DATABASE MANAGEMENT SYSTEMS SOLUTIONS MANUAL THIRD EDITION

Raghu Ramakrishnan
University of Wisconsin
Madison, WI, USA

Johannes Gehrke
Cornell University
Ithaca, NY, USA

Jeff Derstadt, Scott Selikoff, and Lin Zhu
Cornell University
Ithaca, NY, USA

CONTENTS

PREFACE iii

1 INTRODUCTION TO DATABASE SYSTEMS 1

2 INTRODUCTION TO DATABASE DESIGN 6

3 THE RELATIONAL MODEL 16

4 RELATIONAL ALGEBRA AND CALCULUS 28

5 SQL: QUERIES, CONSTRAINTS, TRIGGERS 45

6 DATABASE APPLICATION DEVELOPMENT 63

7 INTERNET APPLICATIONS 66

8 OVERVIEW OF STORAGE AND INDEXING 73

9 STORING DATA: DISKS AND FILES 81

10 TREE-STRUCTURED INDEXING 88

11 HASH-BASED INDEXING 100

12 OVERVIEW OF QUERY EVALUATION 119

13 EXTERNAL SORTING 126

14 EVALUATION OF RELATIONAL OPERATORS 131

15 A TYPICAL QUERY OPTIMIZER 144

16 OVERVIEW OF TRANSACTION MANAGEMENT 159

17 CONCURRENCY CONTROL 167

18 CRASH RECOVERY 179

19 SCHEMA REFINEMENT AND NORMAL FORMS 189

20 PHYSICAL DATABASE DESIGN AND TUNING 204

21 SECURITY 215

PREFACE

It is not every question that deserves an answer.

Publius Syrus, 42 B.C.

I hope that most of the questions in this book deserve an answer. The set of questions is unusually extensive, and is designed to reinforce and deepen students' understanding of the concepts covered in each chapter. There is a strong emphasis on quantitative and problem-solving type exercises.

While I wrote some of the solutions myself, most were written originally by students in the database classes at Wisconsin. I'd like to thank the many students who helped in developing and checking the solutions to the exercises; this manual would not be available without their contributions. In alphabetical order: X. Bao, S. Biao, M. Chakrabarti, C. Chan, W. Chen, N. Cheung, D. Colwell, J. Derstadt, C. Fritz, V. Ganti, J. Gehrke, G. Glass, V. Gopalakrishnan, M. Higgins, T. Jasmin, M. Krishnaprasad, Y. Lin, C. Liu, M. Lusignan, H. Modi, S. Narayanan, D. Randolph, A. Ranganathan, J. Reminga, A. Therber, M. Thomas, Q. Wang, R. Wang, Z. Wang and J. Yuan. In addition, James Harrington and Martin Reames at Wisconsin and Nina Tang at Berkeley provided especially detailed feedback.

Several students contributed to each chapter's solutions, and answers were subsequently checked by me and by other students. This manual has been in use for several semesters. I hope that it is now mostly accurate, but I'm sure **it still contains errors and omissions**. If you are a student and you do not understand a particular solution, contact your instructor; it may be that you are missing something, but it may also be that the solution is incorrect! If you discover a bug, please send me mail (`raghu@cs.wisc.edu`) and I will update the manual promptly.

The latest version of this solutions manual is distributed freely through the Web; go to the home page mentioned below to obtain a copy.

For More Information

The home page for this book is at URL:

Database Management Systems Solutions Manual Third Edition

http://www.cs.wisc.edu/~dbbook

This page is frequently updated and contains information about the book, past and current users, and the software. This page also contains a link to all known errors in the book, the accompanying slides, and the software. *Since the solutions manual is distributed electronically, all known errors are immediately fixed and no list of errors is maintained.* Instructors are advised to visit this site periodically; they can also register at this site to be notified of important changes by email.

1

INTRODUCTION TO DATABASE SYSTEMS

Exercise 1.1 Why would you choose a database system instead of simply storing data in operating system files? When would it make sense *not* to use a database system?

Answer 1.1 A *database* is an integrated collection of data, usually so large that it has to be stored on secondary storage devices such as disks or tapes. This data can be maintained as a collection of operating system files, or stored in a *DBMS* (database management system). The advantages of using a DBMS are:

- *Data independence and efficient access.* Database application programs are independent of the details of data representation and storage. The conceptual and external schemas provide independence from physical storage decisions and logical design decisions respectively. In addition, a DBMS provides efficient storage and retrieval mechanisms, including support for very large files, index structures and query optimization.
- *Reduced application development time.* Since the DBMS provides several important functions required by applications, such as concurrency control and crash recovery, high level query facilities, etc., only application-specific code needs to be written. Even this is facilitated by suites of application development tools available from vendors for many database management systems.
- *Data integrity and security.* The view mechanism and the authorization facilities of a DBMS provide a powerful access control mechanism. Further, updates to the data that violate the semantics of the data can be detected and rejected by the DBMS if users specify the appropriate *integrity constraints*.
- *Data administration.* By providing a common umbrella for a large collection of data that is shared by several users, a DBMS facilitates maintenance and data administration tasks. A good DBA can effectively shield end-users from the chores of fine-tuning the data representation, periodic back-ups etc.

- *Concurrent access and crash recovery.* A DBMS supports the notion of a *transaction*, which is conceptually a single user's sequential program. Users can write transactions as if their programs were running in isolation against the database. The DBMS executes the actions of transactions in an interleaved fashion to obtain good performance, but schedules them in such a way as to ensure that conflicting operations are not permitted to proceed concurrently. Further, the DBMS maintains a continuous log of the changes to the data, and if there is a system crash, it can restore the database to a *transaction-consistent* state. That is, the actions of incomplete transactions are undone, so that the database state reflects only the actions of completed transactions. Thus, if each complete transaction, executing alone, maintains the consistency criteria, then the database state after recovery from a crash is consistent.

If these advantages are not important for the application at hand, using a collection of files may be a better solution because of the increased cost and overhead of purchasing and maintaining a DBMS.

Exercise 1.2 What is logical data independence and why is it important?

Answer 1.2 Answer omitted.

Exercise 1.3 Explain the difference between logical and physical data independence.

Answer 1.3 Logical data independence means that users are shielded from changes in the logical structure of the data, while physical data independence insulates users from changes in the physical storage of the data. We saw an example of logical data independence in the answer to Exercise 1.2. Consider the Students relation from that example (and now assume that it is not replaced by the two smaller relations). We could choose to store Students tuples in a heap file, with a clustered index on the sname field. Alternatively, we could choose to store it with an index on the gpa field, or to create indexes on both fields, or to store it as a file sorted by gpa. These storage alternatives are not visible to users, except in terms of improved performance, since they simply see a relation as a set of tuples. This is what is meant by physical data independence.

Exercise 1.4 Explain the difference between external, internal, and conceptual schemas. How are these different schema layers related to the concepts of logical and physical data independence?

Answer 1.4 Answer omitted.

Exercise 1.5 What are the responsibilities of a DBA? If we assume that the DBA is never interested in running his or her own queries, does the DBA still need to understand query optimization? Why?

Answer 1.5 The DBA is responsible for:

- *Designing the logical and physical schemas, as well as widely-used portions of the external schema.*
- *Security and authorization.*
- *Data availability and recovery from failures.*
- *Database tuning:* The DBA is responsible for evolving the database, in particular the conceptual and physical schemas, to ensure adequate performance as user requirements change.

A DBA needs to understand query optimization even if s/he is not interested in running his or her own queries because some of these responsibilities (database design and tuning) are related to query optimization. Unless the DBA understands the performance needs of widely used queries, and how the DBMS will optimize and execute these queries, good design and tuning decisions cannot be made.

Exercise 1.6 Scrooge McNugget wants to store information (names, addresses, descriptions of embarrassing moments, etc.) about the many ducks on his payroll. Not surprisingly, the volume of data compels him to buy a database system. To save money, he wants to buy one with the fewest possible features, and he plans to run it as a stand-alone application on his PC clone. Of course, Scrooge does not plan to share his list with anyone. Indicate which of the following DBMS features Scrooge should pay for; in each case, also indicate why Scrooge should (or should not) pay for that feature in the system he buys.

1. A security facility.
2. Concurrency control.
3. Crash recovery.
4. A view mechanism.
5. A query language.

Answer 1.6 Answer omitted.

Exercise 1.7 Which of the following plays an important role in *representing* information about the real world in a database? Explain briefly.

1. The data definition language.

2. The data manipulation language.
3. The buffer manager.
4. The data model.

Answer 1.7 Let us discuss the choices in turn.

- The data definition language is important in representing information because it is used to describe external and logical schemas.
- The data manipulation language is used to access and update data; it is not important for representing the data. (Of course, the data manipulation language must be aware of how data is represented, and reflects this in the constructs that it supports.)
- The buffer manager is not very important for representation because it brings arbitrary disk pages into main memory, independent of any data representation.
- The data model is fundamental to representing information. The data model determines what data representation mechanisms are supported by the DBMS. The data definition language is just the specific set of language constructs available to describe an actual application's data in terms of the *data model*.

Exercise 1.8 Describe the structure of a DBMS. If your operating system is upgraded to support some new functions on OS files (e.g., the ability to force some sequence of bytes to disk), which layer(s) of the DBMS would you have to rewrite to take advantage of these new functions?

Answer 1.8 Answer omitted.

Exercise 1.9 Answer the following questions:

1. What is a transaction?
2. Why does a DBMS interleave the actions of different transactions instead of executing transactions one after the other?
3. What must a user guarantee with respect to a transaction and database consistency? What should a DBMS guarantee with respect to concurrent execution of several transactions and database consistency?
4. Explain the strict two-phase locking protocol.
5. What is the WAL property, and why is it important?

Answer 1.9 Let us answer each question in turn:

1. A transaction is any one execution of a user program in a DBMS. This is the basic unit of change in a DBMS.

2. A DBMS is typically shared among many users. Transactions from these users can be interleaved to improve the execution time of users' queries. By interleaving queries, users do not have to wait for other user's transactions to complete fully before their own transaction begins. Without interleaving, if user A begins a transaction that will take 10 seconds to complete, and user B wants to begin a transaction, user B would have to wait an additional 10 seconds for user A's transaction to complete before the database would begin processing user B's request.

3. A user must guarantee that his or her transaction does not corrupt data or insert nonsense in the database. For example, in a banking database, a user must guarantee that a cash withdraw transaction accurately models the amount a person removes from his or her account. A database application would be worthless if a person removed 20 dollars from an ATM but the transaction set their balance to zero! A DBMS must guarantee that transactions are executed fully and independently of other transactions. An essential property of a DBMS is that a transaction should execute atomically, or as if it is the only transaction running. Also, transactions will either complete fully, or will be aborted and the database returned to it's initial state. This ensures that the database remains consistent.

4. Strict two-phase locking uses shared and exclusive locks to protect data. A transaction must hold all the required locks before executing, and does not release any lock until the transaction has completely finished.

5. The WAL property affects the logging strategy in a DBMS. The WAL, Write-Ahead Log, property states that each write action must be recorded in the log (on disk) before the corresponding change is reflected in the database itself. This protects the database from system crashes that happen during a transaction's execution. By recording the change in a log before the change is truly made, the database knows to undo the changes to recover from a system crash. Otherwise, if the system crashes just after making the change in the database but before the database logs the change, then the database would not be able to detect his change during crash recovery.

2

INTRODUCTION TO DATABASE DESIGN

Exercise 2.1 Explain the following terms briefly: *attribute, domain, entity, relationship, entity set, relationship set, one-to-many relationship, many-to-many relationship, participation constraint, overlap constraint, covering constraint, weak entity set, aggregation*, and *role indicator*.

Answer 2.1 Term explanations:

- *Attribute* - a property or description of an entity. A toy department employee entity could have attributes describing the employee's name, salary, and years of service.
- *Domain* - a set of possible values for an attribute.
- *Entity* - an object in the real world that is distinguishable from other objects such as the green dragon toy.
- *Relationship* - an association among two or more entities.
- *Entity set* - a collection of similar entities such as all of the toys in the toy department.
- *Relationship set* - a collection of similar relationships
- *One-to-many relationship* - a key constraint that indicates that one entity can be associated with many of another entity. An example of a one-to-many relationship is when an employee can work for only one department, and a department can have many employees.
- *Many-to-many relationship* - a key constraint that indicates that many of one entity can be associated with many of another entity. An example of a many-to-many relationship is employees and their hobbies: a person can have many different hobbies, and many people can have the same hobby.

- *Participation constraint* - a participation constraint determines whether relationships must involve certain entities. An example is if every department entity has a manager entity. Participation constraints can either be total or partial. A total participation constraint says that every department has a manager. A partial participation constraint says that every employee does not have to be a manager.
- *Overlap constraint* - within an ISA hierarchy, an overlap constraint determines whether or not two subclasses can contain the same entity.
- *Covering constraint* - within an ISA hierarchy, a covering constraint determines where the entities in the subclasses collectively include all entities in the superclass. For example, with an Employees entity set with subclasses HourlyEmployee and SalaryEmployee, does every Employee entity necessarily have to be within either HourlyEmployee or SalaryEmployee?
- *Weak entity set* - an entity that cannot be identified uniquely without considering some primary key attributes of another identifying owner entity. An example is including Dependent information for employees for insurance purposes.
- *Aggregation* - a feature of the entity relationship model that allows a relationship set to participate in another relationship set. This is indicated on an ER diagram by drawing a dashed box around the aggregation.
- *Role indicator* - If an entity set plays more than one role, role indicators describe the different purpose in the relationship. An example is a single Employee entity set with a relation Reports-To that relates supervisors and subordinates.

Exercise 2.2 A university database contains information about professors (identified by social security number, or SSN) and courses (identified by courseid). Professors teach courses; each of the following situations concerns the Teaches relationship set. For each situation, draw an ER diagram that describes it (assuming no further constraints hold).

1. Professors can teach the same course in several semesters, and each offering must be recorded.
2. Professors can teach the same course in several semesters, and only the most recent such offering needs to be recorded. (Assume this condition applies in all subsequent questions.)
3. Every professor must teach some course.
4. Every professor teaches exactly one course (no more, no less).
5. Every professor teaches exactly one course (no more, no less), and every course must be taught by some professor.

6. Now suppose that certain courses can be taught by a team of professors jointly, but it is possible that no one professor in a team can teach the course. Model this situation, introducing additional entity sets and relationship sets if necessary.

Answer 2.2 Answer omitted.

Exercise 2.3 Consider the following information about a university database:

- Professors have an SSN, a name, an age, a rank, and a research specialty.
- Projects have a project number, a sponsor name (e.g., NSF), a starting date, an ending date, and a budget.
- Graduate students have an SSN, a name, an age, and a degree program (e.g., M.S. or Ph.D.).
- Each project is managed by one professor (known as the project's principal investigator).
- Each project is worked on by one or more professors (known as the project's co-investigators).
- Professors can manage and/or work on multiple projects.
- Each project is worked on by one or more graduate students (known as the project's research assistants).
- When graduate students work on a project, a professor must supervise their work on the project. Graduate students can work on multiple projects, in which case they will have a (potentially different) supervisor for each one.
- Departments have a department number, a department name, and a main office.
- Departments have a professor (known as the chairman) who runs the department.
- Professors work in one or more departments, and for each department that they work in, a time percentage is associated with their job.
- Graduate students have one major department in which they are working on their degree.
- Each graduate student has another, more senior graduate student (known as a student advisor) who advises him or her on what courses to take.

Design and draw an ER diagram that captures the information about the university. Use only the basic ER model here; that is, entities, relationships, and attributes. Be sure to indicate any key and participation constraints.

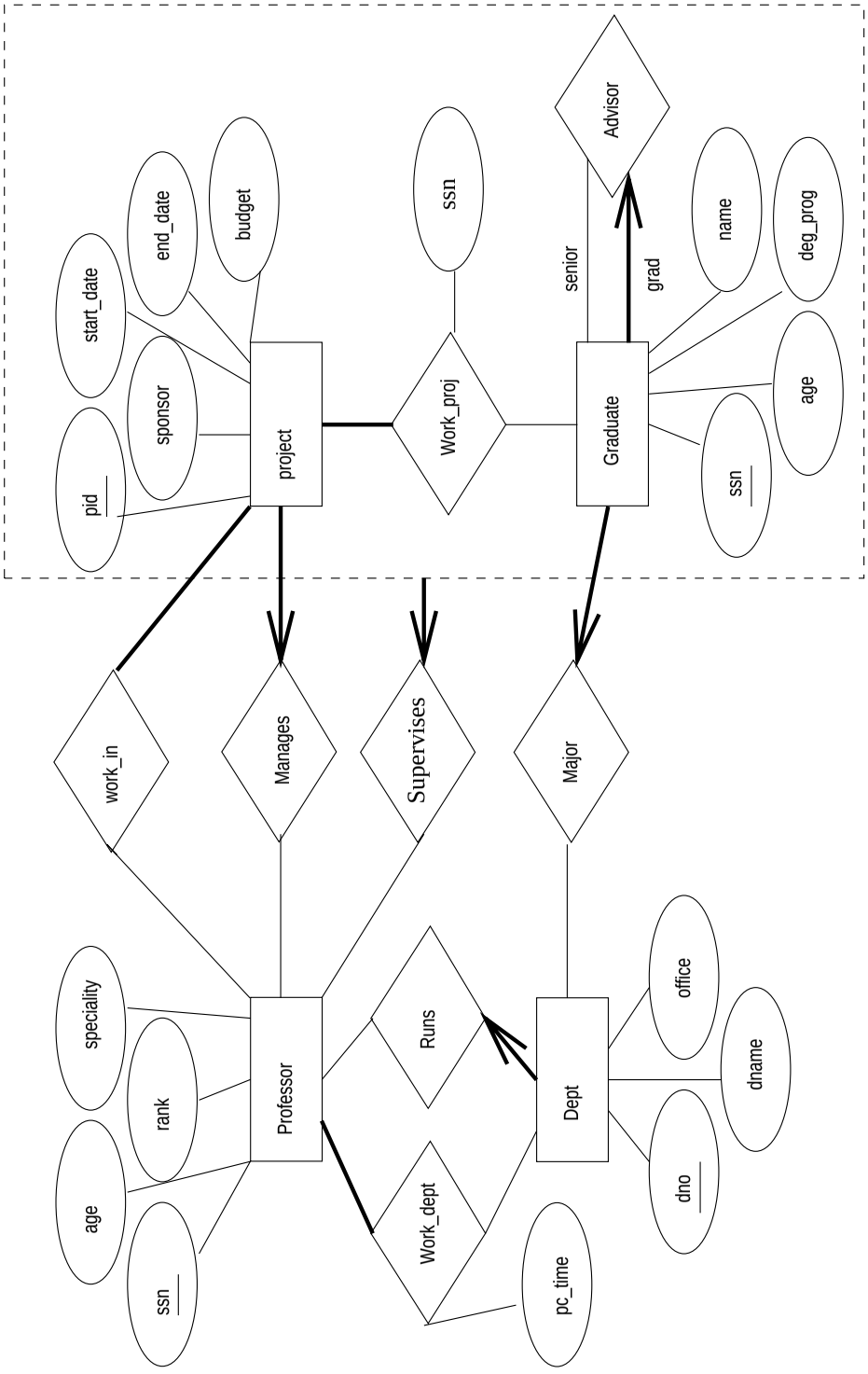


Figure 2.1 ER Diagram for Exercise 2.3

Answer 2.3 The ER diagram is shown in Figure 2.1.

Exercise 2.4 A company database needs to store information about employees (identified by *ssn*, with *salary* and *phone* as attributes), departments (identified by *dno*, with *dname* and *budget* as attributes), and children of employees (with *name* and *age* as attributes). Employees *work* in departments; each department is *managed by* an employee; a child must be identified uniquely by *name* when the parent (who is an employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company.

Draw an ER diagram that captures this information.

Answer 2.4 Answer omitted.

Exercise 2.5 Notown Records has decided to store information about musicians who perform on its albums (as well as other company data) in a database. The company has wisely chosen to hire you as a database designer (at your usual consulting fee of $2500/day).

- Each musician that records at Notown has an SSN, a name, an address, and a phone number. Poorly paid musicians often share the same address, and no address has more than one phone.
- Each instrument used in songs recorded at Notown has a unique identification number, a name (e.g., guitar, synthesizer, flute) and a musical key (e.g., C, B-flat, E-flat).
- Each album recorded on the Notown label has a unique identification number, a title, a copyright date, a format (e.g., CD or MC), and an album identifier.
- Each song recorded at Notown has a title and an author.
- Each musician may play several instruments, and a given instrument may be played by several musicians.
- Each album has a number of songs on it, but no song may appear on more than one album.
- Each song is performed by one or more musicians, and a musician may perform a number of songs.
- Each album has exactly one musician who acts as its producer. A musician may produce several albums, of course.

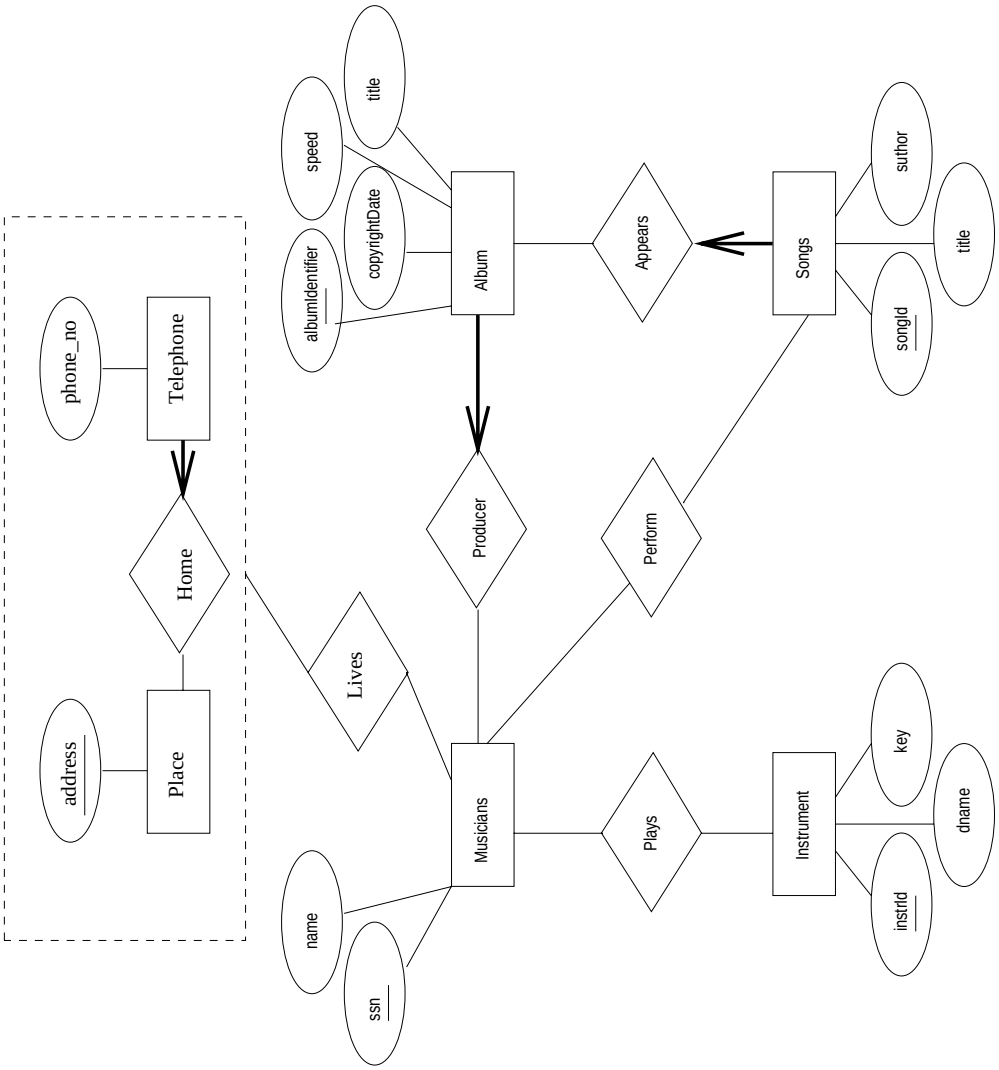


Figure 2.2 ER Diagram for Exercise 2.5

Design a conceptual schema for Notown and draw an ER diagram for your schema. The preceding information describes the situation that the Notown database must model. Be sure to indicate all key and cardinality constraints and any assumptions you make. Identify any constraints you are unable to capture in the ER diagram and briefly explain why you could not express them.

Answer 2.5 The ER diagram is shown in Figure 2.2.

Exercise 2.6 Computer Sciences Department frequent fliers have been complaining to Dane County Airport officials about the poor organization at the airport. As a result, the officials decided that all information related to the airport should be organized using a DBMS, and you have been hired to design the database. Your first task is to organize the information about all the airplanes stationed and maintained at the airport. The relevant information is as follows:

- Every airplane has a registration number, and each airplane is of a specific model.
- The airport accommodates a number of airplane models, and each model is identified by a model number (e.g., DC-10) and has a capacity and a weight.
- A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.
- Each technician is an expert on one or more plane model(s), and his or her expertise may overlap with that of other technicians. This information about technicians must also be recorded.
- Traffic controllers must have an annual medical examination. For each traffic controller, you must store the date of the most recent exam.
- All airport employees (including technicians) belong to a union. You must store the union membership number of each employee. You can assume that each employee is uniquely identified by a social security number.
- The airport has a number of tests that are used periodically to ensure that airplanes are still airworthy. Each test has a Federal Aviation Administration (FAA) test number, a name, and a maximum possible score.
- The FAA requires the airport to keep track of each time a given airplane is tested by a given technician using a given test. For each testing event, the information needed is the date, the number of hours the technician spent doing the test, and the score the airplane received on the test.

1. Draw an ER diagram for the airport database. Be sure to indicate the various attributes of each entity and relationship set; also specify the key and participation constraints for each relationship set. Specify any necessary overlap and covering constraints as well (in English).
2. The FAA passes a regulation that tests on a plane must be conducted by a technician who is an expert on that model. How would you express this constraint in the ER diagram? If you cannot express it, explain briefly.

Answer 2.6 Answer omitted.

Exercise 2.7 The Prescriptions-R-X chain of pharmacies has offered to give you a free lifetime supply of medicine if you design its database. Given the rising cost of health care, you agree. Here's the information that you gather:

- Patients are identified by an SSN, and their names, addresses, and ages must be recorded.
- Doctors are identified by an SSN. For each doctor, the name, specialty, and years of experience must be recorded.
- Each pharmaceutical company is identified by name and has a phone number.
- For each drug, the trade name and formula must be recorded. Each drug is sold by a given pharmaceutical company, and the trade name identifies a drug uniquely from among the products of that company. If a pharmaceutical company is deleted, you need not keep track of its products any longer.
- Each pharmacy has a name, address, and phone number.
- Every patient has a primary physician. Every doctor has at least one patient.
- Each pharmacy sells several drugs and has a price for each. A drug could be sold at several pharmacies, and the price could vary from one pharmacy to another.
- Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and a quantity associated with it. You can assume that, if a doctor prescribes the same drug for the same patient more than once, only the last such prescription needs to be stored.
- Pharmaceutical companies have long-term contracts with pharmacies. A pharmaceutical company can contract with several pharmacies, and a pharmacy can contract with several pharmaceutical companies. For each contract, you have to store a start date, an end date, and the text of the contract.
- Pharmacies appoint a supervisor for each contract. There must always be a supervisor for each contract, but the contract supervisor can change over the lifetime of the contract.

1. Draw an ER diagram that captures the preceding information. Identify any constraints not captured by the ER diagram.
2. How would your design change if each drug must be sold at a fixed price by all pharmacies?
3. How would your design change if the design requirements change as follows: If a doctor prescribes the same drug for the same patient more than once, several such prescriptions may have to be stored.

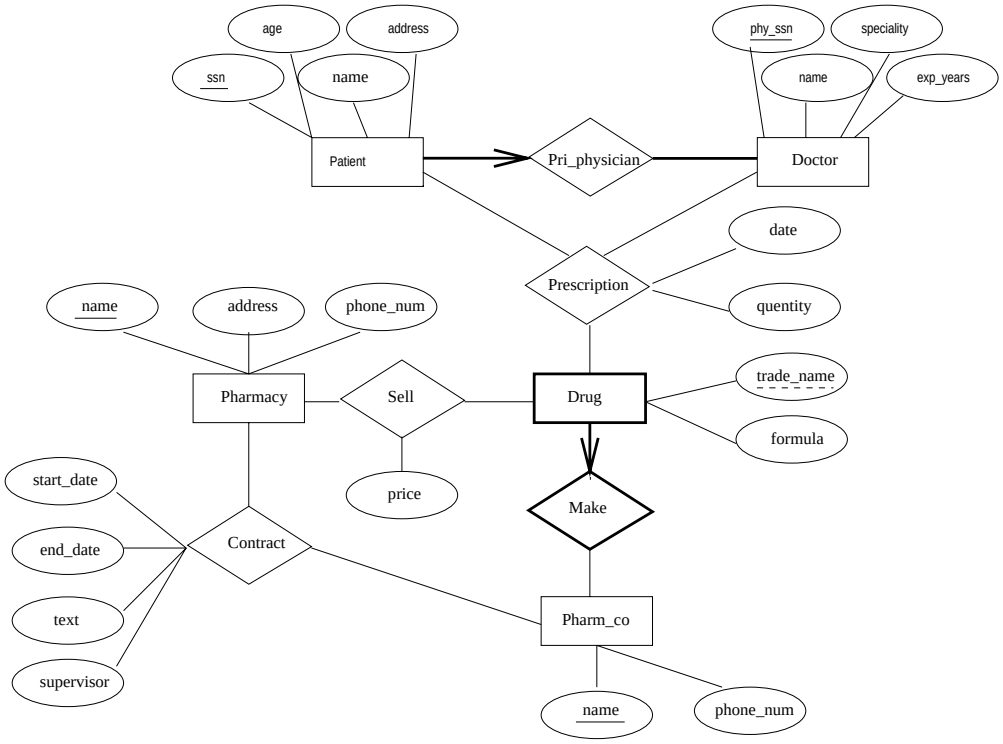


Figure 2.3 ER Diagram for Exercise 2.7

Answer 2.7 1. The ER diagram is shown in Figure 2.3.

2. If the drug is to be sold at a fixed price we can add the price attribute to the Drug entity set and eliminate the price from the Sell relationship set.

3. The date information can no longer be modeled as an attribute of Prescription. We have to create a new entity set called Prescription_date and make Prescription a 4-way relationship set that involves this additional entity set.

Exercise 2.8 Although you always wanted to be an artist, you ended up being an expert on databases because you love to cook data and you somehow confused *database* with *data baste*. Your old love is still there, however, so you set up a database company, ArtBase, that builds a product for art galleries. The core of this product is a database with a schema that captures all the information that galleries need to maintain. Galleries keep information about artists, their names (which are unique), birthplaces, age, and style of art. For each piece of artwork, the artist, the year it was made, its unique title, its type of art (e.g., painting, lithograph, sculpture, photograph), and its price must be stored. Pieces of artwork are also classified into groups of various kinds, for example, portraits, still lifes, works by Picasso, or works of the 19th century; a given piece may belong to more than one group. Each group is identified by a name (like those just given) that describes the group. Finally, galleries keep information about customers. For each customer, galleries keep that person's unique name, address, total amount of dollars spent in the gallery (very important!), and the artists and groups of art that the customer tends to like.

Draw the ER diagram for the database.

Answer 2.8 Answer omitted.

Exercise 2.9 Answer the following questions.

- Explain the following terms briefly: *UML, use case diagrams, statechart diagrams, class diagrams, database diagrams, component diagrams*, and *deployment diagrams*.
- Explain the relationship between ER diagrams and UML.

Answer 2.9 Not yet done.

3

THE RELATIONAL MODEL

Exercise 3.1 Define the following terms: *relation schema, relational database schema, domain, attribute, attribute domain, relation instance, relation cardinality*, and *relation degree.*

Answer 3.1 A *relation schema* can be thought of as the basic information describing a table or *relation.* This includes a set of column names, the data types associated with each column, and the name associated with the entire table. For example, a relation schema for the relation called Students could be expressed using the following representation:

Students(*sid:* `string`, *name:* `string`, *login:* `string`,
age: `integer`, *gpa:* `real`)

There are five fields or columns, with names and types as shown above.

A *relational database schema* is a collection of relation schemas, describing one or more relations.

Domain is synonymous with *data type. Attributes* can be thought of as columns in a table. Therefore, an *attribute domain* refers to the data type associated with a column.

A *relation instance* is a set of tuples (also known as *rows* or *records*) that each conform to the schema of the relation.

The *relation cardinality* is the number of tuples in the relation.

The *relation degree* is the number of fields (or columns) in the relation.

Exercise 3.2 How many distinct tuples are in a relation instance with cardinality 22?

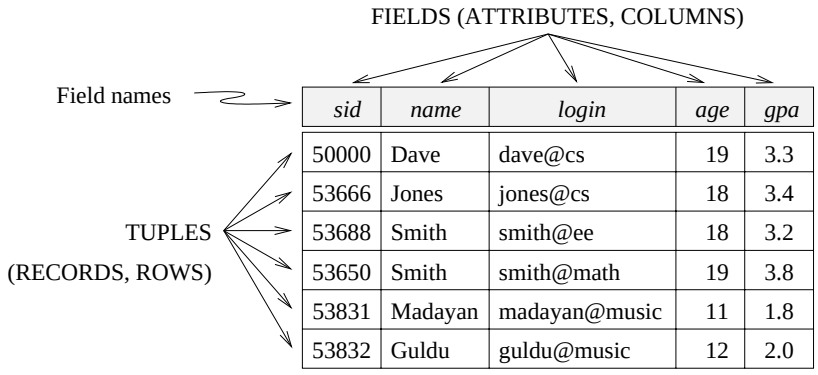
Answer 3.2 Answer omitted.

Exercise 3.3 Does the relational model, as seen by an SQL query writer, provide physical and logical data independence? Explain.

Answer 3.3 The user of SQL has no idea how the data is physically represented in the machine. He or she relies entirely on the relation abstraction for querying. Physical data independence is therefore assured. Since a user can define views, logical data independence can also be achieved by using view definitions to hide changes in the conceptual schema.

Exercise 3.4 What is the difference between a candidate key and the primary key for a given relation? What is a superkey?

Answer 3.4 Answer omitted.



sid	*name*	*login*	*age*	*gpa*
50000	Dave	dave@cs	19	3.3
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2
53650	Smith	smith@math	19	3.8
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure 3.1 An Instance $S1$ of the Students Relation

Exercise 3.5 Consider the instance of the Students relation shown in Figure 3.1.

1. Give an example of an attribute (or set of attributes) that you can deduce is *not* a candidate key, based on this instance being legal.
2. Is there any example of an attribute (or set of attributes) that you can deduce *is* a candidate key, based on this instance being legal?

Answer 3.5 Examples of non-candidate keys include the following: {name}, {age}. (Note that {gpa} can *not* be declared as a non-candidate key from this evidence alone even though common sense tells us that clearly more than one student could have the same grade point average.)

You cannot determine a key of a relation given only one instance of the relation. The fact that the instance is "legal" is immaterial. A candidate key, as defined here, *is a*

key, not something that only *might* be a key. The instance shown is just one possible "snapshot" of the relation. At other times, the same relation may have an instance (or snapshot) that contains a totally different set of tuples, and we cannot make predictions about those instances based only upon the instance that we are given.

Exercise 3.6 What is a foreign key constraint? Why are such constraints important? What is referential integrity?

Answer 3.6 Answer omitted.

Exercise 3.7 Consider the relations Students, Faculty, Courses, Rooms, Enrolled, Teaches, and Meets_In defined in Section 1.5.2.

1. List all the foreign key constraints among these relations.
2. Give an example of a (plausible) constraint involving one or more of these relations that is not a primary key or foreign key constraint.

Answer 3.7 There is no reason for a foreign key constraint (FKC) on the Students, Faculty, Courses, or Rooms relations. These are the most basic relations and must be free-standing. Special care must be given to entering data into these base relations.

In the Enrolled relation, *sid* and *cid* should both have FKCs placed on them. (Real students must be enrolled in real courses.) Also, since real teachers must teach real courses, both the fid and the *cid* fields in the Teaches relation should have FKCs. Finally, Meets_In should place FKCs on both the *cid* and *rno* fields.

It would probably be wise to enforce a few other constraints on this DBMS: the length of *sid*, *cid*, and fid could be standardized; checksums could be added to these identification numbers; limits could be placed on the size of the numbers entered into the credits, capacity, and salary fields; an enumerated type should be assigned to the grade field (preventing a student from receiving a grade of *G*, among other things); etc.

Exercise 3.8 Answer each of the following questions briefly. The questions are based on the following relational schema:

> Emp(*eid:* `integer`, *ename:* `string`, *age:* `integer`, *salary:* `real`)
> Works(*eid:* `integer`, *did:* `integer`, *pcttime:* `integer`)
> Dept(*did:* `integer`, *dname:* `string`, *budget:* `real`, *managerid:* `integer`)

1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?

2. Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints.
3. Define the Dept relation in SQL so that every department is guaranteed to have a manager.
4. Write an SQL statement to add John Doe as an employee with $eid = 101$, $age = 32$ and $salary = 15,000$.
5. Write an SQL statement to give every employee a 10 percent raise.
6. Write an SQL statement to delete the Toy department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.

Answer 3.8 Answer omitted.

sid	*name*	*login*	*age*	*gpa*
53831	Madayan	madayan@music	11	1.8
53832	Guldu	guldu@music	12	2.0

Figure 3.2 Students with $age < 18$ on Instance S

Exercise 3.9 Consider the SQL query whose answer is shown in Figure 3.2.

1. Modify this query so that only the *login* column is included in the answer.
2. If the clause `WHERE` *S.gpa >= 2* is added to the original query, what is the set of tuples in the answer?

Answer 3.9 The answers are as follows:

1. Only *login* is included in the answer:

```
SELECT S.login
FROM   Students S
WHERE  S.age < 18
```

2. The answer tuple for Madayan is omitted then.

Exercise 3.10 Explain why the addition of `NOT NULL` constraints to the SQL definition of the Manages relation (in Section 3.5.3) does not enforce the constraint that each department must have a manager. What, if anything, is achieved by requiring that the *ssn* field of Manages be non-*null*?

Answer 3.10 Answer omitted.

Exercise 3.11 Suppose that we have a ternary relationship R between entity sets A, B, and C such that A has a key constraint and total participation and B has a key constraint; these are the only constraints. A has attributes $a1$ and $a2$, with $a1$ being the key; B and C are similar. R has no descriptive attributes. Write SQL statements that create tables corresponding to this information so as to capture as many of the constraints as possible. If you cannot capture some constraint, explain why.

Answer 3.11 The following SQL statements create the corresponding relations.

```
CREATE TABLE A (  a1      CHAR(10),
                  a2      CHAR(10),
                  b1      CHAR(10),
                  c1      CHAR(10),
                  PRIMARY KEY (a1),
                  UNIQUE (b1),
                  FOREIGN KEY (b1) REFERENCES B,
                  FOREIGN KEY (c1) REFERENCES C )

CREATE TABLE B (  b1      CHAR(10),
                  b2      CHAR(10),
                  PRIMARY KEY (b1) )


CREATE TABLE C (  b1      CHAR(10),
                  c2      CHAR(10),
                  PRIMARY KEY (c1) )
```

The first SQL statement folds the relationship R into table A and thereby guarantees the participation constraint.

Exercise 3.12 Consider the scenario from Exercise 2.2, where you designed an ER diagram for a university database. Write SQL statements to create the corresponding relations and capture as many of the constraints as possible. If you cannot capture some constraints, explain why.

Answer 3.12 Answer omitted.

Exercise 3.13 Consider the university database from Exercise 2.3 and the ER diagram you designed. Write SQL statements to create the corresponding relations and capture as many of the constraints as possible. If you cannot capture some constraints, explain why.

Answer 3.13 The following SQL statements create the corresponding relations.

1. `CREATE TABLE` Professors (
   prof_ssn `CHAR(10)`,
   name `CHAR(64)`,
   age `INTEGER`,
   rank `INTEGER`,
   speciality `CHAR(64)`,
   `PRIMARY KEY` (prof_ssn))

2. `CREATE TABLE` Depts (
   dno `INTEGER`,
   dname `CHAR(64)`,
   office `CHAR(10)`,
   `PRIMARY KEY` (dno))

3. `CREATE TABLE` Runs (
   dno `INTEGER`,
   prof_ssn `CHAR(10)`,
   `PRIMARY KEY` (dno, prof_ssn),
   `FOREIGN KEY` (prof_ssn) `REFERENCES` Professors,
   `FOREIGN KEY` (dno) `REFERENCES` Depts)

4. `CREATE TABLE` Work_Dept (
   dno `INTEGER`,
   prof_ssn `CHAR(10)`,
   pc_time `INTEGER`,
   `PRIMARY KEY` (dno, prof_ssn),
   `FOREIGN KEY` (prof_ssn) `REFERENCES` Professors,
   `FOREIGN KEY` (dno) `REFERENCES` Depts)

   Observe that we would need check constraints or assertions in SQL to enforce the rule that Professors work in at least one department.

5. `CREATE TABLE` Project (
   pid `INTEGER`,
   sponsor `CHAR(32)`,
   start_date `DATE`,
   end_date `DATE`,
   budget `FLOAT`,
   `PRIMARY KEY` (pid))

6. `CREATE TABLE` Graduates (
   grad_ssn `CHAR(10)`,
   age `INTEGER`,
   name `CHAR(64)`,
   deg_prog `CHAR(32)`,

```
                                major     INTEGER,
                                PRIMARY KEY (grad_ssn),
                                FOREIGN KEY (major) REFERENCES Depts )
```

Note that the Major table is not necessary since each Graduate has only one major and so this can be an attribute in the Graduates table.

7.
```
CREATE TABLE Advisor (      senior_ssn CHAR(10),
                            grad_ssn   CHAR(10),
                            PRIMARY KEY (senior_ssn, grad_ssn),
                            FOREIGN KEY (senior_ssn)
                                       REFERENCES Graduates (grad_ssn),
                            FOREIGN KEY (grad_ssn) REFERENCES Graduates )
```

8.
```
CREATE TABLE Manages (      pid        INTEGER,
                            prof_ssn   CHAR(10),
                            PRIMARY KEY (pid, prof_ssn),
                            FOREIGN KEY (prof_ssn) REFERENCES Professors,
                            FOREIGN KEY (pid) REFERENCES Projects )
```

9.
```
CREATE TABLE Work_In (      pid        INTEGER,
                            prof_ssn   CHAR(10),
                            PRIMARY KEY (pid, prof_ssn),
                            FOREIGN KEY (prof_ssn) REFERENCES Professors,
                            FOREIGN KEY (pid) REFERENCES Projects )
```

Observe that we cannot enforce the participation constraint for Projects in the Work_In table without check constraints or assertions in SQL.

10.
```
CREATE TABLE Supervises (   prof_ssn   CHAR(10),
                            grad_ssn   CHAR(10),
                            pid        INTEGER,
                            PRIMARY KEY (prof_ssn, grad_ssn, pid),
                            FOREIGN KEY (prof_ssn) REFERENCES Professors,
                            FOREIGN KEY (grad_ssn) REFERENCES Graduates,
                            FOREIGN KEY (pid) REFERENCES Projects )
```

Note that we do not need an explicit table for the Work_Proj relation since every time a Graduate works on a Project, he or she must have a Supervisor.

Exercise 3.14 Consider the scenario from Exercise 2.4, where you designed an ER diagram for a company database. Write SQL statements to create the corresponding

relations and capture as many of the constraints as possible. If you cannot capture some constraints, explain why.

Answer 3.14 Answer omitted.

Exercise 3.15 Consider the Notown database from Exercise 2.5. You have decided to recommend that Notown use a relational database system to store company data. Show the SQL statements for creating relations corresponding to the entity sets and relationship sets in your design. Identify any constraints in the ER diagram that you are unable to capture in the SQL statements and briefly explain why you could not express them.

Answer 3.15 The following SQL statements create the corresponding relations.

1. `CREATE TABLE` Musicians (ssn `CHAR(10)`,
   name `CHAR(30)`,
   `PRIMARY KEY` (ssn))

2. `CREATE TABLE` Instruments (instrId `CHAR(10)`,
   dname `CHAR(30)`,
   key `CHAR(5)`,
   `PRIMARY KEY` (instrId))

3. `CREATE TABLE` Plays (ssn `CHAR(10)`,
   instrId `INTEGER`,
   `PRIMARY KEY` (ssn, instrId),
   `FOREIGN KEY` (ssn) `REFERENCES` Musicians,
   `FOREIGN KEY` (instrId) `REFERENCES` Instruments)

4. `CREATE TABLE` Songs_Appears (songId `INTEGER`,
   author `CHAR(30)`,
   title `CHAR(30)`,
   albumIdentifier `INTEGER NOT NULL`,
   `PRIMARY KEY` (songId),
   `FOREIGN KEY` (albumIdentifier)
   References Album_Producer,

5. `CREATE TABLE` Telephone_Home (phone `CHAR(11)`,
   address `CHAR(30)`,
   `PRIMARY KEY` (phone),
   `FOREIGN KEY` (address) `REFERENCES` Place,

6. `CREATE TABLE` Lives (
   ssn `CHAR(10)`,
   phone `CHAR(11)`,
   address `CHAR(30)`,
   `PRIMARY KEY` (ssn, address),
   `FOREIGN KEY` (phone, address)
   References Telephone_Home,
   `FOREIGN KEY` (ssn) `REFERENCES` Musicians)

7. `CREATE TABLE` Place (
   address `CHAR(30)`)

8. `CREATE TABLE` Perform (
   songId `INTEGER`,
   ssn `CHAR(10)`,
   `PRIMARY KEY` (ssn, songId),
   `FOREIGN KEY` (songId) `REFERENCES` Songs,
   `FOREIGN KEY` (ssn) `REFERENCES` Musicians)

9. `CREATE TABLE` Album_Producer (
   albumIdentifier `INTEGER`,
   ssn `CHAR(10)`,
   copyrightDate `DATE`,
   speed `INTEGER`,
   title `CHAR(30)`,
   `PRIMARY KEY` (albumIdentifier),
   `FOREIGN KEY` (ssn) `REFERENCES` Musicians)

Exercise 3.16 Translate your ER diagram from Exercise 2.6 into a relational schema, and show the SQL statements needed to create the relations, using only key and null constraints. If your translation cannot capture any constraints in the ER diagram, explain why.

In Exercise 2.6, you also modified the ER diagram to include the constraint that tests on a plane must be conducted by a technician who is an expert on that model. Can you modify the SQL statements defining the relations obtained by mapping the ER diagram to check this constraint?

Answer 3.16 Answer omitted.

Exercise 3.17 Consider the ER diagram that you designed for the Prescriptions-R-X chain of pharmacies in Exercise 2.7. Define relations corresponding to the entity sets and relationship sets in your design using SQL.

Answer 3.17 The statements to create tables corresponding to entity sets Doctor, Pharmacy, and Pharm_co are straightforward and omitted. The other required tables can be created as follows:

1. `CREATE TABLE` Pri_Phy_Patient (ssn `CHAR(11)`,
   name `CHAR(20)`,
   age `INTEGER`,
   address `CHAR(20)`,
   phy_ssn `CHAR(11)`,
   `PRIMARY KEY` (ssn),
   `FOREIGN KEY` (phy_ssn) `REFERENCES` Doctor)

2. `CREATE TABLE` Prescription (ssn `CHAR(11)`,
   phy_ssn `CHAR(11)`,
   date `CHAR(11)`,
   quantity `INTEGER`,
   trade_name `CHAR(20)`,
   pharm_id `CHAR(11)`,
   `PRIMARY KEY` (ssn, phy_ssn),
   `FOREIGN KEY` (ssn) `REFERENCES` Patient,
   `FOREIGN KEY` (phy_ssn) `REFERENCES` Doctor,
   `FOREIGN KEY` (trade_name, pharm_id)
   References Make_Drug)

3. `CREATE TABLE` Make_Drug (trade_name `CHAR(20)`,
   pharm_id `CHAR(11)`,
   `PRIMARY KEY` (trade_name, pharm_id),
   `FOREIGN KEY` (trade_name) `REFERENCES` Drug,
   `FOREIGN KEY` (pharm_id) `REFERENCES` Pharm_co)

4. `CREATE TABLE` Sell (price `INTEGER`,
   name `CHAR(10)`,
   trade_name `CHAR(10)`,
   `PRIMARY KEY` (name, trade_name),
   `FOREIGN KEY` (name) `REFERENCES` Pharmacy,
   `FOREIGN KEY` (trade_name) `REFERENCES` Drug)

5. `CREATE TABLE` Contract (name `CHAR(20)`,
   pharm_id `CHAR(11)`,
   start_date `CHAR(11)`,
   end_date `CHAR(11)`,