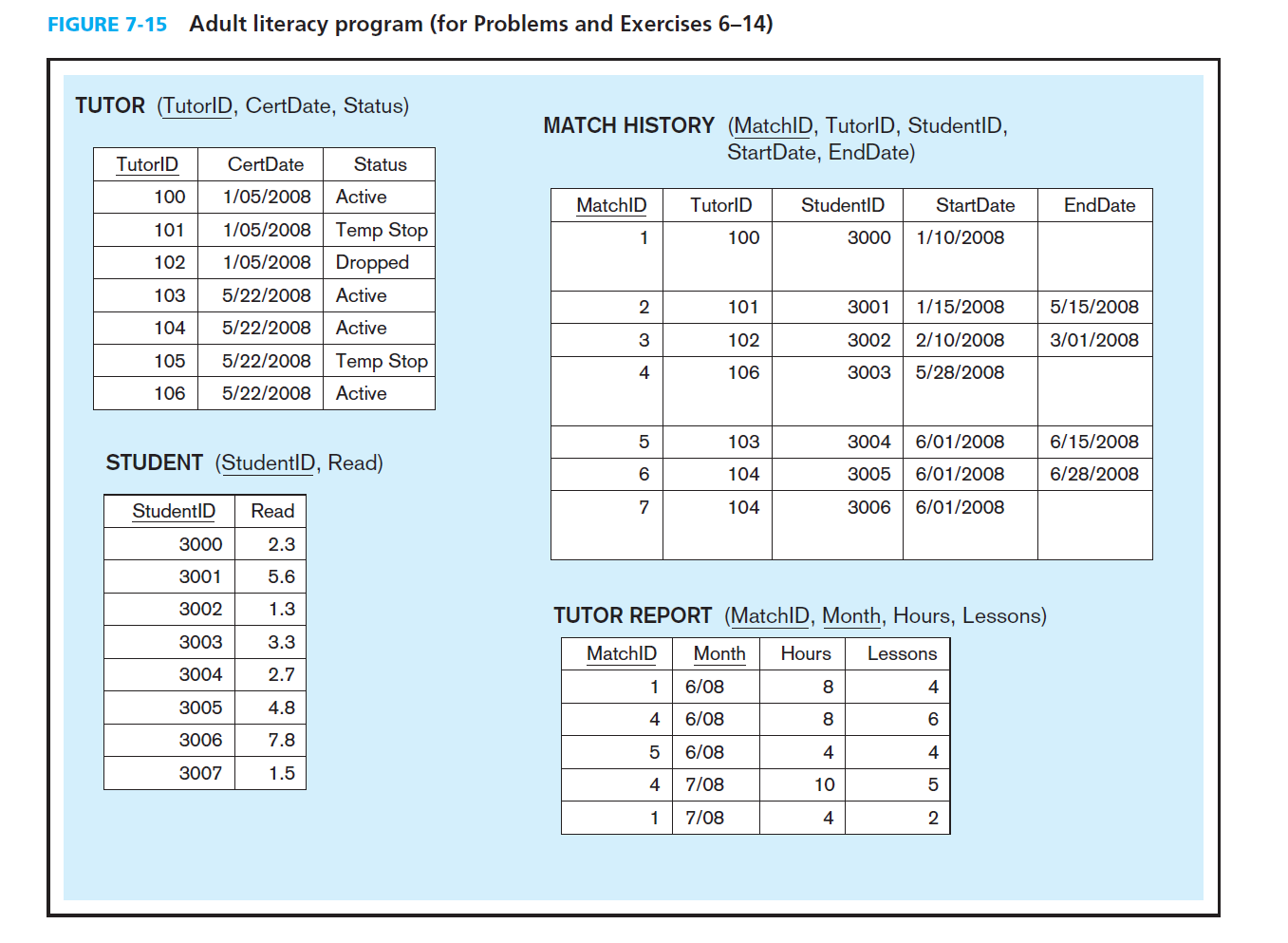
Reading:

* chapter 8: “Stored Procedures and User-Defined Functions” (Petkovic)
* chapter 7: “Advanced SQL” Routines (Hoffer, Ramesh, & Topi)

Homework:



1. Create a database named ““AdultLiteracy” on your RDBMS environment. Using Figure 7-5 above, write DDL commands to create table structures for each entity above. Name your tables the following names: Tutor, Student, MatchHistory, TutorReport

/\* script to create the database from Figure 7-15\*/

create database AdultLiteracy;

go

use AdultLiteracy;

go

create table Tutor

(TutorID int not null PRIMARY KEY,

CertDate date,

[Status] varchar(25)

);

GO

CREATE TABLE Student

( StudentID int not null primary key,

[Read] decimal(2,1)

);

go

CREATE TABLE MatchHistory

( MatchID int not null primary key,

TutorID int not null,

StudentID int not null,

StartDate date,

EndDate date,

CONSTRAINT FK\_TutorID FOREIGN KEY(TutorID) REFERENCES Tutor (TutorID),

CONSTRAINT FK\_StudentID FOREIGN KEY(StudentID) REFERENCES Student (StudentID)

);

go

CREATE TABLE TutorReport

( MatchID int not null,

[Month] DATE not null,

[Hours] INT,

Lessons int,

CONSTRAINT PK\_TutorReport PRIMARY KEY (MatchID,[Month])

);

1. Write SQL scripts to insert sample data from Fig 7-5 into the database.

INSERT INTO Tutor (TutorID, CertDate, [Status]) values (100, '1/5/2008', 'Active');

INSERT INTO Tutor (TutorID, CertDate, [Status]) values (101, '1/5/2008', 'Temp Stop');

INSERT INTO Tutor (TutorID, CertDate, [Status]) values (102, '1/5/2008', 'Dropped');

INSERT INTO Tutor (TutorID, CertDate, [Status]) values (103, '5/22/2008', 'Active');

INSERT INTO Tutor (TutorID, CertDate, [Status]) values (104, '5/22/2008', 'Active');

INSERT INTO Tutor (TutorID, CertDate, [Status]) values (105, '5/22/2008', 'Temp Stop');

INSERT INTO Tutor (TutorID, CertDate, [Status]) values (106, '5/22/2008', 'Active');

GO

INSERT INTO Student (StudentID,[Read]) Values (3000, 2.3);

INSERT INTO Student (StudentID,[Read]) Values (3001, 5.6);

INSERT INTO Student (StudentID,[Read]) Values (3002, 1.3);

INSERT INTO Student (StudentID,[Read]) Values (3003, 3.3);

INSERT INTO Student (StudentID,[Read]) Values (3004, 2.7);

INSERT INTO Student (StudentID,[Read]) Values (3005, 4.8);

INSERT INTO Student (StudentID,[Read]) Values (3006, 7.8);

INSERT INTO Student (StudentID,[Read]) Values (3007, 1.5);

GO

INSERT INTO MatchHistory VALUES (1, 100, 3000, '1/10/2008', NULL);

INSERT INTO MatchHistory VALUES (2, 101, 3001, '1/15/2008', '5/15/2008');

INSERT INTO MatchHistory VALUES (3, 102, 3002, '2/10/2008', '3/01/2008');

INSERT INTO MatchHistory VALUES (4, 106, 3003, '5/28/2008', NULL);

INSERT INTO MatchHistory VALUES (5, 103, 3004, '6/1/2008', '6/15/2008');

INSERT INTO MatchHistory VALUES (6, 104, 3005, '6/1/2008', '6/28/2008');

INSERT INTO MatchHistory VALUES (7, 104, 3006, '6/1/2008', NULL);

GO

INSERT INTO TutorReport VALUES (1, '6/30/2008', 8, 4);

INSERT INTO TutorReport VALUES (4, '6/30/2008', 8, 6);

INSERT INTO TutorReport VALUES (5, '6/30/2008', 4, 4);

INSERT INTO TutorReport VALUES (4, '7/31/2008', 10, 5);

INSERT INTO TutorReport VALUES (1, '7/31/2008', 4, 2);

1. 7. Write the SQL command to add MATH SCORE to the STUDENT table.

ALTER TABLE Student Add Math decimal(2,1);

1. 8. Write the SQL command to add SUBJECT to TUTOR. The only values allowed for SUBJECT will be Reading, Math, and ESL.

ALTER TABLE Tutor Add [Subject] varchar(25) CONSTRAINT CK\_SUBJECT CHECK ( [Subject] IN ('Reading', 'Math','ESL') );

1. 9. What do you need to do if a tutor signs up and wants to tutor in both reading and math? (Don’t need to write SQL).

One approach is to adjust the database design to allow tutors the ability to declare more than one subject preference for tutoring. Based on the prior specifications, the business rule appeared to be that tutors only tutored in one subject, thus the subject could be stored with tutor information. Under this new requirement, the database needs the ability to track more than one subject per tutor. This can be accomplished by adding two tables to the current design (TutorSubject, Subject) and by altering the structure of the MatchHistory table to track the Subject involved in each match of Tutor and Student, as noted in the revised ERD.

1. 10. Write the SQL command to find any tutors who have not submitted a report for July.

-- method 1, using left join

SELECT mh.MatchID, mh.EndDate,

mh.TutorID

FROM MatchHistory mh LEFT JOIN TutorReport tr ON mh.MatchID = tr.MatchID

and tr.[Month] between '7/1/2008' and '7/31/2008'

WHERE (mh.EndDate is null or mh.EndDate>='6/30/2008' )-- active tutoring only

and tr.[Month] is null --- no report for July

ORDER BY mh.MatchID, tr.[Month];

-- method 2, using correlaated subquery

SELECT mh.MatchID, mh.EndDate, mh.TutorID

FROM MatchHistory mh

WHERE (mh.EndDate is null or mh.EndDate>='6/30/2008' )-- active tutors only

and NOT EXISTS

( SELECT 1 FROM TutorReport tr

WHERE mh.MatchID = tr.MatchID

AND tr.[Month] between '7/1/2008' and '7/31/2008'

)

ORDER BY mh.MatchID;

1. Where do you think student and tutor information such as name, address, phone, and e-mail should be kept? Write the necessary SQL commands to capture this information. Make up sample data to populate your newly created table.

Since the newly provided information contains attributes for both Tutor and Student, it makes sense to generalize to a supertype Person.

CREATE TABLE Person (

PersonID int not null

Constraint PERPERSID\_PK PRIMARY KEY,

LastName Varchar(25) not null,

FirstName Varchar(25) not null,

MiddleInit Varchar(1) null,

StrAddress Varchar(20),

City Varchar(20),

State Char(2),

Zip Varchar(10),

Phone Varchar(14),

EMail Varchar(25),

[Type] Char(1));

GO

INSERT INTO Person

VALUES ( 100, 'Yu','Ze', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'ze@gmail.com','S');

INSERT INTO Person

VALUES ( 101, 'Batato','Michel', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'michel@gmail.com','S');

INSERT INTO Person

VALUES ( 102, 'Cai','Chen', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'cai@gmail.com','S');

INSERT INTO Person

VALUES ( 103, 'Gulati','Rohit', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Rohit@gmail.com','S');

INSERT INTO Person

VALUES ( 104, 'Nan','Jiang', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'jiang@gmail.com','S');

INSERT INTO Person

VALUES ( 105, 'Ziye','Ling', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Ling@gmail.com','S');

INSERT INTO Person

VALUES ( 106, 'Ojha','Keerti', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Keerti@gmail.com','S');

go

INSERT INTO Person

VALUES ( 3000, 'Jain','Shaily', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Shaily@gmail.com','S');

INSERT INTO Person

VALUES ( 3001, 'Koli','Vivek', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Koli@gmail.com','S');

INSERT INTO Person

VALUES ( 3002, 'Kori','Preeti', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Preeti@gmail.com','S');

INSERT INTO Person

VALUES ( 3003, 'Kumar','Aishwarya', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Kumar@gmail.com','S');

INSERT INTO Person

VALUES ( 3004, 'Luo','Yitian', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Yitian@gmail.com','S');

INSERT INTO Person

VALUES ( 3005, 'Mora','Jeffrey', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Jeffrey@gmail.com','S');

INSERT INTO Person

VALUES ( 3006, 'Lemos','Stanny', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Lemos@gmail.com','S');

INSERT INTO Person

VALUES ( 3007, 'Pal','Pryanka', NULL,'360 Huntington Ave', 'Boston', 'MA', '02115',

'6173732000', 'Pal@gmail.com','S');

go

/\* ADD FK References to Person\*/

ALTER TABLE Tutor ADD CONSTRAINT FK\_TutorID1

FOREIGN KEY (TutorID) REFERENCES Person(PersonID);

ALTER TABLE Student ADD CONSTRAINT FK\_StudentID1

FOREIGN KEY (StudentID) REFERENCES Student(StudentID);

go

1. List all active students in June by name. (the names you made up above). Include the number of hours students received tutoring and how many lessons they completed. Write the SQL command.

SELECT s.StudentID, mh.EndDate, p.LastName, SUM(tr.[Hours]) AS [Total Hours],

SUM(tr.Lessons) AS [Total Lessons]

FROM Person p

JOIN Student s ON p.PersonID = s.StudentID

JOIN MatchHistory mh ON s.StudentID = mh.StudentID

LEFT JOIN TutorReport tr ON mh.MatchID = tr.MatchID

WHERE (mh.EndDate IS NULL OR mh.EndDate>'6/30/2008')

GROUP BY s.StudentID, mh.EndDate, p.LastName;

1. Which tutors, by name, are available to tutor? Write the SQL command.

This answer assumes any active tutor may be

available to accept a new student: \*/

SELECT Person.LastName, Person.FirstName, Tutor.[Status]

FROM Person JOIN Tutor ON

Person.PersonID = Tutor.TutorID

WHERE Tutor.[Status]='Active';

/\*This answer assumes a tutor is available only if

currently unassigned a student:

\*/

SELECT T.TutorID, Person.LastName, Person.FirstName

FROM Person

JOIN Tutor AS T ON Person.PersonID = T.TutorID

WHERE T.TutorID NOT IN

(SELECT MH.TutorID FROM MatchHistory MH

WHERE EndDate IS NULL)

AND T.[STATUS]='Active';

1. Which tutor needs to be reminded to turn in reports? Write the SQL command

/\* What Needs to be turned in\*/

SELECT p.FirstName, p.LastName, mh.MatchID, mh.EndDate, mh.TutorID

FROM MatchHistory mh

JOIN Person p on p.PersonID =mh.TutorID

WHERE (mh.EndDate is null )-- active tutoring only

and NOT EXISTS

( SELECT 1 FROM TutorReport tr

WHERE mh.MatchID = tr.MatchID

AND tr.[Month] between '7/1/2008' and '7/31/2008'

)

ORDER BY mh.MatchID;

1. Create a stored procedure that returns a result set of all tutors that are available to tutor. No input parameters required

CREATE PROCEDURE AvailablbleTutors AS

BEGIN

SELECT Person.LastName, Person.FirstName, Tutor.[Status]

FROM Person JOIN Tutor ON

Person.PersonID = Tutor.TutorID

WHERE Tutor.[Status]='Active';

END

1. Create a User Define Function which determine if a specific tutor is available to tutor. The function takes a TutorID as an input value and returns a scaler value of ‘Y’ or ‘N’ corresponding to the tutor availability

CREATE FUNCTION IsTutorAvailable (@TutorID int)

returns char(1)

AS

BEGIN

RETURN

(

SELECT CASE WHEN Status ='Active' THEN 'Y' ELSE 'N' END

FROM Tutor WHERE TutorID = @TutorID

)

END;