Assignment for Module 3

The graded problems in Module 3 provide experience with the CREATE TABLE statement. You can use Oracle or MySQL for this assignment. If you use Oracle, you will need to use the Oracle SQL Developer to connect to an Oracle server.

1. Basic CREATE TABLE Statement Requirements

You should use the table descriptions in the Intercollegiate Database background document. You must use the same table and column names as specified in the background document. Here is some advice about data type selections.

- You should use standard SQL data types specified in the notes except for using VARCHAR2 (an Oracle data type) instead of VARCHAR for columns containing varying length character strings. For MySQL, you should use VARCHAR for variable length strings.
- For primary key fields (*CustNo*, *LocNo*, *EventNo*, *PlanNo*, *EmpNo*, *ResNo*, and *FacNo*), use the VARCHAR (or VARCHAR2 in Oracle) data type with length 8. For consistency, corresponding foreign keys (such as *EventRequest.CustNo*) should also be the same data type and length.
- For Oracle, you should use the DATE data type for the columns involving dates or times. The *EventPlanLine.TimeStart* and *EventPlanLine.TimeEnd* columns will store both date and time data so you should use the DATE data type. In MySQL use the DATE data type for columns with just date details (date columns in the *EventRequest* and *EventPlan* tables) and DATETIME for columns with date and time details (time columns in the *EventPlanLine* table).

- Use CHAR(1) for the *Customer.Internal* column as Oracle does not provide a BOOLEAN data type. MySQL has the Boolean data type, but I suggest that you use CHAR(1) instead.

2. Constraints

After writing the basic CREATE TABLE statements, you should modify the statements with constraints. The CONSTRAINT clauses can be either inline in a column definition or separate after column definitions except where noted. You should specify a meaningful name for each CONSTRAINT clause.

- For each primary key, you should specify a PRIMARY KEY constraint clause. For single column primary keys (*CustNo*, *LocNo*, *EventNo*, *PlanNo*, *EmpNo*, *ResNo*, and *FacNo*), the constraint clause can be inline or external. For multiple column primary keys (combination of *PlanNo* and *LineNo*), the CONSTRAINT clause must be external.
- For each foreign key, you should specify a FOREIGN KEY constraint clause. The constraint clauses can be inline or separate.
- Define NOT NULL constraints for all columns except *eventplan.empno*, *EventRequest.DateAuth*, *EventRequest.BudNo*, and *EventPlan.Notes*. Make sure that you define NOT NULL constraints for the PK of each table. Because of MySQL syntax limitations for NOT NULL constraints (inline with no constraint name and no CONSTRAINT keyword), you should define inline NOT NULL constraints.
- Define a named CHECK constraint to restrict the *eventrequest.status* column to have a value of “Pending”, “Denied”, or “Approved”. You can use the IN operator in this constraint. In MySQL, the syntax does not allow the CONSTRAINT keyword and a constraint name for CHECK constraints. You should use the CHECK keyword followed the condition enclosed in parentheses.

- Define named CHECK constraints to ensure that the *resource.rate* and *eventrequest.estaudience* are greater than 0. In MySQL, you cannot use a constraint name and the CONSTRAINT keyword for CHECK constraints. In MySQL, the syntax does not allow the CONSTRAINT keyword and a constraint name for CHECK constraints. You should use the CHECK keyword followed the condition enclosed in parentheses.
- Define a named CHECK constraint involving *EventPlanLine.TimeStart* and *EventPlanLineTimeEnd*. The start time should be smaller (chronologically before) than the end time. This CHECK constraint must be external because it involves two columns. In MySQL, the syntax does not allow the CONSTRAINT keyword and a constraint name for CHECK constraints. You should use the CHECK keyword followed the condition enclosed in parentheses.

3. Populating Tables

The course website contains a text file containing SQL INSERT statements to populate the tables. You need to create the tables before inserting rows in each table. You need to insert rows in parent tables before child tables that reference parent tables. The INSERT statements in the file are in a proper order for populating the tables.

4. Initial CREATE TABLE Statements

To facilitate your work, you can use the CREATE TABLE statements you created in the practice assignment in module 03 for the *Customer*, *Facility*, and *Location* tables. Thus, you only need to write CREATE TABLE statements for the remaining five tables (*ResourceTbl*, *Employee*, *EventRequest*, *EventPlan*, and *EventPlanLine*). You still need to execute the CREATE TABLE statements to create all of the tables and the INSERT statements to populate all tables.

5. Submission

The submission requirements involve CREATE TABLE statements and evidence that you executed the statements and created the tables in Oracle or MySQL. You will submit 5 documents with each document containing a CREATE TABLE statement and screen snapshot to indicate that you created the table in Oracle or MySQL. In each document, you should neatly format your CREATE TABLE statement so that it can be easily graded. You should use the same table and column names as specified in the ICA database background document. You should not put any identifying details about yourself in your submitted document. For the screen snapshot, you need to capture a screen showing most columns and rows of the populated table. You can use a feature of the Oracle or MySQL client to show the rows in a table. Alternatively, you can execute an SQL statement (for example, SELECT * FROM ResourceTbl) to show the columns and rows.

You should submit the documents in this order.

- Employee
- ResourceTbl
- EventRequest
- EventPlan
- EventPlanLine

Extra Problems for Module 3

If you want some additional practice on the CREATE TABLE statement, you can work these problems. The solution document is available in the Module 3 area of the class website.

The problems use the *Customer*, *OrderTbl*, and *Employee* tables of the simplified Order Entry database. The *Customer* table contains clients who have placed orders. The *OrderTbl* contains basic facts about customer orders. The *Employee* table contains facts about employees who take orders. The primary keys of the tables are *CustNo* for *Customer*, *EmpNo* for *Employee*, and *OrdNo* for *OrderTbl*.

Customer

CustNo	CustFirstName	CustLastName	CustCity	CustState	CustZip	CustBal
C0954327	Sheri	Gordon	Littleton	CO	80129-5543	$230.00
C1010398	Jim	Glussman	Denver	CO	80111-0033	$200.00
C2388597	Beth	Taylor	Seattle	WA	98103-1121	$500.00
C3340959	Betty	Wise	Seattle	WA	98178-3311	$200.00
C3499503	Bob	Mann	Monroe	WA	98013-1095	$0.00
C8543321	Ron	Thompson	Renton	WA	98666-1289	$85.00

Employee

EmpNo	EmpFirstName	EmpLastName	EmpPhone	EmpEmail
E1329594	Landi	Santos	(303) 789-1234	LSantos@bigco.com
E8544399	Joe	Jenkins	(303) 221-9875	JJenkins@bigco.com
E8843211	Amy	Tang	(303) 556-4321	ATang@bigco.com
E9345771	Colin	White	(303) 221-4453	CWhite@bigco.com
E9884325	Thomas	Johnson	(303) 556-9987	TJohnson@bigco.com
E9954302	Mary	Hill	(303) 556-9871	MHill@bigco.com

OrderTbl

OrdNo	OrdDate	CustNo	EmpNo
O1116324	01/23/2013	C0954327	E8544399
O2334661	01/14/2013	C0954327	E1329594
O3331222	01/13/2013	C1010398	
O2233457	01/12/2013	C2388597	E9884325
O4714645	01/11/2013	C2388597	E1329594
O5511365	01/22/2013	C3340959	E9884325
O7989497	01/16/2013	C3499503	E9345771
O1656777	02/11/2013	C8543321	
O7959898	02/19/2013	C8543321	E8544399

1. Write a CREATE TABLE statement for the *Customer* table. Choose data types appropriate for the DBMS used in your course. Note that the *CustBal* column contains numeric data. The currency symbols are not stored in the database. The *CustFirstName* and *CustLastName* columns are required (not null).
2. Write a CREATE TABLE statement for the *Employee* table. Choose data types appropriate for the DBMS used in your course. The *EmpFirstName*, *EmpLastName*, and *EmpEMail* columns are required (not null).
3. Write a CREATE TABLE statement for the *OrderTbl* table. Choose data types appropriate for the DBMS used in your course. The *OrdDate* column is required (not null).
4. Identify the foreign keys and 1-M relationships among the *Customer*, *Employee*, and *OrderTbl* tables. For each relationship, identify the parent table and the child table.
5. Extend your CREATE TABLE statement from problem (3) with referential integrity constraints.
6. From examination of the sample data and your common understanding of order entry businesses, are null values allowed for the foreign keys in the *OrderTbl* table? Why or why

not? Extend the CREATE TABLE statement in problem (5) to enforce the null value restrictions if any.

7. Extend your CREATE TABLE statement for the *Employee* table (problem 2) with a unique constraint for *EmpEMail*. Use a named constraint clause for the unique constraint.

Solutions to Extra Problems for Module 3

1. Write a CREATE TABLE statement for the *Customer* table. Choose data types appropriate using standard SQL data types where possible. Note that the *CustBal* column contains numeric data. The currency symbols are not stored in the database. The *CustFirstName* and *CustLastName* columns are required (not null).
2. Write a CREATE TABLE statement for the *Employee* table. Choose data types appropriate using standard SQL data types where possible. The *EmpFirstName*, *EmpLastName*, and *EmpEMail* columns are required (not null).
3. Write a CREATE TABLE statement for the *OrderTbl* table. Choose data types appropriate using standard SQL data types where possible. The *OrdDate* column is required (not null).
4. Identify the foreign keys and 1-M relationships among the Customer, Employee, and OrderTbl tables. For each relationship, identify the parent table and the child table.
5. Extend your CREATE TABLE statement from problem (3) with referential integrity constraints.
6. From examination of the sample data and your common understanding of order entry businesses, are null values allowed for the foreign keys in the *OrderTbl* table? Why or why not? Extend the CREATE TABLE statement in problem (5) to enforce the null value constraints if any.
7. Extend your CREATE TABLE statement for the *Employee* table (problem 2) with a unique constraint for *EmpEMail*. Use a named constraint clause for the unique constraint.

The CREATE TABLE statement solution uses the standard SQL:2011 data types. Your DBMS may provide a different collection of data types.

1.

Oracle

```
CREATE TABLE Customer
( CustNo          CHAR(8),
  CustFirstName   VARCHAR2(20) CONSTRAINT CustFirstNameRequired NOT NULL,
  CustLastName    VARCHAR2(30) CONSTRAINT CustLastNameRequired NOT NULL,
  CustCity        VARCHAR2(30),
  CustState       CHAR(2),
  CustZip         CHAR(10),
  CustBal         DECIMAL(12,2),
 CONSTRAINT PKCustomer PRIMARY KEY (CustNo)  )
```

MySQL

```
CREATE TABLE Customer
( CustNo                  CHAR(8),
  CustFirstName           VARCHAR(20) NOT NULL,
  CustLastName            VARCHAR(30) NOT NULL,
  CustCity                VARCHAR(30),
  CustState               CHAR(2),
  CustZip                 CHAR(10),
 CustBal                  DECIMAL(12,2),
 CONSTRAINT PKCustomer PRIMARY KEY (CustNo)  )
```

2.

Oracle

```
CREATE TABLE Employee
  ( EmpNo         CHAR(8),
    EmpFirstName VARCHAR2(20) CONSTRAINT EmpFirstNameRequired NOT NULL,
    EmpLastName VARCHAR2(30) CONSTRAINT EmpLastNameRequired NOT NULL,
    EmpPhone      CHAR(15),
    EmpEMail      VARCHAR2(50) CONSTRAINT EmpEmailRequired NOT NULL,
CONSTRAINT PKEmployee PRIMARY KEY (EmpNo) )
```

MySQL

```
CREATE TABLE Employee
  ( EmpNo                   CHAR(8),
    EmpFirstName VARCHAR(20) NOT NULL,
    EmpLastName VARCHAR(30) NOT NULL,
    EmpPhone                CHAR(15),
    EmpEMail                VARCHAR(50) NOT NULL,
CONSTRAINT PKEmployee PRIMARY KEY (EmpNo) )
```

3.

Oracle

```
CREATE TABLE OrderTbl
( OrdNo          CHAR(8),
  OrdDate        DATE CONSTRAINT OrdDateRequired NOT NULL,
  CustNo          CHAR(8),
   EmpNo         CHAR(8),
CONSTRAINT PKOrderTbl PRIMARY KEY (OrdNo)  )
```

MySQL

```
CREATE TABLE OrderTbl
( OrdNo                    CHAR(8),
  OrdDate                  DATE NOT NULL,
  CustNo                   CHAR(8),
   EmpNo                   CHAR(8),
CONSTRAINT PKOrderTbl PRIMARY KEY (OrdNo)  )
```

4.
There are two 1-M relationships: (1) Customer (CustNo PK) – OrderTbl (CustNo FK) and (2) Employee (EmpNo PK) – OrderTbl (EmpNo FK).

5.
The CREATE TABLE statement has been extended with foreign keys for *CustNo* and *EmpNo*.

Oracle

```
CREATE TABLE OrderTbl
 ( OrdNo         CHAR(8),
   OrdDate       DATE CONSTRAINT OrdDateRequired NOT NULL,
   CustNo         CHAR(8),
   EmpNo         CHAR(8),
CONSTRAINT PKOrderTbl PRIMARY KEY (OrdNo) ,
CONSTRAINT FKCustNo FOREIGN KEY (CustNo) REFERENCES Customer,
CONSTRAINT FKEmpNo FOREIGN KEY (EmpNo) REFERENCES Employee
 )
```

MySQL

```
CREATE TABLE OrderTbl
 ( OrdNo    CHAR(8),
   OrdDate  DATE NOT NULL,
   CustNo    CHAR(8),
   EmpNo   CHAR(8),
CONSTRAINT PKOrderTbl PRIMARY KEY (OrdNo) ,
CONSTRAINT FKCustNo FOREIGN KEY (CustNo) REFERENCES Customer (CustNo),
CONSTRAINT FKEmpNo FOREIGN KEY (EmpNo) REFERENCES Employee (EmpNo)
        )
```

6.

Null values are not allowed for *CustNo*. The sample data shows that each order has a related customer. In addition, common practice indicates that an order requires a customer. Fraud could result if orders are stored without a related customer. Null values are allowed for the *EmpNo* column. The sample data shows rows without an *EmpNo* value. The null values may correspond to internet orders where no employee takes the order.

Oracle

```
CREATE TABLE OrderTbl
 ( OrdNo          CHAR(8),
   OrdDate        DATE CONSTRAINT OrdDateRequired NOT NULL,
   CustNo          CHAR(8) CONSTRAINT CustNoRequired NOT NULL,
   EmpNo          CHAR(8),
CONSTRAINT PKOrderTbl PRIMARY KEY (OrdNo) ,
CONSTRAINT FKCustNo FOREIGN KEY (CustNo) REFERENCES Customer,
CONSTRAINT FKEmpNo FOREIGN KEY (EmpNo) REFERENCES Employee
  )
```

MySQL

```
CREATE TABLE OrderTbl
 ( OrdNo    CHAR(8),
   OrdDate  DATE NOT NULL,
   CustNo    CHAR(8) NOT NULL,
   EmpNo   CHAR(8),
CONSTRAINT PKOrderTbl PRIMARY KEY (OrdNo) ,
CONSTRAINT FKCustNo FOREIGN KEY (CustNo) REFERENCES Customer (CustNo),
CONSTRAINT FKEmpNo FOREIGN KEY (EmpNo) REFERENCES Employee (EmpNo)
  )
```

7.
Oracle

```
CREATE TABLE Employee
  ( EmpNo          CHAR(8),
    EmpFirstName VARCHAR2(20) CONSTRAINT EmpFirstNameRequired NOT NULL,
    EmpLastName VARCHAR2(30) CONSTRAINT EmpLastNameRequired NOT NULL,
    EmpPhone       CHAR(15),
    EmpEMail       VARCHAR2(50) CONSTRAINT EmpEmailRequired NOT NULL,
CONSTRAINT PKEmployee PRIMARY KEY (EmpNo),
CONSTRAINT UniqueEMail UNIQUE (EmpEMail) )
```

MySQL

```
CREATE TABLE Employee
  ( EmpNo   CHAR(8),
    EmpFirstName VARCHAR(20) NOT NULL,
    EmpLastName VARCHAR(30) NOT NULL,
    EmpPhone                    CHAR(15),
    EmpEMail                    VARCHAR(50) NOT NULL UNIQUE,
CONSTRAINT PKEmployee PRIMARY KEY (EmpNo)
)
```

Module 3 Problems

The problems use the *Customer*, *Facility*, and *Location* tables of the intercollegiate athletic database. The *Customer* table contains clients who initiate event requests. The *Facility* table contains available facilities. The *Location* table contains several locations inside facilities. The primary keys of the tables are *CustNo* for *Customer, FacNo* for *Facility*, and *LocNo* for *Location*.

Customer

custno	custname	address	Internal	contact	phone	city	state	zip
C100	Football	Box 352200	Y	Mary Manager	6857100	Boulder	CO	80309
C101	Men's Basketball	Box 352400	Y	Sally Supervisor	5431700	Boulder	CO	80309
C103	Baseball	Box 352020	Y	Bill Baseball	5431234	Boulder	CO	80309
C104	Women's Softball	Box 351200	Y	Sue Softball	5434321	Boulder	CO	80309
C105	High School Football	123 AnyStreet	N	Coach Bob	4441234	Louisville	CO	80027

Facility

facno	facname
F100	Football stadium
F101	Basketball arena
F102	Baseball field
F103	Recreation room

Location

locno	facno	locname
L100	F100	Locker room
L101	F100	Plaza
L102	F100	Vehicle gate
L103	F101	Locker room
L104	F100	Ticket Booth
L105	F101	Gate
L106	F100	Pedestrian gate

1. Write a CREATE TABLE statement for the *Customer* table. Choose data types appropriate for the DBMS used in your course. All columns are required (not null).

2. Write a CREATE TABLE statement for the *Facility* table. Choose data types appropriate for the DBMS used in your course. All columns are required (not null).
3. Write a CREATE TABLE statement for the *Location* table. Choose data types appropriate for the DBMS used in your course. *LocName* column is required (not null).
4. Identify the foreign key(s) and 1-M relationship(s) among the *Customer, Facility,* and *Location* tables. For each relationship, identify the parent table and the child table.
5. Extend your CREATE TABLE statement from problem (3) with referential integrity constraints.
6. From examination of the sample data and your common understanding of scheduling and operation of events, are null values allowed for the foreign key in the *Location* table? Why or why not? Extend the CREATE TABLE statement in problem (5) to enforce the null value restrictions if any.
7. Extend your CREATE TABLE statement for the *Facility* table (problem 2) with a unique constraint for *FacName*. Use an external named constraint clause for the unique constraint.

Answers to Module 3 Problems

The CREATE TABLE statement solution uses the standard SQL:2011 data types. Your DBMS may provide a different collection of data types.

1.

Oracle:

```
CREATE TABLE Customer
(CustNo VARCHAR2(8) CONSTRAINT CustNoNotNull NOT NULL,
 CustName VARCHAR2(30) CONSTRAINT CustNameNotNull NOT
NULL,
 Address VARCHAR2(50) CONSTRAINT AddressNotNull NOT
NULL,
 Internal CHAR(1) CONSTRAINT InternalNotNull NOT NULL,
 Contact VARCHAR2(35) CONSTRAINT ContractNotNull NOT
NULL,
 Phone VARCHAR2(11) CONSTRAINT CPhoneNotNull NOT NULL,
 City VARCHAR2(30) CONSTRAINT CityNotNull NOT NULL,
 State VARCHAR2(2) CONSTRAINT StateNotNull NOT NULL,
 Zip VARCHAR2(10) CONSTRAINT zipNotNull NOT NULL,
      CONSTRAINT PK_CUSTOMER PRIMARY KEY (CustNo) ) ;
```

MySQL

```
   CREATE TABLE Customer
    (CustNo VARCHAR(8) NOT NULL,
     CustName VARCHAR(30) NOT NULL,
     Address VARCHAR(50) NOT NULL,
     Internal CHAR(1) NOT NULL,
     Contact VARCHAR(35) NOT NULL,
     Phone VARCHAR(11) NOT NULL,
     City VARCHAR(30) NOT NULL,
     State VARCHAR(2) NOT NULL,
     Zip VARCHAR(10) NOT NULL,
     CONSTRAINT PK_CUSTOMER PRIMARY KEY (CustNo) ) ;
```

2. Oracle

```
CREATE TABLE Facility
(FacNo VARCHAR2(8) CONSTRAINT FacNoNotNull NOT NULL,
 FacName VARCHAR2(30) CONSTRAINT FacNameNotNull NOT
NULL,
      CONSTRAINT PK_FACILITY PRIMARY KEY (FacNo) );
```

MySQL

```
    CREATE TABLE Facility
    (FacNo VARCHAR(8) NOT NULL,
     FacName VARCHAR(30) NOT NULL,
     CONSTRAINT PK_FACILITY PRIMARY KEY (FacNo) );
```

3. Oracle

```
    CREATE TABLE Location
(LocNo VARCHAR2(8) CONSTRAINT LocNoNotNull NOT NULL,
 FacNo VARCHAR2(8),
 LocName VARCHAR2(30) CONSTRAINT LocNameNotNull NOT
NULL,
       CONSTRAINT PK_LOCATION PRIMARY KEY (LocNo);
```

MySQL

```
    CREATE TABLE Location
    (LocNo VARCHAR(8) NOT NULL,
     FacNo VARCHAR(8),
     LocName VARCHAR(30) NOT NULL,
     CONSTRAINT PK_LOCATION PRIMARY KEY (LocNo),
```

4.
There is one 1-M relationships: Facility (FacNo PK) – Location (FacNo FK).

5.
The CREATE TABLE statement has been extended with foreign keys for *FacNo*.

Oracle

```
    CREATE TABLE Location
(LocNo VARCHAR2(8) CONSTRAINT LocNoNotNull NOT NULL,
 FacNo VARCHAR2(8),
 LocName VARCHAR2(30) CONSTRAINT LocNameNotNull NOT
NULL,
       CONSTRAINT PK_LOCATION PRIMARY KEY (LocNo),
       CONSTRAINT FK_FACNO FOREIGN KEY (FacNo)
           REFERENCES FACILITY (FacNo) );
```

MySQL

```
    CREATE TABLE Location
    (LocNo VARCHAR(8) NOT NULL,
     FacNo VARCHAR(8),
     LocName VARCHAR(30) NOT NULL,
     CONSTRAINT PK_LOCATION PRIMARY KEY (LocNo),
     CONSTRAINT FK_FACNO FOREIGN KEY (FacNo)
       REFERENCES FACILITY (FacNo) );
```

6.
Null values are not allowed for *FacNo*. The sample data shows that each facility has a related location. In addition, common practice indicates that a location requires a facility. The CREAT TABLE statement has been extended with NOT NULL constraint.

Oracle

```
   CREATE TABLE Location
(LocNo VARCHAR2(8) CONSTRAINT LocNoNotNull NOT NULL,
 FacNo VARCHAR2(8) CONSTRAINT FacNoFKNotNull NOT NULL,
 LocName VARCHAR2(30) CONSTRAINT LocNameNotNull NOT
NULL,
      CONSTRAINT PK_LOCATION PRIMARY KEY (LocNo),
      CONSTRAINT FK_FACNO FOREIGN KEY (FacNo)
       REFERENCES FACILITY (FacNo) );
```

MySQL

```
    CREATE TABLE Location
    (LocNo VARCHAR(8) NOT NULL,
     FacNo VARCHAR(8) NOT NULL,
     LocName VARCHAR(30) NOT NULL,
     CONSTRAINT PK_LOCATION PRIMARY KEY (LocNo),
     CONSTRAINT FK_FACNO FOREIGN KEY (FacNo)
       REFERENCES FACILITY (FacNo) );
```

7. Oracle

```
CREATE TABLE Facility
(FacNo VARCHAR2(8) CONSTRAINT FacNoNotNull NOT NULL,
 FacName VARCHAR2(30) CONSTRAINT FacNameNotNull NOT
NULL,
      CONSTRAINT PK_FACILITY PRIMARY KEY (FacNo)
```

```
       CONSTRAINT Unique_FacName UNIQUE(FacName) );
```

MySQL

```
    CREATE TABLE Facility
    (FacNo VARCHAR(8) NOT NULL,
     FacName VARCHAR(30) NOT NULL,
     CONSTRAINT PK_FACILITY PRIMARY KEY (FacNo)
     CONSTRAINT Unique_FacName UNIQUE(FacName));
```

Intercollegiate Athletic Database Tables

The Intercollegiate Athletic database is used in assignments 1 and 2. This document describes the tables and relationships.

Table Description and Usage

The Intercollegiate Athletic database supports the scheduling and operation of events. Customers initiate event requests with the Intercollegiate Athletic Department. Events are sometimes scheduled several months in advance. The facility and date held are recorded on an event request. If an event request is denied, no additional action is taken. If an event request is approved, one or more event plans are made. Typically, event plans are made for the setup, operation, and clean up of an event. An employee is assigned to manage an event plan before the plan is executed. Initially, there may not be an assigned employee. An event plan consists of one or more event plan lines. In an event plan line, the resource, location, time, and number of resources (*EventPlanLine.Number*) are recorded.

The *EventRequest* table is the hub of the database. An event request represents an event scheduled at a facility. For example, a basketball game may be scheduled at the gymnasium. Holding an event involves plans for allocation of resources including personnel and equipment. The *EventPlan* table contains plans for the setup, operation, and cleanup of an event. The *EventPlanLine* table contains the individual resources required in an event plan. Resources are assigned to specific locations of a facility. For example, guards may be required at the gates of the football stadium.

Population of Sample Tables

Sample rows are shown for each table in the Intercollegiate Athletic Database. You will populate the tables in Assignment 1 and query the tables in Assignment 2 using either Oracle or MySQL. Note that some columns are formatted (such as *rate* and *estcost*), but that you should consider these column values as numeric. Columns with missing values mean that the CREATE TABLE statement should not have NOT NULL constraints. For example, *BudNo* column in the *EventRequest* table has missing values so the *BudNo* column should not have NOT NULL constraint.

Customer

custno	custname	address	Internal	contact	phone	city	state	zip
C100	Football	Box 352200	Y	Mary Manager	6857100	Boulder	CO	80309
C101	Men's Basketball	Box 352400	Y	Sally Supervisor	5431700	Boulder	CO	80309
C103	Baseball	Box 352020	Y	Bill Baseball	5431234	Boulder	CO	80309
C104	Women's Softball	Box 351200	Y	Sue Softball	5434321	Boulder	CO	80309
C105	High School Football	123 AnyStreet	N	Coach Bob	4441234	Louisville	CO	80027

Employee

empno	empname	department	email	phone
E100	Chuck Coordinator	Administration	chuck@colorado.edu	3-1111
E101	Mary Manager	Football	mary@colorado.edu	5-1111
E102	Sally Supervisor	Planning	sally@colorado.edu	3-2222
E103	Alan Administrator	Administration	alan@colorado.edu	3-3333

Facility

facno	facname
F100	Football stadium
F101	Basketball arena
F102	Baseball field
F103	Recreation room

Location

locno	facno	locname
L100	F100	Locker room
L101	F100	Plaza
L102	F100	Vehicle gate
L103	F101	Locker room
L104	F100	Ticket Booth
L105	F101	Gate
L106	F100	Pedestrian gate

ResourceTbl

resno	resname	rate
R100	attendant	$10.00
R101	police	$15.00
R102	usher	$10.00
R103	nurse	$20.00
R104	janitor	$15.00
R105	food service	$10.00

EventRequest

eventno	dateheld	datereq	facno	custno	dateauth	status	estcost	estaudience	budno
E100	25-Oct-2013	06-Jun-2013	F100	C100	08-Jun-2013	Approved	$5,000.00	80000	B1000
E101	26-Oct-2013	28-Jul-2013	F100	C100		Pending	$5,000.00	80000	B1000
E102	14-Sep-2013	28-Jul-2013	F100	C100	31-Jul-2013	Approved	$5,000.00	80000	B1000
E103	21-Sep-2013	28-Jul-2013	F100	C100	01-Aug-2013	Approved	$5,000.00	80000	B1000
E104	03-Dec-2013	28-Jul-2013	F101	C101	31-Jul-2013	Approved	$2,000.00	12000	B1000
E105	05-Dec-2013	28-Jul-2013	F101	C101	01-Aug-2013	Approved	$2,000.00	10000	B1000
E106	12-Dec-2013	28-Jul-2013	F101	C101	31-Jul-2013	Approved	$2,000.00	10000	B1000
E107	23-Nov-2013	28-Jul-2013	F100	C105	31-Jul-2013	Denied	$10,000.00	5000	

EventPlan

planno	eventno	workdate	notes	activity	empno
P100	E100	25-Oct-2013	Standard operation	Operation	E102
P101	E104	03-Dec-2013	Watch for gate crashers	Operation	E100
P102	E105	05-Dec-2013	Standard operation	Operation	E102
P103	E106	12-Dec-2013	Watch for seat switching	Operation	
P104	E101	26-Oct-2013	Standard cleanup	Cleanup	E101
P105	E100	25-Oct-2013	Light cleanup	Cleanup	E101
P199	E102	10-Dec-2013	Standard operation	Operation	E101
P299	E101	26-Oct-2013		Operation	E101
P349	E106	12-Dec-2013		Cleanup	E101
P85	E100	25-Oct-2013	Standard operation	Setup	E102
P95	E101	26-Oct-2013	Extra security	Setup	E102

EventPlanLine

PlanNo	LineNo	TimeStart	TimeEnd	NumberFld	LocNo	ResNo
P100	1	25-Oct-2013 8:00	25-Oct-2013 17:00	2	L100	R100
P100	2	25-Oct-2013 12:00	25-Oct-2013 17:00	2	L101	R101
P100	3	25-Oct-2013 7:00	25-Oct-2013 16:30	1	L102	R102
P100	4	25-Oct-2013 18:00	12-Dec-2013 22:00	2	L100	R102
P101	1	3-Dec-2013 18:00	3-Dec-2013 20:00	2	L103	R100
P101	2	3-Dec-2013 18:30	3-Dec-2013 19:00	4	L105	R100
P101	3	3-Dec-2013 19:00	3-Dec-2013 20:00	2	L103	R103
P102	1	5-Dec-2013 18:00	5-Dec-2013 19:00	2	L103	R100
P102	2	5-Dec-2013 18:00	5-Dec-2013 21:00	4	L105	R100
P102	3	5-Dec-2013 19:00	5-Dec-2013 22:00	2	L103	R103
P103	1	12-Dec-2013 18:00	12-Dec-2013 21:00	2	L103	R100
P103	2	12-Dec-2013 18:00	12-Dec-2013 21:00	4	L105	R100
P103	3	12-Dec-2013 19:00	12-Dec-2013 22:00	2	L103	R103
P104	1	26-Oct-2013 18:00	26-Oct-2013 22:00	4	L101	R104
P104	2	26-Oct-2013 18:00	26-Oct-2013 22:00	4	L100	R104
P105	1	25-Oct-2013 18:00	25-Oct-2013 22:00	4	L101	R104
P105	2	25-Oct-2013 18:00	25-Oct-2013 22:00	4	L100	R104
P199	1	10-Dec-2013 8:00	10-Dec-2013 12:00	1	L100	R100
P349	1	12-Dec-2013 12:00	12-Dec-2013 15:30	1	L103	R100
P85	1	25-Oct-2013 9:00	25-Oct-2013 17:00	5	L100	R100
P85	2	25-Oct-2013 8:00	25-Oct-2013 17:00	2	L102	R101
P85	3	25-Oct-2013 10:00	25-Oct-2013 15:00	3	L104	R100
P95	1	26-Oct-2013 8:00	26-Oct-2013 17:00	4	L100	R100
P95	2	26-Oct-2013 9:00	26-Oct-2013 17:00	4	L102	R101
P95	3	26-Oct-2013 10:00	26-Oct-2013 15:00	4	L106	R100
P95	4	26-Oct-2013 13:00	26-Oct-2013 17:00	2	L100	R103
P95	5	26-Oct-2013 13:00	26-Oct-2013 17:00	2	L101	R104

Primary and Foreign Keys

The primary and foreign keys are depicted in Figure 1. An event request is related to many (one or more) event plans but only one customer. An event plan contains many event plan lines but only one supervising employee. An event plan line references a resource and location. A facility has many locations, but a location is specific to a facility.

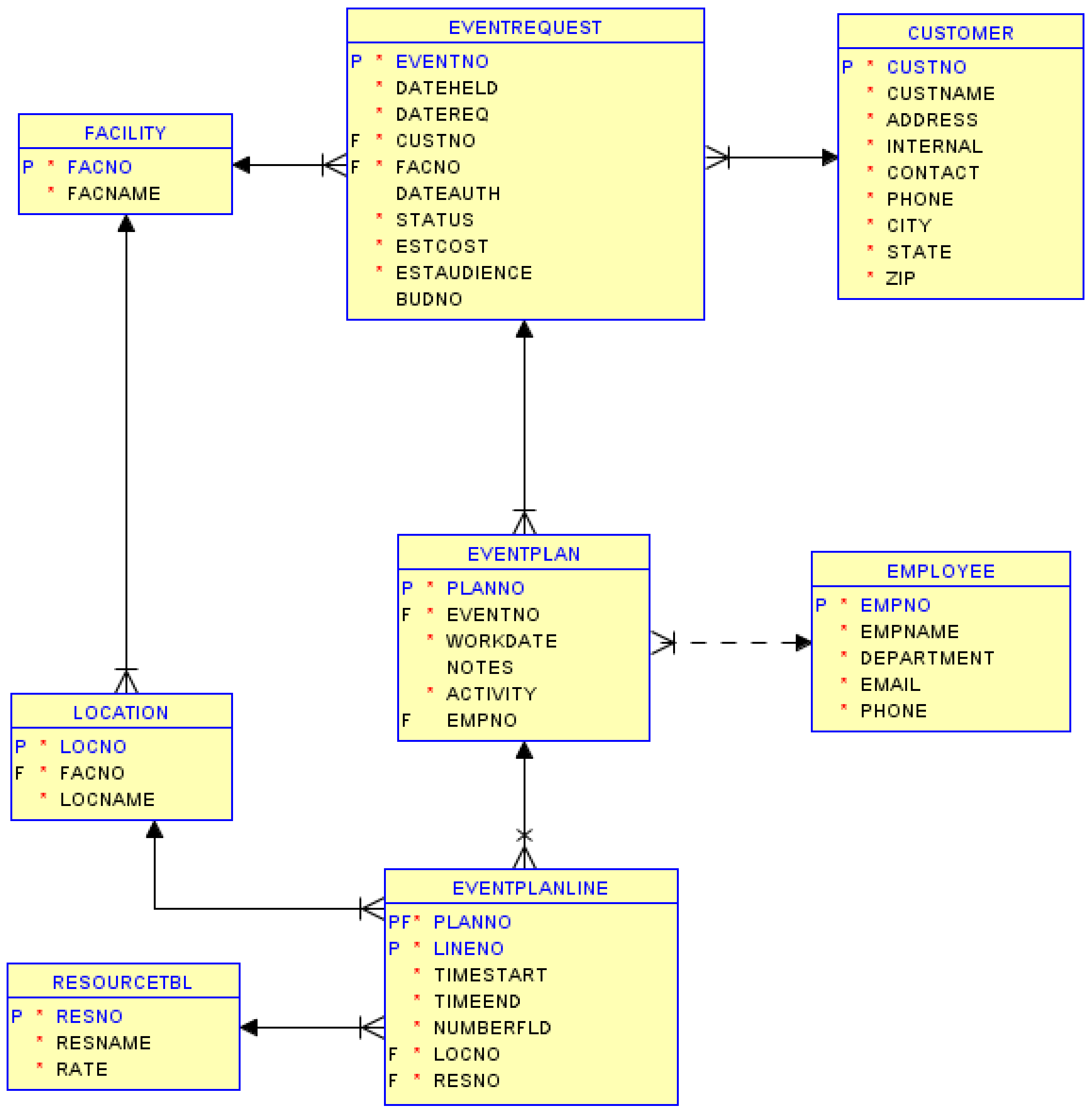

Figure 1: Oracle Relational Database Diagram for the Intercollegiate Athletic Database

All foreign key columns are required except for *EventPlan.EmpNo*. When a column is required, the user must enter a valid value according to the specified integrity rules (including referential integrity). For example when entering a new row in the *EventRequest* table, the user must know the customer number.