# Chapter 1

# RELATIONS AND RELATION SCHEMES

One of the major advantages of the relational model is its uniformity. All data is viewed as being stored in tables, with each row in the table having the same format. Each row in the table summarizes some object or relationship in the real world. Whether the corresponding entities in the real world actually possess the uniformity the relational model ascribes to them is a question that the user of the model must answer. It is a question of the suitability of the model for the application at hand.

Whether or not the relational model is appropriate for a particular set of data shall not concern us. There are plenty of instances where the model is appropriate, and we always assume we are dealing with such instances.

## 1.1 BRASS TACKS

So much for philosophy. Let us consider an example. An airline schedule certainly exhibits regularity. Every flight listed has certain characteristics. It is a flight from an origin to a destination. It is scheduled to depart at a specific time and arrive at a later time. It has a flight number. Part of an airline schedule might appear as in Table 1.1.

What do we observe about this schedule? Each flight is summarized as a set of values, one in each column. There are restrictions on what information may appear in a given column. The FROM column contains names of airports served by the airline, the ARRIVES column contains times of day. The order of the columns is immaterial as far as information content is concerned. The DEPARTS and ARRIVES columns could be interchanged with no change in meaning. Finally, since each flight has a unique number, no flight is represented by more than one row.

The schedule in Table 1.1 is an example of a relation of type FLIGHTS. The format of the relation is determined by the set of column labels {NUMBER, FROM, TO, DEPARTS, ARRIVES}. These column names are

**Table 1.1** FLIGHTS (airline schedule).

| NUMBER | FROM | TO | DEPARTS | ARRIVES |
|---|---|---|---|---|
| 83 | JFK | O'Hare | 11:30a | 1:43p |
| 84 | O'Hare | JFK | 3:00p | 5:55p |
| 109 | JFK | Los Angeles | 9:50p | 2:52a |
| 213 | JFK | Boston | 11:43a | 12:45p |
| 214 | Boston | JFK | 2:20p | 3:12p |

called attribute names. Corresponding to each attribute name is a set of permissible values for the associated column. This set is called the domain of the attribute name. The domain of NUMBER could be the set of all one-, two- or three-digit decimal integers. Each row in the relation is a set of values, one from the domain of each attribute name. The rows of this relation are called 5-tuples, or tuples in general. The tuples of a relation form a set, hence there are no duplicate rows. Finally, there is a subset of the attribute names with the property that tuples can be distinguished by looking only at values corresponding to attribute names in the subset. Such a subset is called a key for the relation. For the relation in Table 1.1, {NUMBER} is a key.

## 1.2 FORMALIZATION OF RELATIONS

We now formalize the definitions of the last section and add a couple of new ones. A *relation scheme* $R$ is a finite set of *attribute names* $\{A_1, A_2, \ldots, A_n\}$. Corresponding to each attribute name $A_i$ is a set $D_i$, $1 \leq i \leq n$, called the *domain* of $A_i$. We also denote the domain of $A_i$ by $dom(A_i)$. Attribute names are sometimes called *attribute symbols* or simply *attributes*, particularly in the abstract. The domains are arbitrary, non-empty sets, finite or countably infinite. Let $\mathbf{D} = D_1 \cup D_2 \cup \cdots \cup D_n$. A *relation* $r$ on relation scheme $R$ is a finite set of mappings $\{t_1, t_2, \ldots, t_p\}$ from $R$ to $\mathbf{D}$ with the restriction that for each mapping $t \in r$, $t(A_i)$ must be in $D_i$, $1 \leq i \leq n$. The mappings are called *tuples*.

**Example 1.1** In Table 1.1 the relation scheme is FLIGHTS = {NUMBER, FROM, TO, DEPARTS, ARRIVES}. The domains for each attribute name might be:

1. *dom*(NUMBER) = the set of one-, two- or three-digit decimal numbers,

2. *dom*(FROM) = *dom*(TO) = {JFK, O'Hare, Los Angeles, Boston, Atlanta},
3. *dom*(DEPARTS) = *dom*(ARRIVES) = the set of times of day.

The relation in Table 1.1 has five tuples. One of them is $t$ defined as $t$(NUMBER) = 84, $t$(FROM) = O'Hare, $t$(TO) = JFK, $t$(DEPARTS) = 3:00p, $t$(ARRIVES) = 5:55p.

Where did the mappings come from? What happened to tables and rows? We use mappings in our formalism to avoid any explicit ordering of the attribute names in the relation scheme. As we noted in the last section, such an ordering adds nothing to the information content of a relation. We do not want to restrict tuples to be sequences of values in a certain order. Rather, a tuple is a set of values, one for each attribute name in the relation scheme.* The mappings we defined are nothing more than correspondences of this type. Now that we have taken the trouble of avoiding any explicit ordering in relations, in nearly every case we shall denote our relations by writing the attributes in a certain order and the tuples as lists of values in the same order.

In either case, it makes sense, given a tuple $t$, to discuss the *value* of $t$ on attribute $A$, alternatively called the A-*value* of $t$. Considering $t$ as a mapping, the $A$-value of $t$ is $t(A)$. Interpreting $t$ as a row in a table, the $A$-value of $t$ is the entry of $t$ in the column headed by $A$. Since $t$ is a mapping, we can restrict the domain of $t$. Let $X$ be a subset of $R$. The usual notation for $t$ restricted to $X$ is $t_{|X}$. We, in our infinite knowledge, shall confuse the issue and write this restriction as $t(X)$ and call it the X-*value* of $t$. Technically, $t(A)$ and $t(\{A\})$ are different objects, but in keeping with the confusing customs of relational database theory, we often write $A$ for the singleton set $\{A\}$. We also blur the distinction between $t(A)$ and $t(\{A\})$, even though one is just a value and the other is a mapping from $A$ to this value. We assume there is some value $\lambda$ such that $t(\emptyset) = \lambda$ for any tuple $t$. Thus $t_1(\emptyset) = t_2(\emptyset)$ for any tuples $t_1$ and $t_2$.

**Example 1.2** Let $t$ be the tuple defined in Example 1.1. The FROM-value of $t$ is $t$(FROM) = O'Hare. The {FROM, TO}-value of $t$ is the tuple $t'$ defined by $t'$(FROM) = O'Hare, $t'$(TO) = JFK. We shall denote such a tuple as ⟨O'Hare:FROM JFK:TO⟩ or simply ⟨O'Hare JFK⟩ where the order of attributes is understood.

We have been treating relations as static objects. However, relations are supposed to abstract some portion of the real world, and this portion of the world may change with time. We consider that relations are time-varying, so that tuples may be added, deleted, or changed. In Table 1.1, flights may be added or dropped, or their times may be changed. We do assume, though,

*Actually, a tuple could be a multiset (a set with duplicates) of values, if domains for different attribute names intersect.

that the relation scheme is time-invariant. Henceforth, when dealing with a relation, we shall think of it as a sequence of relations in the sense already defined, or, in some cases, as potential sequences that the relation might follow, that is, possible states the relation may occupy. We shall discuss restrictions on the states a relation may assume, although nearly all of these restrictions will be *memoryless*: they will depend only on the current state of the relation and not on its history of previous states.

## 1.3 KEYS

A *key* of a relation $r$ on relation scheme $R$ is a subset $K = \{B_1, B_2, \ldots, B_m\}$ of $R$ with the following property. For any two distinct tuples $t_1$ and $t_2$ in $r$, there is a $B \in K$ such that $t_1(B) \neq t_2(B)$. That is, no two tuples have the same value on all attributes in $K$. We could write this condition as $t_1(K) \neq t_2(K)$. Hence, it is sufficient to know the $K$-value of a tuple to identify the tuple uniquely.

**Example 1.3** In Figure 1.1, {NUMBER} and {FROM, TO} are both keys.

Let us formulate some notation for relations, schemes, and keys. Our convention will be to use uppercase letters from the front of the alphabet for attribute symbols, uppercase letters from the back of the alphabet for relation schemes, and lowercase letters for relations. We denote a relation scheme $R = \{A_1, A_2, \ldots, A_n\}$ by $R[A_1A_2 \cdots A_n]$, or sometimes $A_1 A_2 \cdots A_n$ when we are not concerned with naming the scheme. (Another confusing custom of relational database theory is to use concatenation to stand for set union between sets of attributes.) A relation $r$ on scheme $R$ is written $r(R)$ or $r(A_1A_2 \cdots A_n)$. To denote the key of a relation, we underline the attribute names in the key. Relation $r$ on scheme $ABCD$ with $AC$ as a key is written $r(\underline{A}B\underline{C}D)$. We can also incorporate the key into the relation scheme: $R[\underline{A}B\underline{C}D]$. Any relation $r(R)$ is restricted to have $AC$ as a key.

**Example 1.4** We can write the relation scheme for the relation in Table 1.1 as FLIGHTS [$\underline{\text{NUMBER}}$ FROM TO DEPARTS ARRIVES].

If we wish to specify more than one key for a scheme or relation, we must list the keys separately, since the underline notation will not work. The keys explicitly listed with a relation scheme are called *designated keys*. There may be keys other than those listed; they are *implicit keys*. Sometimes we distinguish one of the designated keys as the *primary key*.

Our definition of key is actually a bit too broad. If relation $r(R)$ has key $K'$, and $K' \subseteq K \subseteq R$, then $K$ is also a key for $R$. For tuples $t_1$ and $t_2$ in $r$, if $t_1(K') \neq t_2(K')$, then surely $t_1(K) \neq t_2(K)$. We shall restrict our definition slightly.

**Definition 1.1** A *key* of a relation $r(R)$ is a subset $K$ of $R$ such that for any distinct tuples $t_1$ and $t_2$ in $r$, $t_1(K) \neq t_2(K)$ and no proper subset $K'$ of $K$ shares this property. $K$ is a *superkey* of $r$ if $K$ contains a key of $r$.

The new definition of superkey is the same as the former definition of key. We shall still use the old definition of key in designated key, that is, a designated key may be a superkey.

**Example 1.5** In Table 1.1, {NUMBER} is a key (and a superkey), so {NUMBER, FROM} is a superkey but not a key.

There are some subtleties with keys. As we mentioned in the last section, we consider relations to be time-varying. For any given state of the relation, we can determine the keys and superkeys. Different states of the relation may have different keys. We consider relation schemes, though, to be time-invariant; we would like the keys specified with relation schemes not to vary either. Thus, in determining keys for a relation scheme, we look across all states a relation on the scheme may assume. Keys must remain keys for all permissible data.

**Example 1.6** In Table 1.1, {FROM, TO} is a key for the relation. However, it is likely that there could be two flights between the same origin and destination, although they would undoubtedly leave at different times. Hence {FROM, TO, DEPARTS} is a key for the relation scheme FLIGHTS.

We shall mainly concern ourselves with keys and superkeys of relation schemes, thinking in terms of all permissible states of a relation on the scheme. What is and is not a key is ultimately a semantic question.

## 1.4 UPDATES TO RELATIONS

Now that we have relations, what can be done with them? As noted, the content of a relation varies with time, so we shall consider how to alter a relation. Suppose we wish to put more information into a relation. We perform an *add* operation on the relation. For a relation $r(A_1A_2 \cdots A_n)$, the add operation takes the form

$$\mathrm{ADD}(r; A_1 = d_1, A_2 = d_2, \ldots, A_n = d_n).$$

**Example 1.7** Call the relation in Table 1.1 *sched.* We might perform the update

ADD(*sched*; NUMBER = 117, FROM = Atlanta, TO = Boston,
DEPARTS = 10:05p, ARRIVES = 12:43a).

When there is an order assumed on the attribute names, the shorter version

$$\mathrm{ADD}(r; d_1, d_2, \ldots, d_n)$$

suffices.

**Example 1.8** The short version of Example 1.7 is

ADD(*sched*; 117, Atlanta, Boston, 10:05p, 12:43a).

The intent of the add operation is clear, to add the tuple described to the relation specified. The result of the operation might not agree with the intent for one of the following reasons:

1. The tuple described does not conform to the scheme of the specified relation.
2. Some values of the tuple do not belong to the appropriate domains.
3. The tuple described agrees on a key with a tuple already in the relation.

In any of these cases, we consider $\mathrm{ADD}(r; d_1, d_2, \ldots, d_n)$ to return $r$ unchanged and in some manner indicate the error.

**Example 1.9** If *sched* is the relation in Table 1.1, then

ADD(sched; NUMBER = 117, FROM = Atlanta, TO = Boston,
DATE = 4 March)

is disallowed for reason 1 above. The operation

$$\text{ADD}(\textit{sched};\ \text{NUMBER} = 84, \text{FROM} = \text{O'Hare}, \text{TO} = \text{JFK}, \text{DEPARTS} = 25{:}15\text{p}, \text{ARRIVES} = 6{:}00\text{p})$$

is disallowed for both reasons 2 and 3. (Examine DEPARTS and NUMBER).

We must be able to undo what we do, which calls for a *delete* operation. On a relation $r$ as above, the delete operation takes the form

$$\text{DEL}(r;\ A_1 = d_1, A_2 = d_2, \ldots, A_n = d_n).$$

Again, when there is an assumed order on the attribute names, we abbreviate to

$$\text{DEL}(r;\ d_1, d_2, \ldots, d_n).$$

**Example 1.10** If *sched* is the relation in Table 1.1, we can have

$$\text{DEL}(\textit{sched};\ \text{NUMBER} = 83, \text{FROM} = \text{JFK}, \text{TO} = \text{O'Hare}, \text{DEPARTS} = 11{:}30\text{a}, \text{ARRIVES} = 1{:}43\text{p}),$$

with short version

$$\text{DEL}(\textit{sched};\ 83, \text{JFK}, \text{O'Hare}, 11{:}30\text{a}, 1{:}43\text{p}).$$

Actually, we do not need to give so much information to identify uniquely the tuple to be removed. Specifying the values on some key will suffice. If $K = \{B_1, B_2, \ldots, B_m\}$ is a key, then we may use the form

$$\text{DEL}(r;\ B_1 = e_1, B_2 = e_2, \ldots, B_m = e_m).$$

**Example 1.11** A shorter version of the delete in Example 1.10 is

$$\text{DEL}(\textit{sched};\ \text{FROM} = \text{JFK}, \text{TO} = \text{O'Hare}, \text{DEPARTS} = 11{:}30).$$

If there is a primary designated key, such as {NUMBER}, we could even shorten this form to DEL(*sched*; 83).

The result of the delete operation is as expected. The specified tuple is removed from the relation, except when the tuple is not present in the relation. In this case, the relation is left unchanged and an error condition is signaled. There is no restriction on removing the last tuple from a relation; the empty relation is allowed.

Instead of adding or deleting an entire tuple, we may want to modify only part of a tuple. Modification is achieved with the *change* operation. For a relation $r$ as before, with $\{C_1, C_2, \ldots, C_p\} \subseteq \{A_1, A_2, \ldots, A_n\}$, the change operation takes the form

$$\begin{aligned} &\mathrm{CH}(r; A_1 = d_1, A_2 = d_2, \ldots, A_n = d_n;\\ &\quad C_1 = e_1, C_2 = e_2, \ldots, C_p = e_p). \end{aligned}$$

If $K = \{B_1, B_2, \ldots, B_m\}$ is a key, then we abbreviate to

$$\mathrm{CH}(r; B_1 = d_1, B_2 = d_2, \ldots, B_m = d_m; C_1 = e_1, C_2 = e_2, \ldots, C_p = e_p).$$

**Example 1.12** For the relation *sched* in Table 1.1 we could have

CH(*sched*; NUMBER = 109, FROM = JFK, TO = Los Angeles,
DEPARTS = 9:50p, ARRIVES = 2:52a; DEPARTS = 9:40p,
ARRIVES = 2:42a),

with short version

CH(*sched*; NUMBER = 109; DEPARTS = 9:40p, ARRIVES = 2:42a).

The change operation is mainly a convenience. The same result can be obtained with a delete followed by an add. Therefore, all the possible errors for add and delete apply to the change operation: the specified tuple does not exist, the changes have the wrong format or use values outside the appropriate domain, or the changed tuple has the same key value as a tuple already in the relation.

**Example 1.13** The effect of applying the operations

1. ADD(*sched*; 117, Atlanta, Boston, 10:05p, 12:43a),
2. DEL(*sched*; FROM = JFK, TO = O'Hare, DEPARTS = 11:30a), and
3. CH(*sched*; NUMBER = 109; DEPARTS = 9:40p, ARRIVES = 2:42a)

to Table 1.1 is shown by Table 1.2.

## 1.5 EXERCISES

1.1 (a) Let $R$ be the relation scheme {EMPLOYEE, MANAGER, JOB, SALARY, YEARS-WORKED}, where EMPLOYEE and MANAGER

**Table 1.2** New version of *sched*(FLIGHTS).

| NUMBER | FROM | TO | DEPARTS | ARRIVES |
|---|---|---|---|---|
| 84 | O'Hare | JFK | 3:00p | 5:55p |
| 109 | JFK | Los Angeles | 9:40p | 2:42a |
| 117 | Atlanta | Boston | 10:05p | 12:43a |
| 213 | JFK | Boston | 11:43a | 12:45p |
| 214 | Boston | JFK | 2:20p | 3:12p |

are names, JOB is a job title, SALARY is yearly salary, and YEARS-WORKED is the number of complete years the employee has been at the job. Construct a relation on $R$ based on the following information.

i. Roberts, Ruskin, and Raphael are all ticket agents.
ii. Rayburn is a baggage handler.
iii. Rice is a flight mechanic.
iv. Price manages all ticket agents.
v. Powell manages Rayburn.
vi. Porter manages Rice, Price, Powell and himself.
vii. Powell is head of ground crews and Porter is chief of operations.
viii. Every employee receives a 10% raise for each complete year worked.
ix. Roberts, Ruskin, Raphael, and Rayburn all started at $12,000. Roberts just started work, Ruskin and Raphael have worked for a year and a half, and Rayburn has worked for 2 years.
x. Rice started at $18,000 and now makes $21,780.
xi. Price and Powell started at $16,000 and have both been working for three years.
xii. Porter started at $20,000 and has been around two years longer than anyone else.

(b) Give appropriate update operations for the following changes to the relation for part (a):

i. Ruskin and Raphael complete their second year.
ii. Rice quits.
iii. Powell quits. His duties are assumed by Porter.
iv. Randolph is hired as a ticket agent.

1.2 Consider the relation scheme $R = \{$FLIGHT-NUMBER, DATE, GATE, TIME, DESTINATION$\}$. A tuple $\langle d_1\, d_2\, d_3\, d_4\, d_5 \rangle$ of $r(R)$ has the meaning "flight $d_1$ departs on date $d_2$ from gate $d_3$ at $d_4$ for $d_5$." What are the keys of $R$?

1.3 Let $t$ be a tuple in $r(R)$ and let $X$ and $Y$ be subsets of $R$. When does the expression $t(X)(Y)$ make sense? When it does make sense, how can it be simplified?

1.4 (a) Can the union of two keys be a key?
(b) Is the intersection of two superkeys necessarily a key?

1.5* Given a relation scheme $R[A_1A_2 \cdots A_n]$, what is the maximum number of keys $R$ can have? The maximum number of superkeys?

1.6 What can be said about a relation with a key $K = \emptyset$?

1.7 Let $K = \{B_1, B_2, \ldots, B_m\}$ be a key of the relation scheme $R[A_1A_2 \cdots A_n]$ and let $r$ be a relation on $R$. Consider the operation

$$\begin{aligned}\text{CH}(r; A_1 = d_1, A_2 = d_2, \ldots, A_n = d_n;\\ \text{B}_1 = e_1, B_2 = e_2, \ldots, B_m = e_m).\end{aligned}$$

Suppose that no tuple in $r$ has $K$-value $\langle e_1\, e_2 \cdots e_m\rangle$, there is a tuple $\langle d_1\, d_2 \cdots d_n\rangle$ in $r$, and that $e_i \in \text{dom}(B_i)$, $1 \leq i \leq m$. Is this change operation necessarily legal?

1.8 Let $\Sigma$ be a sequence of update operations to be applied to relation $r$. If the order of the operations is changed in $\Sigma$, will the result necessarily be the same when $\Sigma$ consists of

(a) only add operations?
(b) only delete operations?
(c) add and delete operations?
(d) add and change operations?
(e) only change operations?

## 1.6 BIBLIOGRAPHY AND COMMENTS

The relational data model was originally expounded in a series of papers by Codd [1970, 1971a, 1971b, 1972a]. For background in database systems and the place of the relational model in the scheme of things, the reader is directed to the books by Cardenas [1979], Date [1981], Tsichritzis and Lochovsky [1977], Ullman [1980], and Wiederhold [1977].