Chapter 2: Database Design Fundamentals

Solutions

Answers to Review Questions

1. An entity is a person, place, thing, or event.
2. An attribute is a property of an entity.
3. A relationship is an association between tables (entities). A one-to-many relationship between two tables is a relationship in which each row in the first table can be associated with many rows in the second table, but each row in the second table is associated with only one row in the first table.
4. A repeating group is multiple entries in a single location in a table.
5. A relation is a two-dimensional table in which the entries in the table are single-valued (each location in the table contains a single entry), each column has a distinct name (or attribute name), all values in a column are values of the same attribute, the order of the rows and columns is immaterial, and each row contains unique values.
6. A relational database is a collection of relations.
7. For each table, you write the name of the table and then within parentheses list all of the columns in the table. Underline the primary keys.

```
BRANCH (BRANCH_NUM, BRANCH_NAME, BRANCH_LOCATION, NUM_EMPLOYEES)
PUBLISHER (PUBLISHER_CODE, PUBLISHER_NAME, CITY)
AUTHOR (AUTHOR_NUM, AUTHOR_LAST, AUTHOR_FIRST)
BOOK (BOOK_CODE, TITLE, PUBLISHER_CODE, TYPE, PRICE, PAPERBACK)
WROTE (BOOK_CODE, AUTHOR_NUM, SEQUENCE)
INVENTORY (BOOK_CODE, BRANCH_NUM, ON_HAND)
```

8. To qualify the name of a field, indicate the table in which the field appears. You do this by preceding the name of the field with the name of the table and a period.
9. A column (attribute), B, is functionally dependent on another column, A (or possibly a collection of columns), if at any point in time a value for A determines a single value for B.
10. Column A (or a collection of columns) is the primary key for a table if (1) *All* columns in the table are functionally dependent on A and (2) No subcollection of the columns in A (assuming A is a collection of columns and not just a single column) also has property 1. The primary key of the BRANCH table is the BRANCH_NUM column. The primary key of the PUBLISHER table is the PUBLISHER_CODE column. The primary key of the AUTHOR table is the AUTHOR_NUM column. The primary key of the BOOK table is the BOOK_CODE column. The primary key of the WROTE table is the combination of the BOOK_CODE and AUTHOR_NUM

columns. The primary key of the INVENTORY table is the combination of the BOOK_CODE and BRANCH_NUM columns.

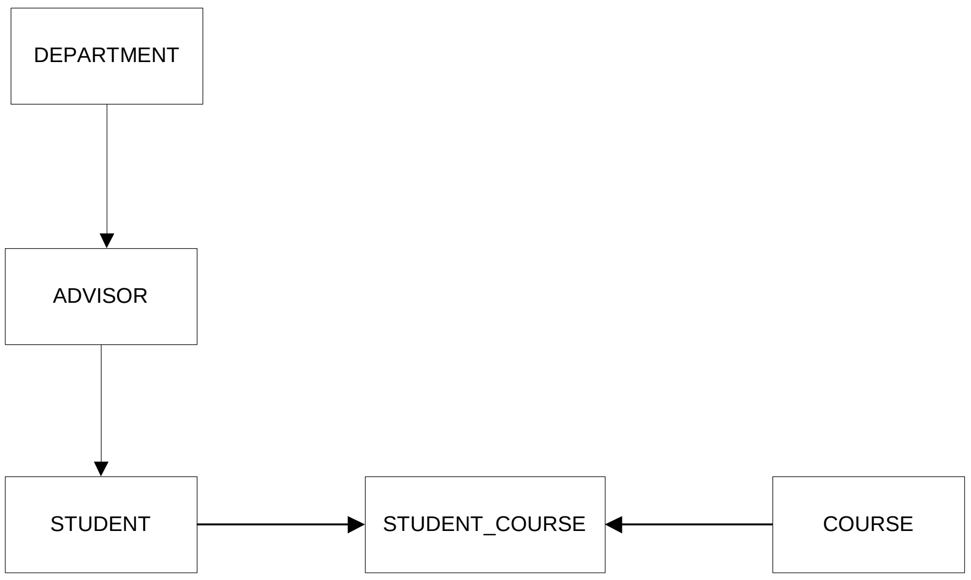
11. Functional dependencies:

```
DEPARTMENT_NUM → DEPARTMENT_NAME
ADVISOR_NUM → ADVISOR_LAST_NAME, ADVISOR_FIRST_NAME, DEPARTMENT_NUM
COURSE_CODE → DESCRIPTION
STUDENT_NUM → STUDENT_LAST_NAME, STUDENT_FIRST_NAME, ADVISOR_NUM
STUDENT_NUM, COURSE_CODE → GRADE
```

Relations:

```
DEPARTMENT (DEPARTMENT_NUM, DEPARTMENT_NAME)
ADVISOR (ADVISOR_NUM, ADVISOR_LAST_NAME, ADVISOR_FIRST_NAME,
      DEPARTMENT_NUM)
COURSE (COURSE_CODE, DESCRIPTION)
STUDENT (STUDENT_NUM, STUDENT_LAST_NAME, STUDENT_FIRST_NAME,
      ADVISOR_NUM
STUDENT_COURSE (STUDENT_NUM, COURSE_CODE, GRADE)
```

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)



12. A table (relation) is in first normal form (1NF) if it does not contain repeating groups.

13. A table (relation) is in second normal form if it is in first normal form and no nonkey column is dependent on only a portion of the primary key. If a table is not in second normal form, the table contains redundancy, which leads to a variety of update anomalies. A change in a value can require not just one change, but several. There is the possibility of inconsistent data. Adding additional data to the database may not be possible without creating artificial values for part of the key. Finally, deletions of certain items can result in inadvertently deleting crucial information from the database.

14. A table is in third normal form if it is in second normal form and if the only determinants it contains are candidate keys. A change in a value can require not just one change, but several. There is the possibility of inconsistent data. Adding certain additional data to the database may not be possible without creating artificial rows in the table. Finally, deletions of certain items can result in inadvertently deleting crucial information from the database.

15.

```
STUDENT (STUDENT_NUM, STUDENT_LAST_NAME, STUDENT_FIRST_NAME,
      ADVISOR_NUM)
ADVISOR (ADVISOR_NUM, ADVISOR_LAST_NAME, ADVISOR_FIRST_NAME)
COURSE (COURSE_CODE, DESCRIPTION)
STUDENT_COURSE  (STUDENT_NUM, COURSE_CODE, GRADE)
```

Answers to Premiere Products Exercises

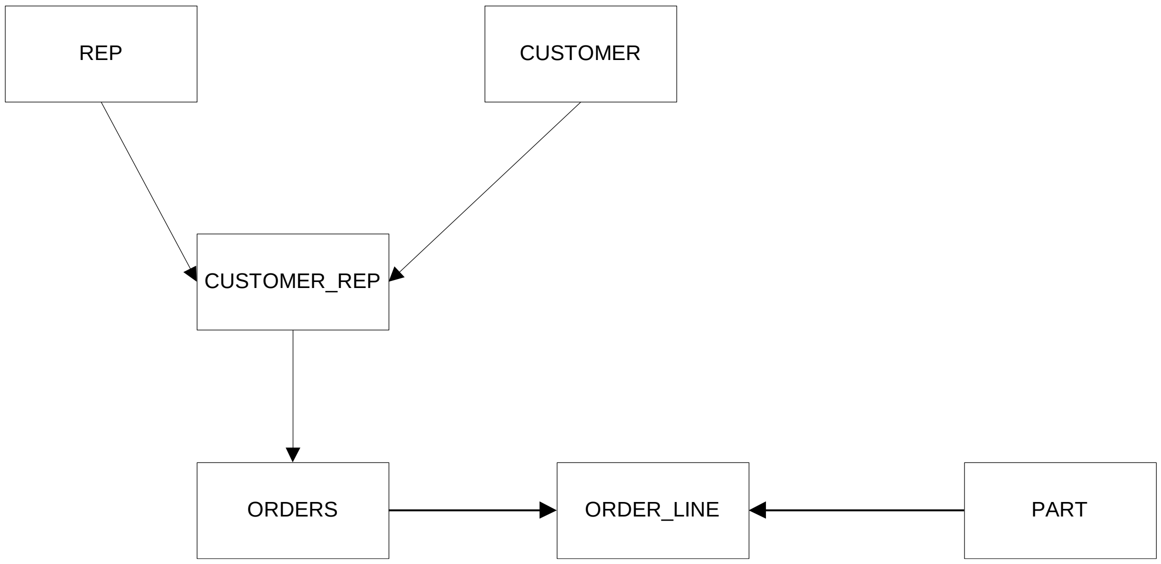
1. **NOTES**: The CUSTOMER_REP table in the following lists implements the relationship between customers and reps. If customer 148, for example, is represented by both rep 20 and rep 35, there would be a row in the table in which the customer number is 148 and the rep number is 20 as well as a row in which the customer number is 148 and the rep number is 35. A row would only be allowed in the order table if the combination of the customer number and the rep number match a row in the CUSTOMER_REP table.

```
REP (REP_NUM, LAST_NAME, FIRST_NAME, STREET,
      CITY, STATE, ZIP, COMMISSION, RATE)
CUSTOMER (CUSTOMER_NUM, CUSTOMER_NAME, STREET,
      CITY, STATE, ZIP, BALANCE, CREDIT_LIMIT)
CUSTOMER_REP (CUSTOMER_NUM, REP_NUM)
ORDERS (ORDER_NUM, ORDER_DATE, CUSTOMER_NUM, REP_NUM)
ORDER_LINE (ORDER_NUM, PART_NUM, NUM_ORDERED,
      QUOTED_PRICE)
PART (PART_NUM, DESCRIPTION, ON_HAND, CLASS,
      WAREHOUSE, PRICE)
```

Indicate the changes that need to be made to the design of the Premiere Products database to support the following: A customer is not necessarily represented by a single sales rep, but can be represented by several sales reps. When a customer places an order, the sales rep who gets the commission on the order must be one of the collection of sales reps who represent the customer.

Relationships: There are one-to-many relationships from REP to CUSTOMER_REP, CUSTOMER to CUSTOMER_REP, CUSTOMER_REP to ORDERS, ORDERS to ORDER_LINE, and PART to ORDER_LINE.

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)

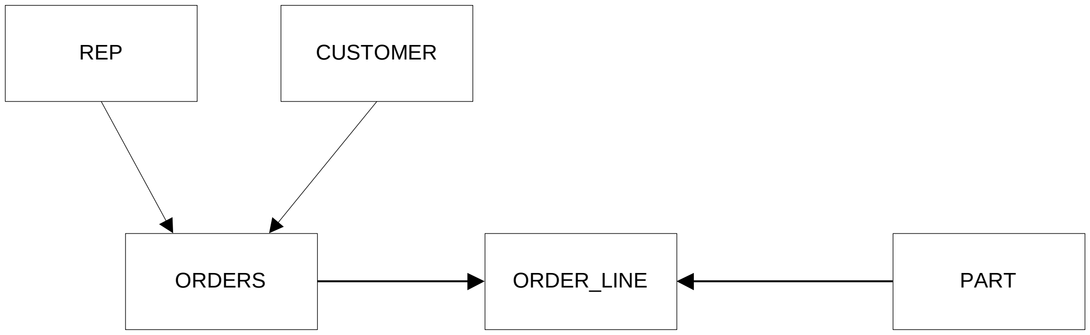


2. **NOTES**: There is no relationship between customers and reps, so there is no REP_NUM column in the CUSTOMER table nor is there an additional table like the CUSTOMER_REP table in Exercise 1. A row can only exist in the ORDERS table if the customer number matches a row in the CUSTOMER table and the rep number matches a row in the REP table.

```
REP (REP_NUM, LAST_NAME, FIRST_NAME, STREET, CITY, STATE, ZIP,
      COMMISSION, RATE)
CUSTOMER (CUSTOMER_NUM, CUSTOMER_NAME, STREET, CITY, STATE,
      ZIP, BALANCE, CREDIT_LIMIT)
ORDERS (ORDER_NUM, ORDER_DATE, CUSTOMER_NUM, REP_NUM)
ORDER_LINE (ORDER_NUM, PART_NUM, NUM_ORDERED, QUOTED_PRICE)
PART (PART_NUM, DESCRIPTION, ON_HAND, CLASS, WAREHOUSE, PRICE)
```

Relationships: There are one-to-many relationships from REP to ORDERS, CUSTOMER to ORDERS, ORDERS to ORDER_LINE, and PART to ORDER_LINE.

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)



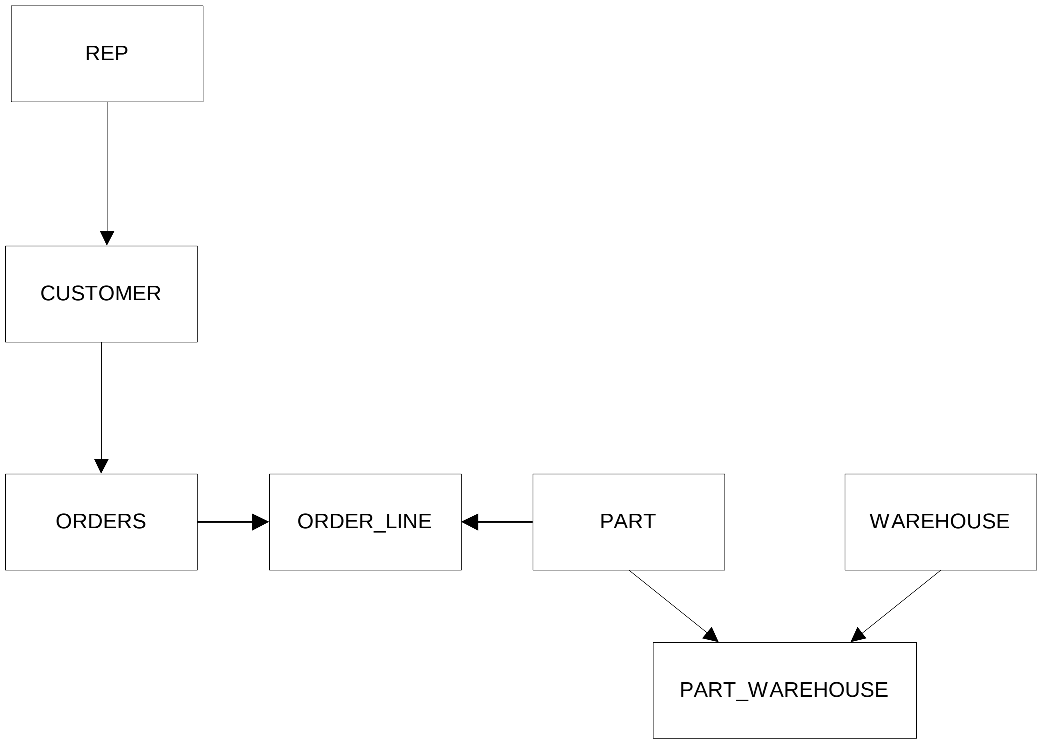
3. **NOTES**: The WAREHOUSE_NUM and ON_HAND columns do not appear in the PART table. There is a WAREHOUSE table, whose key is WAREHOUSE_NUM and which contains the warehouse description. Information about units on hand is stored in a

new table, the PART_WAREHOUSE table, whose key is the combination of the part number and warehouse number. If there are 10 units of part AT94 on hand in warehouse 2, for example, there would be a row in PART_WAREHOUSE on which the part number is AT94, the warehouse number is 2, and the number of units on hand is 10.

```
REP (REP_NUM, LAST_NAME, FIRST_NAME, STREET, CITY, STATE, ZIP,
      COMMISSION, RATE)
CUSTOMER (CUSTOMER_NUM, CUSTOMER_NAME, STREET, CITY, STATE,
      ZIP, BALANCE, CREDIT_LIMIT, REP_NUM)
ORDERS (ORDER_NUM, ORDER_DATE, CUSTOMER_NUM)
ORDER_LINE (ORDER_NUM, PART_NUM, NUM_ORDERED, QUOTED_PRICE)
PART (PART_NUM, DESCRIPTION, CLASS, PRICE)
WAREHOUSE (WAREHOUSE_NUM, WAREHOUSE_DESCRIPTION)
PART_WAREHOUSE (PART_NUM, WAREHOUSE_NUM, ON_HAND)
```

Relationships: There are one-to-many relationships from REP to CUSTOMER, CUSTOMER to ORDERS, ORDERS to ORDER_LINE, PART to ORDER_LINE, PART to PART_WAREHOUSE, and WAREHOUSE to PART_WAREHOUSE.

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)



4. Functional Dependencies:

```
PART_NUM → DESCRIPTION, ON_HAND, CLASS, WAREHOUSE, PRICE
ORDER_NUM → ORDER_DATE, CUSTOMER_NUM
CUSTOMER_NUM → CUSTOMER_NAME
PART_NUM, ORDER_NUM → NUM_ORDERED, QUOTED_PRICE
```

Relations:

```
PART (PART_NUM, DESCRIPTION, ON_HAND, CLASS, WAREHOUSE, PRICE)
ORDERS (ORDER_NUM, ORDER_DATE, CUSTOMER_NUM)
CUSTOMER (CUSTOMER_NUM, CUSTOMER_NAME)
ORDER_LINE (PART_NUM, ORDER_NUM, NUM_ORDERED, QUOTED_PRICE)
```

NOTE: The keys for ORDER_LINE could also have been listed as ORDER_NUM, PART_NUM.

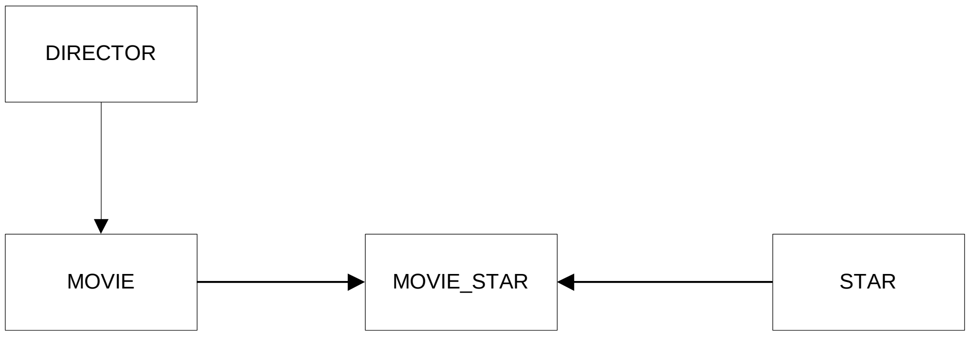
Answers to Henry Books Exercises

1. Tables (Relations):

```
DIRECTOR (DIRECTOR_NUM, DIRECTOR_NAME, YEAR_BORN, YEAR_DIED)
MOVIE (MOVIE_NUM, MOVIE_TITLE, YEAR_MADE, TYPE, DIRECTOR_NUM,
      CRITICS_RATING, MPAA_RATING, NUM_AWARDS_NOM, NUM_AWARDS_WON)
STAR (STAR_NUM, STAR_NAME, BIRTHPLACE, YEAR_BORN, YEAR_DIED)
MOVIE_STAR (MOVIE_NUM, STAR_NUM)
```

Relationships: There are one-to-many relationships from DIRECTOR to MOVIE, MOVIE to MOVIE_STAR, and STAR to MOVIE_STAR.

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)



2. Functional Dependencies:

```
BOOK_CODE → TITLE, TYPE, PRICE
AUTHOR_NUM → AUTHOR_LAST, AUTHOR_FIRST
```

Tables (Relations):

```
BOOK (BOOK_CODE, TITLE, TYPE, PRICE)
AUTHOR (AUTHOR_NUM, AUTHOR_LAST, AUTHOR_FIRST) )
BOOK_AUTHOR (BOOK_CODE, AUTHOR_NUM)
```

NOTE: The BOOK_AUTHOR relation is necessary to relate books and authors. (You could have assigned it any name you like.) The key could have been listed as AUTHOR_NUM, BOOK_CODE rather than BOOK_CODE, AUTHOR_NUM.

3. Functional Dependencies:

```
BOOK_CODE → TITLE, TYPE, PRICE, PUB_CODE
PUB_CODE → PUBLISHER_NAME, CITY
```

Tables (Relations):

```
BOOK (BOOK_CODE, TITLE, TYPE, PRICE, PUB_CODE)
PUBLISHER (PUB_CODE, PUBLISHER_NAME, CITY)
```

Answers to Alexamara Marina Group Exercises

1. **NOTES**: There are two ways to handle the MARINA_SLIP table, the table that contains information about the slips within each marina. You can add a column to identify each slip, for example SLIP_ID (see Figure 1.10 for an illustration of the use of this column). Alternatively, you could do without this column by making the primary key the combination of the marina number and the slip number. (If the SLIP_ID column were missing from Figure 1.10, the primary key would be the combination of the MARINA_NUM and SLIP_NUM columns.) Both approaches are legitimate and are illustrated below.

Tables (Relations):

Option 1 (With SlipID added as a unique identifier for MARINA_SLIP):

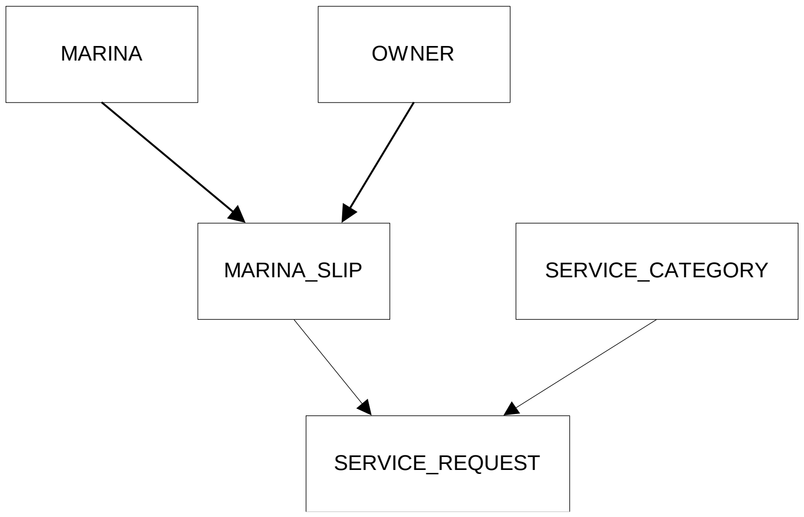
```
MARINA (MARINA_NUM, NAME, ADDRESS, CITY, STATE, ZIP)
OWNER (OWNER_NUM, LAST_NAME, FIRST_NAME, ADDRESS, CITY, STATE, ZIP)
MARINA_SLIP (SLIP_ID, MARINA_NUM, SLIP_NUM, LENGTH, RENTAL_FEE,
      BOAT_NAME, BOAT_TYPE, OWNER_NUM)
SERVICE_CATEGORY (CATEGORY_NUM, CATEGORY_DESCRIPTION)
SERVICE_REQUEST (SERVICE_ID, SLIP_ID, CATEGORY_NUM, DESCRIPTION,
      STATUS, EST_HOURS, SPENT_HOURS, NEXT_SERVICE_DATE)
```

Option 2 (Without SlipID):

```
MARINA (MARINA_NUM, NAME, ADDRESS, CITY, STATE, ZIP)
OWNER (OWNER_NUM, LAST_NAME, FIRST_NAME, ADDRESS, CITY, STATE, ZIP)
MARINA_SLIP (MARINA_NUM, SLIP_NUM, LENGTH, RENTAL_FEE, BOAT_NAME,
      BOAT_TYPE, OWNER_NUM)
SERVICE_CATEGORY (CATEGORY_NUM, CATEGORY_DESCRIPTION)
SERVICE_REQUEST (SERVICE_ID, MARINA_NUM, SLIP_NUM, CATEGORY_NUM,
      DESCRIPTION, STATUS, EST_HOURS, SPENT_HOURS,
      NEXT_SERVICE_DATE)
```

Relationships: There are one-to-many relationships from MARINA to MARINA_SLIP, OWNER to MARINA_SLIP, SERVICE_CATEGORY to SERVICE_REQUEST, and MARINA_SLIP to SERVICE_REQUEST.

Entity-Relationship diagram: (**NOTE**: Your rectangles may be in different positions as long as they are connected by the same arrows.)



2. Functional Dependencies:

```
MARINA_NUM → NAME
MARINA_NUM, SLIP_NUM → LENGTH, RENTAL_FEE, BOAT_NAME
```

Tables (Relations):

MARINA (<u>MARINA_NUM</u>, NAME)
MARINA_SLIP (<u>MARINA_NUM</u>, <u>SLIP_NUM</u>, LENGTH, RENTAL_FEE, BOAT_NAME)

3. Functional Dependencies:

```
ID → MARINA_NUM, SLIP_NUM, LENGTH, RENTAL_FEE, BOAT_NAME,
      BOAT_TYPE, OWNER_NUM, LAST_NAME, FIRST_NAME
OWNER_NUM → LAST_NAME, FIRST_NAME
```

Tables (Relations):

MARINA_SLIP (<u>ID</u>, MARINA_NUM, SLIP_NUM, LENGTH, RENTAL_FEE,
      BOAT_NAME, BOAT_TYPE, OWNER_NUM)
OWNER (<u>OWNER_NUM</u>, LAST_NAME, FIRST_NAME)