The Complete Reference

Includes Unique SQL Join Syntax Summary

SQL

Third Edition

- Comprehensive coverage of SQL capabilities, ANSI standards, usage, and programming
- Includes history, market trends, and feature comparisons of the leading brands of SQL DBMSs
- Updated information on XML; business intelligence; and in-memory, stream, and embedded databases

James R. Groff
Paul N. Weinberg
Andrew J. Oppel

McGraw Hill

Multitable Queries (Joins)

SQL Join Syntax Summary (see Chapter 7)

Type	Standard SQL Syntax	Description
Inner Joins[1]		

Simple equi-join	`SELECT NAME, CITY`
	`  FROM SALESREPS JOIN OFFICES`
	`  ON REP_OFFICE = OFFICE;`
		Forms pairs of rows by matching the contents of related columns based on an exact match between pairs of columns.

Explicit equi-join	`SELECT NAME, CITY`
	`  FROM SALESREPS INNER JOIN OFFICES`
	`  ON REP_OFFICE = OFFICE;`
		Syntax variation using `INNER JOIN` keywords instead of simply `JOIN`.

Parent/child query	`SELECT CITY, NAME, TITLE`
	`  FROM OFFICES JOIN SALESREPS`
	`  ON MGR = EMPL_NUM;`
		An equi-join that matches the primary key in one table with the corresponding foreign key in the other.

Row selection criteria	`SELECT CITY, NAME, TITLE`
	`  FROM OFFICES JOIN SALESREPS`
	`  ON MGR = EMPL_NUM`
	`  WHERE TARGET > 600000.00;`
		Unwanted rows are filtered out from the query results by adding a `WHERE` predicate.

Multiple matching columns	`SELECT ORDER_NUM, AMOUNT, DESCRIPTION`
	`  FROM ORDERS JOIN PRODUCTS`
	`  ON MFR = MFR_ID`
	`  AND PRODUCT = PRODUCT_ID;`
		Multicolumn primary and foreign keys require multiple column matching in the join predicate.

Three-table join	`SELECT ORDER_NUM, AMOUNT, COMPANY, NAME`
	`  FROM ORDERS JOIN CUSTOMERS`
	`  ON CUST = CUST_NUM`
	`  JOIN SALESREPS`
	`  ON REP = EMPL_NUM`
	`  WHERE AMOUNT > 25000.00;`
		More than two tables are joined together by adding additional `JOIN` clauses.

Non-equi-join	`SELECT NAME, QUOTA, TARGET`
	`  FROM SALESREPS JOIN OFFICES`
	`  ON QUOTA > TARGET;`
		The comparison operator in the join predicate is other than equal (=).

Natural join	`SELECT ORDER_NUM, AMOUNT, DESCRIPTION` `  FROM ORDERS NATURAL JOIN PRODUCTS;`	An equi-join that matches rows based on all the columns that share the same name between the joined tables.

Join
with `USING` clause
	`SELECT ORDER_NUM, AMOUNT, DESCRIPTION`
	`  FROM ORDERS JOIN PRODUCTS`
	`  USING (MFR, PRODUCT);`
		An equi-join based on the explicitly identified column names that share the same name in the joined tables.

Self-join	`SELECT EMPS.NAME, MGRS.NAME`
	`  FROM SALESREPS EMPS JOIN SALESREPS MGRS`
	`  ON EMPS.MANAGER = MGRS.EMPL_NUM;`
		An equi-join of a table to itself with each row matched with other rows in the same table.

Outer Joins[2]		

Full outer join	`SELECT *`
	`  FROM GIRLS FULL OUTER JOIN BOYS`
	`  ON GIRLS.CITY = BOYS.CITY;`
		Adds a `NULL`-extended row to the query results for each unmatched row of each joined table.

Natural full outer join	`SELECT *` `  FROM GIRLS NATURAL FULL OUTER JOIN BOYS;`	A full outer join that matches rows based on all the columns that share the same name between the joined tables.

Full outer join
with `USING` clause
	`SELECT *`
	`  FROM GIRLS FULL OUTER JOIN BOYS`
	`  USING (CITY);`
		A full outer join based on the explicitly identified column names that share the same name in the joined tables.

Full outer join
with keyword `OUTER` implied
	`SELECT *`
	`  FROM GIRLS FULL JOIN BOYS`
	`  USING (CITY);`
		Many SQL implementations allow the keyword `OUTER` to be left out because it is implied by the keyword `FULL`.

Left outer join	`SELECT *`
	`  FROM GIRLS LEFT OUTER JOIN BOYS`
	`  ON GIRLS.CITY = BOYS.CITY;`
		Adds a `NULL`-extended row to the query results for each unmatched row of the first (left) table.

Left outer join
with `USING` clause
	`SELECT *`
	`  FROM GIRLS LEFT OUTER JOIN BOYS`
	`  USING (CITY);`
		A left outer join based on the explicitly identified column names that share the same name in the joined tables.

Right outer join	`SELECT *`
	`  FROM GIRLS RIGHT OUTER JOIN BOYS`
	`  ON GIRLS.CITY = BOYS.CITY;`
		Adds a `NULL`-extended row to the query results for each unmatched row of the second (right) table.

Right outer join
with `USING` clause
	`SELECT *`
	`  FROM GIRLS RIGHT OUTER JOIN BOYS`
	`  USING (CITY);`
		A right outer join based on the explicitly identified column names that share the same name in the joined tables.

Other Joins		
Cross join	`SELECT *` `  FROM GIRLS CROSS JOIN BOYS;`	Explicitly requests the Cartesian product, which is the product consisting of all possible pairs of rows from the two tables.

Union "join"	`SELECT *`
	`  FROM GIRLS`
	`  UNION ALL`
	`  SELECT *`
	`  FROM BOYS;`
		Technically not a join; the `SELECT`s process independently and the result sets are then concatenated by the `UNION` operator.

[1] Inner joins have the potential to lose information if the tables being joined contain unmatched rows.

[2] Outer joins do not lose information because they "add back" the unmatched rows.

Also included on pages 160-161

SQL
The Complete Reference, Third Edition

SQL
The Complete Reference, Third Edition

Paul Weinberg
James Groff
Andrew Oppel

Mc Graw Hill

New York Chicago San Francisco
Lisbon London Madrid Mexico City
Milan New Delhi San Juan
Seoul Singapore Sydney Toronto

The McGraw-Hill Companies

Library of Congress Cataloging-in-Publication Data

Weinberg, Paul N.
SQL, the complete reference / Paul N. Weinberg, James R. Groff, Andrew J. Oppel. — 3rd ed.
p. cm.
Cover title: Complete reference SQL
On t.p. of previous ed., James R. Groff's name appeared first.
ISBN 978-0-07-159255-0 (alk. paper)
1. SQL (Computer program language) 2. Relational databases. I. Groff, James R. II. Oppel, Andrew J. III. Groff, James R. SQL, the complete reference; IV. Title. V. Title: Complete reference SQL.
QA76.73.S67G758 2009
005.75'6—dc22

2009027008

McGraw-Hill books are available at special quantity discounts to use as premiums and sales promotions, or for use in corporate training programs. To contact a representative, please e-mail us at bulksales@mcgraw-hill.com.

SQL: The Complete Reference, Third Edition

1234567890 DOC DOC 019

ISBN 978-0-07-159255-0
MHID 0-07-159255-5

Sponsoring Editor
Jane K. Brownlow

Editorial Supervisor
Janet Walden

Project Manager
Smita Rajan, International Typesetting and Composition

Acquisitions Coordinator
Joya Anthony

Technical Editor
Aaron Davenport

Copy Editor
Jan Jue

Proofreader
Mohinder Bhatnagar, International Typesetting and Composition

Indexer
Claire Splan

Production Supervisor
George Anderson

Composition
International Typesetting and Composition

Illustration
International Typesetting and Composition

Art Director, Cover
Jeff Weeks

About the Authors

James R. Groff is CEO of PBworks, whose hosted collaboration software helps teams of people work together more effectively and efficiently. Earlier, Groff was CEO of TimesTen, the leading provider of in-memory SQL databases. He led TimesTen from its early days through eight years of growth and a successful acquisition by Oracle in 2005, where he served as a senior vice president, and Oracle TimesTen became Oracle's flagship real-time database product. Groff was the cofounder, with Paul Weinberg, of Network Innovations Corporation, a developer of SQL-based networking software, and coauthor with him of *Understanding UNIX: A Conceptual Guide* as well as this book. Groff has also held senior division management and marketing positions at Apple Computer and Hewlett-Packard. He holds a BS in Mathematics from the Massachusetts Institute of Technology and an MBA from Harvard University.

Paul N. Weinberg is a senior vice president at SAP, where he runs core MDM (Master Data Management) development. Prior to working at SAP, Weinberg was president of A2i, Inc., which was acquired by SAP in 2004 for its enterprisewide platform for product content management and catalog publishing. Weinberg was the cofounder, with James Groff, of Network Innovations Corporation, a pioneer in client/server database access that was acquired by Apple Computer in 1988, and coauthor with him of *Understanding UNIX: A Conceptual Guide* as well as this book. He has also held software development and marketing positions at Bell Laboratories, Hewlett-Packard, and Plexus Computers. In 1981, he collaborated on *The Simple Solution to Rubik's Cube*, the number-one best-selling book of that year, with over 6 million copies sold. He holds a BS from the University of Michigan and an MS from Stanford University, both in Computer Science.

Andrew J. (Andy) Oppel is lead data modeler at Blue Shield of California. In addition, he has served as a part-time instructor in database technology with the University of California at Berkeley, Extension for more than 20 years. Andy has designed and implemented hundreds of databases for a wide range of applications, including heath care, banking, insurance, apparel manufacturing, telecommunications, wireless communications, and human resources. He is the author of *Databases Demystified*, *SQL Demystified*, and *Databases: A Beginner's Guide* and is coauthor of *SQL: A Beginner's Guide*. He holds a BA in Computer Science from Transylvania University (Lexington, KY).

About the Technical Editor

Aaron Davenport has been working with SQL-based RDBMS technologies for over ten years. He is currently a principal at LCS Technologies, Inc., a Sacramento and San Francisco Bay Area database consulting firm specializing in performance tuning, application development, and database architecture. Prior to joining LCS, Aaron had tenures at Yahoo!, Gap Inc., and Blue Shield of California.

Contents at a Glance

Contents