADE

--ON UPDATE CASCADE

);

CREATE TABLE BOOK\_LOANS

(CardNo INTEGER NOT NULL,

BookId CHAR(20) NOT NULL,

BranchId INTEGER NOT NULL,

DateOut DATE NOT NULL,

DueDate DATE NOT NULL,

PRIMARY KEY (CardNo, BookId, BranchId),

--ON DELETE CASCADE

--ON UPDATE CASCADE

FOREIGN KEY (CardNo) REFERENCES BORROWER (CardNo),

FOREIGN KEY (BranchId) REFERENCES LIBRARY\_BRANCH (BranchId),

FOREIGN KEY (BookId) REFERENCES BOOK (BookId)

ON DELETE CASCADE

);

CREATE TABLE LIBRARY\_BRANCH

(BranchId INTEGER NOT NULL,

BranchName VARCHAR(20) NOT NULL,

Address VARCHAR(40) NOT NULL,

PRIMARY KEY (BranchId)

--ON DELETE CASCADE

--ON UPDATE CASCADE

);

**Triggers**

**The delete triggers for foreign keys are handled in the table DDL code above ^ with the ON DELETE CASCADE statement.**

**This is an example of the AFTER UPDATE triggers for BOOKID, this same pattern is applied to all tables on the primary keys.**

CREATE OR REPLACE TRIGGER BOOKID

AFTER UPDATE OF BOOKID ON BOOK FOR EACH ROW

BEGIN

-- update BOOK\_AUTHORS table with new foreign key

-- the new. and old. are part of the trigger variables under the hood

-- you don't have to define new and old, you can use them in your trigger though

UPDATE BOOK\_AUTHORS

SET bookid = :new.bookid

WHERE bookid = :old.bookid;

END;

**Trigger Test Set**

This is the test script I used to test my Trigger above

--- start new inserts

INSERT INTO "CS4347F18"."PUBLISHER" (NAME, ADDRESS, PHONE) VALUES ('pub1', 'addy1', '1235557729');

INSERT INTO "CS4347F18"."PUBLISHER" (NAME, ADDRESS, PHONE) VALUES ('pub2', 'addy2', '1235557729');

INSERT INTO "CS4347F18"."PUBLISHER" (NAME, ADDRESS, PHONE) VALUES ('pub3', 'addy3', '1235557729');

INSERT INTO "CS4347F18"."PUBLISHER" (NAME, ADDRESS, PHONE) VALUES ('pub4', 'addy4', '1235557729');

INSERT INTO "CS4347F18"."BOOK" (BOOKID, TITLE, PUBLISHERNAME) VALUES ('b1', 'Art of War', 'pub1');

INSERT INTO "CS4347F18"."BOOK" (BOOKID, TITLE, PUBLISHERNAME) VALUES ('b2', 'Think and Grow Rich', 'pub2');

INSERT INTO "CS4347F18"."BOOK\_AUTHORS" (BOOKID, AUTHORNAME) VALUES ('b1', 'Sun Tzu');

INSERT INTO "CS4347F18"."BOOK\_AUTHORS" (BOOKID, AUTHORNAME) VALUES ('b2', 'Napoleon Hill');

--- update query

UPDATE BOOK SET BOOKID = 'b90'

WHERE BOOKID = 'b1';

--- delete query

DELETE FROM BOOK

WHERE BOOKID = 'b90';

**6.9.** How can the key and foreign key constraints be enforced by the DBMS? Is the enforcement technique you suggest difficult to implement? Can the constraint checks be executed efficiently when updates are applied to the database?

Index the attributes that make each key. Before any insert, check all indexes to see if the key already exists.

Same thing for updates and deletes. Use the index to check all relations to see if either operation will cause conflicts.

**6.12.** Specify the following queries in SQL on the database schema of Figure 1.2.

(a) Retrieve the names of all senior students majoring in 'COSC' (computer science).  
SELECT Name

FROM STUDENT

WHERE Major='COSC'

(b) Retrieve the names of all courses taught by professor King in 85 and 86.

SELECT CourseName

FROM COURSE, SECTION

WHERE COURSE.CourseNumber=SECTION.CourseNumber AND Instructor='King'

AND (Year='85' OR Year='86')

(c) For each section taught by professor King, retrieve the course number, semester, year, and number of students who took the section.

SELECT CourseNumber, Semester, Year, COUNT(\*)

FROM SECTION, GRADE\_REPORT

WHERE Instructor='King' AND SECTION.SectionIdentifier=GRADE\_REPORT.SectionIdentifier

GROUP BY CourseNumber, Semester, Year

(d) Retrieve the name and transcript of each senior student (Class=5) majoring in COSC. Transcript includes course name, course number, credit hours, semester, year, and grade for each course completed by the student.  
SELECT Name, CourseName, C.CourseNumber, CreditHours, Semester, Year, Grade

FROM STUDENT ST, COURSE C, SECTION S, GRADE\_REPORT G

WHERE Class=5 AND Major='COSC' AND ST.StudentNumber=G.StudentNumber AND

G.SectionIdentifier=S.SectionIdentifier AND S.CourseNumber=C.CourseNumber

(e) Retrieve the names and major departments of all straight A students (students who have a grade of A in all their courses).  
SELECT Name, Major

FROM STUDENT

WHERE NOT EXISTS ( SELECT \*

FROM GRADE\_REPORT

WHERE StudentNumber= STUDENT.StudentNumber AND NOT(Grade='A'))

(f) Retrieve the names and major departments of all students who do not have any grade of A in any of their courses.

SELECT Name, Major

FROM STUDENT

WHERE NOT EXISTS ( SELECT \*

FROM GRADE\_REPORT

WHERE StudentNumber= STUDENT.StudentNumber AND Grade='A' )

**6.13.** Write SQL update statements to do the following on the database schema shown in Figure 1.2.

(a) INSERT INTO STUDENT

VALUES ('Johnson', 25, 1, 'MATH')

(b) UPDATE STUDENT

SET CLASS = 2

WHERE Name='Smith'

(c) INSERT INTO COURSE

VALUES ('Knowledge Engineering','COSC4390', 3,'COSC')

(d) DELETE FROM STUDENT

WHERE Name='Smith' AND StudentNumber=17

**6.16.** Write SQL statements to create a table EMPLOYEE\_BACKUP to back up the EMPLOYEE table shown in Figure 5.6.

create table EMPLOYEE\_BACKUP as select \* from EMPLOYEE where 1=0;