RD2. A Tiny Database and Two Extensions to it

Here is a tiny database with just four relations. You will use it, and extensions to it throughout the book - with data words such as Creature, S_code and Person, are capitalized.

Work → Before reading on, you should study this tiny database. Study the names of things and count how many of various things you see. (Do not skip this exercise!) Try to broadly characterize the data. For example, the Score values are about evenly distributed.

Advice: Put a tab here so you can easily refer back to the tiny database and its extensions.

Creature

C_id	C_name	C_type
1	Bannon	Person
2	Myers	Person
3	Neff	Person
4	Neff	Person
5	Mieska	Person
6	Carlis	Person
7	Kermit	Frog
8	Godzilla	Monster

Skill

S_code	S_description
A	Float
E	Swim
O	Sink
U	Walk on Water
Z	Gargle

Achievement

C_id	S_code	Score
1	A	1
1	E	3
1	Z	3
2	A	3
3	A	2
3	Z	1
4	A	2
4	E	2
5	Z	3
7	E	1
8	O	1

Aspiration

C_id	S_code	Score
1	A	1
1	E	3
1	Z	1
2	A	3
3	A	2
3	Z	2
4	E	2
5	Z	3
6	Z	3
7	E	3
8	O	1

Interesting Data:

Here are some counts. The tiny database has four relations: Creature, Skill, Achievement and Aspiration. Creature has eight rows of data and three columns named C_id, C_name and C_type. Creature has one identifier with one column, C_id, in that set. Skill has five rows and two columns named S_code and S_description. Skill has one identifier with one column, S_code, in that set. Achievement has eleven rows and three columns named C_id, S_code and Score. Achievement has one identifier with two columns, C_id and S_code, in that set. Aspiration has the same number of columns (3) and rows (11), the same column names, the same number of identifiers (one) and the same set of identifying column names – but it has different semantic content.

Stop! Some novices count incorrectly and when asked "How many identifiers does Achievement have" reply with "Two." That is wrong. Achievement has one identifier consisting of a set of columns with a set size of two. In other words, C_id and S_code together identify Achievement. So while a Creature can achieve many Skills and a Skill can be achieved by many Creatures, the identifier for Achievement asserts that there is at most one Achievement for a given Creature-Skill pair.

Each of these relations has a one-word name. Creature and Skill have obvious bases (Creature and Skill), and implicit column and row modifiers (every specified column and every current row). Achievement and Aspiration have implicit column modifiers and have hidden bases and noun forms of their row modifiers. Their long names would have been "achieved Creature – Skill Pair" and "aspired-to Creature – Skill Pair."

The relations are connected: C_id appears in Creature, Achievement and Aspiration, while S_code appears in Skill, Achievement and Aspiration. So the database remembers: about each Achievement its Creature and its Skill; about each Aspiration its Creature and its Skill; and about each Creature its Achievements and Aspirations. Notice that the last two terms are plural – think *set*.

Be careful – while like-named columns generally do connect relations, they may not, especially if you blindly integrate different operational databases for analytic tasks. Also, as you will see soon, differently-named columns can connect relations.

For any database, users (in this case your author is the user) make burden-shifting boundary decisions — what is remembered in the database, and what is outside, not remembered and therefore not analyzable. Here I decided to store Score numbers, but not what they mean. (You, not the DBMS, bear the burden of remembering that: 1 means Good; 2 means Fair; and 3 means Poor.) However, I did store what the S_codes mean, e.g., "A" means "Float." Boundary decisions affect how much you need to know in order to query the database. So now you can directly ask the DBMS "who Floats," you cannot ask directly it "who achieves Poorly" unless you have remembered, outside the database, that a Score of 3 means Poor.

Because you need to know the meaning of Score values, you could highlight the previous paragraph, copy it for handy reference, or shift a burden onto the DBMS by adding the relation below.

Score_entry

Score	Score_meaning
1	Good
2	Fair
3	Poor

`Work →` Before reading on, use your finger (not just your eyeball) to examine the tiny database and confirm that each of the following statements is true. Notice how much of the database you touched for each statement, and how you searched. Look for shortcuts or oddities in the statements, e.g., people instead of Persons and Frogs where there is only one.

- There are six people.
- Myers is a Person who Floats with Score 3.
- There are two Neffs, and every Neff Floats, but not every Neff Swims.
- Neff (#4) Floats, but does aspire to do so.
- Nobody is worse than Bannon at Gargling.
- While only Godzilla Sinks, several people Float, although not equally well.
- On average, Frogs Swim better than people.
- There are no NULL values, Carlis does not achieve (hmmm, a worrisome tidbit), and nobody Walks on Water.
- Bannon achieves, sort of, what he aspires to.

The last two statements deserve highlighting. First, while a Creature can have *many Achievements*, she can achieve each *Skill* only *once*. This restriction is due to Achievement's identifier which asserts that the database cannot remember, for example, that Bannon now Gargles poorly but used to be a fair Gargler. (You will see such data in Part V.) Also, while a Creature can achieve a Skill at most once, she can achieve a Score *many* times. Second, while a Creature *can* have *many* Achievements, an individual might *actually* have just *one*, or *zero*. Remembering to incorporate degenerates like Carlis in your querying and analysis is part of mastery. For example, because somebody does not achieve, the average number of Skills per Creature (11/8) differs from the average number of Skills per achieving Creature (11/7).

Because this tiny database is tiny, when you look at it you easily can *peruse* it, that is, intensely study all of it. However, in a real database you will have more and larger relations, and will only be able to *browse* the database, that is, skim moderate portions of it, intensely eyeball only small portions. Therefore, you need operators to work for you.

Two Extensions to the Tiny Database.

Where possible you will see just the original tiny database relations, but in order to explain some of the operators, you will need to see different data. (--including little chuckles, e.g., it's really relational since all the people, and just the people, are related to Carlis; we do the Minnesota thing and "go up North to Bemidji to Float and Swim"; and the walk-on-able water in Embarrass is frozen much of the year.) Later, in Chapter ME of Part V, you will see several more extensions which will help illustrate non-matching, "range" operators. (You may want to read that Chapter after reading this one.)

Extension 1

The first extension adds a new column to each original relation and two new relations. Now the database can remember that each:

- Town has an id and a name.
- Creature resides in a Town.
- Achievement tested for in a Town.
- Skill originated in a Town.
- Aspiration tested for in a Town.
- Town can have a Mayor who is a Creature, but only Anoka does.
- Town can have a biggest rival Town. Some do; some don't: some reciprocate; some form a cycle larger than 2.
- Town can be associated with Creatures, Skills, Achievements and Aspirations. (Notice the plurals.)

Suppose I assert that Test_T_id in Achievement has a foreign key association with T_id in Town. (You can infer that Test_T_id in Aspiration does not have one because of the NULL value.) Do you agree with this foreign key? You may object because you think that Godzilla sinks in the ocean not in any Town, not even Tokyo, and should be NULL.

Here is a new kind of information. Normally with a non-identifying column such as Mayor several different Towns could have the same Mayor – just like several Creatures could have the C_type of Person. However, I assert that a Creature can be the mayor of *at most one*

Town. This extra constraint allows me a choice: either a "Mayor_C_id" column in Town, or a "Mayored_T_id" column in Creature, or both. Unfortunately, you cannot look at the Town relation and see the at-most-one constraint.

Beware! In general, most people are surprised at how few constraints truly apply to all instances. I added this constraint because I wanted to create the data for a seldom used Part IV Match Join circumstance.

Notice that I asserted that a Creature can be the Mayor of at most one Town. However, I did not say when, or by means, that constraint would be enforced. It could be accomplished by some user interface software at the time when the data comes off the users fingertips. It could be enforced by the DBMS when that software (or the user directly) asks the DBMS to modify the database. It could be just left to the users to behave themselves, with, perhaps, an occasional report on anomalies.

Creature

C_id	C_name	C_type	Reside_T_id
1	Bannon	Person	p
2	Myers	Person	a
3	Neff	Person	b
4	Neff	Person	c
5	Mieska	Person	d
6	Carlis	Person	p
7	Kermit	Frog	h
8	Godzilla	Monster	t

Skill

S_code	S_description	Origin_T_id
A	Float	b
E	Swim	b
O	Sink	t
U	Walk on Water	em
Z	Gargle	p

Town

T_id	T_name	Mayor_C_id	Biggest_Rival_T_id
a	Anoka	2	be
b	Bemidji	NULL	d
be	Blue Earth	NULL	c
c	Chaska	NULL	a
d	Duluth	NULL	b
e	Edina	NULL	h
em	Embarrass	NULL	NULL
h	Hollywood	NULL	p
p	Philly	NULL	d
s	Swampville	NULL	NULL
t	Tokyo	NULL	NULL

Achievement

C_id	S_code	Score	Test_T_id
1	A	1	a
1	E	3	a
1	Z	3	p
2	A	3	b
3	A	2	b
3	Z	1	p
4	A	2	c
4	E	2	c
5	Z	3	d
7	E	1	s
8	O	1	t

Aspiration

C_id	S_code	Score	Test_T_id
1	A	1	a
1	E	3	a
1	Z	1	be
2	A	3	NULL
3	A	2	b
3	Z	2	be
4	E	2	c
5	Z	3	d
6	Z	3	e
7	E	3	s
8	O	1	t

Extension 2

The second extension adds new relations and new columns to relations already defined.

As shown below, Creature now has two new columns. One is defined over the same domain as S_code; the other is defined over the same domain as C_id. It has an extra column in Skill. It also has two new relations: "Creature – Reside Town of Friend Pair" and "Forbidden Town Achievement." With Creature and Skill now the database remembers that each:

- Creature can have one particular Skill that he/she is just not interested in, that is, a Least-aspired-to Skill.
- Creature can have a Boss Creature, but some do not.