Tutorial letter 202/1/2016

Database Design

ICT3621

Semesters 1

SOLUTIONS TO ASSIGNMENT 02

School of Computing

IMPORTANT INFORMATION:

This tutorial letter contains important information about your module.

BAR CODE

Solutions to Assignment 02 [TOTAL MARKS (100)]

Due date	Tutorial matter covered in prescribed book
10 April 2015	Chapter 1: The Database Approach Chapter 2: Data Models Chapter 3: Relational Model Characteristics Chapter 5: Data Modelling with Entity Relationship (ER) Diagrams Chapter 6: Data Modelling Advanced Concepts Chapter 7: Normalizing Database Designs Chapter 10: Database Development Process Chapter 11: Conceptual, Logical and Physical Database Design
Unique number: 588078	

Questions and Answers:

Questions 1 (31 Marks)

1. Discuss the lack of data independence in file systems. **(3 marks)**
   File systems exhibit data dependence because file access is dependent on a file's data characteristics. Therefore, any time the file data characteristics are changed, the programs that access the data within those files must be modified. Data independence exists when changes in the data characteristics don't require changes in the programs that access those data. ***(Chapter 1, page 268)***
2. What is data independence, and why is it important? **(3 Marks)** ***(Chapter 1, page 183)***
   Data independence exists when data access programs are not subject to change when any of the file's data characteristics change. Data independence is important because it substantially decreases programming effort and program maintenance costs. (Chapter 1, page 183)
3. What is a business rule, and what is its purpose in data modeling? **(3 Marks)** ***(Chapter 2, page 230)***
   A business rule is a brief, precise, and unambigous description of a policy, procedure, or principle within a specific organization's environment. In a sense, business rules are misnamed: they apply to any organization -- a business, a government unit, a religious group, or a research laboratory; large or small -- that stores and uses data to generate information.

Business rules are derived from a description of operations. As its name implies, a description of operations is a detailed narrative that describes the operational nvironment of an organization. Such a description requires great precision and detail. If the description of operations is incorrect or inomplete, the business rules derived from it will not reflect the real world data environment accurately, thus leading to poorly defined data models, which lead to poor database designs. In turn, poor database designs lead to poor applications, thus setting the stage for poor decision making – which may ultimately lead to the demise of the organization.

Note especially that business rules help to create and enforce actions within that organization's environment. Business rules must be rendered in writing and updated to reflect any change in the organization's operational environment. Properly written business rules are used to define entities, attributes, relationships, and constraints. Because these components form the basis for a database design, the careful derivation and definition of business rules is crucial to good database design. (Chapter 2, page 260)

4. Give an example of each of the three types of relationships**. (3 Marks)**

1:1

An academic department is chaired by one professor; a professor may chair only one academic department.

1:*

A customer may generate many invoices; each invoice is generated by one customer.

:

An employee may have earned many degrees; a degree may have been earned by many employees. (Chapter 2, page 186)

5. Define and describe the basic characteristics of a NoSQL database. **(4 Marks)**

Every time you search for a product on Amazon, send messages to friends in Facebook, watch a video in YouTube or search for directions in Google Maps, you are using a NoSQL database. NoSQL refers to a new generation of databases that address the very specific challenges of the "big data" era and have the following general characteristics:

- **Not based on the relational model.**

These databases are generally based on a variation of the key-value data model rather than in the relational model, hence the NoSQL name. The key-value data model is based on a structure composed of two data elements: a key and a value; in which for every key there is a corresponding value (or a set of values). The key-value data model is also referred to as the attribute-value or associative data model. In the key-value data model, each row represents one attribute of one entity instance. The "key" column points to an attribute and the "value" column contains the actual value for the attribute. The data type of the "value" column is generally a long string to accommodate the variety of actual data types of the values that are placed in the column.

- **Support distributed database architectures.**

One of the big advantages of NoSQL databases is that they generally use a distributed architecture. In fact, several of them (Cassandra, Big Table) are designed to use low cost commodity servers to form a complex network of distributed database nodes.

- **Provide high scalability, high availability and fault tolerance.**

NoSQL databases are designed to support the ability to add capacity (add database nodes to the distributed database) when the demand is high and to do it transparently and without downtime. Fault tolerant means that if one of the nodes in the distributed database fails, the database will keep operating as normal.

- **Support very large amounts of sparse data.**

Because NoSQL databases use the key-value data model, they are suited to handle very high volumes of sparse data; that is for cases where the number of attributes is very large but the number of actual data instances is low.

- **Geared toward performance rather than transaction consistency.**

One of the biggest problems of very large distributed databases is to enforce data consistency. Distributed databases automatically make copies of data elements at multiple nodes – to ensure high availability and fault tolerance. If the node with the requested data goes down, the request can be served from any other node with a copy of the data. However, what happen if the network goes down during a data update? In a relational database, transaction updates are guaranteed to be consistent or the transaction is rolled back. NoSQL databases sacrifice consistency in order to attain high levels of performance. NoSQL databases provide eventual consistency. *Eventual consistency* is a feature of NoSQL databases that indicates that data are not guaranteed to be consistent immediately after an update (across all copies of the data) but rather, that updates will propagate through the system and eventually all data copies will be consistent.. *(Chapter 2, pages 264)*

6. Why are entity integrity and referential integrity important in a database? **(4 Marks)** ***(Chapter 3, page 239)***

   Entity integrity and referential integrity are important because they are the basis for expressing and implementing relationships in the entity relationship model. Entity integrity ensures that each row is uniquely identified by the primary key. Therefore, entity integrity means that a proper search for an *existing* tuple (row) will always be successful. (And the failure to find a match on a row search will always mean that the row for which the search is conducted does not exist in that table.) Referential integrity means that, if the foreign key contains a value, that value refers to an existing valid tuple (row) in another relation. Therefore, referential integrity ensures that it will be impossible to assign a non-existing foreign key value to a table. *(Chapter 3, page 239)*

7. What is a composite entity, and when is it used? (3 Marks)

   A composite entity is generally used to transform *:* relationships into 1:* relationships. (Review the discussion that accompanied Figures 5.26 through 5.28) A composite entity, also known as a bridge entity, is one that has a primary key composed of multiple attributes. The PK attributes are inherited from the entities that it relates to one another. (Chapter 5, pages 189)

8. What is a derived attribute? Give an example **(3)**

A derived attribute is an attribute whose value is calculated (derived) from other attributes. The derived attribute need not be physically stored within the database; instead, it can be derived by using an algorithm. For example, an employee's age, EMP_AGE, may be found by computing the integer value of the difference between the current date and the EMP_DOB. If you use MS Access, you would use INT((DATE() – EMP_DOB)/365).

Similarly, a sales clerk's total gross pay may be computed by adding a computed sales commission to base pay. For instance, if the sales clerk's commission is 1%, the gross pay may be computed by EMP_GROSSPAY = INV_SALES*1.01 + EMP_BASEPAY

Or the invoice line item amount may be calculated by

LINE_TOTAL = LINE_UNITS*PROD_PRICE (chapter 5 Page 37)

9. What is an overlapping subtype? Give an example. (3)
Overlapping subtypes are subtypes that contain non-unique subsets of the supertype entity set; that is, each entity instance of the supertype may appear in more than one subtype. For example, in a university environment, a person may be an employee or a student or both. In turn, an employee may be a professor as well as an administrator. Because an employee also may be a student, STUDENT and EMPLOYEE are overlapping subtypes of the supertype PERSON, just as LECTURER and ADMINISTRATOR are overlapping subtypes of the supertype EMPLOYEE. The text's Figure 6.4 (reproduced next for your convenience) illustrates overlapping subtypes.

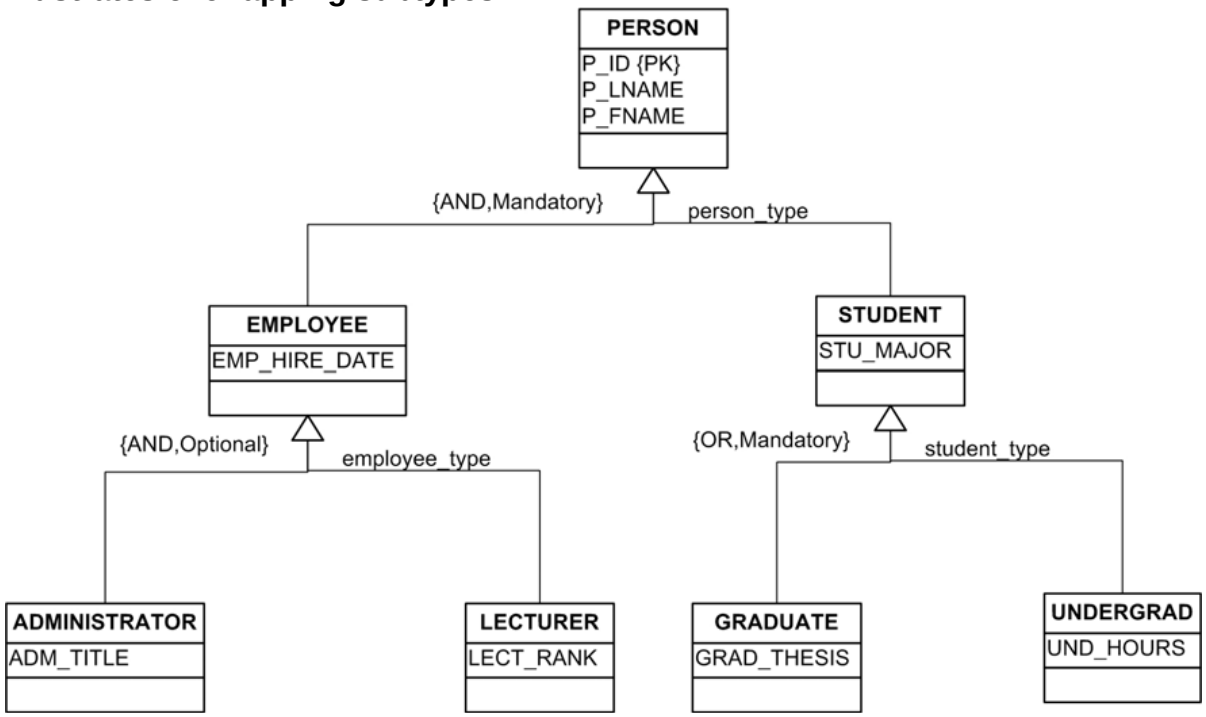


FIGURE 6.4 Specialization Hierarchy with Overlapping Subtypes (Chapter 6 Page 233)

10. What is a partial dependency? With what normal form is it associated? (2 Marks)
A partial dependency exists when an attribute is dependent on only a portion of the primary key. This type of dependency is associated with 1NF. (Chapter 7, pages 189)

Question 2 (16 Marks)

a. What are the activities of Database initial study? (4)
Answer:

- Analize the company situation
- Define problems and constraints
- Define objectives
- Define scope and boundaries

a. What is data encryption and why is it important to data security? (2 Marks)

It is carried out by an algorithm to render data useless to unauthorized users who might have violated some of the database security measures. ***(Chapter 10, page 532)***

b. What are business rules? Why are they important to a database designer? **(5 Marks)**
Business rules are narrative descriptions of the business policies, procedures, or principles that are derived from a detailed description of operations. Business rules are particularly valuable to database designers, because they help define:
- ***Entities***
- ***Attributes***
- ***Relationships (1:1, 1:*, *:*, expressed through connectivities and cardinalities)***
- ***Constraints***

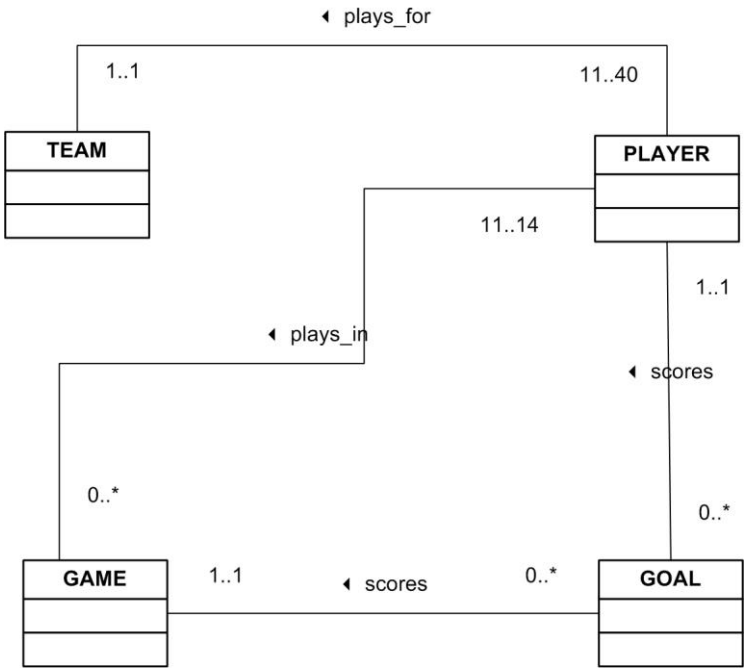
To develop an accurate data model, the database designer must have a thorough and complete understanding of the organization's data requirements. The business rules are very important to the designer because they enable the designer to fully understand how the business works and what role is played by data within company operations. (Chapter 11, page 183)

c. Describe the logical database design process. **(5 Marks)**
The aim of the logical design stage is to map the conceptual model into a logical model which can then be implemented on a relational DBMS. The logical design stage consists of the following phases:
1. Creating the logical data model
2. Validating the logical data model using normalization
3. Assigning and validating integrity constraints
4. Merging logical models constructed for different parts for the database together.
5. Reviewing the logical data model with the user
A full description of the logical design stage can be found in section 11.2. (Chapter 11, page 260)

Question 3 (10 Marks)

3. Given the following business rules, create the appropriate ERD using UML notation.(10)
    a. A football team has at least 11 players and may have up to 40 players.
    b. Each player may or may not play one or more games.
    c. A minimum of 11 players and a maximum of 14 players may participate in one game.
    d. A player may or may not score one or more goals.
    e. Each game may have zero or more goals.

The solution is presented in the solution is presented in the follwoing Figure:



Question 4 (30)

Suppose that you have been given the table structure and data shown in below Table, which was imported from an Excel spreadsheet. The data reflect that a professor can have multiple advisees, can serve on multiple committees, and can edit more than one journal.

Table :Sample Records

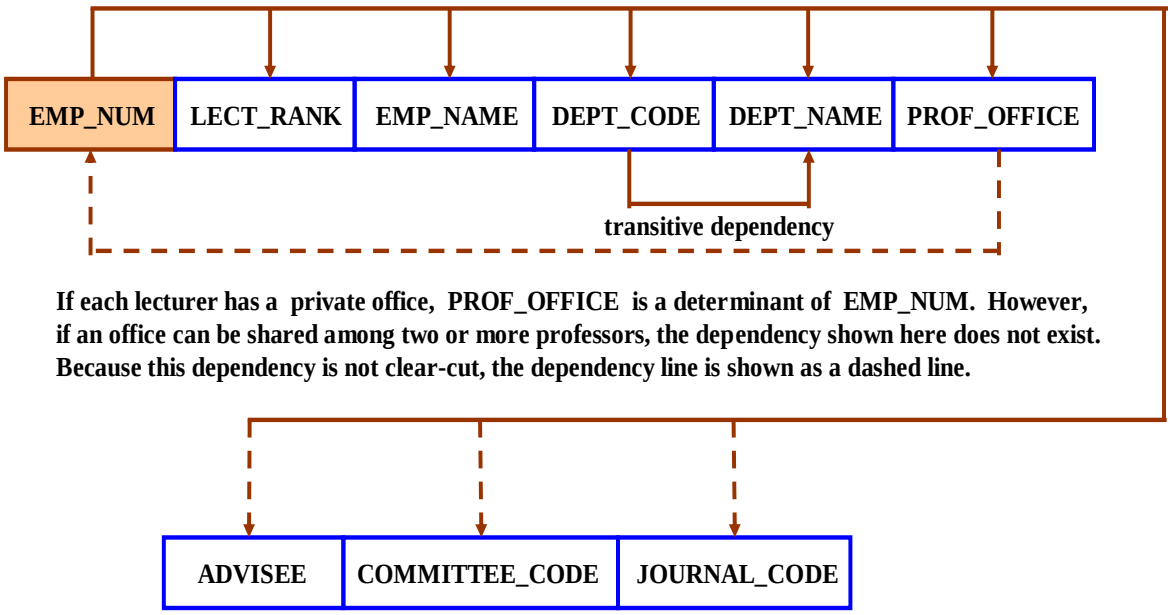
Attribute Name	Sample Value	Sample Value	Sample Value	Sample Value
EMP_NUM	123	104	118	
LECT_RANK	Professor	Asst. Lecturer	Assoc. Lecturer	Assoc. Lecturer
EMP_NAME	Ghee	Rankin	Ortega	Smith
DEPT_CODE	CIS	CHEM	CIS	ENG
DEPT_NAME	Computer Info. Systems	Chemistry	Computer Info. Systems	English
PROF_OFFICE	KDD-567	BLF-119	KDD-562	PRT-345
ADVISEE	1215, 2312, 3233, 2218, 2098	3102, 2782, 3311, 2008, 2876, 2222, 3745, 1783, 2378	2134, 2789, 3456, 2002, 2046, 2018, 2764	2873, 2765, 2238, 2901, 2308
COMMITTEE_CODE	PROMO,	DEV	SPR, TRAF	PROMO,

	TRAF APPL, DEV			SPR DEV
JOURNAL_CODE	JMIS, QED, JMGT		JCIS, JMGT	

Given the information in above Table:

a. Draw the dependency diagram. (10)

Figure 4a: The Dependency Diagram for Problem 4a



Note that Figure 4a reflects several ambiguities. For example, although each PROF_OFFICE value shown in Table P7.26 is unique, does that limited information indicate that each lecturer has a private office? If so, the office number identifies the lecturer who uses that office. This condition yields a dependency. However, this dependency is *not* a transitive one, because a non-key attribute, PROF_OFFICE, determines the value of a key attribute, EMP_NUM. (We have indicated this *potential* transitive dependency through a dashed dependency line.)

NOTE
The assumption that PROF_OFFICE → EMP_CODE is a rather restrictive one, because it would mean that professors cannot share an office. One could safely assume that administrators at all levels would not care to be tied by such a restrictive office assignment requirement. Therefore, we will remove this restriction in the remaining problem solutions.

Also, note that there is no reliable way to identify the effect of multivalued attributes on the dependencies. For example, EMP_NUM = 123 could identify any one of five advisees. Therefore, knowing the EMP_NUM does not identify a *specific* ADVISEE value. The same is true for the COMMITTEE_CODE and JOURNAL_CODE attributes. *Therefore, these attributes are not marked with a solid arrow line*. However, if you know that EMP_NUM = 123, you will also know all five advisees, all four committee codes, and all three journal codes for that employee number value. But you do not have a *unique* identification for each of those attribute values. Therefore, you *cannot* conclude that EMP_NUM → ADVISEE, nor can you conclude that EMP_NUM → COMMITTEE_CODE or that EMP_NUM → JOURNAL_CODE.

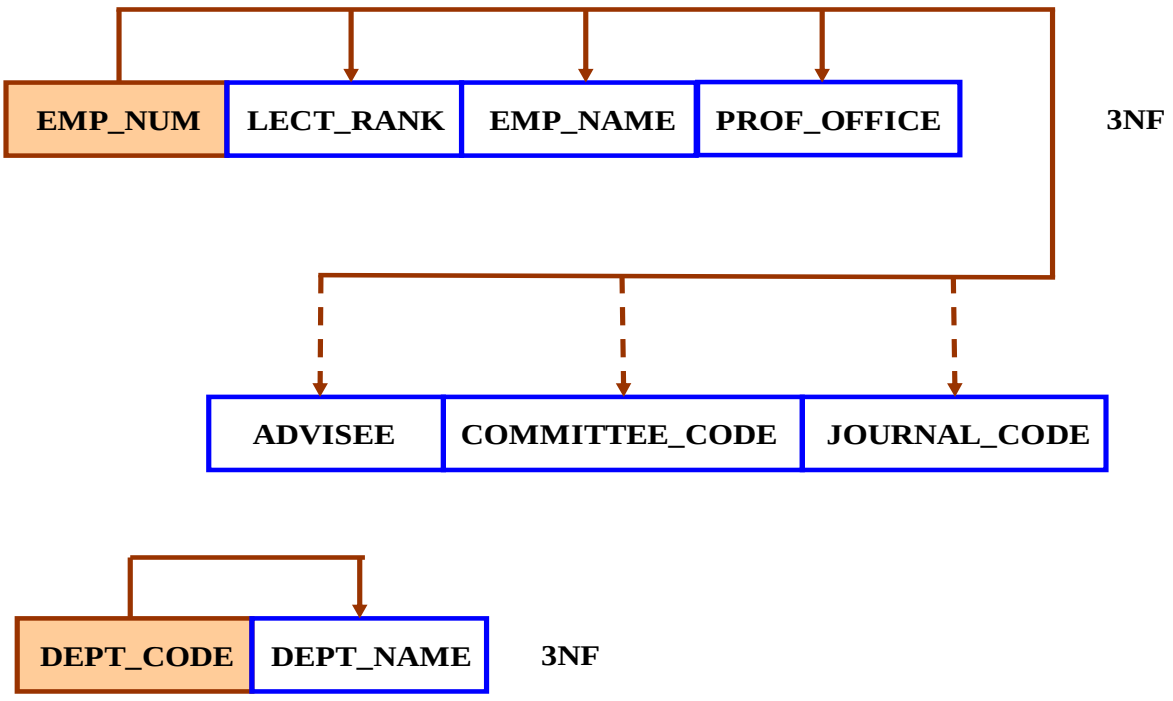
b. Identify the multivalued dependencies. (3)

Table above shows several lecturer attributes – ADVISEE, COMMITTEE_CODE, and JOURNAL_CODE -- that represent multivalued dependencies.

c. Create the dependency diagrams to yield a set of table structures in 3NF. (7)

The dependency diagrams are shown in Figure P7.26c. Note that we have assumed that it is possible that professors can share an office.

Figure c : The Dependency Diagram for Problem c

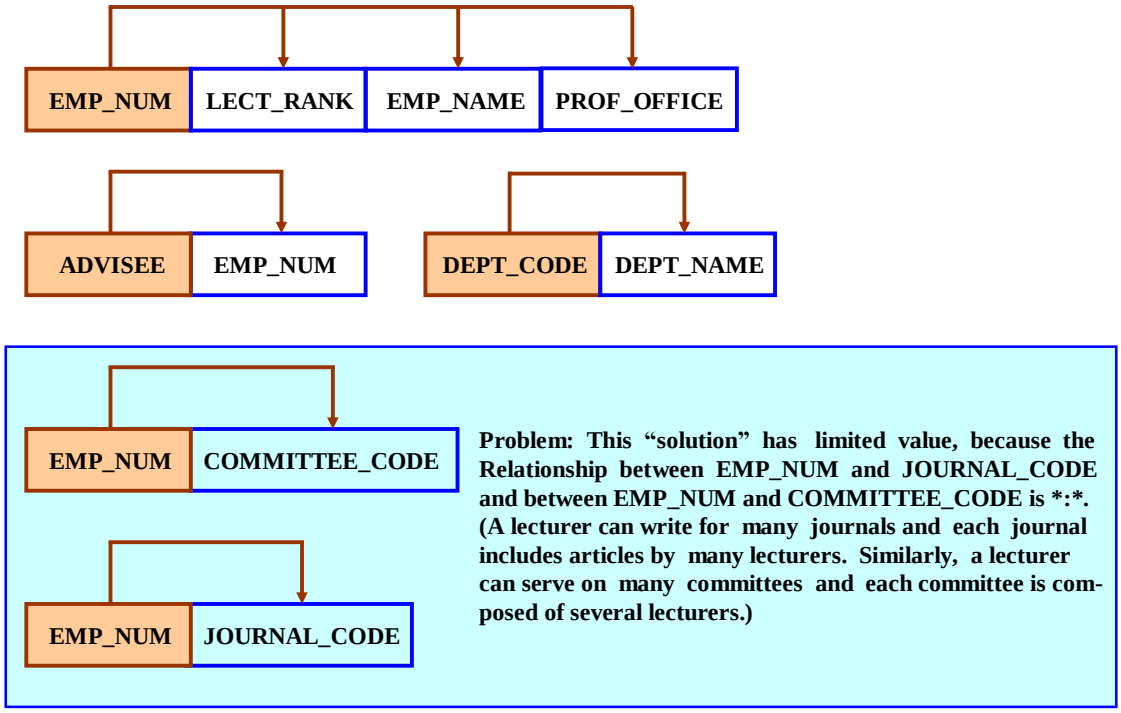


d. Eliminate the multivalued dependencies by converting the affected table structures to 4NF. (10)

answer

The structures shown in Figure d1 conform to the 4NF requirement. Yet this normalization does not yield a viable database design. Here is another opportunity to stress that normalization without data modeling is a poor way to generate useful databases. (Note that we have assumed that an advisee can have only one advisor, but that an advisor can have many advisees.)

Figure d: The Initial Dependency Diagrams for Problem 4d



The dependency diagrams shown in Figure d constitute an attempt to eliminate the shortcomings of the "system" shown in Figure P7.26c. Unfortunately, while this solution meets the normalization requirements, it lacks the ability to properly link the lecturers to committees and journals. (That's because the relationships between lecturers and journals and between lecturers and committees are *:*.) This solution would yield tables P7.26d1 and P7.26d2. (One would expect a lecturers to be an employee, so it's reasonable to assume that – at some point -- we'll have to create a supertype/subtype relationship between employee and lecturers. (To save space, we show only the first three EMP_NUM value sets from Sample Table.)

Table d1 Implementation of the *:* Relationship between EMP_NUM and COMMITTEE_CODE

EMP_NUM	COMMITTEE_CODE
123	PROMO
123	TRAF
123	APPL
123	JMGT
104	DEV
118	SPR
118	TRAF

The PK of the table shown in Table d1 is EMP_NUM + COMMITTEE_CODE.

Table d2 Implementation of the *:* Relationship between EMP_NUM and JOURNAL_CODE

EMP_NUM	JOURNAL_CODE
123	JMIS
123	QED
123	JMGT
118	JCIS
118	JMGT

The PK of the table shown in Table d2 is EMP_NUM + JOURNAL_CODE. Because EMP_CODE = 104 does not show any entries in the JOURNAL_CODE, the employee code does not occur in Table d2.

The preceding table structures create multiple redundancies. Therefore, this solution is not acceptable. Here is yet another indication that normalization, while very useful, is not always (usually?) capable of producing implementable solutions. For example, the preceding examples illustrate that multivalued attributes and *:* relationships cannot be effectively modeled without first using the ERD. (After the ERD has done its work, you should, of course, use dependency diagrams to check for data redundancies!) the below Figure on question 5 shows a more practical solution to the problem and its structures all conform to the normalization requirements.

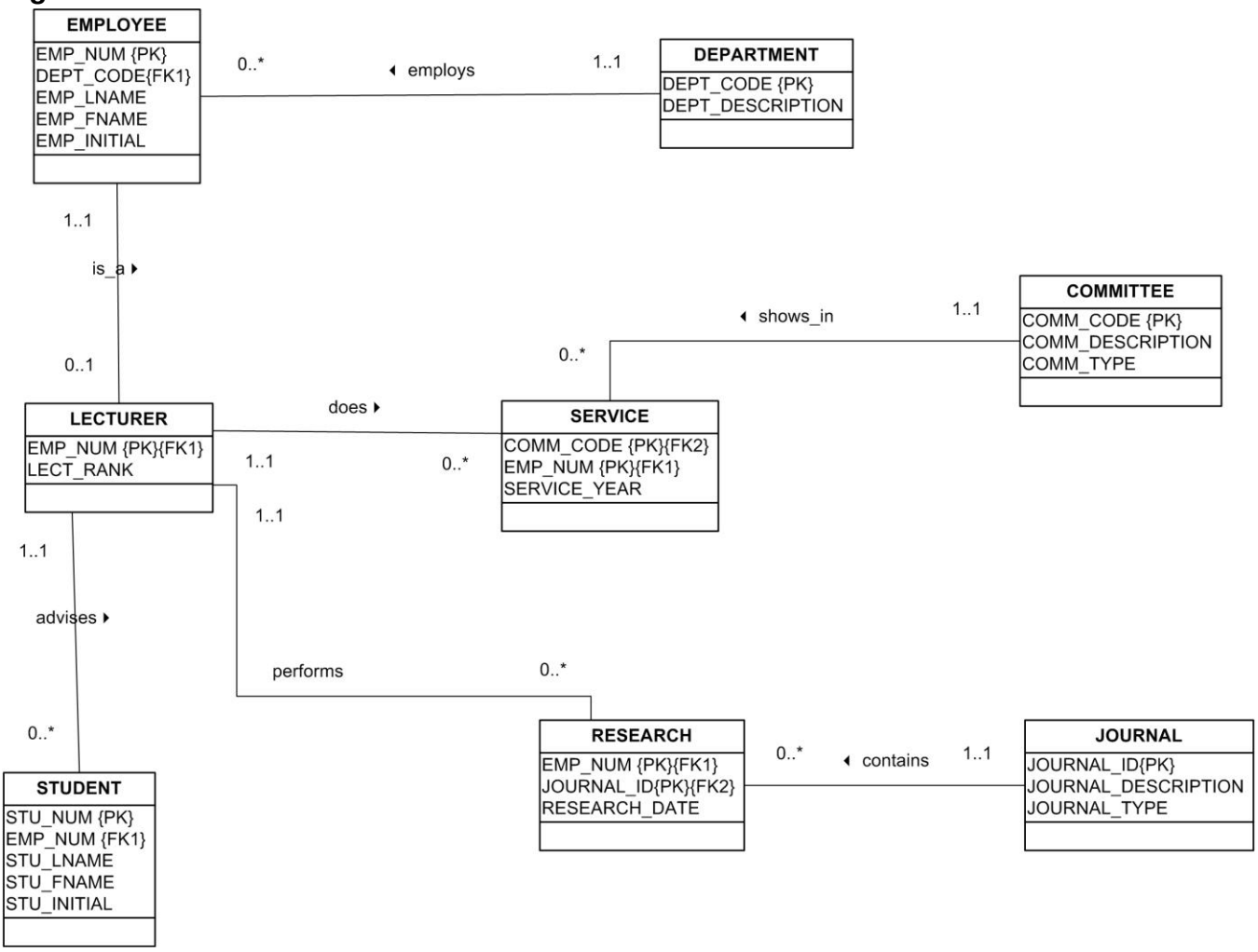
Question 5 (20 Marks)

b. Draw the ERD to reflect the dependency diagrams you drew in Question 4a and 4c. (Note: You may have to create additional attributes to define the proper PKs and FKs. Make sure that all of your attributes conform to the naming conventions.) (20)

Answer:

Given the discussion in the previous problem segment d, we have incorporated additional features in the ERD shown in Figure 5a. Note that we have eliminated the *:* relationships in this design by creating composite entities. This design is implementable and it meets design standards. Normalization was part of the process that led to this solution, but it was *only* a part of that solution. **Normalization does not replace design!**

Figure 5a: The ERD for Problem a



==========

Due date	Tutorial matter covered in prescribed book
10 April 2016	Chapter 1: The Database Approach Chapter 2: Data Models Chapter 3: Relational Model Characteristics Chapter 5: Data Modelling with Entity Relationship (ER) Diagrams Chapter 6: Data Modelling Advanced Concepts Chapter 7: Normalizing Database Designs Chapter 10: Database Development Process Chapter 11: Conceptual, Logical and Physical Database Design
Unique number: xxxxxxxxxx	

Questions and Answers:

Questions 1 (31 Marks)

11. Discuss the lack of data independence in file systems. **(3 marks)**
    File systems exhibit data dependence because file access is dependent on a file's data characteristics. Therefore, any time the file data characteristics are changed, the programs that access the data within those files must be modified. Data independence exists when changes in the data characteristics don't require changes in the programs that access those data. *(Chapter 1, page 268)*

12. What is data independence, and why is it important? **(3 Marks)** ***(Chapter 1, page 183)***
Data independence exists when data access programs are not subject to change when any of the file's data characteristics change. Data independence is important because it substantially decreases programming effort and program maintenance costs. (Chapter 1, page 183)

13. What is a business rule, and what is its purpose in data modeling? **(3 Marks)** ***(Chapter 2, page 230)***
A business rule is a brief, precise, and unambigous description of a policy, procedure, or principle within a specific organization's environment. In a sense, business rules are misnamed: they apply to any organization -- a business, a government unit, a religious group, or a research laboratory; large or small -- that stores and uses data to generate information.
Business rules are derived from a description of operations. As its name implies, a description of operations is a detailed narrative that describes the operational nvironment of an organization. Such a description requires great precision and detail. If the description of operations is incorrect or inomplete, the business rules derived from it will not reflect the real world data environment accurately, thus leading to poorly defined data models, which lead to poor database designs. In turn, poor database designs lead to poor applications, thus setting the stage for poor decision making – which may ultimately lead to the demise of the organization.
Note especially that business rules help to create and enforce actions within that organization's environment. Business rules must be rendered in writing and updated to reflect any change in the organization's operational environment. Properly written business rules are used to define entities, attributes, relationships, and constraints. Because these components form the basis for a database design, the careful derivation and definition of business rules is crucial to good database design. (Chapter 2, page 260)

14. Give an example of each of the three types of relationships**. (3 Marks)**
1:1
An academic department is chaired by one professor; a professor may chair only one academic department.

1:*
A customer may generate many invoices; each invoice is generated by one customer.

:
An employee may have earned many degrees; a degree may have been earned by many employees. (Chapter 2, page 186)

15. Define and describe the basic characteristics of a NoSQL database. **(4 Marks)**

Every time you search for a product on Amazon, send messages to friends in Facebook, watch a video in YouTube or search for directions in Google Maps, you are using a NoSQL database. NoSQL refers to a new generation of databases that address the very specific challenges of the "big data" era and have the following general characteristics:
- **Not based on the relational model.**

These databases are generally based on a variation of the key-value data model rather than in the relational model, hence the NoSQL name. The key-value data model is based on a structure composed of two data elements: a key and a value; in which for every key there is a corresponding value (or a set of values). The key-value data model is also referred to as the attribute-value or associative data model. In the key-value data model, each row represents one attribute of one entity instance. The "key" column points to an attribute and the "value" column contains the actual value for the attribute. The data type of the "value" column is generally a long string to accommodate the variety of actual data types of the values that are placed in the column.

 Support distributed database architectures.

One of the big advantages of NoSQL databases is that they generally use a distributed architecture. In fact, several of them (Cassandra, Big Table) are designed to use low cost commodity servers to form a complex network of distributed database nodes.

 Provide high scalability, high availability and fault tolerance.

NoSQL databases are designed to support the ability to add capacity (add database nodes to the distributed database) when the demand is high and to do it transparently and without downtime. Fault tolerant means that if one of the nodes in the distributed database fails, the database will keep operating as normal.

 Support very large amounts of sparse data.

Because NoSQL databases use the key-value data model, they are suited to handle very high volumes of sparse data; that is for cases where the number of attributes is very large but the number of actual data instances is low.

 Geared toward performance rather than transaction consistency.

One of the biggest problems of very large distributed databases is to enforce data consistency. Distributed databases automatically make copies of data elements at multiple nodes – to ensure high availability and fault tolerance. If the node with the requested data goes down, the request can be served from any other node with a copy of the data. However, what happen if the network goes down during a data update? In a relational database, transaction updates are guaranteed to be consistent or the transaction is rolled back. NoSQL databases sacrifice consistency in order to attain high levels of performance. NoSQL databases provide eventual consistency. *Eventual consistency* is a feature of NoSQL databases that indicates that data are not guaranteed to be consistent immediately after an update (across all copies of the data) but rather, that updates will propagate through the system and eventually all data copies will be consistent.. *(Chapter 2, pages 264)*

16. Why are entity integrity and referential integrity important in a database? **(4 Marks)** ***(Chapter 3, page 239)***

    Entity integrity and referential integrity are important because they are the basis for expressing and implementing relationships in the entity relationship model. Entity integrity ensures that each row is uniquely identified by the primary key. Therefore, entity integrity means that a proper search for an *existing* tuple (row) will always be successful. (And the failure to find a match on a row search will always mean that the row for which the search is conducted does not exist in that table.) Referential integrity means that, if the foreign key contains a value, that value refers to an existing valid tuple (row) in another relation. Therefore, referential integrity ensures that it will be impossible to assign a non-existing foreign key value to a table. *(Chapter 3, page 239)*

17. What is a composite entity, and when is it used? (3 Marks)

A composite entity is generally used to transform *:* relationships into 1:* relationships. (Review the discussion that accompanied Figures 5.26 through 5.28) A composite entity, also known as a bridge entity, is one that has a primary key composed of multiple attributes. The PK attributes are inherited from the entities that it relates to one another. (Chapter 5, pages 189)

18. What is a derived attribute? Give an example **(3)**

A derived attribute is an attribute whose value is calculated (derived) from other attributes. The derived attribute need not be physically stored within the database; instead, it can be derived by using an algorithm. For example, an employee's age, EMP_AGE, may be found by computing the integer value of the difference between the current date and the EMP_DOB. If you use MS Access, you would use INT((DATE() – EMP_DOB)/365).

Similarly, a sales clerk's total gross pay may be computed by adding a computed sales commission to base pay. For instance, if the sales clerk's commission is 1%, the gross pay may be computed by EMP_GROSSPAY = INV_SALES*1.01 + EMP_BASEPAY

Or the invoice line item amount may be calculated by LINE_TOTAL = LINE_UNITS*PROD_PRICE (chapter 5 Page 37)

19. What is an overlapping subtype? Give an example. (3)

Overlapping subtypes are subtypes that contain non-unique subsets of the supertype entity set; that is, each entity instance of the supertype may appear in more than one subtype. For example, in a university environment, a person may be an employee or a student or both. In turn, an employee may be a professor as well as an administrator. Because an employee also may be a student, STUDENT and EMPLOYEE are overlapping subtypes of the supertype PERSON, just as LECTURER and ADMINISTRATOR are overlapping subtypes of the supertype EMPLOYEE. The text's Figure 6.4 (reproduced next for your convenience) illustrates overlapping subtypes.

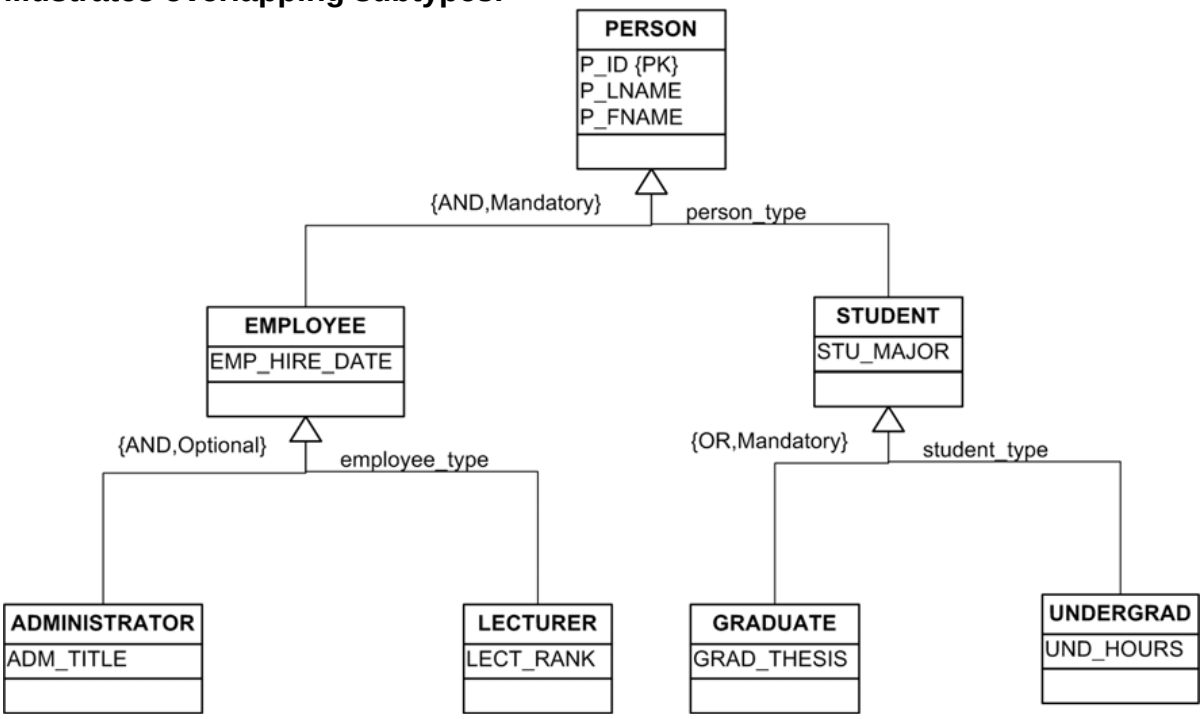


FIGURE 6.4 Specialization Hierarchy with Overlapping Subtypes (Chapter 6 Page 233)

20. What is a partial dependency? With what normal form is it associated? (2 Marks)

A partial dependency exists when an attribute is dependent on only a portion of the primary key. This type of dependency is associated with 1NF. (Chapter 7, pages 189)

3.1. What three (often conflicting) database requirements must be addresses in a database design? **(3)**

Answer: pg209, chap 5
Design elegance
Processing speed
Information requirements

Question 2 (17 Marks)

a. What is the data dictionary's function in database design? (4)

A good data dictionary provides a precise description of the characteristics of all the entities and attributes found within the database. The data dictionary thus makes it easier to check for the existence of synonyms and homonyms, to check whether all attributes exist to support required reports, to verify appropriate relationship representations, and so on. The data dictionary's contents are both developed and used during the six DBLC phases:

DATABASE DESIGN

The data dictionary contents are used to verify the database design components: entities, attributes, and their relationships. The designer also uses the data dictionary to check the database design for homonyms and synonyms and verifies that the entities and attributes will support all required query and report requirements.

IMPLEMENTATION AND LOADING

The DBMS's data dictionary helps to resolve any remaining attribute definition inconsistencies.

TESTING AND EVALUATION

If problems develop during this phase, the data dictionary contents may be used to help restructure the basic design components to make sure that they support all required operations.

OPERATION

If the database design still yields (the almost inevitable) operational glitches, the data dictionary may be used as a quality control device to ensure that operational modifications to the database do not conflict with existing components.

MAINTENANCE AND EVOLUTION

As users face inevitable changes in information needs, the database may be modified to support those needs. Perhaps entities, attributes, and relationships must be added, or relationships must be changed. If new database components are fit into the design, their introduction may produce conflict with existing components. The data dictionary turns out to be a very useful tool to check whether a suggested change invites conflicts within the database design and, if so, how such conflicts may be (Chapter 5, page 554)

b. What is the impact of threats on database security? Give examples. **(6)**

Threats are any set of circumstances that have the potential to cause loss, misuse or harm to the system and/or its data. Threats can cause:

- **The loss of the integrity of data through unauthorized modification. For example a person gaining unauthorized access to a bank account and removing some money from the account.**
- **The loss of availability of the data. For example some adversary causes the database system from being operational which stops authorized users of the data from accessing it.**
- **The loss of confidentiality of the data (also referred to as the privacy of data). This could be caused by a person gaining access to private information such as a password or a bank account balance. (Chapter 10, page 529*)***

c. What are the stages of the conceptual database design? (4)

Answer

The stages of conceptual database design are:

- Data analysis and requirements
- Entity relationship modeling and normalization
- Data model verification
- Distributed database design ***(Chapter 11, page 563)***

d. Describe the purpose of a B-tree. (3)

Answer:

Within a DBMS, indexes are often stored in a data structure known as a tree. Trees are generally more efficient in storing indexes as they reduce the time of the search compared with other data structures such as lists. These trees are often referred to as Balanced or B-trees and are used to maintain an ordered set of indexes or data to allow efficient operations to select, delete and insert data. ***(Chapter 11, page 596)***

Question 3 (10 Marks)

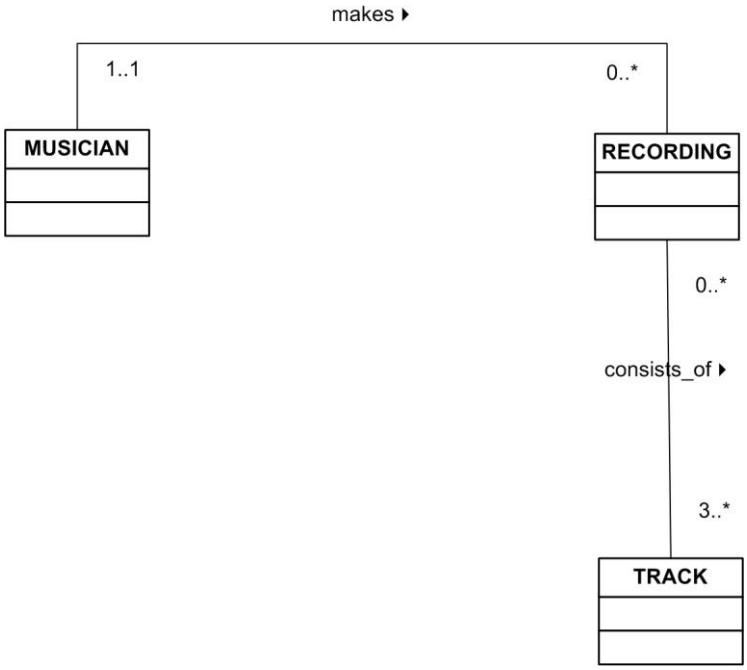
Given the following business rules, create an initial ERD using UML notation.

a. A musician makes at least one recording, but may over a period of time make many recordings.

b. One recording consists if at least 3 or more tracks.

c. A track can appear on more than one recording.

The solution is presented in the following Figure:



Question 4 (25)

To keep track of office furniture, computers, printers, and so on, the FOUNDIT Company uses the table structure shown in the following Table:

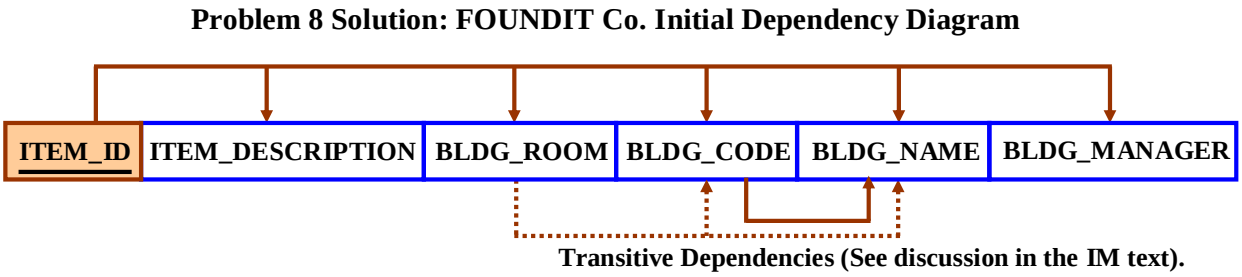
Table: Sample ITEM Records

Attribute Name	Sample Value	Sample Value	Sample Value
ITEM_ID	231134-678	342245-225	254668-449
ITEM_LABEL	HP DeskJet 895Cse	HP Toner	DT Scanner
ROOM_NUMBER	325	325	123
BLDG_CODE	NTC	NTC	CSF
BLDG_NAME	Nottooclear	Nottoclear	Canseefar
BLDG_MANAGER	I. B. Rightonit	I. B. Rightonit	May B. Next

a. Given that information, write the relational schema and draw the dependency diagram. Make sure that you label the transitive and/or partial dependencies. (10)

The answers to this problem are shown in Figure P7.8 and the relational schema definition below the figure..

Figure P7.8 The FOUNDIT Co. Initial Dependency Diagram



The dotted transitive dependency lines indicate that these transitive dependencies are subject to interpretation. We will address these dependencies in the discussion that accompanies Problem b's solution.

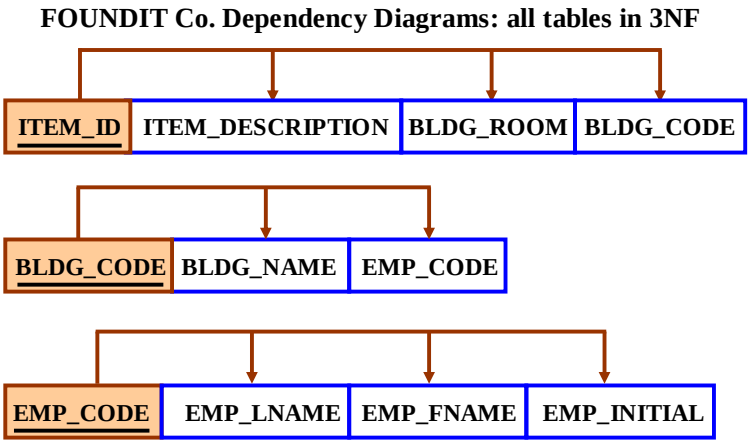
The relational schema may be written as follows:

ITEM(ITEM_ID, ITEM_DESCRIPTION, BLDG_ROOM, BLDG_CODE, BLDG_NAME, BLDG_MANAGER)

b. Using the answer to Problem 8, create the relational schemas and create a set of dependency diagrams that meet 3NF requirements. Rename attributes to meet the naming conventions, and create new entities and attributes as necessary. (15)

The dependency diagrams are shown in the below Figure. The dependency diagrams reflect the notion that one employee manages each building.

Figure: FOUNDIT Co. 3NF and Its Relational Diagram



The relational schemas are written as follows:

EMPLOYEE(EMP_CODE, EMP_LNAME, EMP_FNAME, EMP_INITIAL)

BUILDING(BLDG_CODE, BLDG_NAME, EMP_CODE)

ITEM(ITEM_ID, ITEM_DESCRIPTION, ITEM_ROOM, BLDG_CODE)

As you discuss the dependency diagrams in above Figure, remember that BLDG_CODE is not a determinant of BLDG_ROOM. A building can have many rooms, so knowing the building code will not tell you what the room in that building is.

If the room is numbered to reflect the building it is in – for example, HE105 indicates room 105 in the Heinz building – one might argue that the BLDG_ROOM value is the determinant of the BLDG_CODE and the BLDG_NAME values. You will learn in Chapter 8, "Introduction to Structured Query Language (SQL)," that you can create a query to find a building by looking at room prefixes. However, if you define dependencies in strictly relational algebra terms, you might argue that partitioning the attribute value to "create" a dependency indicates that the partitioned attribute is not (in that strict sense) a determinant. Although we have indicated a transitive dependency from BLDG_ROOM to BLDG_CODE and BLDG_NAME, we have used a dotted line to indicate that there is room for argument in this set of transitive dependencies. In any case, the (arguable) dependency BLDG_ROOM $\rightarrow$ BLDG_CODE does not create any problems in a practical sense, so we have not identified it in the Problem 9 solution.

Clearly, BLDG_CODE is a determinant of BLDG_NAME. Therefore, the transitive dependency is marked properly in the Problem 8 solution.

Question 5 (20 Marks)

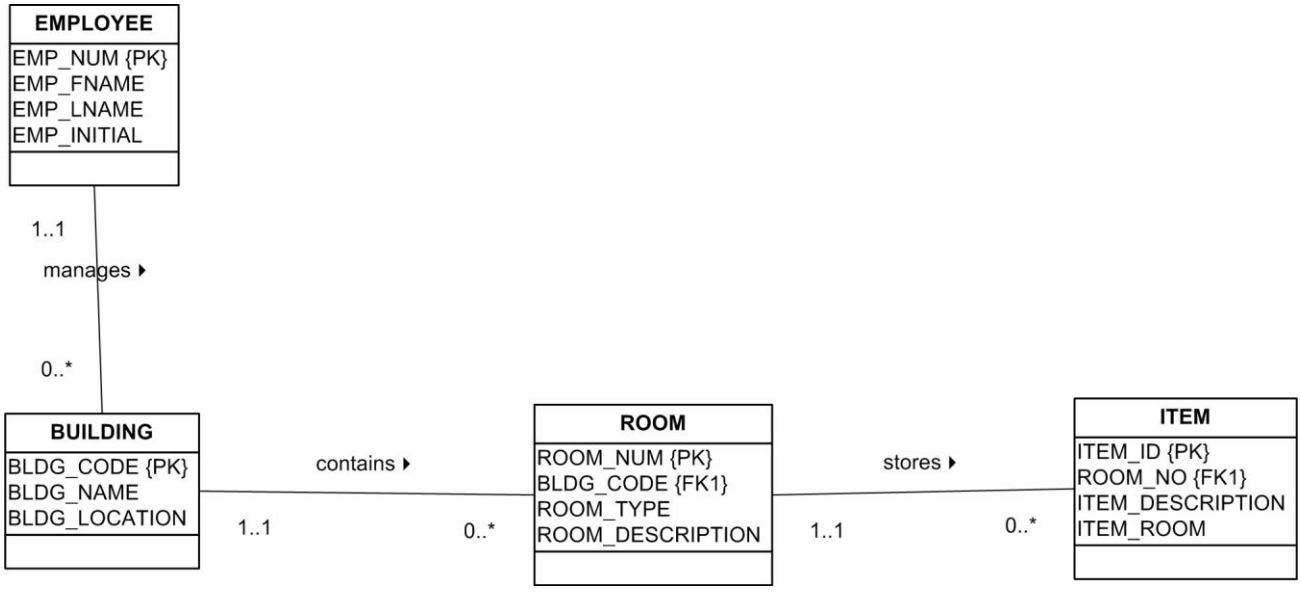
Using the results of Problems of Question 4.a-b above, now draw/create the ERD containing all primary keys and main attributes. (20)

Answer:

The following diagram show that, in this case, the ER diagram reflects the business rule that one employee can manage many (or at least more than one) building. Because all employees are not *required* to manage buildings, BUILDING is optional to EMPLOYEE in the *manages* relationship. Once again, the nature of this relationship is not and cannot be reflected in the dependency diagram.

NOTE
We also assume here that each item has a unique item code and that, therefore, an item can be located in only one place at a time. However, we demonstrate in Appendixes B and C that inventory control requirements usually cover both durable and consumable items. Although durables such as tables, desks, lamps, computers, printers, etc. would be uniquely identified by an assigned inventory code, consumables such as individual reams of paper would clearly not be so identified. Therefore, a given inventory description such as "8.5 inch x 11 inch laser printer paper" could describe reams of paper located in many different buildings and in rooms within those buildings. We demonstrate in Appendixes B and C how such a condition may be properly handled.

Figure: The FOUNDIT Co. ERD



As you examine the above Figure, note that the BLDG_ROOM is actually an ITEM entity attribute, so it is appropriate to rename it ITEM_ROOM. Also, keep in mind that a room may be related to the building in which it is located. (A BUILDING may contain many ROOMs. Each ROOM is located in a single building.) Therefore, you can expand the design shown in the above Figure. This solution assumes that a room is directly traceable to a building. For example, room SC-508 would be located in the Science (SC) Building and room BA-305 would be located in the Business Administration (BA) building. Note that we have made ROOM optional to BUILDING to reflect the likelihood that some buildings – such as storage sheds -- may not contain designated (numbered) rooms. Although optionalities make excellent default conditions, it is always wise to establish the optionality based on a business rule. In any case, the designer must *ask* about the nature of the room/building relationship.

==========