PART 1: Chapter 13 – Database Tuning from Beginning SQL

1. Create an index on the LastName and FirstName in the columns in the MemberDetails table

```
CREATE INDEX idx_MemberFirstName_MemberLastName
ON MemberDetails (FirstName, LastName)
```

2. Create an index on the State column in the MemberDetails table

```
CREATE INDEX idx_MemberState
ON MemberDetails (State)
```

3. Create a unique index on the MemberId column in the MemberDetails table

```
CREATE UNIQUE INDEX idx_MemberId
ON MemberDetails (MemberId)
```

4. Delete all the indexes on the MemberDetails table

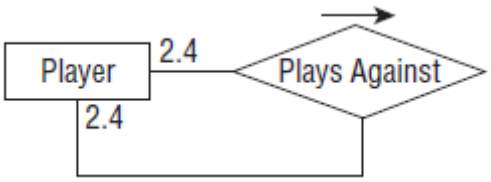
```
DROP INDEX MemberFirstName_MemberLastName, MemberState, MemberId
ON MemberDetails
```

5. Rebuild all the indexes on the MemberDetails table

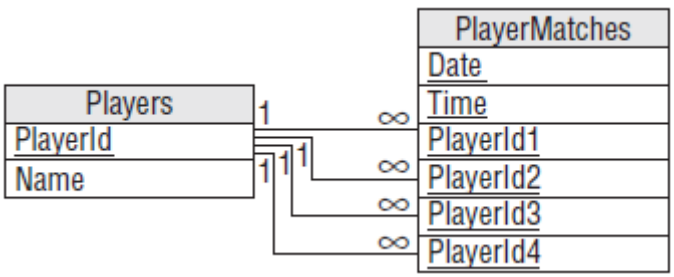
```
CREATE INDEX idx_MemberFirstName_MemberLastName, idx_MemberState, idx_MemberId
ON MemberDetails (FirstName, LastName), (State), (MemberId)
```

PART 2: Chapter 9 – Common Design Patterns

1. Parcheesi is a board game for two to four players. Make an ER diagram to record information about Parcheesi matches



2. Build a relational model to record information about Parcheesi matches. Be sure to include a way to tell who finished first through fourth

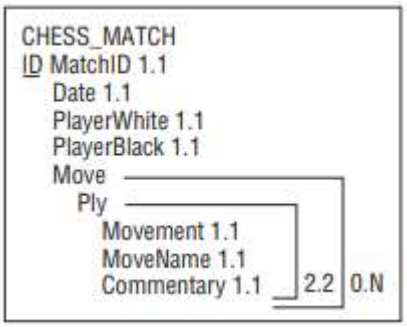


3. Chess enthusiasts often like to walk through the moves of past matches to study how the play developed. They even give names to the most famous of these matches. For example, the ‘‘Immortal Game’’ was played on June 21, 1851 by Adolf Anderssen and Lionel Kieseritzky (see http://en.wikipedia.org/wiki/Immortalgame).

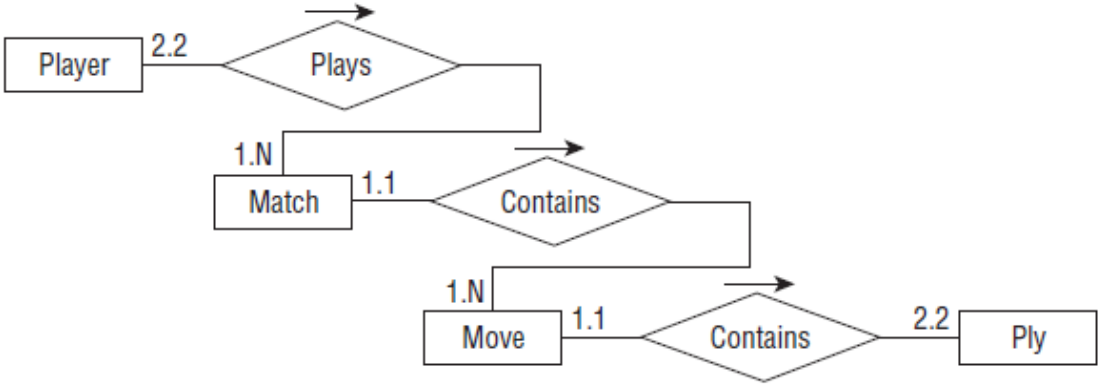
   The following text shows the first six moves in the Immortal Game:

   e4 e5 f4 exf4 Bc4 Qh4+?! Kf1 b5?! Bxb5 Nf6 Nf3 Qh6

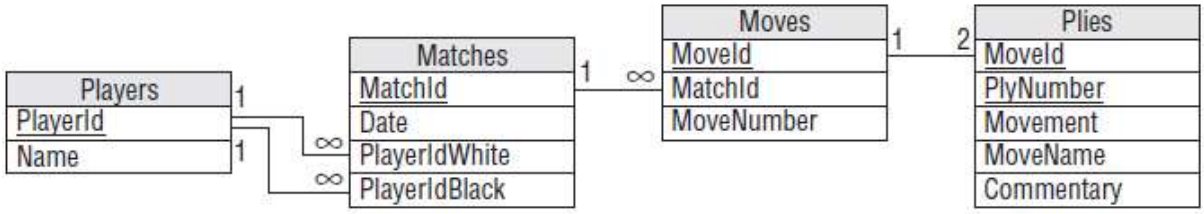
   (If someone showed me this string and I wasn’t thinking about chess at the time I’m not sure whether I would guess it was an assembly program, encrypted data, or some new variant of Leet. See http://en.wikipedia.org/wiki/Leet.) Of course, a database shouldn’t store multiple pieces of information in a single field, so the stream of move data should be broken up and stored in separate fields. In chess terms, a ply is when one player moves a piece and a move is when both players complete a ply. Figure 9-28 shows a semantic object model for a CHESS_MATCH class that stores the move information as a series of Move attributes, each containing two Ply attributes. The Movement field holds the actual move information (Qh4+?!) and MoveName is something like ‘‘The Sierpinski Gambit’’ or ‘‘The Hilbert Defense.’’ Commentary is where everyone else pretends they know what the player had in mind when he made the move.



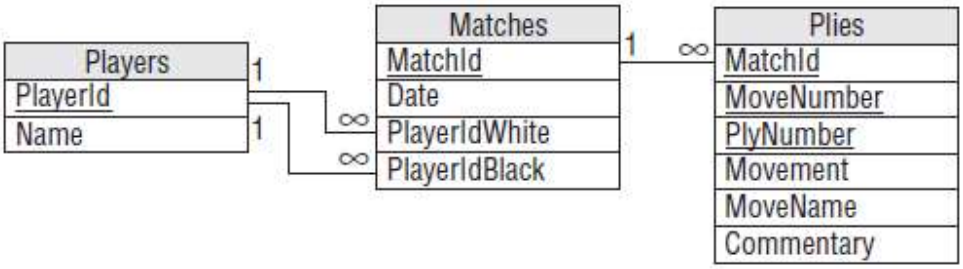
   Draw an ER diagram to show the relationships between the Player, Match, Move, and Ply entity sets



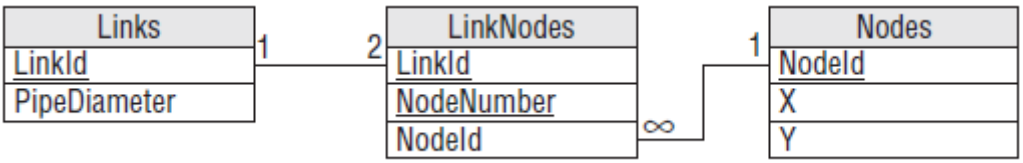
4. Build a relational model to represent chess relationships by using the tables Players, Matches, Moves, and Plies. How can you represent the one-to-two relationship between Moves and Plies within the tables' fields? How would you implement this in the database?



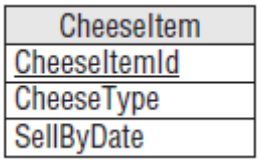
5. Consider the relational model you built for Exercise 4. The Moves table doesn't contain any data of its own except for MoveNumber. Build a new relational model that eliminates the Moves table. (Hint: collapse its data into the Plies table.) How does the new model affect the one-to-two relationship between Moves and Plies?



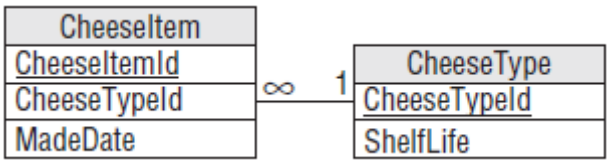
6. Suppose you are modeling a network of pipes and you want to record each pipe's diameter. Design a relational model that can hold this kind of pipe network data



7. Suppose you run a wine and cheese shop. Wine seems to get more expensive the longer it sits on your shelves, but most cheeses don't last forever. Build a relational model to store cheese inventory information that includes the date by which it must be sold. Assume that each batch of cheese gets a separate sell-by date



8. Modify the design you made for Exercise 7 assuming each type of cheese has the same shelf-life



Chapter 10 – Common Design Pitfalls

1. Suppose your client runs a ski rental shop and he's the most dreaded of all clients: one who thinks he knows something about databases. He has designed a Customers table that looks like this:

Name	Address	City	Zip	Phone1	Phone2	Stuff
Sue Rank	2832 Shush Ln. #2090	Boiler	72010	704-291-2039		Downhill, Snowboarding
Mark Bosc	276 1st Ave	East Fork	72013	704-829-1928	606-872-3982	X-country

The Stuff field contains a list of Downhill, X-country, Snowboarding, and Telemark. Your client plans to get the customer's state from the Zip value when he's sending out mailings. Your client wants you to build Orders and other tables to go with this one. For this exercise, explain to your client what's wrong with this table, paying particular attention to the ideas covered in this chapter.

This database has many multivalued fields that can create a problem when adding more records. By normalizing the data and making sure "Stuff" can be separated into "Sport 1", "Sport 2", "Sport 3" will make searching for a person easier. Also by separating the "Name" into "FirstName" & "LastName" and creating an auto incrementing "ID" column that is the primary key will ease adding records or new tables that relate to this one, in the future.

2. Suppose you're building a system to track rentals for a company that owns two Blu-ray rental stores and plans to open a third next year. What special considerations do you need to ponder that might not be as important if the client were, for example, a well-established party rental store. (They rent chairs, punch bowls, big tents, and other stuff for large parties.)

As a new business that is growing with the new business, more testing will need to be carried out to ensure the design can meet increasing performance requirements. Well established companies do not see the same level of growth as startup's do, their systems are designed for performance but require long term efficiencies rather than performance for sudden spikes in growth.

3. What's wrong with an Addresses table that includes these fields:

- ❑ CustomerId
- ❑ StreetName
- ❑ StreetNumber
- ❑ StreetPrefix
- ❑ StreetSuffix
- ❑ StreetPreDirectional

- ❑ StreetPostDirectional
- ❑ ApartmentOrSuite
- ❑ Floor
- ❑ City
- ❑ Neighborhood
- ❑ State
- ❑ Zip
- ❑ PlusFour

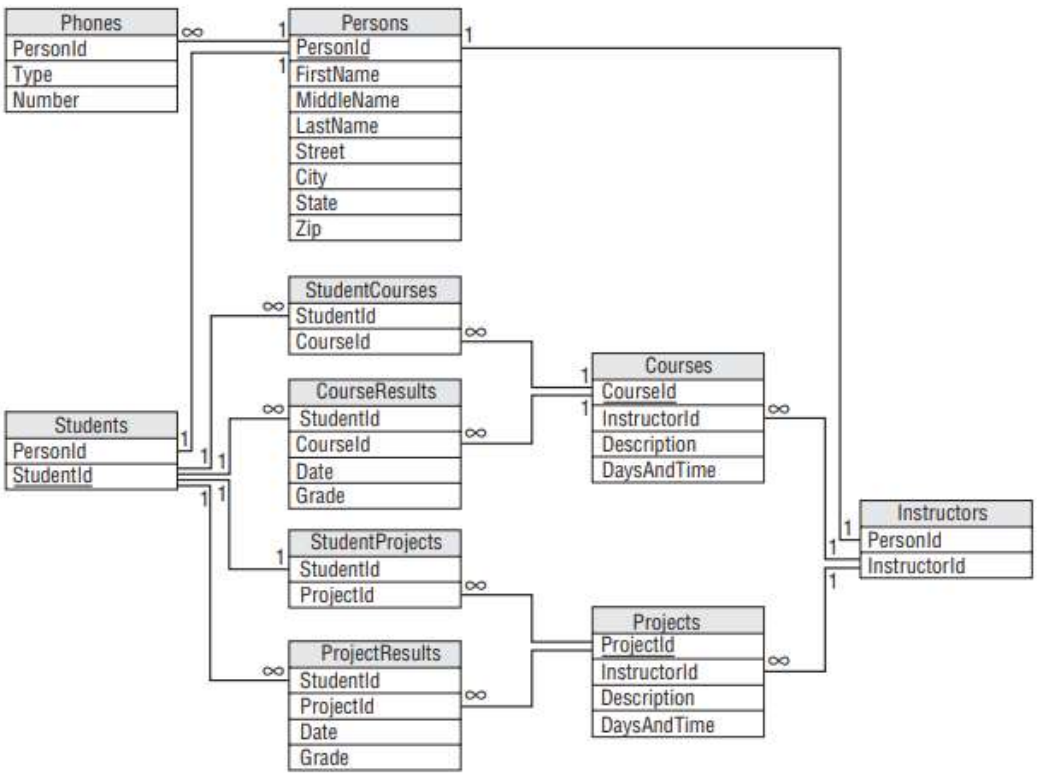
The Zip and PlusFour fields hold detailed ZIP Code data. For example, if a customer's ZIP+4 is 02536-2918, the Zip field would hold 02536 and the PlusFour field would hold 2918

There are too many unecessary fields, "StreetPreDirection", "StreetPostDirection", "StreetPrefix", and "StreetSuffix" are all unnecessary fields as the address will derive those fields. Also not all Zip Codes may have a "PlusFour", and those that do, do not require them to be filled in order to find them, also "Floor" is a field that can only be filled by customers that live in apartments, as the "ApartmentOrSuite" field will find this a "Floor" field is not necessary. Removing these columns will reduce the size of a database, making queries in the future faster to read.

A new list could be made, like this:

- ❑ CustomerId
- ❑ Street
- ❑ City
- ❑ State
- ❑ Zip

4. Consider the relational design shown in Figure below



List the database constraints that you would place on the fields in this model and explain how you would implement each of those constraints. (Feel free to add new tables if necessary.)

The phones table in unconstrained as it allows the "Person" to have any number of phone and phone types.

Field	Constraints	Implementation
PersonId	Exists	FK to match Persons.PersonId
Number	Format	Values in format of ***-***-****
Type	Enumerable Value	FK to match new PhoneTypes table

In Persons Table, every field but for MiddleName is required and the table can implement the following

Field	Constraints	Implementation
State	Enumerable Value	FK to match new State Table
Zip	Format	Values in format of **** or ****-****

The Students and Instructors tables should require all fields. They should also have a foreign key constraint requiring that their PersonId fields have values that exist in the Persons table.

StudentCourses and StudentProjects are linking tables used to implement many-to-many relationships. FK should verify that its values exist in the corresponding tables. CourseResults and ProjectResults are linking tables that implement many-to-many relationships. FK should verify that its ID values exist in the corresponding tables.

Check that the Date fields in the ProjectResults and CourseResults tables come after the corresponding student's enrolment date.

PART 3: Chapter 11 – User Needs & Requirements

1. Make a table showing the data integrity needs for the Course entity. Note any special requirements and conditions, which fields are required, and any relationships with other entities.

Field	**Required**	**Data Type**	**Domain**
Title	Yes	String	Any String
Description	Yes	String	Any String
MaxParticipants	Yes	Integer	>0
AnimalType	Yes	String	Dog, Cat, Bird, Fish, etc
Date	Yes	String	Date List
Time	Yes	Time	Between 8am and 11pm
Location	Yes	String	Room 1, Room 2, Pond 1, Cage 1, etc
Price	Yes	Currency	>0
Student	No	Reference	Customer Table
Trainer	No	Reference	Employee teacher

1. Make a table showing the data integrity needs for the Employee entity. Note any special requirements and conditions, which fields are required, and any relationships with other entities.

Field	Required	Data Type	Domain
FirstName	Yes	String	First Name
LastName	Yes	String	Last Name
Street	Yes	String	Street Name, Street Number. Not Validated
City	Yes	String	City Name
State	Yes	String	FK to States Table
Zip	Yes	String	Zip Code, Valid
HomePhone	No	String	10-digit phone number, Valid
MobilePhone	No	String	10-digit phone number, Valid
Email	No	String	Email address, Valid. Send emails
SSN	Yes	String	10-digit SSN, Valid
Specialties	No	String	Dogs, Cats, Birds, Snakes, Horses, Fish, etc

2. Make a table showing the data integrity needs for the Shift entity. Note any special requirements and conditions, which fields are required, and any relationships with other entities.

Field	Required	Data Type	Domain
Employee	Yes	Reference	Assignments Employee
Date	Yes	Date	Valid Date
StartTime	Yes	Time	>= 6am
StopTime	Yes	Time	<= 11pm, and >= StartTime + 1 Hour

3. Make a table showing the data integrity needs for the Customer entity. Note any special requirements and conditions, which fields are required, and any relationships with other entities

Field	Required	Data Type	Domain
FirstName	Yes	String	First Name
LastName	Yes	String	Last Name
Street	Yes	String	Street Name, Street Number. Not Validated
City	Yes	String	City Name
State	Yes	String	FK to States Table
Zip	Yes	String	Zip Code, Valid
HomePhone	Yes	String	10-digit phone number, Valid
MobilePhone	No	String	10-digit phone number, Valid
Email	No	String	Email address, Valid. Send emails
Pets	No	String	Pet Age, Name, and Breed/Type

4. Make a table showing the data integrity needs for the TimeEntry entity. Note any special requirements and conditions, which fields are required, and any relationships with other entities.

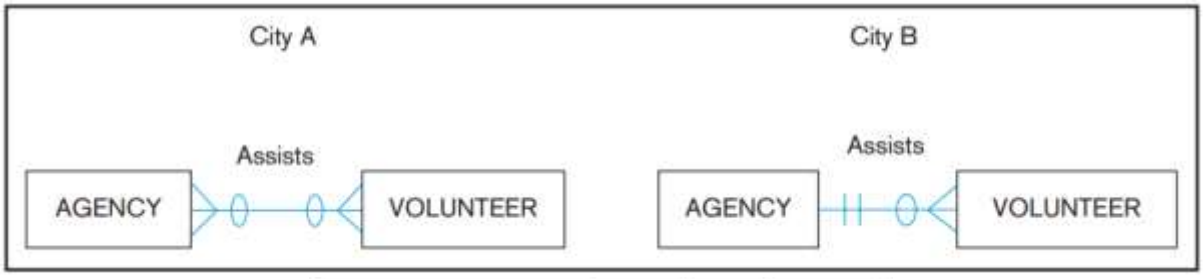
Field	Required	Data Type	Domain
Employee	Yes	Reference	Employee worked
Date	Yes	Date	<=current date
StartTime	Yes	Time	<=current time
StopTime	Yes	Time	<= current time, and > StartTime
PaidDate	No	Date	<=current date

5. Make a table showing the data integrity needs for the Vendor entity. Note any special requirements and conditions, which fields are required, and any relationships with other entities.

Field	Required	Data Type	Domain
CompanyName	Yes	String	Name of Company
FirstName	Yes	String	First Name of Contact
LastName	Yes	String	Last Name of Contact
Street	Yes	String	Street Name and Number, Not Validated
City	Yes	String	City Name
State	Yes	String	FK to State Table
Zip	Yes	String	Valid Zip Code
Email	No	String	Valid Email address
Phone	Yes	String	10-digit Phone number, Valid
Notes	No	String	Instructions or notes/other

PART 4: Chapter 2 – Modern Database Management

1. Consider the two E-R diagrams in Figure 2-24, which represent a database of community service agencies and volunteers in two different cities (A and B). For each of the following three questions, place a check mark under City A, City B, or Can't Tell for the choice that is the best answer



	City A	City B	Can't Tell
a. Which city maintains data about only those volunteers who currently assist agencies?			◊
b. In which city would it be possible for a volunteer to assist more than one agency?	◊		
c. In which city would it be possible for a volunteer to change which agency or agencies he or she assists?	◊		

2. The entity type STUDENT has the following attributes: Student Name, Address, Phone, Age, Activity, and No of Years. Activity represents some campus-based student activity, and No of Years represents the number of years the student has engaged in this activity. A given student may engage in more than one activity. Draw an ERD for this situation. What attribute or attributes did you designate as the identifier for the STUDENT entity? Why?

3. Are associative entities also weak entities? Why or why not? If yes, is there anything special about their "weakness"?

4. Because Visio does not explicitly show associative entities, it is not clear in Figure 2-22 which entity types are associative. List the associative entities in this figure. Why are there so many associative entities in Figure 2-22?

5. Figure 2-25 shows a grade report that is mailed to students at the end of each semester. Prepare an ERD reflecting the data contained in the grade report. Assume that each course is taught by one instructor. Also, draw this data model using the tool you have been told to use in the course. Explain what you chose for the identifier of each entity type on your ERD.

6. Add minimum and maximum cardinality notation to each of the following figures, as appropriate: a. Figure 2-5 b. Figure 2-10a c. Figure 2-11b d. Figure 2-12 (all parts) e. Figure 2-13c f. Figure 2-14

7. The Is Married To relationship in Figure 2-12a would seem to have an obvious answer in Problem and Exercise 9d— that is, until time plays a role in modeling data. Draw a data model for the PERSON entity type and the Is Married To relationship for each of the following variations by showing the appropriate cardinalities and including, if necessary, any attributes:

a. All we need to know is who a person is currently married to, if anyone. (This is likely what you represented in your answer to Problem and Exercise 9d.)
b. We need to know who a person has ever been married to, if anyone.
c. We need to know who a person has ever been married to, if anyone, as well as the date of their marriage and the date, if any, of the dissolution of their marriage.
d. The same situation as in c, but now assume (which you likely did not do in c) that the same two people can remarry each other after a dissolution of a prior marriage to each other.
e. In history, and even in some cultures today, there may be no legal restriction on the number of people to whom one can be currently married. Does your answer to part c of this Problem and Exercise handle this situation or must you make some changes (if so, draw a new ERD).

PART 5: Chapter 3 – Modern Database Management

1.

PART 6: Chapter 12 – SQL Security

1. Create DataEntry, Supervisor, and Management groups.

```
EXECUTE sp_addrole @rolename = 'DataEntry'
EXECUTE sp_addrole @rolename = 'Management'
EXECUTE sp_addrole @rolename = 'Supervisor'
```

2. Create users John, Joe, Fred, Lynn, Amy, and Beth

```
EXECUTE sp_addlogin @loginame = 'John', @passwd = 'pass1'
EXECUTE sp_addlogin @loginame = 'Joe', @passwd = 'pass2'
EXECUTE sp_addlogin @loginame = 'Fred', @passwd = 'pass3'
EXECUTE sp_addlogin @loginame = 'Lynn', @passwd = 'pass4'
EXECUTE sp_addlogin @loginame = 'Amy', @passwd = 'pass5'
EXECUTE sp_addlogin @loginame = 'Beth', @passwd = 'pass6'
EXECUTE sp_adduser 'John', 'John'
EXECUTE sp_adduser 'Joe', 'Joe'
EXECUTE sp_adduser 'Fred', 'Fred'
EXECUTE sp_adduser 'Lynn', 'Lynn'
EXECUTE sp_adduser 'Amy', 'Amy'
EXECUTE sp_adduser 'Beth', 'Beth'
```

3. Add John, Joe, and Lynn to the DataEntry group, add Fred to the Supervisor group, and add Amy and Beth to the Management group

```
EXECUTE sp_addrolemember @rolename = 'DataEntry', @membername = 'John', 'Joe', 'Lynn'
EXECUTE sp_addrolemember @rolename = 'Supervisor', @membername = 'Fred'
EXECUTE sp_addrolemember @rolename = 'Management', @membername = 'Amy', 'Beth'
```

4. Give the DataEntry group SELECT, INSERT, and UPDATE privileges on the Attendance table.

```
GRANT SELECT, INSERT, UPDATE
ON Attendance
TO DataEntry
```

5. Give the Supervisor group SELECT and DELETE privileges on the Attendance table.

```
GRANT SELECT, DELETE
ON Attendance
TO Supervisor
```

6. Give the Management group SELECT privileges on the Attendance table.

```
GRANT SELECT
ON Attendance
TO Management
```