What Is "Functional" About Functional Dependencies?

$A_1A_2\cdots A_n \to B$ is called a "functional" dependency because in principle there is a function that takes a list of values, one for each of attributes $A_1, A_2, \ldots, A_n$ and produces a unique value (or no value at all) for B. For instance, in the `Movies1` relation, we can imagine a function that takes a string like `"Star Wars"` and an integer like 1977 and produces the unique value of `length`, namely 124, that appears in the relation `Movies1`. However, this function is not the usual sort of function that we meet in mathematics, because there is no way to compute it from first principles. That is, we cannot perform some operations on strings like `"Star Wars"` and integers like 1977 and come up with the correct length. Rather, the function is only computed by lookup in the relation. We look for a tuple with the given `title` and `year` values and see what value that tuple has for `length`.

3.1.3 Superkeys

A set of attributes that contains a key is called a *superkey*, short for "superset of a key." Thus, every key is a superkey. However, some superkeys are not (minimal) keys. Note that every superkey satisfies the first condition of a key: it functionally determines all other attributes of the relation. However, a superkey need not satisfy the second condition: minimality.

Example 3.3: In the relation of Example 3.2, there are many superkeys. Not only is the key

{`title`, `year`, `starName`}

a superkey, but any superset of this set of attributes, such as

{`title`, `year`, `starName`, `length`, `studioName`}

is a superkey. □

3.1.4 Exercises for Section 3.1

Exercise 3.1.1: Consider a relation about people in the United States, including their name, Social Security number, street address, city, state, ZIP code, area code, and phone number (7 digits). What FD's would you expect to hold? What are the keys for the relation? To answer this question, you need to know something about the way these numbers are assigned. For instance, can an area

Other Key Terminology

In some books and articles one finds different terminology regarding keys. One can find the term "key" used the way we have used the term "superkey," that is, a set of attributes that functionally determine all the attributes, with no requirement of minimality. These sources typically use the term "candidate key" for a key that is minimal — that is, a "key" in the sense we use the term.

code straddle two states? Can a ZIP code straddle two area codes? Can two people have the same Social Security number? Can they have the same address or phone number?

!! **Exercise 3.1.2:** Suppose R is a relation with attributes $A_1, A_2, \ldots, A_n$. As a function of n, tell how many superkeys R has, if:

a) The only key is A_1.

b) The only keys are A_1 and A_2.

c) The only keys are $\{A_1, A_2\}$ and $\{A_1, A_3\}$.

d) The only keys are $\{A_1, A_2\}$ and $\{A_3, A_4\}$.

Exercise 3.1.3: Consider a relation representing the present position of molecules in a closed container. The attributes are an ID for the molecule, the u, v, and w coordinates of the molecule, and its velocity in the u, v, and w dimensions. What FD's would you expect to hold? What are the keys?

3.2 Rules About Functional Dependencies

In this section, we shall learn how to *reason* about FD's. That is, suppose we are told of a set of FD's that a relation satisfies. Often, we can deduce that the relation must satisfy certain other FD's. This ability to discover additional FD's is essential when we discuss the design of good relation schemas in Section 3.3.

3.2.1 Reasoning About Functional Dependencies

Let us begin with a motivating example that will show us how we can infer a functional dependency from other given FD's.

Example 3.4: If we are told that a relation $R(A, B, C)$ satisfies the FD's $A \to B$ and $B \to C$, then we can deduce that R also satisfies the FD $A \to C$. How does that reasoning go? To prove that $A \to C$, we must consider two tuples of R that agree on A and prove they also agree on C.

- If we already know that the closure of some set X is all attributes, then we cannot discover any new FD's by closing supersets of X.

Thus, we may start with the closures of the singleton sets, and then move on to the doubleton sets if necessary. For each closure of a set X, we add the FD $X \to E$ for each attribute E that is in X^+ and in the schema of R_1, but not in X.

First, $\{A\}^+ = \{A, B, C, D\}$. Thus, $A \to C$ and $A \to D$ hold in R_1. Note that $A \to B$ is true in R, but makes no sense in R_1 because B is not an attribute of R_1.

Next, we consider $\{C\}^+ = \{C, D\}$, from which we get the additional FD $C \to D$ for R_1. Since $\{D\}^+ = \{D\}$, we can add no more FD's, and are done with the singletons.

Since $\{A\}^+$ includes all attributes of R_1, there is no point in considering any superset of $\{A\}$. The reason is that whatever FD we could discover, for instance $AC \to D$, follows from an FD with only A on the left side: $A \to D$ in this case. Thus, the only doubleton whose closure we need to take is $\{C, D\}^+ = \{C, D\}$. This observation allows us to add nothing. We are done with the closures, and the FD's we have discovered are $A \to C$, $A \to D$, and $C \to D$.

If we wish, we can observe that $A \to D$ follows from the other two by transitivity. Therefore a simpler, equivalent set of FD's for R_1 is $A \to C$ and $C \to D$. This set is, in fact, a minimal basis for the FD's of R_1. □

3.2.9 Exercises for Section 3.2

Exercise 3.2.1: Consider a relation with schema $R(A, B, C, D)$ and FD's $BC \to D$, $D \to A$, and $A \to B$.

a) What are all the nontrivial FD's that follow from the given FD's? You should restrict yourself to FD's with single attributes on the right side.

b) What are all the keys of R?

c) What are all the superkeys for R that are not keys?

Exercise 3.2.2: Repeat Exercise 3.2.1 for the following schemas and sets of FD's:

i) $S(A, B, C, D)$ with FD's $A \to B$, $A \to C$, and $C \to D$.

ii) $T(A, B, C, D)$ with FD's $A \to B$, $B \to C$, $C \to D$, and $D \to A$.

iii) $U(A, B, C, D)$ with FD's $AD \to B$, $AB \to C$, $BC \to D$, and $CD \to A$.

Exercise 3.2.3: Show that the following rules hold, by using the closure test of Section 3.2.4.

a) *Augmenting left sides.* If $A_1 A_2 \cdots A_n \to B$ is an FD, and C is another attribute, then $A_1 A_2 \cdots A_n C \to B$ follows.

b) *Full augmentation.* If $A_1A_2\cdots A_n \to B$ is an FD, and C is another attribute, then $A_1A_2\cdots A_nC \to BC$ follows. Note: from this rule, the "augmentation" rule mentioned in the box of Section 3.2.7 on "A Complete Set of Inference Rules" can easily be proved.

c) *Pseudotransitivity.* Suppose FD's $A_1A_2\cdots A_n \to B_1B_2\cdots B_m$ and

$$C_1C_2\cdots C_k \to D$$

hold, and the B's are each among the C's. Then

$$A_1A_2\cdots A_nE_1E_2\cdots E_j \to D$$

holds, where the E's are all those of the C's that are not found among the B's.

d) *Addition.* If FD's $A_1A_2\cdots A_n \to B_1B_2\cdots B_m$ and

$$C_1C_2\cdots C_k \to D_1D_2\cdots D_j$$

hold, then FD $A_1A_2\cdots A_nC_1C_2\cdots C_k \to B_1B_2\cdots B_mD_1D_2\cdots D_j$ also holds. In the above, we should remove one copy of any attribute that appears among both the A's and C's or among both the B's and D's.

! **Exercise 3.2.4:** Let X and Y be sets of attributes. Show that if $X \subseteq Y$, then $X^+ \subseteq Y^+$, where the closures are taken with respect to the same set of FD's.

! **Exercise 3.2.5:** Prove that $(X^+)^+ = X^+$.

! **Exercise 3.2.6:** Show that each of the following are *not* valid rules about FD's by giving example relations that satisfy the given FD's (following the "if") but not the FD that allegedly follows (after the "then").

a) If $AB \to C$, then $A \to C$ or $B \to C$.

b) If $A \to B$ then $B \to A$.

c) If $AB \to C$ and $A \to C$, then $B \to C$.

! **Exercise 3.2.7:** Show that if a relation has no attribute that is functionally determined by all the other attributes, then the relation has no nontrivial FD's at all.

!! **Exercise 3.2.8:** We say a set of attributes X is *closed* (with respect to a given set of FD's) if $X^+ = X$. Consider a relation with schema $R(A, B, C, D)$ and an unknown set of FD's. If we are told which sets of attributes are closed, we can discover the FD's. What are the FD's if:

a) All sets of the four attributes are closed.

b) The only closed sets are $\emptyset$ and $\{A, B, C, D\}$.

c) The closed sets are $\emptyset$, {A,B}, and $\{A, B, C, D\}$.

! **Exercise 3.2.9:** Suppose we have relation $R(A, B, C, D, E)$, with some set of FD's, and we wish to project those FD's onto relation $S(A, B, C)$. Give the FD's that hold in S if the FD's for R are:

a) $A \to C$, $C \to B$, $B \to D$, $D \to E$, and $E \to A$.

b) $BC \to DE$, $A \to E$, $D \to A$, and $E \to B$.

c) $C \to D$, $AD \to E$, $BC \to E$, and $DE \to A$.

d) $AB \to E$, $AC \to D$, $BC \to E$, $E \to A$, and $D \to B$.

In each case, it is sufficient to give a minimal basis for the full set of FD's of S.

!! **Exercise 3.2.10:** Show that if an FD F follows from some given FD's, then we can prove F from the given FD's using Armstrong's axioms (defined in the box "A Complete Set of Inference Rules" in Section 3.2.7). *Hint*: Examine Algorithm 3.7 and show how each step of that algorithm can be mimicked by inferring some FD's by Armstrong's axioms.

! **Exercise 3.2.11:** Find all the minimal bases for the FD's and relation of Example 3.11.

3.3 Design of Relational Database Schemas

Careless selection of a relational database schema can lead to redundancy and related anomalies. For instance, consider the relation in Fig. 3.2, which we reproduce here as Fig. 3.6. Notice that the length and genre for *Star Wars* and *Wayne's World* are each repeated, once for each star of the movie. The repetition of this information is redundant. It also introduces the potential for several kinds of errors, as we shall see.

In this section, we shall tackle the problem of design of good relation schemas in the following stages:

1. We first explore in more detail the problems that arise when our schema is poorly designed.
2. Then, we introduce the idea of "decomposition," breaking a relation schema (set of attributes) into two smaller schemas.
3. Next, we introduce "Boyce-Codd normal form," or "BCNF," a condition on a relation schema that eliminates these problems.
4. These points are tied together when we explain how to assure the BCNF condition by decomposing relation schemas.

{title, year, studioName}
{studioName, president}
{president, presAddr}

□

In general, we must keep applying the decomposition rule as many times as needed, until all our relations are in BCNF. We can be sure of ultimate success, because every time we apply the decomposition rule to a relation R, the two resulting schemas each have fewer attributes than that of R. As we saw in Example 3.17, when we get down to two attributes, the relation is sure to be in BCNF; often relations with larger sets of attributes are also in BCNF. The strategy is summarized below.

Algorithm 3.20: BCNF Decomposition Algorithm.

INPUT: A relation R_0 with a set of functional dependencies S_0.

OUTPUT: A decomposition of R_0 into a collection of relations, all of which are in BCNF.

METHOD: The following steps can be applied recursively to any relation R and set of FD's S. Initially, apply them with $R = R_0$ and $S = S_0$.

1. Check whether R is in BCNF. If so, nothing more needs to be done. Return $\{R\}$ as the answer.
2. If there are BCNF violations, let one be $X \to Y$. Use Algorithm 3.7 to compute X^+. Choose $R_1 = X^+$ as one relation schema and let R_2 have attributes X and those attributes of R that are not in X^+.
3. Use Algorithm 3.12 to compute the sets of FD's for R_1 and R_2; let these be S_1 and S_2, respectively.
4. Recursively decompose R_1 and R_2 using this algorithm. Return the union of the results of these decompositions.

□

3.3.5 Exercises for Section 3.3

Exercise 3.3.1: For each of the following relation schemas and sets of FD's:

a) $R(A, B, C, D)$ with FD's $B \to A$, $C \to B$, $D \to C$, and $A \to D$.

b) $R(A, B, C, D)$ with FD's $BC \to D$, $D \to A$, and $A \to B$.

c) $R(A, B, C, D)$ with FD's $A \to B$ and $A \to C$.

d) $R(A, B, C, D)$ with FD's $AB \to D$, $BD \to C$, $CD \to A$, and $AC \to B$.

e) $R(A, B, C, D, E)$ with FD's $AB \to C$, $C \to E$, $E \to A$, and $E \to D$.

f) $R(A, B, C, D, E)$ with FD's $AB \to C$, $DE \to C$, and $B \to E$.

do the following:

i) Indicate all the BCNF violations. Do not forget to consider FD's that are not in the given set, but follow from them. However, it is not necessary to give violations that have more than one attribute on the right side.

ii) Decompose the relations, as necessary, into collections of relations that are in BCNF.

! **Exercise 3.3.2:** Suppose we have a relation schema $R(A, B, C)$ with FD $B \to C$. Suppose also that we decide to decompose this schema into $S(A, C)$ and $T(B, C)$. Give an example of an instance of relation R whose projection onto S and T and subsequent rejoining as in Section 3.4.1 does not yield the same relation instance. That is, $\pi_{B,C}(R) \bowtie \pi_{A,C}(R) \neq R$.

Exercise 3.3.3: We mentioned in Section 3.3.4 that we would exercise our option to expand the right side of an FD that is a BCNF violation if possible. Consider a relation R whose schema is the set of attributes $\{A, B, C, D\}$ with FD's $A \to B$ and $A \to C$. Either is a BCNF violation, because the only key for R is $\{A, D\}$. Suppose we begin by decomposing R according to $A \to B$. Do we ultimately get the same result as if we first expand the BCNF violation to $A \to BC$? Why or why not?

! **Exercise 3.3.4:** Let R be as in Exercise 3.3.3, but let the FD's be $A \to B$ and $B \to C$. Again compare decomposing using $A \to B$ first against decomposing by $A \to BC$ first.

3.4 Decomposition: The Good, Bad, and Ugly

So far, we observed that before we decompose a relation schema into BCNF, it can exhibit anomalies; after we decompose, the resulting relations do not exhibit anomalies. That's the "good." But decomposition can also have some bad, if not downright ugly, consequences. In this section, we shall consider three distinct properties we would like a decomposition to have.

1. *Elimination of Anomalies* by decomposition as in Section 3.3.
2. *Recoverability of Information.* Can we recover the original relation from the tuples in its decomposition?
3. *Preservation of Dependencies.* If we check the projected FD's in the relations of the decomposition, can we can be sure that when we reconstruct the original relation from the decomposition by joining, the result will satisfy the original FD's?

3.4.5 Exercises for Section 3.4

Exercise 3.4.1: Let $R(A, B, C, D, E)$ be decomposed into relations with the following three sets of attributes: $\{A, B, C\}$, $\{B, C, D\}$, and $\{A, C, E\}$. For each of the following sets of FD's, use the chase test to tell whether the decomposition of R is lossless. For those that are not lossless, give an example of an instance of R that returns more than R when projected onto the decomposed relations and rejoined.

a) $BC \to D$ and $AC \to E$.

b) $B \to E$, $E \to D$, and $B \to E$.

c) $B \to E$, $CE \to D$, and $D \to E$.

d) $A \to D$ and $CD \to B$.

! **Exercise 3.4.2:** For each of the sets of FD's in Exercise 3.4.1, are dependencies preserved by the decomposition?

3.5 Third Normal Form

The solution to the problem illustrated by Example 3.25 is to relax our BCNF requirement slightly, in order to allow the occasional relation schema that cannot be decomposed into BCNF relations without our losing the ability to check the FD's. This relaxed condition is called "third normal form." In this section we shall give the requirements for third normal form, and then show how to do a decomposition in a manner quite different from Algorithm 3.20, in order to obtain relations in third normal form that have both the lossless-join and dependency-preservation properties.

3.5.1 Definition of Third Normal Form

A relation R is in *third normal form* (3NF) if:

- Whenever $A_1 A_2 \cdots A_n \to B_1 B_2 \cdots B_m$ is a nontrivial FD, either

  $$\{A_1, A_2, \ldots, A_n\}$$

  is a superkey, or those of $B_1, B_2, \ldots, B_m$ that are not among the A's, are each a member of some key (not necessarily the same key).

An attribute that is a member of some key is often said to be *prime*. Thus, the 3NF condition can be stated as "for each nontrivial FD, either the left side is a superkey, or the right side consists of prime attributes only."

Note that the difference between this 3NF condition and the BCNF condition is the clause "is a member of some key (i.e., prime)." This clause "excuses" an FD like `theater` $\to$ `city` in Example 3.25, because the right side, `city`, is prime.

3.5.4 Exercises for Section 3.5

Exercise 3.5.1: For each of the relation schemas and sets of FD's of Exercise 3.3.1:

i) Indicate all the 3NF violations.

ii) Decompose the relations, as necessary, into collections of relations that are in 3NF.

Exercise 3.5.2: Verify, using the chase, that the decomposition of Example 3.27 has a lossless join. Eksempel 3.27: Se neste to sider

Exercise 3.5.3: Consider the relation `Courses`(C, T, H, R, S, G), whose attributes may be thought of informally as course, teacher, hour, room, student, and grade. Let the set of FD's for `Courses` be $C \to T$, $HR \to C$, $HT \to R$, $HS \to R$, and $CS \to G$. Intuitively, the first says that a course has a unique teacher, and the second says that only one course can meet in a given room at a given hour. The third says that a teacher can be in only one room at a given hour, and the fourth says the same about students. The last says that students get only one grade in a course.

a) What are all the keys for `Courses`?

b) Verify that the given FD's are their own minimal basis.

c) Use the 3NF synthesis algorithm to find a lossless-join, dependency-preserving decomposition of R into 3NF relations. Are any of the relations not in BCNF?

Exercise 3.5.4: Consider a relation `Stocks`(B, O, I, S, Q, D), whose attributes may be thought of informally as broker, office (of the broker), investor, stock, quantity (of the stock owned by the investor), and dividend (of the stock). Let the set of FD's for `Stocks` be $S \to D$, $I \to B$, $IS \to Q$, and $B \to O$. Repeat Exercise 3.5.3 for the relation `Stocks`.

!! **Exercise 3.5.5:** Suppose we modified Algorithm 3.20 (BNCF decomposition) so that instead of decomposing a relation R whenever R was not in BCNF, we only decomposed R if it was not in 3NF. Provide a counterexample to show that this modified algorithm would not necessarily produce a 3NF decomposition with dependency preservation.

3.6 Multivalued Dependencies

A "multivalued dependency" is an assertion that two attributes or sets of attributes are independent of one another. This condition is, as we shall see, a generalization of the notion of a functional dependency, in the sense that

Other Normal Forms

If there is a "third normal form," what happened to the first two "normal forms"? They indeed were defined, but today there is little use for them. *First normal form* is simply the condition that every component of every tuple is an atomic value. *Second normal form* is a less restrictive verison of 3NF. There is also a "fourth normal form" that we shall meet in Section 3.6.

3.5.2 The Synthesis Algorithm for 3NF Schemas

We can now explain and justify how we decompose a relation R into a set of relations such that:

a) The relations of the decomposition are all in 3NF.

b) The decomposition has a lossless join.

c) The decomposition has the dependency-preservation property.

Algorithm 3.26: Synthesis of Third-Normal-Form Relations With a Lossless Join and Dependency Preservation.

INPUT: A relation R and a set F of functional dependencies that hold for R.

OUTPUT: A decomposition of R into a collection of relations, each of which is in 3NF. The decomposition has the lossless-join and dependency-preservation properties.

METHOD: Perform the following steps:

1. Find a minimal basis for F, say G.
2. For each functional dependency $X \to A$ in G, use XA as the schema of one of the relations in the decomposition.
3. If none of the sets of relations from Step 2 is a superkey for R, add another relation whose schema is a key for R.

□

Example 3.27: Consider the relation $R(A, B, C, D, E)$ with FD's $AB \to C$, $C \to B$, and $A \to D$. To start, notice that the given FD's are their own minimal basis. To check, we need to do a bit of work. First, we need to verify that we cannot eliminate any of the given dependencies. That is, we show, using Algorithm 3.7, that no two of the FD's imply the third. For example, we must take the closure of $\{A, B\}$, the left side of the first FD, using only the

second and third FD's, $C \to B$ and $A \to D$. This closure includes D but not C, so we conclude that the first FD $AB \to C$ is not implied by the second and third FD's. We get a similar conclusion if we try to drop the second or third FD.

We must also verify that we cannot eliminate any attributes from a left side. In this simple case, the only possibility is that we could eliminate A or B from the first FD. For example, if we eliminate A, we would be left with $B \to C$. We must show that $B \to C$ is not implied by the three original FD's, $AB \to C$, $C \to B$, and $A \to D$. With these FD's, the closure of $\{B\}$ is just B, so $B \to C$ does not follow. A similar conclusion is drawn if we try to drop B from $AB \to C$. Thus, we have our minimal basis.

We start the 3NF synthesis by taking the attributes of each FD as a relation schema. That is, we get relations $S_1(A, B, C)$, $S_2(B, C)$, and $S_3(A, D)$. It is never necessary to use a relation whose schema is a proper subset of another relation's schema, so we can drop S_2.

We must also consider whether we need to add a relation whose schema is a key. In this example, R has two keys: $\{A, B, E\}$ and $\{A, C, E\}$, as you can verify. Neither of these keys is a subset of the schemas chosen so far. Thus, we must add one of them, say $S_4(A, B, E)$. The final decomposition of R is thus $S_1(A, B, C)$, $S_3(A, D)$, and $S_4(A, B, E)$. □

3.5.3 Why the 3NF Synthesis Algorithm Works

We need to show three things: that the lossless-join and dependency-preservation properties hold, and that all the relations of the decomposition are in 3NF.

1. *Lossless Join.* Start with a relation of the decomposition whose set of attributes K is a superkey. Consider the sequence of FD's that are used in Algorithm 3.7 to expand K to become K^+. Since K is a superkey, we know K^+ is all the attributes. The same sequence of FD applications on the tableau cause the subscripted symbols in the row corresponding to K to be equated to unsubscripted symbols in the same order as the attributes were added to the closure. Thus, the chase test concludes that the decomposition is lossless.

2. *Dependency Preservation.* Each FD of the minimal basis has all its attributes in some relation of the decomposition. Thus, each dependency can be checked in the decomposed relations.

3. *Third Normal Form.* If we have to add a relation whose schema is a key, then this relation is surely in 3NF. The reason is that all attributes of this relation are prime, and thus no violation of 3NF could be present in this relation. For the relations whose schemas are derived from the FD's of a minimal basis, the proof that they are in 3NF is beyond the scope of this book. The argument involves showing that a 3NF violation implies that the basis is not minimal.

3.6.6 Relationships Among Normal Forms

As we have mentioned, 4NF implies BCNF, which in turn implies 3NF. Thus, the sets of relation schemas (including dependencies) satisfying the three normal forms are related as in Fig. 3.12. That is, if a relation with certain dependencies is in 4NF, it is also in BCNF and 3NF. Also, if a relation with certain dependencies is in BCNF, then it is in 3NF.

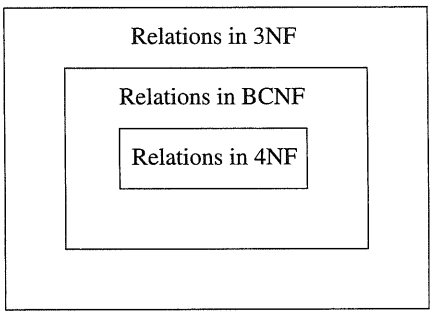

Figure 3.12: 4NF implies BCNF implies 3NF

Another way to compare the normal forms is by the guarantees they make about the set of relations that result from a decomposition into that normal form. These observations are summarized in the table of Fig. 3.13. That is, BCNF (and therefore 4NF) eliminates the redundancy and other anomalies that are caused by FD's, while only 4NF eliminates the additional redundancy that is caused by the presence of MVD's that are not FD's. Often, 3NF is enough to eliminate this redundancy, but there are examples where it is not. BCNF does not guarantee preservation of FD's, and none of the normal forms guarantee preservation of MVD's, although in typical cases the dependencies are preserved.

Property	3NF	BCNF	4NF
Eliminates redundancy due to FD's	No	Yes	Yes
Eliminates redundancy due to MVD's	No	No	Yes
Preserves FD's	Yes	No	No
Preserves MVD's	No	No	No

Figure 3.13: Properties of normal forms and their decompositions

3.6.7 Exercises for Section 3.6

Exercise 3.6.1: Suppose we have a relation $R(A, B, C)$ with an MVD $B \twoheadrightarrow$

C. If we know that the tuples (a_1, b, c_1), (a_2, b, c_2), and (a_3, b, c_3) are in the current instance of R, what other tuples do we know must also be in R?

Exercise 3.6.2: For each of the following relation schemas and dependencies

a) $R(A, B, C, D)$ with MVD's $D \twoheadrightarrow A$ and $A \twoheadrightarrow BC$.

b) $R(A, B, C, D)$ with MVD $AB \twoheadrightarrow D$ and FD $B \rightarrow C$.

c) $R(A, B, C, D, E)$ with MVD's $A \twoheadrightarrow C$ and $AC \twoheadrightarrow D$ and FD's $A \rightarrow E$ and $AC \rightarrow B$.

d) $R(A, B, C, D)$ with MVD's $B \twoheadrightarrow C$ and $B \twoheadrightarrow D$.

do the following:

i) Find all the 4NF violations.

ii) Decompose the relations into a collection of relation schemas in 4NF.

Exercise 3.6.3: Suppose we have a relation in which we want to record for each person their name, Social Security number, and birthdate. Also, for each child of the person, the name, Social Security number, and birthdate of the child, and for each automobile the person owns, its serial number and make. To be more precise, this relation has all tuples

$$(n, s, b, cn, cs, cb, as, am)$$

such that

1. n is the name of the person with Social Security number s.
2. b is n's birthdate.
3. cn is the name of one of n's children.
4. cs is cn's Social Security number.
5. cb is cn's birthdate.
6. as is the serial number of one of n's automobiles.
7. am is the make of the automobile with serial number as.

For this relation:

a) Tell the functional and multivalued dependencies we would expect to hold.

b) Suggest a decomposition of the relation into 4NF.

Exercise 3.6.4: Give informal arguments why we would not expect any of the five attributes in Example 3.28 to be functionally determined by the other four.

1. It is surely not necessary to check the trivial FD's and MVD's.
2. For FD's, we can restrict ourselves to looking for FD's with a singleton right side, because of the combining rule for FD's.
3. An FD or MVD whose left side does not contain the left side of any given dependency surely cannot hold, since there is no way for its chase test to get started. That is, the two rows with which you start the test are unchanged by the given dependencies.

3.7.5 Exercises for Section 3.7

Exercise 3.7.1: Use the chase test to tell whether each of the following dependencies hold in a relation $R(A, B, C, D, E)$ with the dependencies $A \twoheadrightarrow BC$, $B \rightarrow E$, and $C \twoheadrightarrow D$.

a) $A \rightarrow D$.

b) $A \twoheadrightarrow D$.

c) $A \rightarrow E$.

d) $A \twoheadrightarrow E$.

! **Exercise 3.7.2:** If we project the relation R of Exercise 3.7.1 onto $S(A, C, D)$, what nontrivial FD's and MVD's hold in S?

! **Exercise 3.7.3:** Show the following rules for MVD's. In each case, you can set up the proof as a chase test, but you must think a little more generally than in the examples, since the set of attributes are arbitrary sets X, Y, Z, and the other unnamed attributes of the relation in which these dependencies hold.

a) *Removing attributes shared by left and right side.* If $X \twoheadrightarrow Y$ holds, then $X \twoheadrightarrow (Y - X)$ holds.

b) The *Union Rule.* If X, Y, and Z are sets of attributes, $X \twoheadrightarrow Y$, and $X \twoheadrightarrow Z$, then $X \twoheadrightarrow (Y \cup Z)$.

c) The *Intersection Rule.* If X, Y, and Z are sets of attributes, $X \twoheadrightarrow Y$, and $X \twoheadrightarrow Z$, then $X \twoheadrightarrow (Y \cap Z)$.

d) The *Difference Rule.* If X, Y, and Z are sets of attributes, $X \twoheadrightarrow Y$, and $X \twoheadrightarrow Z$, then $X \twoheadrightarrow (Y - Z)$.

! **Exercise 3.7.4:** Give counterexample relations to show why the following rules for MVD's do *not* hold. *Hint*: apply the chase test and see what happens.

a) If $A \twoheadrightarrow BC$, then $A \twoheadrightarrow C$.

b) If $BC \twoheadrightarrow A$, then $B \twoheadrightarrow A$.

c) If $A \twoheadrightarrow B$, then $A \rightarrow B$.