

**Information
Modeling**

David Edmond

Javvin Technologies Inc. Distribution

Information Modeling

Information Modeling

David Edmond

Javvin Technologies Inc. Distribution

<http://www.javvin.com>

<http://www.networkdictionary.com>

Table of Contents

Chapter 1 Introduction	1
1.1 Why Compute?	1
1.2 Facts and Knowledge	2
1.3 Inside a Bank	4
1.4 Next Please!.....	9
1.5 The NextPlease Program.....	12
1.6 Summary.....	15
Exercises.....	17
Chapter 2 Specific Facts	18
2.1 Introduction	18
2.2 The Plain Facts	19
2.3 Facts as Relationships.....	20
2.3.1 Relations	20
2.3.2 Defining Fact Types	21
2.3.3 Domains and Ranges	22
2.3.4 Base Types	22
2.3.5 Formalizing Sentences	23
2.4 One-to-many Relationships	24
2.4.1 Functions	24
2.4.2 Partial Functions.....	26
2.5 One-to-one Relationships	27
2.6 The Construction of Simple Sentences.....	29
2.6.1 Function Application	29
2.6.2 Terms	31
2.6.3 Variables.....	32
2.6.4 Infix and Prefix Form	33
2.7 The Circle Database	35
2.8 Compound Sentences	36
2.8.1 Operations on Sentences	36
2.8.2 Negation	37
2.8.3 Conjunction: When both sentences must be true	37
2.8.4 Disjunction: When at least one of the sentences must be true.....	38
2.8.5 Sentence Construction	39
2.8.6 Evaluating Sentences.....	40
2.8.7 Phrasing Sentences	41
2.9 Summary.....	42
Exercises.....	43
Chapter 3 Sets	48
3.1 Introduction	48

3.2 Sets and Everyday Language	48
3.3 Set Extension	49
3.4 A Sample Database	51
3.5 Set Comprehension	52
3.5.1 Form 1: {Declaration Predicate}.....	52
3.5.2 Form 2: {Declaration Predicate • Term}	55
3.5.3 Form 3: {Declaration • Term}.....	56
3.6 Set Operations	57
3.7 Higher Order Sets	59
3.7.1 Power sets	59
3.7.2 Declarations.....	60
3.8 Product sets	62
3.9 Sets, Relations and Functions	63
3.9.1 Type Construction.....	63
3.9.2 Relations and Functions	64
3.9.3 Deriving New Relations	65
3.9.4 Deriving New Functions.....	66
3.10 Set Terms	68
3.11 Summary	69
Exercises.....	70

Chapter 4 Relations.....77

4.1 Introduction	77
4.2 Merging Facts	77
4.3 Relations	80
4.4 Tuples	81
4.4.1 Form Filling.....	81
4.4.2 Tuple or Aggregate Objects	82
4.4.3 A Definition.....	84
4.4.4 Identifying Individual Tuples	84
4.5 Domains.....	85
4.6 Problems with the Automatic.....	87
4.6.1 Solving the Problem of Repetition	88
4.6.2 Solving the Composite Domain Problem.....	89
4.6.3 Solving the Set Valued Domain Problem.....	90
4.7 The Cars Database	91
4.8 Anatomy of a Database.....	92
4.8.1 The SUBJECT Database.....	92
4.8.2 Keys.....	93
4.9 Relational Languages	94
4.9.1 Relational Algebra	95
4.9.2 Relational Calculus.....	96
4.9.3 The Select Operation.....	97
4.9.4 The Project Operation.....	98

4.9.5 The Product Operation	99
4.9.6 The Join Operation	100
4.9.7 Relational Expressions	101
4.9.8 Relational Calculus Summary	101
4.10 The Circle Database	102
4.10.1 Circle Record Types	102
4.10.2 Comparing the Two Views of the Circle	104
4.11 Summary	106
Exercises.....	107
Chapter 5 Introducing SQL.....	111
5.1 Introduction	111
5.2 SQL Databases.....	112
5.3 Database Definition.....	113
5.4 Database Retrieval	114
5.5 Database Modification	117
5.6 Database Security.....	119
5.7 Using SQL.....	119
5.8 Summary.....	120
Exercises.....	121
Chapter 6 SQL Retrieval	124
6.1 Introduction	124
6.2 Simple Queries	125
6.3 Join Queries.....	126
6.4 Statistical Queries	132
6.5 “Group by” Queries	133
6.6 Multi-table “Group by” Queries.....	136
6.7 Product Queries	138
6.8 Pattern Matching.....	139
6.9 Summary.....	140
Exercises.....	142
Chapter 7 SQL Modularization	149
7.1 Introduction	149
7.2 Query Nesting	149
7.3 Simple Nesting.....	151
7.4 “In” Queries	152
7.5 “All–Any” Queries	153
7.6 Correlated Subqueries	155
7.7 “Exists” Queries.....	156
7.8 Subquery Usage	158
7.9 The Union Operator	158
7.10 Union Usage	161

7.11 Views	162
7.12 View Usage	164
7.13 Summary	164
Exercises.....	166
Chapter 8 Facts and Relations	170
8.1 Introduction	170
8.2 Facts	171
8.3 A Simple Design	173
8.4 An Experiment.....	174
8.5 Another Experiment	175
8.6 Uniqueness Constraints	179
8.7 Single and Many-valued Fact Types	181
8.8 Irreducible Facts	183
8.9 Nested Fact Types	185
8.10 Aggregation	186
8.10.1 Determinants	189
8.10.2 Record Types.....	190
8.10.3 Attribute Naming	190
8.10.4 Looking for Nulls	191
8.11 Establishing the Database.....	195
8.12 Summary	197
Exercises.....	198
Chapter 9 Uncovering Facts	203
9.1 Introduction	203
9.2 Defining Syntax	203
9.3 Analyzing a View	204
9.4 Another Analysis	206
9.5 A Summary of the Notation	206
9.6 Some More Examples.....	207
9.7 View Analysis	209
9.8 Deriving View Relations	209
9.8.1 Flattening Structures	209
9.8.2 Separating Alternatives.....	210
9.8.3 Gather Them Together.....	210
9.9 Extracting Elementary Fact Types	211
9.10 Further Abstraction.....	213
9.11 Summary	216
Exercises.....	217
Chapter 10 Fact-based Analysis	221
10.1 Introduction	221
10.2 The Problem.....	223

10.3 Step 1: Uncover the fact types	223
10.3.1 Derive View Structures	224
10.3.2 Derive View Relations.....	225
10.3.3 Extract Elementary Fact Types.....	225
10.4 Step 2: Look for uniqueness constraints	228
10.5 Step 3: Construct record types	230
10.6 Step 4: Decide which attributes may be null	233
10.7 Step 5: Define the database	235
10.8 Step 6: Review the design	237
10.9 Summary.....	238
Exercises.....	239
Chapter 11 Entity-relationship Modeling.....	241
11.1 Introduction.....	241
11.2 An Example	242
11.2.1 Entities	242
11.2.2 Relationships	243
11.2.3 Attributes	246
11.2.4 Dependent or Weak Entity Types	248
11.2.5 Recursive Relationships	249
11.3 Database Design.....	252
11.4 The Conversion Process	253
11.5 Issues in ER Modeling.....	258
11.5.1 Entity or Attribute?.....	258
11.5.2 Entity or Relationship?	259
11.5.3 Naming.....	259
11.5.4 Optional and Mandatory Roles	261
11.6 Summary	261
Exercises.....	263
Chapter 12 Knowledge.....	267
12.1 Introduction	267
12.2 The Predicate Calculus	268
12.2.1 Simple Sentences.....	268
12.2.2 Terms	269
12.2.3 Compound Sentences	270
12.3 Quantification	271
12.3.1 Existential Quantification	271
12.3.2 The One-point Rule	274
12.3.3 Universal Quantification.....	275
12.3.4 Implication	275
12.3.5 A Summary of Quantification	277
12.3.6 Quantifier Equivalences.....	278
12.4 Defining New Symbols.....	280

12.4.1 Introducing New Total Functions	283
12.4.2 Introducing New Partial Functions	285
12.5 Generic Functions and Relations	286
12.6 Describing Change	291
12.6.1 Adding New Facts	291
12.6.2 Removing Facts	293
12.6.3 Modifying Facts	294
12.7 Abbreviations	295
12.8 Sequences	298
12.8.1 Sequence Construction	300
12.8.2 Sequence Decomposition	301
12.8.3 Operations on Sequences	301
12.9 Summary	302
Exercises	303
Chapter 13 The Knowledge Base	309
13.1 Introduction	309
13.2 Information Systems Development	310
13.3 Knowledge	310
13.4 Representing Organizational Knowledge	311
13.5 A look at Z	312
13.6 Signatures	314
13.6.1 Declaration	315
13.6.2 Type Introductions	316
13.6.3 Sets	317
13.6.4 Set Extension	317
13.6.5 Set Comprehension	318
13.6.6 Type Construction	318
13.6.7 Set Operations	319
13.6.8 Special Set Operations	319
13.6.9 Fact Types	320
13.6.10 Sequences and Sequence Operations	321
13.7 Predicates	321
13.7.1 The Structure of a Predicate	322
13.7.2 Simple Predicates	322
13.7.3 Compound Predicates	323
13.7.4 Quantified Predicates	324
13.8 Kinds of Schema	324
13.8.1 Process Descriptions	324
13.8.2 State Descriptions	326
13.8.3 Type Descriptions	326
13.9 Summary	328
Exercises	330

Chapter 14 From Specification to Implementation.....	331
14.1 Introduction	331
14.2 The State Schema	331
14.3 Schema Inclusion.....	332
14.4 Schema Decoration	333
14.5 State Transition	333
14.6 Operation Schemas	334
14.6.1 Enrolling a New Student.....	335
14.6.2 Award a Mark.....	336
14.6.3 Amend a Mark	337
14.6.4 A Student Drops Out.....	338
14.7 Read-only Transactions	339
14.8 Maintaining the State Invariant	339
14.8 Maintaining the State Invariant	341
14.10 Implementation	343
14.11 Developing the Database.....	344
14.12 The State Schema and the Database.....	346
14.13 Implementing an Operation.....	348
14.14 From Operation to Program.....	349
14.15 Summary.....	350
Exercises.....	352
Chapter 15 Database Definition in SQL.....	358
15.1 Introduction	358
15.2 Tables.....	358
15.2.1 Table Creation.....	358
15.2.2 Table Alteration	361
15.2.3 Table Removal.....	362
15.3 SQL Datatypes.....	362
15.3.1 Datatypes	362
15.3.2 Numbers	363
15.3.3 Character Strings.....	365
15.3.4 The Date Datatype.....	368
15.3.5 Conversion Between Datatypes	370
15.4 Referential Integrity and Other Constraints.....	370
15.5 Views.....	372
15.6 Indexes	374
15.6.1 Unique or Primary Indexes	374
15.6.2 Secondary Indexes.....	376
15.6.3 The Role of Indexes in a Join Operation	377
15.6.4 Advantages and Disadvantages of Indexes	379
15.7 Summary.....	380
Exercises.....	381

Chapter 16 Database Manipulation in SQL	386
16.1 Introduction	386
16.2 Adding New Rows	386
16.2.1 Single Row Insert	386
16.2.2 Multi-row Insert	387
16.3 Modifying Existing Rows	388
16.4 Removing Rows	391
16.5 Transactions	392
16.6 Referential Integrity	393
16.7 View Update	395
16.8 Controlling Database Access	397
16.8.1 Granting Access	397
16.8.2 Revoking Privileges	398
16.9 Summary	399
Exercises	400
Chapter 17 Application Programming	405
17.1 Introduction	405
17.2 Using SQL	406
17.3 Host Language Interface	407
17.3.1 Introduction	407
17.3.2 Pre-processing	408
17.3.3 The Enrol Program	408
17.3.4 The Declare Section	411
17.3.5 The SQL Communications Area	411
17.3.6 Exception Handling	412
17.3.7 Assignment	414
17.3.8 The SQLcode Variable	415
17.3.9 The Classlist Program	415
17.3.10 Cursors	418
17.3.11 Indicator Variables	419
17.4 Form-based Application Development	420
17.4.1 Transaction Processing	420
17.4.2 Using Forms	421
17.4.3 Using Automated Forms	422
17.4.4 Other Points on the Form	424
17.4.5 Triggered Actions	425
17.4.6 Awarding a Mark	426
17.5 Summary	428
Exercises	429
Chapter 18 Case Studies	430
18.1 Introduction	430
18.2 The League Table	430

18.2.1 Introduction	430
18.2.2 Defining the League	432
18.2.3 Adding New Results	434
18.2.4 Producing a Summary Table.....	435
18.2.5 The League Database	436
18.3 The Rocky Concrete Company	443
18.3.1 Developing a Specification	445
18.3.2 The Rocky State	448
18.3.3 Adding a New Customer	449
18.3.4 Taking a New Order	450
18.3.5 Making a Request for Production	453
18.3.6 The Database	454
18.3.7 Implementing the AddCustomer Operation	456
18.3.8 Implementing the TakeOrder Operation.....	457
Exercises.....	463
Additional Cases	464
Chapter 19 Refinement	470
19.1 Introduction	470
19.2 The Abstract Specification.....	471
19.2.1 The class Situation	471
19.2.2 The Individual Student	472
19.2.3 Assessment	473
19.2.4 The Lecturer	473
19.3 Operations on Student Records.....	474
19.3.1 A Student Submits Some Work.....	474
19.3.2 A Student Is Awarded a Mark.....	476
19.3.3 A Mark is Amended	477
19.4 The Concrete Specification	478
19.4.1 The Tables Used	478
19.4.2 Mapping Between Representations.....	481
19.4.3 The Award Operation Re-specified	483
19.5 A Review	484
19.6 Verification	485
19.7 Verifying the Award Operation	487
19.7.1 The One-point Rule Revisited.....	487
19.7.2 Applicability	488
19.7.3 Correctness	489
19.7.4 The Initial State.....	490
19.8 The External Interface.....	491
19.9 Translating the Award ExE Schema into SQL.....	492
19.10 Summary.....	493

Table of Figures

Figure 1.1 The Great Computing Divide	2
Figure 1.2 In the bank	5
Figure 2.1 Defining a fact type	22
Figure 2.2 The hasage relation.....	24
Figure 2.3 Relation or function?	26
Figure 2.4 The drives partial function	27
Figure 2.5 The left total injection	28
Figure 2.6 The spouse partial injection.....	29
Figure 3.1 The KIDS Database	53
Figure 3.2 Set Evaluation	55
Figure 4.1 An easy merger	78
Figure 4.2 Bad and good mergers.....	79
Figure 4.3 The database anatomy.....	94
Figure 4.4 Links between tables.....	95
Figure 4.5 The select operation.....	97
Figure 4.6 The project operation	98
Figure 8.1 Merging fact types	174
Figure 8.2 Merging single-valued facts	175
Figure 8.3 Introducing uniqueness constraints.....	178
Figure 8.4 Single and many-valued fact types	182
Figure 8.5 An irreducible fact type.....	184
Figure 8.6 A uniqueness constraint on two roles.....	185
Figure 8.7 A nested fact type.....	186
Figure 8.8 Multiple nested facts	187
Figure 8.9 The final conceptual schema.....	188
Figure 8.10 Looking for nulls	192
Figure 8.11 Looking at a nested facts for nulls	194
Figure 10.1 An outline of fact-based analysis.....	222
Figure 10.2 The first-draft conceptual schema diagram	227
Figure 10.3 An irreducible fact type	230
Figure 10.4 A nested fact type	231
Figure 10.5 The final schema	232
Figure 11.1 The campus entity type	242
Figure 11.2 And now we have two!.....	243
Figure 11.3 Faculties are divided into schools.....	243
Figure 11.4 The Science Faculty is divided	243
Figure 11.5 Introducing the campus entity type.....	244
Figure 11.6 The story so far.....	245
Figure 11.7 Faculty attributes	246
Figure 11.8 Relationships may also have attributes	247
Figure 11.9 Attributes of a many-to-many relationship	247

Figure 11.10 Composite attributes.....	248
Figure 11.11 A set-valued attribute	249
Figure 11.12 Weak or dependent entity types.....	250
Figure 11.13 A recursive relationship.....	250
Figure 11.14 A hierarchical relationship.....	250
Figure 11.15 Moreton Bay University	251
Figure 11.16 Record types based on entity types (part 1).....	254
Figure 11.17 Record types based on entity types (part 2).....	255
Figure 11.18 A dependent entity	255
Figure 11.19 The many-to-many relationships	256
Figure 11.20 The One-to-many Relationships.....	257
Figure 11.21 Resolving set-valued attributes	258
Figure 11.22 The Attempt entity.....	260
Figure 11.23 Every lecturer belongs to a school	261
Figure 12.1 Existential quantification with conjunction	273
Figure 12.2 Universal quantification with implication.....	276
Figure 12.3 Implication	277
Figure 12.4 General statements	278
Figure 12.5 Equivalence.....	282
Figure 12.6 Defining new relations.....	283
Figure 12.7 Defining new total functions	284
Figure 12.8 Defining new partial functions	286
Figure 12.9 In the bank	290
Figure 14.1 Three views of the classroom.....	342
Figure 14.2 Conceptual schema diagram	344
Figure 14.3 Entity-relationship diagram.....	345
Figure 14.4 The Class Database.....	346
Figure 14.5 Abstract and concrete states.....	346
Figure 14.6 From specification to implementation.....	351
Figure 14.7 Full circle	351
Figure 15.1 Revised create table syntax	370
Figure 17.1 Modes of SQL Usage	406
Figure 17.2 Pre-processing	408
Figure 17.3 An electronic form	423
Figure 18.1 Catalog of products and prices	443
Figure 18.2 List of Stock Report.....	444
Figure 18.3 Production Request.....	444
Figure 18.4 List of Customers	445
Figure 18.5 Order Form.....	446
Figure 18.6 Before and after	447
Figure 18.7 The AddCustomer screen.....	456
Figure 18.8 The TakeOrder screen.....	458
Figure 19.1 What the user thinks	485
Figure 19.2 What the programmer thinks.....	486

Chapter 1

Introduction

1.1 Why Compute?

What *are* computers for? What is their purpose? Suppose your life depended upon coming up with a word or phrase that most accurately summed up what computing is all about. What would your answer be?

Would you say that computing is about . . .

sex?
drugs?
rock'n'roll?

No, there's not too much of that in computing.

Well then, perhaps it's about . . .

money?
power?
food?
gambling?

No, these topics are hardly ever discussed in computing magazines.

Chapter 2

Specific Facts

2.1 Introduction

Computers are *not* magical. They are *marvellous*, but they are not magical. They may be extremely fast, with computation speeds measured in millions of instructions per second. They may have huge amounts of memory, measured in billions of characters. But there is nothing happening inside them that we could not contemplate doing ourselves. We may take a lot longer; we may get bored and make mistakes, but we *must* believe that we could. We must think of the computer as doing things that we could do with pencil and paper or with a blackboard and some chalk. If we cannot do this, then we are resigned to thinking of the computer as something beyond our comprehension. As a consequence of this necessary act of faith, it is the things that *we* can express (in conversation with a friend or on a piece of paper, say) that are of importance. And, unless we are day-dreaming, these expressions have some meaning. They are attempting to say something about reality. The **sentence** is the unit of language that allows us to say things about the world in which we live. Sentences, however, come in all shapes and sizes; there are commands, questions, forecasts and opinions to name just a few. This book will focus on one particular category consisting of what are called declarative sentences or, more simply, **facts**. A declarative sentence is one that is capable of being true or false. Consider the following sentences:

Stop, in the name of love!
Big girls don't cry.
Will you still love me tomorrow?

Only one of these three is some kind of statement about the world. Only one is a representation. Only one can be added to the end of:

I declare that:

Chapter 3

Sets

3.1 Introduction

Suppose someone writes down a list of people's names and hands that list to you. Then you are asked what these people have in common.

It is fairly safe to claim that one way or another you would find something to connect these people. Even if the names were as unlikely as John, Paul, George and Ringo. You would probably feel frustrated and disappointed with yourself if you were unable to discern some common feature.

A **set** is a collection of objects, with the objects usually sharing some property. The formation of a set allows us, mentally, to gather things that seem to belong together, and to provide them with a collective being. This process of **generalization** is a means of conquering complexity. Defining a set is a way of enforcing order upon our world and because of that order we can have reasonable expectations. We anticipate certain kinds of behavior and not others.

By isolating an object and stating that this thing is a "man", for example, we accomplish two things:

1. We provide a number of properties that can be ascribed to that object – beards, beer and baldness perhaps.
2. We group this person with other men – all the people who share these properties.

Having decided that a person is a man or a woman or a singer or a computer programmer we would expect a whole range of associated behavior patterns.

3.2 Sets and Everyday Language

There are two ways to specify a set: **set extension** and **set comprehension**. These two methods form an essential part of our everyday language.

Chapter 4

Relations

4.1 Introduction

In this chapter we take a step towards the implementation of our specific facts. In previous chapters, we attempted to represent situations in reasonably natural, if formal, way. We would usually consider a person's age and a person's father to be separate facts about that person; and so, in our specification, we would probably want to treat them separately. Don't forget that the specification is a description written for *our* benefit. An implementation, however, is a description written with automation in mind. While a specification may be written with a relatively free hand, an implementation is usually required to be efficient and effective, using a minimum amount of storage space and providing an acceptable response time.

This chapter provides a continuation of the formal notions of relations and sets that were introduced in the two previous chapters. It allows us to gather these ideas in a theoretical manner before discussing their implementation in a "real-live" computer language, namely SQL.

The chapter introduces the **relational model** of data. Using this approach, facts are combined to produce larger storage structures called relations. A relational database is a cohesive collection of relations. We use the relational model (1) because it allows us to access and to manipulate facts in a relatively easy manner, and (2) because there are many commercially available database management systems that support the relational model.

4.2 Merging Facts

The idea of a relation was introduced in Chapter 2 where it was described as a set of pairs. In that chapter, relations were frequently shown in the form of a two-column table. For that reason, we might call them **binary** relations to distinguish them from the more general relations that are the subject of this chapter. But, because each binary relation corresponds to a particular type of fact, we will also refer to binary relations as **fact types** to make their origin clear. Here are two examples of these fact types that were introduced in that chapter.

Chapter 5

Introducing SQL

5.1 Introduction

In this chapter we introduce one of the most important computer languages so far developed, SQL. It represents a major departure from the languages we usually think of in connection with computer programming. These more conventional languages are primarily concerned with giving instructions to a computer. SQL is different.

SQL is, first and foremost, a **means of communication**, a means of expressing our requirements. These requirements are passed to a complex software product known as a database management system (DBMS). This software is designed to control access to and usage of the database. SQL is a means of telling the DBMS what we want done. Because the nature of the language allows us to concentrate on specifying the information to be retrieved from our database, there is a consequential load placed upon the DBMS. It must be able to determine a sufficiently rapid means of accessing the data, sufficiently rapid, that is, to satisfy our need for the data.

SQL is an acronym for **Structured Query Language**, and the key word is **query**. This word is to be taken in a more general sense than simply "retrieval". The central idea in SQL is that of identifying the portion of the database that interests you. Having done that, you may apply some operation to that portion: you may display it, you may update it, or you may delete it.

This chapter is intended to provide a brief look at some of the language's major features. These features are divided into four groups concerned with:

- **database definition**, whereby the major components of the database may be defined, modified or discarded;
- **database retrieval**, whereby the portion of the database that meets certain conditions may be identified and examined;
- **database manipulation**, whereby some part of the database may be extended, updated or deleted;

Chapter 6

SQL Retrieval

6.1 Introduction

This chapter contains a series of examples of database retrieval using SQL. The examples attempt to show the basic retrieval capabilities of the language.

There are three basic ways in which information may be extracted or derived from a table. These relate to the ways that we ourselves might extract information presented to us in tabular form.

Sometimes we are interested in detailed information. We scan down particular columns looking for values that interest us, stopping when we find such a value. Then we will examine the rest of the row upon which we found the value. This is how people look up telephone numbers or exam results or sports results or a timetable. The search operation will be repeated until we have, for example, noted our own exam results and those of our friends.

There is another kind of retrieval. This kind is performed when, essentially, we are looking for one particular value. The value may be one that can be extracted from the table, or it may be a derived value. The situations when we scan a table in this way are, for example, when looking for the lowest mark in an exam or the total number of people who passed or the time of the last train or bus.

The third kind of retrieval is the kind performed when we want to compare one group of figures with another. Did chemistry students perform better than computing students? Are there more trains to town than buses?

These are the basic means of retrieval offered by SQL. There is nothing performed by SQL that we could not contemplate doing ourselves. SQL is a language, after all; it is a means of expressing our wishes.

All examples are based on the `SUBJECT` database introduced in Chapter 4. This database contains three tables:

- `Students`, which contains the names of students enrolled in the single subject offered;
- `Assess`, which contains details of assessment involved in the subject; and

Chapter 7

SQL Modularization

7.1 Introduction

In computing, a **module** is the name we give to an item of work. A program module is a discrete component of that program. It performs a particular task, such as finding the minimum of a set of numbers. All the various modules of a large program are put together in such a way as to achieve the program's overall goal.

This process of conquering complexity is sometimes called **modularization**. Using this technique a complex task may be reduced to a number of relatively simpler tasks. Suitable program modules are then built to accomplish each of these tasks.

This chapter is concerned with how the fundamental query building methods of Chapter 6 may be combined in different ways. In creating these more complex queries we can answer more complex questions.

Three mechanisms are discussed. They are:

- query **nesting** whereby the results of one query are fed into another;
- the **union** operator which allows the results of two or more queries to be merged to produce a single result table; and
- the **view** which allows the results of a query to be given a name and subsequently treated as just another database table.

7.2 Query Nesting

The first kind of query modularization considered in this chapter is query nesting. This involves passing results from one query directly into another. Thus two queries may be executed, one after the other, with no "manual" intervention required.

Example 7.1 What is the Id of the student who got the lowest mark in the first assignment? We could issue the following command:

Chapter 8

Facts and Relations

8.1 Introduction

A **fact** is a declarative sentence; that is, it is a statement which may be either true or false. It describes a particular relationship between two or more things or **entities**; for example:

Billy Connolly was born in Scotland.

We use computers when we have lots of similar facts to remember.

Bill Cosby was born in the USA.
John Cleese was born in England.
Barry Humphries was born in Australia.
: : : : :

When we recognize that certain facts are similar, we can generalize them into a **fact type** which is a relationship between two types of entity rather than between individual entities. In this case the relationship is between people and countries. Or is it? Perhaps it's between comedians and countries, or between men and countries? To be more certain we need to investigate the *universe of discourse* or UoD which is simply the situation that we intend representing in our information system. In attempting to design the most appropriate database structure for a given situation, we need to know the kinds of things that are involved and the kinds of facts that relate them.

Fact types are not stored individually; rather, they are embedded within **relations** with each relation dedicated to representing a fixed number of fact types. This chapter looks at some of the problems we face when deciding where to place a fact type when designing a database. We will find that each fact type may be merged or grouped only with certain others. Some we will be unable to merge; they must remain on their own. We will use **conceptual schema diagrams** to help achieve this grouping. These diagrams are used to depict the knowledge we need to design a database.

Chapter 9

Uncovering Facts

9.1 Introduction

Suppose we are required to design a database to support a new information system. In the preceding chapter some rules were formulated regarding which facts may and which may not be merged into relations. Once we have, in front of us, the kinds of facts that are to be stored in the database then it is a relatively mechanical process to follow these rules and to arrive at a design for the database.

Unfortunately, this information is rarely presented to us in a neatly packaged and labeled way. In other words, the basic facts types do not usually show themselves clearly and obviously. We, the designers, must identify them.

The people who are going to use this new system will want the computer to extract information from the database, to sort it, to merge it with other information, to summarize it, and so on. They are most unlikely to be interested in receiving long lists of quite trivial facts. They have sophisticated ideas of how the organization works and may want these ideas reflected in complex reports.

A report is simply a view of the organization. This chapter introduces a language that may be used to describe the structure of such views. From these descriptions, the underlying simple facts may be uncovered.

9.2 Defining Syntax

The **syntax** of a language is a set of rules that govern exactly what may be said in that language. This definition applies as much to programming languages as it does to any other kind.

The language to be presented in this chapter is a special language used to describe the syntax of programming languages. It is the language in which we write the rules of syntax. The syntax of SQL, for example, tells us that "Select * From Students" is legal SQL whereas "From Students Select *" is not. Syntax is concerned with the superficial

Chapter 10

Fact-based Analysis

10.1 Introduction

This chapter is presented as a worked example in a technique which we will call **fact-based analysis**. This is a way of designing a relational database. In particular, it is concerned with developing a design that guarantees that in any resulting database each fact is stored *just once*. Here are the stages that we follow.

1. Uncover the relevant entity types and the fact types that join them.
In this step, we apply the techniques of Chapter 9 to find the relevant elementary fact types.
2. Look for any uniqueness constraints involved in each fact type.
In this step, we apply the question and answer technique of Section 8.6 to decide whether a fact type is a many-to-many, a many-to-one or a one-to-one relationship.
3. Construct record types by merging fact types, where appropriate.
In this step, we merge fact type according to the rule that permits the merging of single-valued facts about the same kind of thing.
4. Decide which attributes may be null.
In this step we process each record type in turn, examining each non-key attribute of that record. Those that may contain null values are flagged.
5. Define the database.
In this step, we provide an outline of the database.
6. Review the design.
Finally, we should check that the database design is satisfactory. Has any computable or derivable information slipped through into our design? Using SQL, can the major views be reproduced with this design?

Chapter 11

Entity-relationship Modeling

11.1 Introduction

Entity-relationship modeling is a very popular method for designing databases. ER modeling, as it is often called, may be described as a **top-down** approach in that it encourages to look at the “big picture” first. We begin by describing the world in terms of **entity** types that are **related** to one another in various ways. We may then refine that picture to show the **attributes** of each entity type. Thus we start by looking for the major kinds of things that populate the situation to be modeled. These entity types will give rise, eventually, to the major relations in our database. In a hospital situation, for example, the entity types might be:

- patients
- wards
- beds
- surgeons
- nurses

We then establish any relationships that exist between these entity types, such as:

- Patients are operated on by surgeons.
- Patients are located in beds.
- Beds are placed in wards.
- Nurses are allocated to wards.

Chapter 12

Knowledge

12.1 Introduction

This chapter is an introduction to the **Z Notation** which is a language that allows us to express our understanding of any given situation in a concise and precise way. In Chapter 1, it was suggested that if we write down all that we know about something, then our statements may be divided into (1) simple specific statements that were termed **facts** and (2) more general statements that were termed **knowledge**. This chapter is concerned with the more general statements, that is, it is about knowledge representation. Z is the language we will use to express that knowledge.

The generality of knowledge is achieved by the statements involved saying something about whole classes or sets of objects. Such knowledge will eventually be encoded as computer programs, so Z will be used to **specify** these programs. The resulting specification will then be **implemented** using a programming language. It is important, however, to realize that the implementation is yet another description of the *same* situation that was portrayed in the original specification. The implementation is a kind of re-specification of that situation, this time written in a specification language that the computer can follow and obey or execute. This implementation is, in effect, an **executable** specification; that is, it will be written using a programming language such as C or COBOL or SQL or some combination of these.

The Z Notation is a particular style of writing two mathematical languages, **set theory** and **predicate calculus**. Z is an amalgam of these. Most of the ideas relating to set construction and manipulation were discussed in Chapter 3. In this chapter, we will see how to write set expressions in Z. Mostly, this involves using a special symbol rather than a word. For example, Z uses the symbol \cup for set union whereas in Chapter 3 and in the coverage of SQL that followed, we would have used the word `union` instead.

The second element of Z is predicate calculus, and many of the ideas behind this theory have also been discussed previously. This was done in Chapter 2 where we discussed basic simple facts and their construction. In Z, basic facts are also constructed in the same way,

Chapter 13

The Knowledge Base

13.1 Introduction

Documentation is one of the most disliked features of computing. This is rather unfortunate since program documentation is where we store our knowledge in its most (human) readable form. As an example, consider a simple rule stating that a customer's current balance must not be allowed to exceed his or her credit limit. A rule like this is typically specified using a program design technique such as pseudocode or a decision table. The rule might then be encoded within a COBOL program. Thus the program becomes the rule enforcer. The rule is specified in pseudocode and implemented in COBOL.

What happens next? If we are honest with ourselves we know that from now on all attention turns to the program code; the specification takes on a secondary role of documentation. Changes to the rule occur, such as amendments, extensions, special cases and so on. These are implemented directly in the program and the documentation becomes increasingly obsolete, simply confirming the programmer's prejudice against documentation. The result is that the database is encapsulated by a collection of programs, with each program implementing any number of undocumented rules. The database starts to suffer from hardening of its arteries. Organizational knowledge becomes buried in programs. When the system is to be replaced, all this knowledge must be rediscovered by the next generation of systems developers.

One answer to this problem is to keep the documentation up to date and to make the programs subordinate to this documentation. In doing so, we would move towards a more evolutionary style of systems development. Maybe the term *specification* should be discarded. In the minds of many programmers, any specification is a disposable means to an end; the end being an executable program. A better approach is to consider that the programs constitute a knowledge-base of some kind; and that knowledge needs to be expressed in at least two forms: (1) in a way that humans can understand; and (2) in a way that machines can execute. Both forms are necessary; the first for us, and the second for the machines. A knowledge base of the kind being proposed is simply continuously updated

Chapter 14

From Specification to Implementation

14.1 Introduction

In this chapter we will look at how the various schemas that form a Z specification might be put together. The situation to be modeled is that of a class of students who are studying a particular subject. The specification covers such typical activities as students being enrolled, being awarded marks, having marks amended and, hard to believe, students dropping the subject. We will begin by introducing a **state schema** which provides a static picture of the classroom. Based on that picture we specify a number of **operation schemas** which describe the ways in which the classroom may change. Then we return to the state schema and discuss how it might be developed. Finally, we discuss the relationship between this **specification**, which consists of a state schema and a number of operation schemas, and the **implementation**, which consists of a database and a number of programs.

14.2 The State Schema

Suppose we describe the classroom in the following way.

<i>Class</i>
<i>students</i> : Set of Person <i>last</i> : Person \leftrightarrow Name <i>mark</i> : Person \leftrightarrow 0..100
<i>dom last</i> = <i>students</i> <i>dom mark</i> \subseteq <i>students</i>

Chapter 15

Database Definition in SQL

15.1 Introduction

This chapter describes how to define the major objects that may appear in an SQL database.

- There are **tables** without which the database would be empty. These are sometimes referred to as **base** tables.
- There are **views** which define **virtual** tables.
- There are **indexes** which enable the DBMS to respond to queries within an acceptable period of time.

The word definition is used in the general sense of describing or delimiting the properties of these objects. So, this chapter will discuss the creation, alteration and, in the extreme case, removal of these properties.

Information about the properties of database objects is stored in the **system catalog** or dictionary. The catalog itself takes the form of a set of tables which we may examine ourselves using SQL. It forms a relational database that resides alongside our own.

15.2 Tables

15.2.1 Table Creation

A new table is introduced into the database by means of the `create table` statement which, in its simplest form, only requires that we name the table and then name and provide a datatype for each column in the table.

Chapter 16

Database Manipulation in SQL

16.1 Introduction

In this chapter we look at the three data manipulation commands of SQL.

- The `insert` statement enables one or more new rows to be added to a table.
- The `update` statement allows one or more existing rows to be modified in some way.
- The `delete` statement allows one or more existing rows to be removed from a table.

These statements allow the user's model (or information system) to be *manipulated* so as to reflect changes that have occurred in the user's environment.

16.2 Adding New Rows

The `insert` statement allows one or more rows to be inserted into a table.

There are two forms of the statement, one that allows us to insert a single row, and one that allows the insertion of a set of rows.

16.2.1 Single Row Insert

This form allows one new row to be placed in the table specified.

Example 16.1 We want to add a new student number 999, with the name Meg Murphy. To do this we construct the row with a `values` clause and then insert that into the `Students` table.

```
Insert
Into Students
Values(999, 'Meg', 'Murphy')
```

Chapter 17

Application Programming

17.1 Introduction

This chapter is concerned with how we program our information systems or **application**. The system will, typically, consist of a database and a set of programs. This split reflects that division first discussed in Chapter 1.

- The database contains simple specific facts concerning the organisation.
- The programs contain more general statements or knowledge.

The programs themselves can be divided into two groups according to the kind of knowledge they encode. Some of them are **report** programs. They inspect the database, make calculations using the data retrieved, and report on the results of their calculations. The other group of programs **process transactions**, that is, they allow events and changes in the real world to be represented in the database. Each of these latter programs will be dedicated to handling one particular kind of event.

A large part of the work involved in operating an organisational information system is concerned with this latter group of programs. The information system *must* be able to record changes in the organisation's environment and circumstances.

This chapter is mainly concerned with such transaction processing. It examines how the operation schemas that are used to *specify* the transactions are *implemented* as SQL programs.

Each operation schema could be divided into two separate sets of conditions.

- There are the pre-conditions which, collectively, state what conditions must apply before some event may truly be said to have occurred.
- There are the post-conditions which, on the basis of the pre-conditions being satisfied, say how the situation changes as a result of this event.

Chapter 18

Case Studies

18.1 Introduction

In this chapter we will look at two data processing situations.

- The first situation involves deriving a compact yet complex report or view from some simple data.
- The second situation involves monitoring the handling of orders made on a small manufacturer.

For both situations, we will (1) present an informal introduction, (2) describe it formally, and (3) look at an implementation in SQL.

18.2 The League Table

18.2.1 Introduction

Last year, a number of rugby clubs agreed to take part in a competition to decide the best team in the district. The competition is to take place over a number of weeks. Every week there will be a round of matches with each team playing one match per round. There are six clubs altogether, so each round will involve three games. In the first round, the results were as follows:

Round 1 Results			
Home Team		Away Team	
Wiseacres	12	Shinhackers	8
Rosewell	8	Witsend	20
Rovers	5	Jeeps	5

The convention about presenting results is that the home team and its score are given first and the away team and its score second. Each team will play all the others twice, once at

Chapter 19

Refinement

19.1 Introduction

This chapter is about how we implement our specification, that is, it is about how we turn it into a collection of computer programs operating upon a database. The situation we want to reach is one where we will have two quite distinct pictures *of the same situation*.

We start with one picture, the one provided by the specification. This will be stated or expressed in a language that tries to describe the situation as *we* see it. The other picture is a re-statement of that same situation; but this time the language used is ambivalent. It can be taken as just another way of perceiving the problem, but it can also be thought of as providing instructions to a machine in order for that machine to create an animated equivalent of the original specification. It is a version that is *executable* by the machine. In other words we have made the original problem tractable to information technology.

This use of different forms of language is not restricted to computing. We would give a stranger to town instructions expressed differently from those given to an obvious local asking directions. We would talk to the local in terms of shared knowledge such as familiar streets and landmarks. Conversely, we would talk to the stranger in physical terms – “turn left”, “straight ahead for 2 km”, “third on the left” and so on.

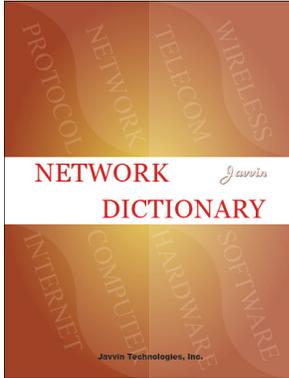
In this chapter we will take a relatively abstract specification such as shown in Chapter 14 and show how to map that to another specification this time expressed in the relational calculus or tuple oriented set comprehension of Chapter 4. This language is the basis for SQL and it will be assumed that the transformation to SQL is straightforward.

This chapter provides a worked example of how to move, formally, from a specification to its implementation. The technique used is *data refinement*, and here it is used on a database system.

A small situation is described along with some of the events that might impinge upon it. The description or specification is written using the Z notation. The intention is to explain the situation as the user sees it.

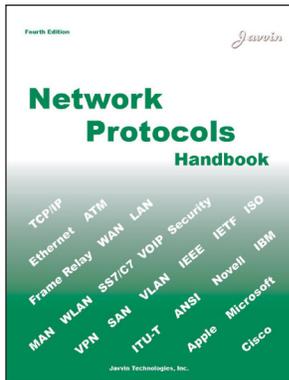
That same situation is then recast in terms of tables and the events in terms of operations

Javvin Networking Technology Series



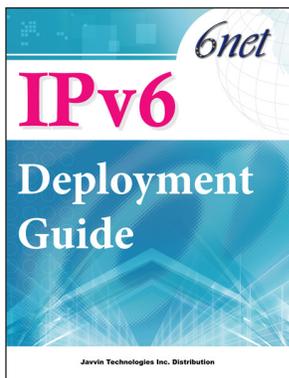
Network Dictionary

Networking, Internet, telecom, wireless, computer, hardware and software - multiple dictionaries in one. A “Must have” reference for IT/Networking professionals and students!



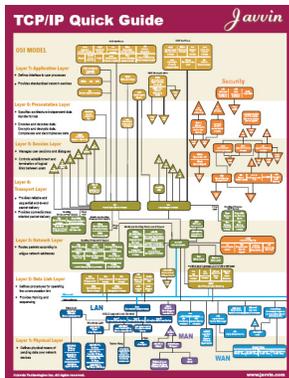
Network Protocols Handbook

Fully explains and reviews all active protocols. Illustrates latest networking technologies.



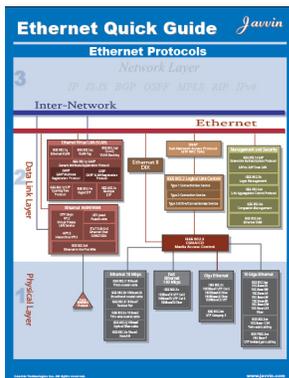
IPv6 Deployment Guide

IPv6 is replacing IPv4 to dominate the networking world. This deployment guide will enable you to fully harness the power of IPv6. A “Must have” reference for IT/Networking professionals and students!



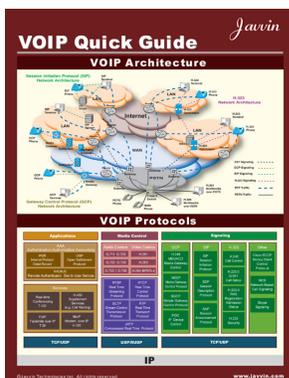
TCP/IP Quick Guide

Practical TCP/IP information extracted from hundreds of pages of TCP/IP books. A comprehensive and clear map of all TCP/IP protocols in OSI 7 layers model. A portable tool for you to carry, insert into a folder or put on your desk.



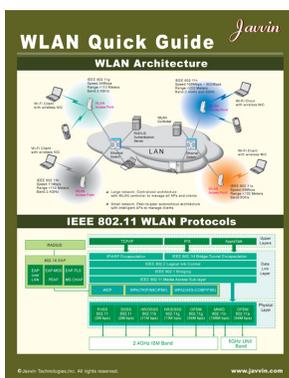
Ethernet Quick Guide

Practical Ethernet information extracted from hundreds of pages of Ethernet books. A comprehensive and clear map of all Ethernet protocols. A portable tool for you to carry, insert into a folder or put on your desk.



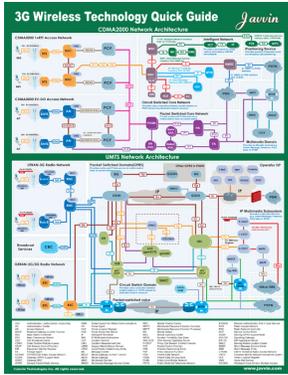
VOIP Quick Guide

All must known VOIP technologies included in this comprehensive yet portable quick reference.



WLAN (WiFi) Quick Guide

A comprehensive quick reference to assist you in WiFi WLAN implementation, learning and operation.



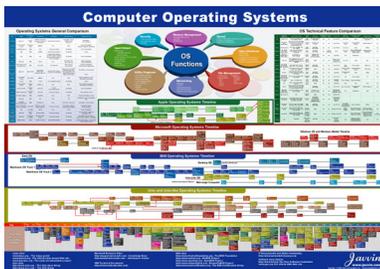
3G Wireless Tech Quick Guide

Highlights the third generation wireless technologies in one portable quick guide.



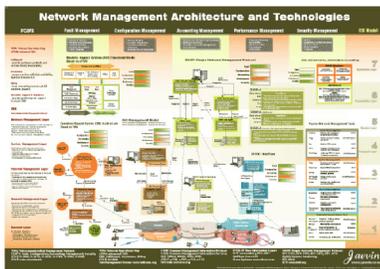
Windows Vista Security Quick Guide

All you must to know about Windows Vista Security on this handy quick guide...useful for all Windows Vista users!



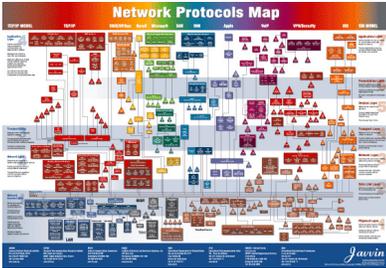
Computer Operating Systems (OS) Poster

Illustrates the computer operating systems now and their evolution path over the past 60 years.



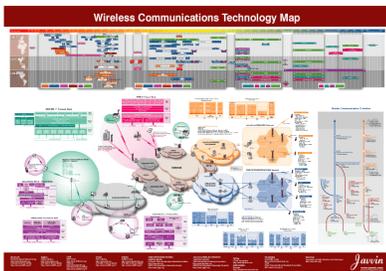
Network Management Architecture and Technology Map

All network management architecture and technologies for both telecom and data communications displayed on one chart.



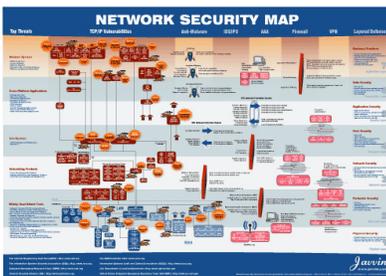
The Network Protocols Map Poster

All network protocols illustrated on one Chart. A “must have” for all networking, IT and Telecom professionals and students.



Wireless Communications Technology Map Poster

All major wireless technologies displayed in one chart: WLAN, WiMAX (WMAN), WPAN and mobile wireless technologies (WWAN)...



Network Security Map

All you must know about network security on one chart! A unique gift for yourself, your colleagues, partners and customers.



www.NetworkDictionary.com

Free networking technology library, comprehensive network protocol and network security knowledge, telecom encyclopedia, computer hardware and software terms and white-papers.

A site for you to learn and share knowledge, ask experts, write blogs and get connected with peers.

Information Modeling

ISBN: 978-1-60267-015-0



9 781602 670150 >