

Second Edition

DB2

Design & Development Guide

Second Edition

DB2

Design & Development Guide

Gabrielle Wiorkowski
David Kulk



ADDISON-WESLEY PUBLISHING COMPANY, INC.

Reading, Massachusetts • Menlo Park, California • New York
Don Mills, Ontario • Wokingham, England • Amsterdam • Bonn
Sydney • Singapore • Tokyo • Madrid • San Juan

Library of Congress Cataloging-in-Publication Data

Wiorkowski, Gabrielle.

DB2: design and development guide / Gabrielle Wiorkowski, David Kull. —2nd ed.

p. cm.

ISBN 0-201-50735-8

1. Data base management. 2. IBM Database 2 (Computer system)

I. Kull, David. II. Title.

QA76.9.D3W58 1990

005.75'65—dc20

89-17706

CIP

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and Addison-Wesley was aware of a trademark claim, the designations have been printed in caps or initial caps.

The programs and applications presented in this book have been included for their instructional value. They have been tested with care, but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations, nor does it accept any liabilities with respect to the programs or applications.

Reprinted with corrections June, 1990

Copyright © 1990, 1988 by Addison-Wesley Publishing Company, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

BCDEFGHIJ-MA-943210

Preface

DB2 has matured considerably in the year and a half since this book was first published. In that time IBM added support, if only partial, for distributed database management and referential integrity enforcement and made a number of other significant enhancements. DB2 users also matured in their use of the product. Many now are no longer learning DB2 as much as they are discovering its nuances. For that reason, in addition to updating the book to cover features delivered in the newer releases—up through Version 2.2—we have also added considerable detail to previously covered subjects. Along with new chapters on referential integrity and distributed database management there are expanded sections on composite indexes, date and time calculations, mass updates in batch programs, data type conversions, and so forth. We hope that this additional depth will be useful to veteran developers without impeding the learning of beginners.

The book still attempts to address an essential condition of DB2. Its power and the ease with which it can be used mask a number of subtleties that must be mastered before it can be used to its potential. When implemented fully, E.F. Codd's relational database model, upon which DB2 is based, will free developers and users from many of the concerns addressed in this book. The model's conceptual elegance has already opened the world of database management to more data-processing and business professionals than ever before. Although DB2 implements the model more completely than other database management systems, however, its failure to adhere to the model fully means that users must approach the product with considerable knowledge. This edition attempts to provide an introduction to DB2 for newcomers to database management and an indepth exploration for

DB2 veterans—information both groups need to get the best possible performance from their applications.

The book is meant to operate on several levels. It is designed as a text for learning about database management, as a guide to DB2, and as a resource for its continued use. When a book attempts to cover such wide territory, its structure is important. Readers interested in a resource will probably not need the background information required by a learner. We dealt with this problem in two ways. First we provided some of the background information in separate chapters. Then we combined teaching about basic principles with detailed information about the specific subject to which the principles apply as unobtrusively as we could.

For example, two chapters focus on SQL. Chapter 7 describes how to use SQL's data manipulation facilities at a beginner's level. Those familiar with the language may want to skip this chapter or use it for reference or review. Chapter 9 describes how to use DB2's implementation of SQL for optimal performance, providing crucial information for both SQL learners and experts. Although knowing how to normalize a database design is essential to the use of DB2, those already familiar with normalization should have to visit Chapter 2 on that subject only briefly.

We also felt that the blending of underlying principles with details of DB2's use is not only an effective approach to teaching the principles but also a spur to the proper use of DB2. Although users need not know the reason for a prescribed action in order to take it, most feel more comfortable when they do know. Understanding the principles allows users to apply them to problems they confront that may not be specifically addressed in this book or in any other. We have assumed that readers have a minimal knowledge of programming. We trust that parenthetical definitions of basic terms and occasional longer expositions on basic material will not hinder more experienced professionals.

Because we believe the days of DB2 Version 1 are rapidly slipping into data processing history, we have made little attempt to identify differences between the earlier and current releases. In a few cases, we did contrast the old with the new as a way of illustrating a point. In general, however, we have assumed users are moving to current releases and have little interest in outmoded facilities.

ACKNOWLEDGMENTS

DB2 is a relatively new product. Few people have had a chance to gain a deep understanding of its concepts and facilities. In preparing this book for publication, therefore, we were fortunate to have the advice and support of the foremost experts in the field. Thanks in particular to Marilyn Bohl

and Barbara von Halle for their valuable assistance with this second edition and to Chris Loosley and Colin White. Their suggestions led to many improvements. Thanks also to John Wiorkowski, who was particularly helpful in reviewing and helping to shape some of the book's more mathematically intense portions. We are also indebted to Chris Date for sharing his Suppliers, Parts, and Jobs tables, which are used in most of this book's examples, and for his advice and support during the project. Finally we wish to thank Sharon Weinberg and E.F. Codd, whose guidance helped shape both the structure and content of this book. And finally, thanks to the editorial and support staff at Addison-Wesley for their assistance in preparing this edition.

Gabrielle Wiorkowski
David Kull

Contents

CHAPTER 1 / Concepts and Components	1
1.1 Introduction	1
1.2 DB2's Key Components	3
1.3 An Introduction to SQL	6
1.4 Introduction to Referential Integrity	8
1.5 Table Naming	10
1.6 A Word on Performance	10
1.7 Summary	11
Exercises	12
 CHAPTER 2 / Review of Normalization	 13
2.1 Introduction	13
2.2 The First Normal Form	15
2.3 The Key Issue	17
2.4 The Second Normal Form	19
2.5 The Third Normal Form	20
2.6 The Fourth Normal Form	21
2.7 The Fifth Normal Form	24
2.8 Minimizing Joins	25
2.9 Summary	28
Exercise	28
 CHAPTER 3 / Creating Physical Objects	 31
3.1 Introduction	31
3.2 Databases	32

3.3 Tablespace and Indexspace.....	36
3.4 Free Space.....	42
3.5 Locksize.....	48
3.6 Segmented Tablespaces.....	50
3.7 Additional Tablespace Parameters.....	54
3.8 Create Index.....	57
3.9 Partitioning and Performance.....	60
3.10 Summary.....	64
Exercises.....	64
 CHAPTER 4 / Creating Tables and Views	67
4.1 Introduction.....	67
4.2 Creating Tables.....	67
4.3 Views.....	83
4.4 Dropping Objects.....	91
4.5 Summary.....	91
Exercises.....	91
 CHAPTER 5 / Indexing for Performance	93
5.1 Introduction.....	93
5.2 The B-Tree.....	93
5.3 Balancing the B-Tree.....	97
5.4 Selecting Columns to Index.....	99
5.5 Use of Multiple Indexes.....	105
5.6 The Value of Clustering.....	106
5.7 Index Cardinality and Composite Indexes.....	109
5.8 Redundant Index Cost.....	115
5.9 Maintaining Clustering.....	118
5.10 Index Monitoring.....	120
5.11 Generating Primary Keys.....	121
5.12 Summary.....	122
Exercises.....	123
 CHAPTER 6 / Concurrency Control	125
6.1 Introduction.....	125
6.2 Transactions, Commits, Rollbacks.....	126
6.3 Lock Sizes and Types.....	130
6.4 Isolation Level.....	133
6.5 Intent Locks.....	136
6.6 Acquire and Release Parameters.....	137

6.7 Explicit Lock Statements.....	142
6.8 Summary	143
Exercises.....	143
 CHAPTER 7 / Using SQL for Data Manipulation	 145
7.1 Introduction.....	145
7.2 Select Statements	147
7.3 Negating a Predicate.....	151
7.4 Aggregates and Other Built-In Functions	151
7.5 Date and Time Calculations	156
7.6 Selecting by Groups.....	161
7.7 Sorting the Answer Set.....	163
7.8 Subselects.....	164
7.9 Joins	167
7.10 Union.....	171
7.11 Outer Joins.....	172
7.12 Intersection and Difference	174
7.13 Inserting, Updating, or Deleting Data	176
7.14 Summary.....	178
Exercises	178
 CHAPTER 8 / Program Development	 179
8.1 Introduction.....	179
8.2 Application Preparation and Execution	179
8.3 Static SQL.....	182
8.4 Restartable Batch Programs	197
8.5 The SQL Communications Area.....	201
8.6 DB2 Interactive	205
8.7 The DSN Processor.....	215
8.8 Dynamic SQL	218
8.9 Sample Host-Language Programs.....	222
8.10 Summary.....	222
Exercises	222
 CHAPTER 9 / Programming for Performance	 223
9.1 Introduction.....	223
9.2 Efficient Joins.....	224
9.3 Joins versus Correlated Subselects	229
9.4 Incomplete Statements	237
9.5 Efficient Updating.....	239

9.6	Joining a Table with Itself	242
9.7	Functions Requiring Ordered Rows.....	242
9.8	Indexes and Negations	245
9.9	Stage 1 versus Stage 2 Predicates.....	246
9.10	The BETWEEN Operator	251
9.11	Managing a Skewed Distribution of Data.....	252
9.12	Order of Predicates	253
9.13	Capping the Costs	256
9.14	Use of Efficient Views	258
9.15	Indexes Are Not Always Best	258
9.16	Performance Results.....	259
9.17	Summary.....	261
	Exercises	261
CHAPTER 10 / The Optimizer		263
10.1	Introduction.....	263
10.2	Parsing	268
10.3	Optimization.....	268
10.4	Code Generation.....	276
10.5	Execution	276
10.6	Fooling the Optimizer	278
10.7	Summary.....	287
	Exercises	287
CHAPTER 11 / Referential Integrity		289
11.1	Introduction.....	289
11.2	The Rules	291
11.3	DB2's Referential Integrity Support	292
11.4	Data Manipulation Constraints	300
11.5	Application Enforced Integrity.....	307
11.6	Tracking Integrity Rules	310
11.7	Summary.....	313
	Exercises	313
CHAPTER 12 / The EXPLAIN Command		315
12.1	Introduction.....	315
12.2	Requesting the Explanation.....	316
12.3	Preparation for EXPLAIN	317
12.4	Analyzing the Plan Table.....	320
12.5	Sample Explanations.....	323

12.6 Summary.....	336
Exercises	336
 CHAPTER 13 / Application Monitoring	 337
13.1 Introduction.....	337
13.2 Querying the Catalog.....	337
13.3 Sample Queries Related to Performance	343
13.4 When to Reorganize	353
13.5 Space Management.....	356
13.6 Tracking Referential Integrity Structures	360
13.7 Determining Function and Object Privileges	361
13.8 Documentation.....	363
13.9 Summary.....	365
Exercises	365
 CHAPTER 14 / Security Control	 367
14.1 Introduction	367
14.2 The System Administrator	370
14.3 Implicit Privileges	370
14.4 GRANT and REVOKE Statements.....	370
14.5 Granting and Revoking Table Privileges.....	371
14.6 Subsystem Privileges	373
14.7 Database Privileges	374
14.8 Use and Plan Privileges	374
14.9 Revoke Cascades	375
14.10 The AUTHID.....	376
14.11 Security in Production.....	379
14.12 Restricting Access to Parts of Tables	380
14.13 Summary	383
Exercises.....	384
 CHAPTER 15 / Utilities	 385
15.1 Introduction.....	385
15.2 The Load Utility	386
15.3 Loading Referential Integrity Structures.....	400
15.4 RUNSTATS.....	405
15.5 Reorganization	409
15.6 Copy, Mergecopy, and Recover	411
15.7 Recovering Referential Structures	414
15.8 Summary.....	420
Exercises	420

CHAPTER 16 / Distributed Database	423
16.1 Introduction.....	423
16.2 Three Types of Distributed Capability	424
16.3 Distributed Request	425
16.4 Location Transparency	426
16.5 Local Autonomy	427
16.6 Security.....	430
16.7 Program Development in a Distributed Environment.....	431
16.8 Distributed Enhancements.....	433
16.9 Summary.....	433
Exercises	433
 APPENDIX A / Tablespace Size Estimations	 435
 APPENDIX B / Estimating Index Sizes	 439
 APPENDIX C / Using SPUFI	 443
 Answers to Exercises	 447
 Index	 457

Concepts and Components

1.1 INTRODUCTION

A database management system (DBMS) physically stores information in electromagnetic form and makes it available for sharing by users or programs. The DBMS must provide ways for developers to define the data to the system and for users or programs to retrieve or change them. A DBMS must also provide facilities for security, recovery, and other tasks dictated by practical necessity. And it should relieve users of the work of maintaining the information's consistency throughout the database as parts of it are changed. But the data definition, physical storage, and access capabilities are the essential business of a DBMS.

The database's physical structure, constrained by requirements of storage devices and input/output processes, must be relatively fixed and usually very different from the structure of the information as database users, including programmers, think of it. The DBMS translates a data request—ideally made in the user's terms—into a search through the physical structure to fulfill the request. To be as useful as possible, the DBMS should make it easy for users to formulate their requests, requiring them to know little or nothing about the database's physical structure. The relational model, proposed by E. F. Codd at IBM in 1969, frees users from having to know the database structure or refer to it in their database requests. This is a valuable advance over the design of earlier DBMSs, which require users

to include details of the physical database structure in their requests for data.

DB2 and the Relational Model

The relational database model is the foundation upon which DB2 has been built and in fact provides its reason for being. Because of its advantages, the relational model has become the technology of choice for DBMSs. In 1981, IBM announced SQL/DS, a DBMS based on the relational model for computers running under the DOS/VSE and later the VM/CMS operating systems. Other companies have also based their DBMS products on the model. But IBM's announcement in 1983 of DB2 for computers running under MVS, its most advanced operating system for large computers, gave the movement toward the relational model perhaps its greatest impetus.

The model provides an uncomplicated, intuitive way for developers and users to think about and work with the information the DBMS manages. The model's basic elements are tables in which columns represent things and the attributes that describe them and rows represent specific instances of the things described. In Fig. 1.1, for example, the columns show that the table contains information about employees—their names, ages, project assignments, and salaries. Each row provides that information for a given employee.

The model also provides for operators for generating new tables from old, which is the way users manipulate the database and retrieve information from it. The language providing these operators for DB2 is SQL (structured query language), which has become the standard for all relational DBMSs. SQL also provides for the definition and control of DB2 databases.

EMPLOYEE TABLE

EMPL_NUMBER	NAME	AGE	SALARY	ASSIGNMENT
29145	SMITH, J	55	45000	BOOSTER
36201	THOMAS, H	29	32000	BOOSTER
12596	GEORGE, R	46	94000	ACCOUNTING
42578	SMITH, T	32	50000	DESIGN
23820	JONES, L	34	25000	REENTRY
26899	RALPH, T	50	36000	ACCOUNTING
41020	PETERS, J	41	26000	DESIGN

Fig. 1.1 A relational table

The tables and operators are the basic elements of the relational model. The model includes many more elements, of course, many relating to the need for the DBMS to maintain various kinds of logical consistency as the data are manipulated. No current DBMS, including DB2, fully implements the model as Codd defined it, although IBM and other vendors have said they are working toward it.

We will cover SQL's data-manipulation capabilities in detail in two chapters and provide two brief introductory examples in this chapter. We will also review normalization, the database design method that is closely related to the relational model. And although a detailed examination of the relational model is beyond this book's scope, we will visit a number of its elements in the discussion of a variety of DB2 features.

Although this book provides a grounding in the relational model, its primary purpose is to describe how to employ effectively DB2's implementation of the relational model, including SQL. Our examinations will cover facilities and techniques for creating and controlling the use of databases, loading and backing up data, programming DB2 applications, and monitoring and tuning for performance.

1.2 DB2'S KEY COMPONENTS

DB2 cooperates with—*attaches to* is the technical term—any of three MVS subsystem environments: IMS, CICS, and TSO. These subsystems cooperate with DB2 facilities to provide such services as data communications and control of transactions, which are groups of database operations that must be coordinated to avoid the introduction of errors. CICS is a teleprocessing monitor, a program for controlling online transactions—those that execute as they are entered from a terminal—allowing users to interact with the computer. IMS/DB/DC is a well-established nonrelational DBMS, which includes a teleprocessing monitor. DB2 can use CICS only for online applications, but DB2 transactions running under CICS and IMS may access DB2 and IMS databases simultaneously (Fig. 1.2). A fourth attach, the call attach facility, allows an application to interact with DB2 without the help of the three subsystems, leaving many transaction management activities to the developer. TSO also contains a teleprocessing monitor that can be used by DB2. DB2 applications running under TSO may be online or batch, and they may not access IMS databases:

DB2 provides two primary online facilities: Query Management Facility (QMF) under the call attach facility and DB2 Interactive (DB2I) under TSO. QMF, an optional product, is designed primarily to help novices or occasional users formulate database requests and format reports. It has also proved useful for testing during application development. DB2I comes with

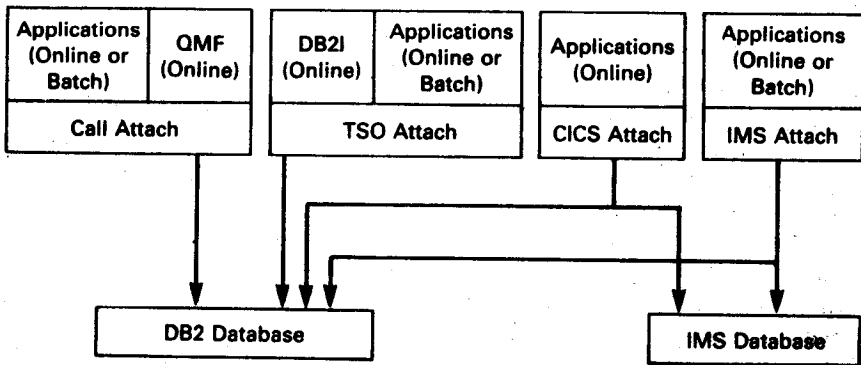


Fig. 1.2 Subsystem attaches

DB2 and is directed primarily at allowing data processing professionals to enter SQL statements interactively through a component called SPUFI and at helping them prepare programs and utilities for execution.

Detailed knowledge of the subsystem environments is not required, however, to understand DB2 and develop databases and applications for it. Nor is it necessary for developers to have a detailed understanding of DB2's internal components. Subsequent chapters will present the necessary details during discussions of how to develop applications using DB2. The following overview of components provides a groundwork for those discussions.

Bind

DB2 translates database requests into code for fulfilling them through a process called *bind*, illustrated in Fig. 1.3. Bind reads and analyzes an SQL statement, determines an access path through the database's physical structure to the data in question, and generates machine code calls for locating and acting on the data, called an *application plan*. The access path depends on a number of characteristics of the data's structure—the sizes of tables, the number of distinct values in particular columns, and so on.

Bind automatically creates an application plan for each interactive query or update entered from a terminal. For application programs, the application plan covers all of the program's SQL statements. The plan is saved in a component called a *directory*, to be used when needed as a program runs. Initially the developer invokes bind while preparing a program for execution. On subsequent executions of the program, if the data's structure has been changed in a way that makes the previous application plan

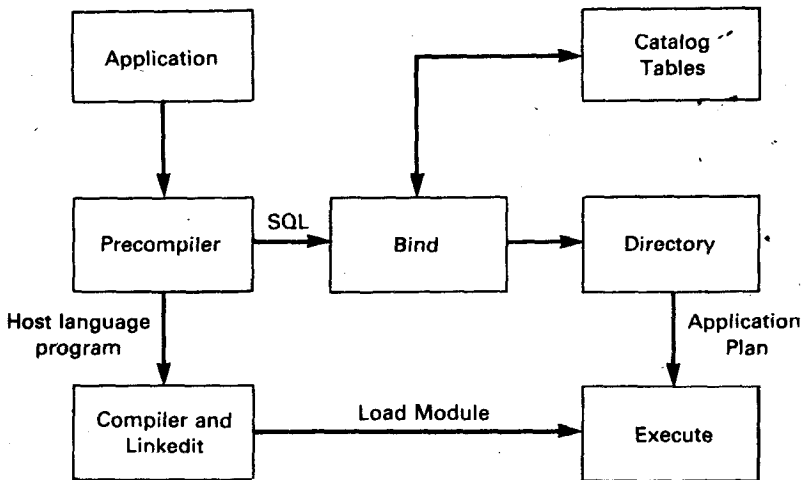


Fig. 1.3 The bind process

unusable, DB2 automatically rebinds the plan to create a new one. It knows about the change by consulting with a component called the *catalog*, which keeps track of such things.

If the developer believes that the data's structure has been changed to provide a more efficient access path, he or she can choose to rebind the plan to take advantage of it. Because SQL statements are separated from the rest of a program's code in a precompile process, an entire program need not be recompiled when an application plan is changed. The plan is simply rebound.

Catalog Tables, Statistics, and the Optimizer

The bind process eases the maintenance burden on programmers by allowing the database to be changed without requiring the programs that use it to be changed. It also saves programmers the trouble of determining access paths for their requests. Bind's intelligence for carrying out these tasks comes from a component called the *optimizer*, which estimates the most efficient access path for each request. It does this by applying rules for weighing database access options to statistics describing the particular database being accessed.

When tables and other elements of a DB2 database, called *objects*, are created, altered, or dropped, DB2 stores information about them in catalog