Preface

Purpose

This book is a quick reference for the SQL dialect supported by the Teradata Database.

Audience

All users of Teradata SQL who need information about how to structure an SQL statement.

Supported Software Release

This book supports Teradata® Database 13.0.

Prerequisites

You should be familiar with basic computer technology, the Teradata Database, and the Teradata SQL language.

It may be helpful to review the following books:

• Introduction to Teradata
• The SQL book set

Changes to This Book

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata Database 13.0</td>
<td>• Updated syntax diagrams throughout the book</td>
</tr>
<tr>
<td>April 2009</td>
<td>• Added the following new syntax diagrams to Chapter 1:</td>
</tr>
<tr>
<td></td>
<td>• Period Literals</td>
</tr>
<tr>
<td></td>
<td>• PERIOD(DATE) Data Type</td>
</tr>
<tr>
<td></td>
<td>• PERIOD(TIME) Data Type</td>
</tr>
<tr>
<td></td>
<td>• PERIOD(TIME WITH TIME ZONE) Data Type</td>
</tr>
<tr>
<td></td>
<td>• PERIOD(TIMESTAMP) Data Type</td>
</tr>
</tbody>
</table>
### Changes to This Book

<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
</table>
| Teradata Database 13.0 (Continued) | • Added the following new syntax diagrams to Chapter 1:  
  • PERIOD(TIMESTAMP WITH TIME ZONE) Data Type  
  • VARIANT_TYPE UDT  
  • Geospatial Data Types  
  • Added the following new syntax diagrams to Chapter 2:  
  • CURRENT_ROLE  
  • CURRENT_USER  
  • CONTAINS  
  • IS UNTIL_CHANGED/IS NOT UNTIL_CHANGED  
  • MEETS  
  • PRECEDES  
  • SUCCEEDS  
  • BEGIN  
  • END  
  • LAST  
  • INTERVAL  
  • PRIOR  
  • NEXT  
  • P_INTERSECT  
  • LDIFF  
  • RDIFF  
  • P_NORMALIZE  
  • Period Value Constructor  
  • Arithmetic Operators  
  • Scalar UDF Expression  
  • Aggregate UDF Expression  
  • NEW VARIANT_TYPE  
  • Added the following new syntax diagrams to Chapter 3:  
  • COLLECT STATISTICS (Recollect Statistics)  
  • CREATE GLOP SET  
  • CREATE REPLICATION RULESET/REPLACE REPLICATION RULESET  
  • DROP GLOP SET  
  • DROP REPLICATION RULESET  
  • SHOW QUERY LOGGING  
  • Added the following new syntax diagrams to Chapter 4:  
  • GRANT CONNECT THROUGH  
  • REVOKE CONNECT THROUGH  
  • Added the following new syntax diagrams to Chapter 5:  
  • HASH BY Clause  
  • LOCAL ORDER BY Clause |
<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Teradata Database 13.0 (Continued)</td>
<td>• Added the following new syntax diagrams to Chapter 7:</td>
</tr>
<tr>
<td></td>
<td>• SIGNAL</td>
</tr>
<tr>
<td></td>
<td>• RESIGNAL</td>
</tr>
<tr>
<td></td>
<td>• GET DIAGNOSTICS</td>
</tr>
<tr>
<td>Teradata Database 12.0 September 2007</td>
<td>• Updated existing syntax diagrams in Chapters 2, 3, 4, 5, 6, 7, 8, 9 and 10</td>
</tr>
<tr>
<td></td>
<td>• Added the following new syntax diagrams to Chapter 1:</td>
</tr>
<tr>
<td></td>
<td>• Hexadecimal Name Literals</td>
</tr>
<tr>
<td></td>
<td>• Unicode Character String Literals</td>
</tr>
<tr>
<td></td>
<td>• Unicode Delimited Identifier</td>
</tr>
<tr>
<td></td>
<td>• Added the following new syntax diagrams to Chapter 2:</td>
</tr>
<tr>
<td></td>
<td>• DEGREES, RADIANS</td>
</tr>
<tr>
<td></td>
<td>• STRING_CS</td>
</tr>
<tr>
<td></td>
<td>• Modified the syntax diagrams for Window Aggregate Functions in Chapter 2 to support the following aggregate functions:</td>
</tr>
<tr>
<td></td>
<td>• CORR</td>
</tr>
<tr>
<td></td>
<td>• COVAR_POP</td>
</tr>
<tr>
<td></td>
<td>• COVAR_SAMP</td>
</tr>
<tr>
<td></td>
<td>• REGR_AVGX</td>
</tr>
<tr>
<td></td>
<td>• REGR_AVGY</td>
</tr>
<tr>
<td></td>
<td>• REGR_COUNT</td>
</tr>
<tr>
<td></td>
<td>• REGR_INTERCEPT</td>
</tr>
<tr>
<td></td>
<td>• REGR_R2</td>
</tr>
<tr>
<td></td>
<td>• REGR_SLOPE</td>
</tr>
<tr>
<td></td>
<td>• REGR_SXX</td>
</tr>
<tr>
<td></td>
<td>• REGR_SXY</td>
</tr>
<tr>
<td></td>
<td>• REGR_SYY</td>
</tr>
<tr>
<td></td>
<td>• STDDEV_POP</td>
</tr>
<tr>
<td></td>
<td>• STDDEV_SAMP</td>
</tr>
<tr>
<td></td>
<td>• VAR_POP</td>
</tr>
<tr>
<td></td>
<td>• VAR_SAMP</td>
</tr>
<tr>
<td></td>
<td>• Added the following new syntax diagrams to Chapter 3:</td>
</tr>
<tr>
<td></td>
<td>• CREATE ERROR TABLE</td>
</tr>
<tr>
<td></td>
<td>• Java Simple and Object Mapped Data Types</td>
</tr>
<tr>
<td></td>
<td>• DROP ERROR TABLE</td>
</tr>
<tr>
<td></td>
<td>• LOGGING ONLINE ARCHIVE OFF</td>
</tr>
<tr>
<td></td>
<td>• LOGGING ONLINE ARCHIVE ON</td>
</tr>
<tr>
<td></td>
<td>• SET QUERY_BAND</td>
</tr>
</tbody>
</table>
Teradata Database 12.0
(Continued)

- Added the following new syntax diagrams to Chapter 5:
  - INITIATE PARTITION ANALYSIS
  - DIAGNOSTIC COSTPRINT
  - DIAGNOSTIC DUMP COSTS
  - DIAGNOSTIC HELP COSTS
  - DIAGNOSTIC SET COSTS
  - DIAGNOSTIC HELP PROFILE
  - DIAGNOSTIC SET PROFILE
- Added the following new syntax diagrams to Chapter 6:
  - HELP COLUMN - Syntaxes 6, 7 and 8
  - HELP ERROR TABLE

## Additional Information

<table>
<thead>
<tr>
<th>URL</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.info.teradata.com">http://www.info.teradata.com</a></td>
<td>Use the Teradata Information Products Publishing Library site to:</td>
</tr>
<tr>
<td></td>
<td>- View or download a manual:</td>
</tr>
<tr>
<td></td>
<td>1 Under <strong>Online Publications</strong>, select <strong>General Search</strong>.</td>
</tr>
<tr>
<td></td>
<td>2 Enter your search criteria and click <strong>Search</strong>.</td>
</tr>
<tr>
<td></td>
<td>- Download a documentation CD-ROM:</td>
</tr>
<tr>
<td></td>
<td>1 Under <strong>Online Publications</strong>, select <strong>General Search</strong>.</td>
</tr>
<tr>
<td></td>
<td>2 In the <strong>Title or Keyword</strong> field, enter <strong>CD-ROM</strong>, and click <strong>Search</strong>.</td>
</tr>
<tr>
<td></td>
<td>- Order printed manuals:</td>
</tr>
<tr>
<td></td>
<td>Under <strong>Print &amp; CD Publications</strong>, select <strong>How to Order</strong>.</td>
</tr>
<tr>
<td><a href="http://www.teradata.com">http://www.teradata.com</a></td>
<td>The Teradata home page provides links to numerous sources of information about Teradata. Links include:</td>
</tr>
<tr>
<td></td>
<td>- Executive reports, case studies of customer experiences with Teradata, and thought leadership</td>
</tr>
<tr>
<td></td>
<td>- Technical information, solutions, and expert advice</td>
</tr>
<tr>
<td></td>
<td>- Press releases, mentions and media resources</td>
</tr>
<tr>
<td><a href="http://teradatauniversitynetwork.com">http://teradatauniversitynetwork.com</a></td>
<td>Teradata University Network fosters education on data warehousing, business intelligence (BI) and database administration (DBA).</td>
</tr>
</tbody>
</table>
To maintain the quality of our products and services, we would like your comments on the accuracy, clarity, organization, and value of this document. Please e-mail: teradata-books@lists.teradata.com

References to Microsoft Windows and Linux

This book refers to “Microsoft Windows” and “Linux.” For Teradata Database 13.0, these references mean:

- “Windows” is Microsoft Windows Server 2003 64-bit.
- “Linux” is SUSE Linux Enterprise Server 9 and SUSE Linux Enterprise Server 10.
# Table of Contents

## Preface
- Purpose ............................................. 3
- Audience ........................................ 3
- Supported Software Release ......................... 3
- Prerequisites .................................... 3
- Changes to This Book ................................. 3
- Additional Information .................................. 6
- References to Microsoft Windows and Linux ............. 7

## Chapter 1: Data Types and Literals
- Byte and BLOB Data Types ............................. 21
- Character and CLOB Data Types ......................... 22
- Data Literals ......................................... 23
- DateTime and Interval Data Types ....................... 29
- Decimal/Numeric Data Types ........................... 32
- PERIOD Data Types .................................. 33
- UDT Data Types .................................... 34
- Geospatial Data Types ................................ 35
- Default Value Control Phrases ......................... 37
- Output Format Phrases ................................ 38

## Chapter 2: SQL Functions and Expressions
- Aggregate Functions ................................ 39
- Arithmetic Operators and Functions/Trigonometric/Hyperbolic Functions .................... 42
- Trigonometric Functions ................................ 45
- Hyperbolic Functions ................................ 45
- Attribute Functions .................................. 46
- Built-In Functions .................................... 47
<table>
<thead>
<tr>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE Expressions</td>
<td>49</td>
</tr>
<tr>
<td>Comparison Operators</td>
<td>50</td>
</tr>
<tr>
<td>Data Type Conversions</td>
<td>51</td>
</tr>
<tr>
<td>Byte Conversion</td>
<td>51</td>
</tr>
<tr>
<td>Character-to-Character Conversion</td>
<td>52</td>
</tr>
<tr>
<td>Character-to-DATE Conversion</td>
<td>53</td>
</tr>
<tr>
<td>Character-to-INTERVAL Conversion</td>
<td>53</td>
</tr>
<tr>
<td>Character-to-Period Conversion</td>
<td>54</td>
</tr>
<tr>
<td>Character-to-Numeric Conversion</td>
<td>54</td>
</tr>
<tr>
<td>Character-to-TIME Conversion</td>
<td>54</td>
</tr>
<tr>
<td>Character-to-TIMESTAMP Conversion</td>
<td>55</td>
</tr>
<tr>
<td>Character-to-UDT Conversion</td>
<td>56</td>
</tr>
<tr>
<td>DATE-to-Character Conversion</td>
<td>56</td>
</tr>
<tr>
<td>DATE-to-DATE Conversion</td>
<td>57</td>
</tr>
<tr>
<td>DATE-to-Numeric Conversion</td>
<td>57</td>
</tr>
<tr>
<td>DATE-to-Period Conversion</td>
<td>58</td>
</tr>
<tr>
<td>DATE-to-TIMESTAMP Conversion</td>
<td>58</td>
</tr>
<tr>
<td>DATE-to-UDT Conversion</td>
<td>59</td>
</tr>
<tr>
<td>INTERVAL-to-Character Conversion</td>
<td>59</td>
</tr>
<tr>
<td>INTERVAL-to-INTERVAL Conversion</td>
<td>60</td>
</tr>
<tr>
<td>INTERVAL-to-Numeric Conversion</td>
<td>61</td>
</tr>
<tr>
<td>INTERVAL-to-UDT Conversion</td>
<td>61</td>
</tr>
<tr>
<td>Numeric-to-Character Conversion</td>
<td>61</td>
</tr>
<tr>
<td>Numeric-to-DATE Conversion</td>
<td>62</td>
</tr>
<tr>
<td>Numeric-to-INTERVAL Conversion</td>
<td>62</td>
</tr>
<tr>
<td>Numeric-to-Numeric Conversion</td>
<td>63</td>
</tr>
<tr>
<td>Numeric-to-UDT Conversion</td>
<td>63</td>
</tr>
<tr>
<td>Period-to-Character Conversion</td>
<td>64</td>
</tr>
<tr>
<td>Period-to-DATE Conversion</td>
<td>64</td>
</tr>
<tr>
<td>Period-to-Period Conversion</td>
<td>64</td>
</tr>
<tr>
<td>Period-to-TIME Conversion</td>
<td>65</td>
</tr>
<tr>
<td>Period-to-TIMESTAMP Conversion</td>
<td>65</td>
</tr>
<tr>
<td>TIME-to-Character Conversion</td>
<td>65</td>
</tr>
<tr>
<td>TIME-to-DATE Conversion</td>
<td>66</td>
</tr>
<tr>
<td>TIME-to-Period Conversion</td>
<td>66</td>
</tr>
<tr>
<td>TIME-to-TIME Conversion</td>
<td>66</td>
</tr>
<tr>
<td>TIME-to-TIMESTAMP Conversion</td>
<td>67</td>
</tr>
<tr>
<td>TIME-to-UDT Conversion</td>
<td>68</td>
</tr>
<tr>
<td>TIMESTAMP-to-Character Conversion</td>
<td>68</td>
</tr>
</tbody>
</table>
### Table of Contents

- TIMESTAMP-to-DATE Conversion ........................................... 69
- TIMESTAMP-to-Period Conversion ....................................... 69
- TIMESTAMP-to-TIME Conversion ....................................... 69
- TIMESTAMP-to-TIMESTAMP Conversion ................................ 70
- TIMESTAMP-to-UDT Conversion ......................................... 71
- UDT-to-Byte Conversion ................................................. 71
- UDT-to-Character Conversion ........................................... 72
- UDT-to-DATE Conversion .................................................. 72
- UDT-to-INTERVAL Conversion ......................................... 73
- UDT-to-Numeric Conversion .............................................. 73
- UDT-to-TIME Conversion .................................................. 74
- UDT-to-TIMESTAMP Conversion ....................................... 75
- UDT-to-UDT Conversion ................................................... 75
- DateTime and Interval Functions and Expressions .................. 75
- Hash-Related Functions ................................................... 78
- Logical Predicates ........................................................ 79
- Ordered Analytical Functions ......................................... 83
- Period Functions and Operators ....................................... 87
- Set Operators .................................................................. 89
- String Operator and Functions ......................................... 90
- UDF Expressions ............................................................. 92
- UDT Expressions and Methods .......................................... 93

### Chapter 3: SQL Data Definition Language ......................... 95

- ALTER FUNCTION .......................................................... 95
- ALTER METHOD ............................................................ 95
- ALTER PROCEDURE (External Form) .................................... 96
- ALTER PROCEDURE (SQL Form) ........................................ 96
- ALTER REPLICATION GROUP ............................................ 96
- ALTER TABLE ............................................................... 97
- ALTER TRIGGER ........................................................... 102
- ALTER TYPE ............................................................... 103
- BEGIN LOGGING .......................................................... 104
- BEGIN QUERY LOGGING ................................................. 105
- COLLECT STATISTICS (Optimizer Form) .............................. 106
- COLLECT STATISTICS (Alternate Optimizer Form) ................. 107
COLLECT STATISTICS (Recollect Statistics) ......................................................... 107
COMMENT (Comment Placing Form) ............................................................ 108
CREATE AUTHORIZATION/
    REPLACE AUTHORIZATION ................................................................. 108
CREATE CAST/
    REPLACE CAST ................................................................................. 109
CREATE DATABASE ...................................................................................... 110
CREATE ERROR TABLE .............................................................................. 110
CREATE FUNCTION/
    REPLACE FUNCTION ................................................................. 111
CREATE FUNCTION (Table Form) ................................................................. 115
CREATE GLOBAL TEMPORARY TRACE TABLE ................................................ 118
CREATE GLOP SET ................................................................................... 121
CREATE HASH INDEX ............................................................................... 121
CREATE INDEX .......................................................................................... 122
CREATE JOIN INDEX ............................................................................... 123
CREATE MACRO/
    REPLACE MACRO ................................................................. 125
CREATE METHOD ..................................................................................... 126
CREATE ORDERING/
    REPLACE ORDERING ................................................................. 128
CREATE PROCEDURE (External Form)/
    REPLACE PROCEDURE (External Form) .................................................. 129
CREATE PROCEDURE (SQL Form)/
    REPLACE PROCEDURE ........................................................................ 131
CREATE PROFILE ..................................................................................... 138
CREATE RECURSIVE VIEW/
    REPLACE RECURSIVE VIEW .......................................................... 139
CREATE REPLICATION GROUP ................................................................... 140
CREATE REPLICATION RULESET/
    REPLACE REPLICATION RULESET ...................................................... 141
CREATE ROLE .......................................................................................... 141
CREATE TABLE ......................................................................................... 142
CREATE TABLE (Queue Table Form) ............................................................ 151
CREATE TRANSFORM/
    REPLACE TRANSFORM ..................................................................... 156
CREATE TRIGGER/
    REPLACE TRIGGER ........................................................................... 157
CREATE TYPE (Distinct Form) ................................................................... 158
CREATE TYPE (Structured Form) ................................................................. 160
CREATE USER ............................................................................................ 164
<table>
<thead>
<tr>
<th>SQL Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE VIEW/</td>
<td>165</td>
</tr>
<tr>
<td>REPLACE VIEW</td>
<td></td>
</tr>
<tr>
<td>DATABASE</td>
<td>166</td>
</tr>
<tr>
<td>DELETE DATABASE</td>
<td></td>
</tr>
<tr>
<td>DELETE USER</td>
<td>166</td>
</tr>
<tr>
<td>DROP AUTHORIZATION</td>
<td>166</td>
</tr>
<tr>
<td>DROP CAST</td>
<td>166</td>
</tr>
<tr>
<td>DROP DATABASE</td>
<td>168</td>
</tr>
<tr>
<td>DROP ERROR TABLE</td>
<td>168</td>
</tr>
<tr>
<td>DROP FUNCTION</td>
<td>169</td>
</tr>
<tr>
<td>DROP GLOP SET</td>
<td>171</td>
</tr>
<tr>
<td>DROP HASH INDEX</td>
<td>171</td>
</tr>
<tr>
<td>DROP INDEX</td>
<td>172</td>
</tr>
<tr>
<td>DROP JOIN INDEX</td>
<td>172</td>
</tr>
<tr>
<td>DROP MACRO/</td>
<td></td>
</tr>
<tr>
<td>DROP PROCEDURE/</td>
<td></td>
</tr>
<tr>
<td>DROP TABLE/</td>
<td></td>
</tr>
<tr>
<td>DROP TRIGGER/</td>
<td></td>
</tr>
<tr>
<td>DROP VIEW</td>
<td>172</td>
</tr>
<tr>
<td>DROP ORDERING</td>
<td>173</td>
</tr>
<tr>
<td>DROP PROFILE</td>
<td>173</td>
</tr>
<tr>
<td>DROP REPLICATION GROUP</td>
<td>174</td>
</tr>
<tr>
<td>DROP REPLICATION RULESET</td>
<td>174</td>
</tr>
<tr>
<td>DROP ROLE</td>
<td>174</td>
</tr>
<tr>
<td>DROP STATISTICS (Optimizer Form)</td>
<td>175</td>
</tr>
<tr>
<td>DROP TRANSFORM</td>
<td>176</td>
</tr>
<tr>
<td>DROP TYPE</td>
<td>176</td>
</tr>
<tr>
<td>DROP USER</td>
<td>176</td>
</tr>
<tr>
<td>END LOGGING</td>
<td>177</td>
</tr>
<tr>
<td>END QUERY LOGGING</td>
<td>178</td>
</tr>
<tr>
<td>LOGGING ONLINE ARCHIVE OFF</td>
<td>178</td>
</tr>
<tr>
<td>LOGGING ONLINE ARCHIVE ON</td>
<td>178</td>
</tr>
<tr>
<td>MODIFY DATABASE</td>
<td>179</td>
</tr>
<tr>
<td>MODIFY PROFILE</td>
<td>180</td>
</tr>
<tr>
<td>MODIFY USER</td>
<td>181</td>
</tr>
<tr>
<td>RENAME FUNCTION</td>
<td>182</td>
</tr>
<tr>
<td>RENAME MACRO/</td>
<td></td>
</tr>
<tr>
<td>RENAME PROCEDURE/</td>
<td></td>
</tr>
<tr>
<td>RENAME TABLE/</td>
<td></td>
</tr>
<tr>
<td>Command/Method</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>RENAME TRIGGER/</td>
<td>184</td>
</tr>
<tr>
<td>RENAME VIEW.</td>
<td></td>
</tr>
<tr>
<td>REPLACE METHOD</td>
<td>185</td>
</tr>
<tr>
<td>SET QUERY_BAND</td>
<td>187</td>
</tr>
<tr>
<td>SET ROLE</td>
<td>188</td>
</tr>
<tr>
<td>SET SESSION</td>
<td>188</td>
</tr>
<tr>
<td>SET SESSION ACCOUNT</td>
<td></td>
</tr>
<tr>
<td>SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL</td>
<td>189</td>
</tr>
<tr>
<td>SET SESSION COLLATION</td>
<td>189</td>
</tr>
<tr>
<td>SET SESSION DATABASE</td>
<td>189</td>
</tr>
<tr>
<td>SET SESSION DATEFORM</td>
<td></td>
</tr>
<tr>
<td>SET SESSION FUNCTION TRACE</td>
<td>190</td>
</tr>
<tr>
<td>SET SESSION OVERRIDE REPLICATION</td>
<td>190</td>
</tr>
<tr>
<td>SET SESSION SUBSCRIBER</td>
<td>190</td>
</tr>
<tr>
<td>SET TIME ZONE</td>
<td>190</td>
</tr>
<tr>
<td>HELP</td>
<td>191</td>
</tr>
<tr>
<td>HELP CAST</td>
<td>193</td>
</tr>
<tr>
<td>HELP COLUMN</td>
<td>193</td>
</tr>
<tr>
<td>HELP CONSTRAINT</td>
<td>194</td>
</tr>
<tr>
<td>HELP DATABASE/</td>
<td></td>
</tr>
<tr>
<td>HELP USER</td>
<td>195</td>
</tr>
<tr>
<td>HELP ERROR TABLE</td>
<td>195</td>
</tr>
<tr>
<td>HELP FUNCTION</td>
<td>195</td>
</tr>
<tr>
<td>HELP HASH INDEX</td>
<td>197</td>
</tr>
<tr>
<td>HELP INDEX</td>
<td>197</td>
</tr>
<tr>
<td>HELP JOIN INDEX</td>
<td>198</td>
</tr>
<tr>
<td>HELP MACRO/</td>
<td></td>
</tr>
<tr>
<td>HELP TABLE/</td>
<td></td>
</tr>
<tr>
<td>HELP VIEW</td>
<td>198</td>
</tr>
<tr>
<td>HELP METHOD</td>
<td>198</td>
</tr>
<tr>
<td>HELP PROCEDURE</td>
<td>199</td>
</tr>
<tr>
<td>HELP REPLICATION GROUP</td>
<td>199</td>
</tr>
<tr>
<td>HELP SESSION</td>
<td>199</td>
</tr>
<tr>
<td>HELP STATISTICS (Optimizer Form)</td>
<td>199</td>
</tr>
<tr>
<td>HELP STATISTICS (QCD Form)</td>
<td>200</td>
</tr>
<tr>
<td>HELP TRANSFORM</td>
<td>200</td>
</tr>
<tr>
<td>HELP TRIGGER</td>
<td>200</td>
</tr>
<tr>
<td>HELP TYPE</td>
<td>201</td>
</tr>
<tr>
<td>HELP VOLATILE TABLE</td>
<td>201</td>
</tr>
<tr>
<td>SQL Command</td>
<td>Page</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>HELP (Online Form)</td>
<td>201</td>
</tr>
<tr>
<td>SHOW</td>
<td>202</td>
</tr>
<tr>
<td>SHOW CAST/</td>
<td></td>
</tr>
<tr>
<td>SHOW ERROR TABLE/</td>
<td></td>
</tr>
<tr>
<td>SHOW FUNCTION/</td>
<td></td>
</tr>
<tr>
<td>SHOW HASH INDEX/</td>
<td></td>
</tr>
<tr>
<td>SHOW JOIN INDEX/</td>
<td></td>
</tr>
<tr>
<td>SHOW MACRO/</td>
<td></td>
</tr>
<tr>
<td>SHOW METHOD/</td>
<td></td>
</tr>
<tr>
<td>SHOW PROCEDURE/</td>
<td></td>
</tr>
<tr>
<td>SHOW REPLICATION GROUP/</td>
<td></td>
</tr>
<tr>
<td>SHOW TABLE/</td>
<td></td>
</tr>
<tr>
<td>SHOW TRIGGER/</td>
<td></td>
</tr>
<tr>
<td>SHOW TYPE/</td>
<td></td>
</tr>
<tr>
<td>SHOW VIEW</td>
<td>202</td>
</tr>
<tr>
<td>SHOW QUERY LOGGING</td>
<td>208</td>
</tr>
</tbody>
</table>

**Chapter 4: SQL Data Control Language** .......................... 209

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIVE</td>
<td>209</td>
</tr>
<tr>
<td>GRANT</td>
<td>209</td>
</tr>
<tr>
<td>GRANT CONNECT THROUGH</td>
<td>212</td>
</tr>
<tr>
<td>GRANT LOGON</td>
<td>213</td>
</tr>
<tr>
<td>REVOKE</td>
<td>213</td>
</tr>
<tr>
<td>REVOKE CONNECT THROUGH</td>
<td>216</td>
</tr>
<tr>
<td>REVOKE LOGON</td>
<td>217</td>
</tr>
</tbody>
</table>

**Chapter 5: SQL Data Manipulation Language** ...................... 219

<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT</td>
<td>220</td>
</tr>
<tr>
<td>SELECT AND CONSUME</td>
<td>223</td>
</tr>
<tr>
<td>WITH [RECURSIVE] Request Modifier</td>
<td>224</td>
</tr>
<tr>
<td>DISTINCT, ALL, .ALL Options</td>
<td>226</td>
</tr>
<tr>
<td>TOP n Operator</td>
<td>227</td>
</tr>
<tr>
<td>FROM Clause</td>
<td>227</td>
</tr>
<tr>
<td>HASH BY Clause</td>
<td>228</td>
</tr>
<tr>
<td>LOCAL ORDER BY Clause</td>
<td>228</td>
</tr>
<tr>
<td>WHERE Clause</td>
<td>228</td>
</tr>
<tr>
<td>Subqueries in Search Conditions</td>
<td>228</td>
</tr>
<tr>
<td>Clause</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>GROUP BY Clause</td>
<td></td>
</tr>
<tr>
<td>HAVING Clause</td>
<td></td>
</tr>
<tr>
<td>QUALIFY Clause</td>
<td></td>
</tr>
<tr>
<td>SAMPLE Clause</td>
<td></td>
</tr>
<tr>
<td>SAMPLEID Expression</td>
<td></td>
</tr>
<tr>
<td>ORDER BY Clause</td>
<td></td>
</tr>
<tr>
<td>WITH Clause</td>
<td></td>
</tr>
<tr>
<td>Outer Join</td>
<td></td>
</tr>
<tr>
<td>Null</td>
<td></td>
</tr>
<tr>
<td>ABORT</td>
<td></td>
</tr>
<tr>
<td>BEGIN TRANSACTION</td>
<td></td>
</tr>
<tr>
<td>CALL</td>
<td></td>
</tr>
<tr>
<td>CHECKPOINT</td>
<td></td>
</tr>
<tr>
<td>COMMENT (Comment-Retrieving Form)</td>
<td></td>
</tr>
<tr>
<td>COMMIT</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td></td>
</tr>
<tr>
<td>ECHO</td>
<td></td>
</tr>
<tr>
<td>END TRANSACTION</td>
<td></td>
</tr>
<tr>
<td>EXECUTE</td>
<td></td>
</tr>
<tr>
<td>INSERT/INSERT ... SELECT</td>
<td></td>
</tr>
<tr>
<td>LOCKING Request Modifier</td>
<td></td>
</tr>
<tr>
<td>MERGE</td>
<td></td>
</tr>
<tr>
<td>ROLLBACK</td>
<td></td>
</tr>
<tr>
<td>UPDATE</td>
<td></td>
</tr>
<tr>
<td>USING Request Modifier</td>
<td></td>
</tr>
<tr>
<td>COLLECT DEMOGRAPHICS</td>
<td></td>
</tr>
<tr>
<td>COLLECT STATISTICS (QCD Form)</td>
<td></td>
</tr>
<tr>
<td>DROP STATISTICS (QCD Form)</td>
<td></td>
</tr>
<tr>
<td>DUMP EXPLAIN</td>
<td></td>
</tr>
<tr>
<td>EXPLAIN Request Modifier</td>
<td></td>
</tr>
<tr>
<td>INITIATE INDEX ANALYSIS</td>
<td></td>
</tr>
<tr>
<td>INITIATE PARTITION ANALYSIS</td>
<td></td>
</tr>
<tr>
<td>INSERT EXPLAIN</td>
<td></td>
</tr>
<tr>
<td>RESTART INDEX ANALYSIS</td>
<td></td>
</tr>
<tr>
<td>DIAGNOSTIC COSTPRINT</td>
<td></td>
</tr>
<tr>
<td>DIAGNOSTIC DUMP COSTS</td>
<td></td>
</tr>
<tr>
<td>DIAGNOSTIC HELP COSTS</td>
<td></td>
</tr>
<tr>
<td>DIAGNOSTIC SET COSTS</td>
<td></td>
</tr>
</tbody>
</table>
**Table of Contents**

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIAGNOSTIC HELP PROFILE</td>
<td>245</td>
</tr>
<tr>
<td>DIAGNOSTIC SET PROFILE</td>
<td>245</td>
</tr>
<tr>
<td>DIAGNOSTIC DUMP SAMPLES</td>
<td>246</td>
</tr>
<tr>
<td>DIAGNOSTIC HELP SAMPLES</td>
<td>246</td>
</tr>
<tr>
<td>DIAGNOSTIC SET SAMPLES</td>
<td>246</td>
</tr>
<tr>
<td>DIAGNOSTIC “Validate Index”</td>
<td>247</td>
</tr>
</tbody>
</table>

**Chapter 6: SQL Cursor Control** ......................................................... 249

<table>
<thead>
<tr>
<th>Statement</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLOSE</td>
<td>249</td>
</tr>
<tr>
<td>DECLARE CURSOR</td>
<td>249</td>
</tr>
<tr>
<td>DELETE</td>
<td>250</td>
</tr>
<tr>
<td>FETCH</td>
<td>251</td>
</tr>
<tr>
<td>OPEN</td>
<td>252</td>
</tr>
<tr>
<td>POSITION</td>
<td>252</td>
</tr>
<tr>
<td>PREPARE</td>
<td>252</td>
</tr>
<tr>
<td>REWIND</td>
<td>253</td>
</tr>
<tr>
<td>SELECT ... INTO</td>
<td>253</td>
</tr>
<tr>
<td>SELECT AND CONSUME ... INTO</td>
<td>254</td>
</tr>
<tr>
<td>UPDATE (Positioned Form)</td>
<td>254</td>
</tr>
</tbody>
</table>

**Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling** ....................................................... 255

<table>
<thead>
<tr>
<th>Statement</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN - END Statement</td>
<td>255</td>
</tr>
<tr>
<td>CASE</td>
<td>255</td>
</tr>
<tr>
<td>DECLARE</td>
<td>260</td>
</tr>
<tr>
<td>FOR</td>
<td>260</td>
</tr>
<tr>
<td>IF</td>
<td>265</td>
</tr>
<tr>
<td>ITERATE</td>
<td>269</td>
</tr>
<tr>
<td>LEAVE</td>
<td>269</td>
</tr>
<tr>
<td>LOOP</td>
<td>269</td>
</tr>
<tr>
<td>REPEAT</td>
<td>273</td>
</tr>
<tr>
<td>SET</td>
<td>274</td>
</tr>
<tr>
<td>WHILE</td>
<td>274</td>
</tr>
<tr>
<td>DECLARE CONDITION</td>
<td>278</td>
</tr>
<tr>
<td>DECLARE HANDLER (Basic Syntax)</td>
<td>279</td>
</tr>
</tbody>
</table>
## Table of Contents

- SIGNAL .................................................................................................................. 279
- RESIGNAL ............................................................................................................... 279
- GET DIAGNOSTICS ................................................................................................. 280

### Chapter 8: Static Embedded SQL Statements ......................................................... 281
- BEGIN DECLARE SECTION .......................................................................................... 281
- COMMENT .................................................................................................................. 281
- DATABASE .................................................................................................................. 281
- DECLARE STATEMENT ............................................................................................... 282
- DECLARE TABLE ........................................................................................................ 282
- END DECLARE SECTION ............................................................................................. 282
- END-EXEC Statement Terminator ............................................................................. 282
- EXEC ........................................................................................................................ 283
- EXEC SQL Statement Prefix ..................................................................................... 283
- INCLUDE ..................................................................................................................... 283
- INCLUDE SQLCA ...................................................................................................... 283
- INCLUDE SQLDA ...................................................................................................... 284
- WHENEVER ............................................................................................................... 284

### Chapter 9: Dynamic Embedded SQL Statements .................................................... 285
- DESCRIBE ................................................................................................................ 285
- EXECUTE ................................................................................................................... 285
- EXECUTE IMMEDIATE .............................................................................................. 286
- PREPARE ................................................................................................................... 286

### Chapter 10: SQL Client-Server Connectivity Statements ...................................... 287
- CONNECT .................................................................................................................. 287
- GET CRASH .............................................................................................................. 287
- LOGOFF ..................................................................................................................... 287
- LOGON ....................................................................................................................... 288
- SET BUFFERSIZE ..................................................................................................... 288
- SET CHARSET ......................................................................................................... 288
- SET CONNECTION ................................................................................................. 288
CHAPTER 1 Data Types and Literals

Byte and BLOB Data Types

BLOB Data Type

BYTE Data Type

VARBYTE Data Type
Character and CLOB Data Types

CASESPECIFIC Phrase

CHARACTER SET Phrase

CHARACTER Data Type

CLOB Data Type
UPPERCASE Phrase

VARCHAR Data Type

Data Literals

CHARACTER String Literals
Chapter 1: Data Types and Literals

Data Literals

**DATE Literals**

\[
\text{DATE Literal} \quad \text{'} \text{string} \text{'}
\]

**DECIMAL Literals**

\[
\pm n. \\
\pm n \\
\pm n.n
\]

**FLOATING POINT Literals**

\[
\pm n\pm m \\
\pm n\pm m \\
\pm n.n\pm m \\
\pm n.n\pm m
\]

**GRAPHIC Literals**

\[
G \text{'}< \text{ABC} >'\]

**Hexadecimal Byte Literals**

\[
'\text{hexadecimal digits}' \quad \text{XB}
\]

**Hexadecimal Character Literals**

\[
\text{character_set} '\text{hexadecimal digits}' \quad \text{XC}
\]
Chapter 1: Data Types and Literals

Data Literals

**Hexadecimal Integer Literals**

```
  'hexadecimal digits'  X
```

**Hexadecimal Name Literals**

```
  'hexadecimal_name_body'  XN
```

**INTEGER Literals**

```
± n
```

**INTERVAL DAY Literals**

```
INTERVAL 'string'  DAY
```

**INTERVAL DAY TO HOUR Literals**

```
INTERVAL 'string'  DAY TO HOUR
```
Chapter 1: Data Types and Literals
Data Literals

**INTERVAL DAY TO MINUTE Literals**

```
| INTERVAL [sign] 'string' --- DAY TO MINUTE |
```

1101A028

**INTERVAL DAY TO SECOND Literals**

```
| INTERVAL [sign] 'string' --- DAY TO SECOND |
```

1101A029

**INTERVAL HOUR Literals**

```
| INTERVAL [sign] 'string' --- HOUR |
```

1101A030

**INTERVAL HOUR TO MINUTE Literals**

```
| INTERVAL [sign] 'string' --- HOUR TO MINUTE |
```

1101A037

**INTERVAL HOUR TO SECOND Literals**

```
| INTERVAL [sign] 'string' --- HOUR TO SECOND |
```

1101A038
INTERVAL MINUTE Literals

```
 INTERVAL 'string' MINUTE sign
```

1101A031

INTERVAL MINUTE TO SECOND Literals

```
 INTERVAL 'string' MINUTE TO SECOND sign
```

1101A032

INTERVAL MONTH Literals

```
 INTERVAL 'string' MONTH sign
```

1101A025

INTERVAL SECOND Literals

```
 INTERVAL 'string' SECOND sign
```

1101A033

INTERVAL YEAR Literals

```
 INTERVAL 'string' YEAR sign
```

1101A023
Chapter 1: Data Types and Literals

Data Literals

**INTERVAL YEAR TO MONTH Literals**

```
INTERVAL sign 'string' YEAR TO MONTH
```

**Period Literals**

```
PERIOD (beginning_bound UNTIL_CHANGED ending_bound)
```

**Time Literals**

```
TIME 'string'
```

**Timestamp Literals**

```
TIMESTAMP 'string'
```

**Unicode Character String Literals**

```
character_set U& 'Unicode_string_body' U& ' ' 'Unicode_string_body'
```

```
UESCAPE 'Unicode_esc_char'
```
Unicode Delimited Identifier

--- U" Unicode_delimiter_body " — UESCAPE — 'Unicode_esc_char' ---

1101A495

DateTime and Interval Data Types

DATE Data Type

INTERVAL DAY Data Type

INTERVAL DAY TO HOUR Data Type

INTERVAL DAY TO MINUTE Data Type
INTERVAL DAY TO SECOND Data Type

INTERVAL HOUR Data Type

INTERVAL HOUR TO MINUTE Data Type

INTERVAL HOUR TO SECOND Data Type

INTERVAL MINUTE Data Type
INTERVAL MINUTE TO SECOND Data Type

INTERVAL MONTH Data Type

INTERVAL SECOND Data Type

INTERVAL YEAR Data Type

INTERVAL YEAR TO MONTH Data Type

TIME Data Type
Chapter 1: Data Types and Literals
Decimal/Numeric Data Types

TIME WITH TIME ZONE Data Type

TIMESTAMP Data Type

TIMESTAMP WITH TIME ZONE Data Type

Decimal/Numeric Data Types

BIGINT Data Type

BYTEINT Data Type
DECIMAL/NUMERIC Data Types

FLOAT/REAL/DOUBLE PRECISION Data Types

INTEGER Data Type

SMALLINT Data Type

PERIOD Data Types

PERIOD(DATE) Data Type
Chapter 1: Data Types and Literals
UDT Data Types

<table>
<thead>
<tr>
<th>PERIOD(TIME) Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD(TIME)</td>
</tr>
<tr>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>attributes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PERIOD(TIME WITH TIME ZONE) Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD(TIME)</td>
</tr>
<tr>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>WITH TIME ZONE</td>
</tr>
<tr>
<td>attributes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PERIOD(TIMESTAMP) Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD(TIMESTAMP)</td>
</tr>
<tr>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>attributes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PERIOD(TIMESTAMP WITH TIME ZONE) Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD(TIMESTAMP)</td>
</tr>
<tr>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>WITH TIME ZONE</td>
</tr>
<tr>
<td>attributes</td>
</tr>
</tbody>
</table>

UDT Data Types

UDT Data Type

<table>
<thead>
<tr>
<th>UDT Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>udt_name</td>
</tr>
<tr>
<td>SYSUDTLIB.</td>
</tr>
<tr>
<td>attribute</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VARIANT_TYPE UDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter_name</td>
</tr>
<tr>
<td>VARIANT_TYPE</td>
</tr>
</tbody>
</table>

34
Geospatial Data Types

**MBR Type**

```
  MBR
  └── SYSUDTLIB.MBR    
        └── attribute
```

**ST_Geometry Type**

```
  ST_GEOMETRY
  └── SYSUDTLIB.ST_GEOMETRY    
        └── attribute
```

**Well-Known Text Representation**

```
  point
  └── curve
  └── surface
  └── multipoint
  └── multicurve
  └── multisurface
      └── geometrycollection
          └── geosequence

  point
  └── POINT EMPTY
      (x y)

  curve
  └── LINESTRING EMPTY
      (x y)
```
Chapter 1: Data Types and Literals
Geospatial Data Types

surface

- POLYGON
  - EMPTY
  - ( EMPTY
    - ( x y ) )

multipoint

- MULTIPOINT
  - EMPTY
  - ( EMPTY
    - ( x y ) )

c multicurve

- MULTILINESTRING
  - EMPTY
  - ( EMPTY
    - ( x y ) )

multisurface

- MULTIPOLYGON
  - EMPTY
  - ( EMPTY
    - ( x y ) )

geometrycollection

- GEOMETRYCOLLECTION
  - EMPTY
  - ( point
    - curve
    - surface
    - multipoint
    - multicurve
    - multisurface
    - geometrycollection
    - geosequence )
Chapter 1: Data Types and Literals

Default Value Control Phrases

COMPRESS Phrase

DEFAULT Phrase

NOT NULL Phrase

WITH DEFAULT Phrase
Output Format Phrases

**AS**

\[
\text{value_expression} \quad \text{AS} \quad \text{name}
\]

**FORMAT**

\[
\text{FORMAT} \quad \text{format_string}'
\]

**NAMED**

\[
\text{(expression)} \quad \text{(NAMED \ \text{name})}
\]

**TITLE**

\[
\text{TITLE} \quad \text{quotestring}
\]
CHAPTER 2 SQL Functions and Expressions

Aggregate Functions

AVG

\[
\text{AVG}(\text{value_expression}) \\
\text{AVG}(\text{DISTINCT value_expression}) \\
\text{AVG}(\text{ALL value_expression})
\]

CORR

\[
\text{CORR}(\text{value_expression}_1, \text{value_expression}_2)
\]

COUNT

\[
\text{COUNT}(\text{DISTINCT value_expression}) \\
\text{COUNT}(\text{DISTINCT ALL value_expression})
\]

COVAR_POP

\[
\text{COVAR_POP}(\text{value_expression}_1, \text{value_expression}_2)
\]

COVAR_SAMP

\[
\text{COVAR_SAMP}(\text{value_expression}_1, \text{value_expression}_2)
\]

GROUPING

\[
\text{GROUPING}(\text{expression})
\]
Chapter 2: SQL Functions and Expressions

Aggregate Functions

KURTOSIS

\[
\text{KURTOSIS} \ ( value\_expression )
\]

MAX

\[
\text{MAX} \ ( value\_expression )
\]

MIN

\[
\text{MIN} \ ( value\_expression )
\]

REGR_AVGX

\[
\text{REGR\_AVGX} \ ( \text{dependent\_variable\_expression}, \text{independent\_variable\_expression} )
\]

REGR_AVGY

\[
\text{REGR\_AVGY} \ ( \text{dependent\_variable\_expression}, \text{independent\_variable\_expression} )
\]

REGR_COUNT

\[
\text{REGR\_COUNT} \ ( \text{dependent\_variable\_expression}, \text{independent\_variable\_expression} )
\]

REGR_INTERCEPT

\[
\text{REGR\_INTERCEPT} \ ( \text{dependent\_variable\_expression}, \text{independent\_variable\_expression} )
\]

REGR_R2

\[
\text{REGR\_R2} \ ( \text{dependent\_variable\_expression}, \text{independent\_variable\_expression} )
\]
Chapter 2: SQL Functions and Expressions
Aggregate Functions

REGR_SLOPE

--- REGR_SLOPE --- ( dependent_variable_expression, independent_variable_expression ) ---

1101B419

REGR_SXX

--- REGR_SXX --- ( dependent_variable_expression, independent_variable_expression ) ---

1101B420

REGR_SXY

--- REGR_SXY --- ( dependent_variable_expression, independent_variable_expression ) ---

1101B421

REGR_SYY

--- REGR_SYY --- ( dependent_variable_expression, independent_variable_expression ) ---

1101B422

SKEW

--- SKEW --- ( value_expression ) ---

DISTINCT ALL

1101B428

STDDEV_POP

--- STDDEV_POP --- ( value_expression ) ---

DISTINCT ALL

1101B424

STDDEV_SAMP

--- STDDEV_SAMP --- ( value_expression ) ---

DISTINCT ALL

1101B425

SUM

--- SUM --- ( value_expression ) ---

DISTINCT ALL

1101B423
### VAR_POP

\[
\text{VAR\_POP} \quad (\text{DISTINCT/ALL}\quad \text{value\_expression})
\]

### VAR_SAMP

\[
\text{VAR\_SAMP} \quad (\text{DISTINCT/ALL}\quad \text{value\_expression})
\]

## Arithmetic Operators and Functions/Trigonometric/Hyperbolic Functions

### Arithmetic Operators

Teradata Database supports the following arithmetic operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>**</td>
<td>Exponentiate</td>
</tr>
<tr>
<td></td>
<td>This is a Teradata extension to the ANSI SQL-99 standard.</td>
</tr>
<tr>
<td>*</td>
<td>Multiply</td>
</tr>
<tr>
<td>/</td>
<td>Divide</td>
</tr>
<tr>
<td>MOD</td>
<td>Modulo (remainder).</td>
</tr>
<tr>
<td></td>
<td>MOD calculates the remainder in a division operation.</td>
</tr>
<tr>
<td></td>
<td>For example, 60 MOD 7 = 4: 60 divided by 7 equals 8, with a remainder of 4. The result takes the sign of the dividend, thus:</td>
</tr>
<tr>
<td></td>
<td>-17 MOD 4 = -1</td>
</tr>
<tr>
<td></td>
<td>-17 MOD -4 = -1</td>
</tr>
<tr>
<td></td>
<td>17 MOD -4 = 1</td>
</tr>
<tr>
<td></td>
<td>17 MOD 4 = 1</td>
</tr>
<tr>
<td></td>
<td>This is a Teradata extension to the ANSI SQL-99 standard.</td>
</tr>
<tr>
<td>+</td>
<td>Add</td>
</tr>
<tr>
<td>-</td>
<td>Subtract</td>
</tr>
<tr>
<td>+</td>
<td>Unary plus (positive value)</td>
</tr>
<tr>
<td>-</td>
<td>Unary minus (negative value)</td>
</tr>
</tbody>
</table>
Chapter 2: SQL Functions and Expressions

Arithmetic Operators and Functions/Trigonometric/Hyperbolic Functions

--- ABS ---

ABS — ( arg ) ---

--- CASE_N ---

CASE_N — ( conditional_expression ) ---

NO CASE

OR UNKNOWN

UNKNOWN

--- DEGREES/RADIANS ---

DEGREES ( arg ) ---

RADIANS ---

--- EXP ---

EXP — ( arg ) ---

--- LN ---

LN — ( arg ) ---

--- LOG ---

LOG — ( arg ) ---
### NULLIFZERO

- **NULLIFZERO** — (arg) —

### RANDOM

- **RANDOM** — (lower_bound, upper_bound) —

### RANGE_N

- **RANGE_N** — (test_expression) — BETWEEN

- **A**
  - start_expression — AND — end_expression
  - start_expression — AND — end_expression

- **B**
  - start_expression — AND — end_expression
  - end_expression

- **C**
  - range_list

- **D**
  - start_expression — AND — end_expression

### SQRT

- **SQRT** — (arg) —
Chapter 2: SQL Functions and Expressions

Trigonometric Functions

**WIDTH_BUCKET**

\[\text{WIDTH\_BUCKET} \left( \text{value\_expression, lower\_bound, upper\_bound, partition\_count} \right)\]

**ZEROIFNULL**

\[\text{ZEROIFNULL} \left( \text{arg} \right)\]

Hyperbolic Functions

**COSH, SINH, TANH, ACOSH, ASINH, ATANH**

**COS, SIN, TAN, ACOS, ASIN, ATAN, ATAN2**
Attribute Functions

BYTES

\[
\text{BYTES}(\text{byte_expression})
\]

CHARACTERS

\[
\text{CHARACTERS}(\text{string_expression})
\]

\[
\text{CHARS}
\]

\[
\text{CHAR}
\]

CHARACTER_LENGTH

\[
\text{CHARACTER_LENGTH}(\text{string_expression})
\]

\[
\text{CHAR_LENGTH}
\]

DEFAULT

\[
\text{DEFAULT}(\text{column_name})
\]

FORMAT

\[
\text{FORMAT}(\text{column_name})
\]

OCTET_LENGTH

\[
\text{OCTET_LENGTH}(\text{string_expression}, \text{character_set_name})
\]

TITLE

\[
\text{TITLE}(\text{expression})
\]
Chapter 2: SQL Functions and Expressions
Built-In Functions

**TYPE**

```
TYPE ( expression )
```

**ACCOUNT**

```
ACCOUNT
```

**CURRENT_DATE**

```
CURRENT_DATE
```

**CURRENT_ROLE**

```
CURRENT_ROLE
```

**CURRENT_TIME**

```
CURRENT_TIME (fractional_precision)
```

**CURRENT_TIMESTAMP**

```
CURRENT_TIMESTAMP (fractional_precision)
```

**CURRENT_USER**

```
CURRENT_USER
```
Chapter 2: SQL Functions and Expressions
Built-In Functions

DATABASE

---- DATABASE ----

FF07R002

DATE

---- DATE ----

FF07D134

PROFILE

---- PROFILE ----

KZ01A006

ROLE

---- ROLE ----

KZ01A007

SESSION

---- SESSION ----

FF07R003

TIME

---- TIME ----

FF07D271

USER

---- USER ----

FF07D272
CASE Expressions

Valued CASE Expression

```sql
CASE value_expression_1
  WHEN value_expression_n THEN scalar_expression_n
  ELSE scalar_expression_m
END
```

Searched CASE Expression

```sql
CASE
  WHEN search_condition_n THEN scalar_expression_n
  ELSE scalar_expression_m
END
```

COALESCE Expression

```sql
COALESCE (scalar_expression_n, scalar_expression_2)
```

NULLIF Expression

```sql
NULLIF (scalar_expression1, scalar_expression2)
```
## Comparison Operators

Teradata Database supports the following comparison operators:

<table>
<thead>
<tr>
<th>ANSI Operator</th>
<th>Teradata Extensions</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>EQ</td>
<td>Tests for equality.</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>^=</td>
<td>Tests for inequality.</td>
</tr>
<tr>
<td></td>
<td>NE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>NOT=</td>
<td></td>
</tr>
<tr>
<td>&lt;</td>
<td>LT</td>
<td>Tests for less than.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>LE</td>
<td>Tests for less than or equal.</td>
</tr>
<tr>
<td>&gt;</td>
<td>GT</td>
<td>Tests for greater than.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>GE</td>
<td>Tests for greater than or equal.</td>
</tr>
</tbody>
</table>

### Comparison Operators in Logical Expressions

- `scalar_expression comparison_operator scalar_expression`
- `expression_1 operator expression_2`
- `expression_1 operator quantifier ( constant )`
- `expression_1 operator quantifier ( subquery )`
- `expression_1 operator quantifier ( subquery )`
Data Type Conversions

CAST

```sql
CAST ( expression AS ansi_sql_data_type )
```

Teradata Conversion Syntax

```sql
expression ( data_type, data_attribute, data_type, data_attribute )
```

Byte Conversion

CAST

```sql
CAST ( byte_expression AS byte_data_type )
```

```sql
UDT_data_type
```

```sql
data_attribute
```
Chapter 2: SQL Functions and Expressions
Character-to-Character Conversion

Teradata Conversion

\[
\text{byte_expression} \leftarrow \left( \text{byte_data_type}, \text{data_attribute}, \text{byte_data_type}, \text{data_attribute} \right)
\]

Character-to-Character Conversion

CAST

\[
\text{CAST} \leftarrow \left( \text{character_expression} \rightarrow \text{AS} \text{character_data_type} \right)
\]

Teradata Conversion

\[
\text{character_expression} \leftarrow \left( \text{character_data_type}, \text{data_attribute}, \text{character_data_type}, \text{data_attribute} \right)
\]
Character-to-DATE Conversion

CAST

```
CAST ( character_expression AS DATE )
```

Teradata Conversion

```
( character_expression ) AS DATE ( data_attribute , data_attribute )
```

Character-to-INTERVAL Conversion

CAST

```
CAST ( character_expression AS interval_data_type )
```

Teradata Conversion

```
( character_expression ) interval_data_type ( data_attribute , data_attribute )
```
Character-to-Period Conversion

CAST

CAST (character_expression AS period_data_type)

Character-to-Numeric Conversion

CAST

CAST (character_expression AS numeric_data_type)

Teradata

CAST (character_expression (numeric_data_type))

Character-to-TIME Conversion

CAST

CAST (character_expression AS TIME (fractional_seconds_precision) WITH TIME ZONE time_data_attribute)
Teradata Conversion

```
- character_expression -> ( data_attribute , TIME ( fractional_seconds_precision )
  )
  , data_attribute
  WITH TIME ZONE
```

Character-to-TIMESTAMP Conversion

CAST

```
- CAST -> ( - character_expression -> AS TIMESTAMP ( fractional_seconds_precision )
  )
  , WITH TIME ZONE
  , timestamp_data_attribute
```

Teradata Conversion

```
- character_expression -> ( data_attribute , TIME ( fractional_seconds_precision )
  )
  , data_attribute
  WITH TIME ZONE
```
Character-to-UDT Conversion

CAST

```
CAST (character_expression AS UDT_data_definition)
```

DATE-to-Character Conversion

CAST

```
CAST (date_expression AS character_data_type)
```

CHARACTER SET server_character_set

Teradata Conversion

```
data_expression (character_data_type, data_attribute)
```

CHARACTER SET server_character_set
DATE-to-DATE Conversion

CAST

CAST (date_expression AS DATE)

Teradata Conversion

--- date_expression --- AS --- DATE ---

, data_attribute

, DATE

, data_attribute

DATE-to-Numeric Conversion

CAST

CAST (date_expression AS numeric_data_type)

numeric_data_attribute
Chapter 2: SQL Functions and Expressions

DATE-to-Period Conversion

**Teradata Conversion**

```
CAST ( date_expression numeric_data_type )
```

**DATE-to-Period Conversion**

**CAST**

```
CAST ( date_expression AS period_data_type )
```

**DATE-to-TIMESTAMP Conversion**

**CAST**

```
CAST ( date_expression AS TIMESTAMP (fractional_seconds_precision) WITH TIME ZONE )
```
**Teradata Conversion**

```sql
CAST

date_expression

(data_attribute, WITH TIME ZONE)

TIMESTAMP

(fractional_seconds_precision)
```

**DATE-to-UDT Conversion**

**CAST**

```sql
CAST (date_expression AS UDT_data_definition)
```

**INTERVAL-to-Character Conversion**

**CAST**

```sql
CAST (interval_expression AS character_data_type)

CHARACTER SET server_character_set

character_data_attribute
```
INTERVAL-to-INTERVAL Conversion

CAST

Teradata Conversion

INTERVAL-to-INTERVAL Conversion

CAST

Teradata Conversion
INTERVAL-to-Numeric Conversion

CAST

```
CAST ( interval_expression AS numeric_data_type )
```

Teradata Conversion

```
interval_expression ( data_attribute , numeric_data_type )
```

INTERVAL-to-UDT Conversion

CAST

```
CAST ( interval_expression AS UDT_data_definition )
```

Numeric-to-Character Conversion

CAST

```
CAST ( numeric_expression AS character_data_type )
```
Chapter 2: SQL Functions and Expressions

Numeric-to-DATE Conversion

Teradata Conversion

```
numeric_expression character_data_type(data_attribute, )
```

```
numeric_expression CAST AS DATE(data_attribute, )
```

```
numeric_expression CAST AS numeric_expression interval_data_type(interval_data_attribute, )
```

Numeric-to-DATE Conversion

CAST

```
CAST (numeric_expression AS DATE(data_attribute, )
```

Numeric-to-INTERVAL Conversion

CAST

```
CAST (numeric_expression AS interval_data_type(interval_data_attribute, )
```

62 SQL Quick Reference
Teradata Conversion

- \( \text{numeric} \text{expression} \rightarrow \{ \text{interval_data_type} \text{, data_attribute} \text{, data_attribute} \} \)

Numeric-to-Numeric Conversion

CAST

- CAST \( \rightarrow \) \( \text{numeric} \text{expression} \rightarrow \text{AS} \text{numeric} \text{data_type} \text{numeric} \text{data_attribute} \)

Teradata Conversion

- \( \text{numeric} \text{expression} \rightarrow \{ \text{numeric} \text{data_type} \text{, data_attribute} \text{, data_attribute} \} \)

Numeric-to-UDT Conversion

CAST

- CAST \( \rightarrow \) \( \text{numeric} \text{expression} \rightarrow \text{AS} \text{UDT_data_definition} \)
Period-to-Character Conversion

CAST

```
CAST ( period_expression AS character_data_type )
```

CHARACTER SET server_character_set character_data_attribute

Period-to-DATE Conversion

CAST

```
CAST ( period_expression AS DATE )
```

date_data_attribute

Period-to-Period Conversion

CAST

```
CAST ( period_expression AS period_data_type )
```

period_data_attribute
**Period-to-TIME Conversion**

CAST

```
CAST (period_expression AS TIME
(fractional_seconds_precision) WITH TIME ZONE time_data_attribute)
```

**Period-to-TIMESTAMP Conversion**

CAST

```
CAST (period_expression AS TIMESTAMP
(fractional_seconds_precision)
WITH TIME ZONE timestamp_data_attribute)
```

**TIME-to-Character Conversion**

CAST

```
CAST (time_expression AS character_data_type
CHARACTER SET server_character_set character_data_attribute)
```
**Teradata Conversion**

```
CAST
```

```
<table>
<thead>
<tr>
<th>time_expression</th>
<th>(</th>
<th>character_data_type</th>
<th>data_attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>,</td>
<td>data_attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CHARACTER SET</td>
<td>server_character_set</td>
</tr>
</tbody>
</table>
```

**TIME-to-Period Conversion**

```
CAST
```

```
<table>
<thead>
<tr>
<th>CAST</th>
<th>(</th>
<th>time_expression</th>
<th>AS</th>
<th>period_data_type</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>period_data_attribute</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

**TIME-to-TIME Conversion**

```
CAST
```

```
<table>
<thead>
<tr>
<th>CAST</th>
<th>(</th>
<th>time_expression</th>
<th>AS</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>(fractional_seconds_precision)</td>
<td>WITH TIME ZONE</td>
<td>time_data_attribute</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>time_data_attribute</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Teradata Conversion

```
TIME-to-TIMESTAMP Conversion

CAST

--time_expression-- ( TIME
  (fractional_seconds_precision)
, data_attribute
  WITH TIME ZONE
, data_attribute
, TIME
  (fractional_seconds_precision)
) AS
  TIMESTAMP
  (timestamp_data_attribute
  WITH TIME ZONE)
```

TIME-to-TIMESTAMP Conversion

CAST

```
-- time_expression -- AS -- TIMESTAMP
  (fractional_seconds_precision)

WITH TIME ZONE
  timestamp_data_attribute
```
Chapter 2: SQL Functions and Expressions

TIME-to-UDT Conversion

CAST

CAST (time_expression AS UDT_data_definition)

TIMESTAMP-to-Character Conversion

CAST

CAST (timestamp_expression AS character_data_type)

CHARACTER SET server_character_set

Teradata Conversion

timestamp_expression (character_data_type data_attribute

CHARACTER SET server_character_set)

data_attribute

1101A340

1101A269

1101B277
TIMESTAMP-to-DATE Conversion

CAST

```
CAST (timestamp_expression AS DATE) date_data_attribute
```

Teradata Conversion

```
( timestamp_expression AS (data_attribute , data_attribute ) )
```

TIMESTAMP-to-Period Conversion

CAST

```
CAST (timestamp_expression AS period_data_type ) period_data_attribute
```

TIMESTAMP-to-TIME Conversion

CAST

```
CAST (timestamp_expression AS TIME WITH TIME ZONE)
```

1101A270

1101B278

1101A608

1101A271
Teradata Conversion

\[
\text{TIMESTAMP-to-TIMESTAMP Conversion}\\
\text{CAST}\\
\]

\[
\text{CAST} \left( \text{\texttt{timestamp_expression}} \right) \rightarrow \text{\texttt{TIME} (data_attribute, \frac{\text{fractional_seconds_precision}}{10000}) WITH TIME ZONE}
\]

TIMESTAMP-to-TIMESTAMP Conversion

\[
\text{CAST} \left( \text{\texttt{timestamp_expression}} \rightarrow \text{\texttt{TIMESTAMP (data_attribute, \frac{\text{fractional_seconds_precision}}{10000}) WITH TIME ZONE}}
\]

\[
\text{data_attribute}
\]

\[
\text{1101B279}
\]

\[
\text{1101A272}
\]
**TIMESTAMP-to-UDT Conversion**

**CAST**

```
CAST ( timestamp_expression AS UDT_data_definition )
```

**UDT-to-Byte Conversion**

**CAST**

```
CAST ( UDT_expression AS byte_data_definition )
```
UDT-to-Character Conversion

CAST

CAST — ( UDT_expression — AS — character_data_definition — )

UDT-to-DATE Conversion

CAST

CAST — ( UDT_expression — AS — DATE — date_data_attribute — )
UDT-to-INTERVAL Conversion

**CAST**

\[
\text{CAST} \ ( \text{UDT} \text{expression} \text{ AS } \text{interval} \text{ data} \text{ definition} )
\]

---

UDT-to-Numeric Conversion

**CAST**

\[
\text{CAST} \ ( \text{UDT} \text{expression} \text{ AS } \text{numeric} \text{ data} \text{ definition} )
\]

---
Teradata Conversion

UDT-to-TIME Conversion

CAST

Teradata Conversion
Chapter 2: SQL Functions and Expressions

UDT-to-TIMESTAMP Conversion

CAST

CAST ( UDT_expression AS TIMESTAMP (fractional_seconds_precision) )

WITH TIME ZONE

timestamp_data_attribute

Teradata Conversion

CAST ( UDT_expression AS TIMESTAMP (fractional_seconds_precision) )

data_attribute ,

WITH TIME ZONE

data_attribute

UDT-to-UDT Conversion

CAST

CAST ( UDT_expression AS UDT_data_definition )

DateTime and Interval Functions and Expressions

Arithmetic Operators and Result Types

The following arithmetic operations are permitted for DateTime and Interval data types:
Chapter 2: SQL Functions and Expressions

DateTime and Interval Functions and Expressions

### ANSI DateTime Expressions

**date_time_expression Syntax**

\[
\text{date_time_term} \\
\text{interval_expression} \rightarrow + \text{date_time_term} \\
\text{date_time_expression} \rightarrow \pm \text{interval_term}
\]

**date_time_term Syntax**

\[
\text{date_time_primary} \\
\text{AT} \quad \text{LOCAL} \\
\text{TIME ZONE} \rightarrow \text{interval_expression}
\]

### ANSI Interval Expressions

**interval_expression Syntax**

\[
\text{interval_term} \\
\text{interval_expression} \rightarrow \pm \text{interval_term} \\
(\text{date_time_expression} \rightarrow \text{date_time_term} \rightarrow \text{start}) \rightarrow \text{TO} \rightarrow \text{end}
\]
**interval_term Syntax**

\[\pm interval\_primary \]

\[\frac{interval\_term \times numeric\_factor}{interval\_term \times interval\_factor} \]

**numeric_term Syntax**

\[numeric\_factor \]

\[\frac{numeric\_term \times interval\_factor}{numeric\_term \times numeric\_factor} \]

**numeric_factor Syntax**

\[\pm numeric\_primary \]

**ADD_MONTHS (DATE Syntax)**

\[ADD\_MONTHS\ (date\_expression, integer\_expression) \]

**ADD_MONTHS (TIMESTAMP Syntax)**

\[ADD\_MONTHS\ (timestamp\_expression, integer\_expression) \]

**EXTRACT**

\[EXTRACT\ (YEAR \__FROM\ value) \]

\[MONTH\]

\[DAY\]

\[HOUR\]

\[MINUTE\]

\[SECOND\]

\[TIMEZONE\_HOUR\]

\[TIMEZONE\_MINUTE\]
Hash-Related Functions

**HASHAMP**

```
HASHAMP(expression)
```

**HASHBAKAMP**

```
HASHBAKAMP(expression)
```

**HASHBUCKET**

```
HASHBUCKET(expression)
```

**HASHROW**

```
HASHROW(expression)
```
Logical Predicates

ANY/ALL/SOME Quantifiers

BETWEEN/NOT BETWEEN

CONTAINS

EXISTS/NOT EXISTS
**IN/NOT IN (Syntax 1)**

```
expression_1 NOT IN expression_2
```

```
constant OR (signed_constant_1 TO signed_constant_2)
```

```
datetime_literal
```

**IN/NOT IN (Syntax 2)**

```
expression NOT IN (subquery)
```

```
NOT (expression)
```

**IS NULL/IS NOT NULL**

```
expression IS NULL
```

```
NOT
```

**IS UNTIL_CHANGED/IS NOT UNTIL_CHANGED**

```
END(period_value_expression) IS UNTIL_CHANGED
```

```
NOT
```

1101A309

1101A639

HH01A042
**LIKE**

```
expression LIKE pattern_expression ESCAPE escape_character
```

```
expression NOT LIKE (subquery) ESCAPE escape_character
```

```
(expression) NOT LIKE ALL | ANY | SOME (subquery) ESCAPE escape_character
```

```
(expression) NOT LIKE ALL | ANY | SOME pattern_expression ESCAPE escape_character
```

**MEETS**

```
period_expression NOT MEETS period_expression
```

```
datetime_expression NOT MEETS period_expression
```

**OVERLAPS**

```
(row_subquery period_expression)
```

```
(row_subquery datetime_expression, interval_expression)
```

```
(row_subquery datetime_expression, interval_expression)
```

```
(row_subquery period_expression)
```
Chapter 2: SQL Functions and Expressions
Logical Predicates

<table>
<thead>
<tr>
<th>PRECEDES</th>
</tr>
</thead>
<tbody>
<tr>
<td>period_expression</td>
</tr>
<tr>
<td>NOT</td>
</tr>
<tr>
<td>datetime_expression</td>
</tr>
<tr>
<td>PRECEDES</td>
</tr>
<tr>
<td>period_expression</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SUCCEEDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>period_expression</td>
</tr>
<tr>
<td>NOT</td>
</tr>
<tr>
<td>datetime_expression</td>
</tr>
<tr>
<td>SUCCEEDS</td>
</tr>
<tr>
<td>period_expression</td>
</tr>
</tbody>
</table>

PRECEDES

SUCCEEDS
Ordered Analytical Functions

Window Aggregate Functions (AVG, CORR, COUNT, COVAR_POP, COVAR_SAMP, MAX, MIN, REGR_AVGX, REGR_AVGY, REGR_COUNT, REGR_INTERCEPT, REGR_R2, REGR_SLOPE, REGR_SXX, REGR_SXY, REGR_SYY, STDDEV_POP, STDDEV_SAMP, SUM, VAR_POP, VAR_SAMP)

AVG - (value_expression)
COUNT - (value_expression)
COVAR_POP - (value_expression_1, value_expression_2)
COVAR_SAMP - (value_expression_1, value_expression_2)
CORR - (value_expression_1, value_expression_2)
MAX - (value_expression)
MIN - (value_expression)
REGR_AVGX - (dependent_variable_expression, independent_variable_expression)
REGR_AVGY - (dependent_variable_expression, independent_variable_expression)
REGR_COUNT - (dependent_variable_expression, independent_variable_expression)
REGR_INTERCEPT - (dependent_variable_expression, independent_variable_expression)
REGR_R2 - (dependent_variable_expression, independent_variable_expression)
REGR_SLOPE - (dependent_variable_expression, independent_variable_expression)
REGR_SXX - (dependent_variable_expression, independent_variable_expression)
REGR_SXY - (dependent_variable_expression, independent_variable_expression)
REGR_SYY - (dependent_variable_expression, independent_variable_expression)
STDDEV_POP - (value_expression)
STDDEV_SAMP - (value_expression)
SUM - (value_expression)
VAR_POP - (value_expression)
VAR_SAMP - (value_expression)
Chapter 2: SQL Functions and Expressions
Ordered Analytical Functions

```
window

OVER (PARTITION BY column_reference

ORDER BY value_expression ASC DESC

RESET WHEN condition

ROWS UNBOUNDED PRECEDING

value PRECEDING

CURRENT ROW

rows BETWEEN UNBOUNDED PRECEDING AND value PRECEDING

value FOLLOWING

CURRENT ROW

value FOLLOWING

value FOLLOWING

value FOLLOWING

CSUM

CSUM (value_expression, sort_expression ASC DESC)

1101A398

1101B464
```
Chapter 2: SQL Functions and Expressions
Ordered Analytical Functions

MAVG

\[
\mathrm{MAVG} \left( -value\_expression, -width, -sort\_expression \right) \quad \text{ASC} \quad \text{DESC}
\]

MDIFF

\[
\mathrm{MDIFF} \left( -value\_expression, -width, -sort\_expression \right) \quad \text{ASC} \quad \text{DESC}
\]

MLINREG

\[
\mathrm{MLINREG} \left( -value\_expression, -width, -sort\_expression \right) \quad \text{ASC} \quad \text{DESC}
\]

MSUM

\[
\mathrm{MSUM} \left( -value\_expression, -width, -sort\_expression \right) \quad \text{ASC} \quad \text{DESC}
\]

PERCENT_RANK

\[
\text{PERCENT\_RANK() \quad OVER \quad \left( \right) \quad \text{PARTITION BY} \quad \text{column\_reference} \quad \text{ORDER BY} \quad value\_expression \quad \text{ASC} \quad \text{DESC} \quad \text{RESET WHEN} \quad \text{condition}
\]

Chapter 2: SQL Functions and Expressions
Ordered Analytical Functions

**QUANTILE**

\[
\text{QUANTILE} - (\text{quantile_constant}, \text{sort_expression})^\text{ASC/DESC}
\]

**RANK (Teradata-Specific Function)**

\[
\text{RANK} - (\text{sort_expression})^\text{ASC/DESC}
\]

**RANK (SQL:2008 Window Function)**

\[
\text{RANK}() \quad \text{OVER} \quad (\text{PARTITION BY} \quad \text{column_reference}) \quad \text{ORDER BY} \quad \text{value_expression}^\text{ASC/DESC} \quad \text{RESET WHEN} \quad \text{condition}
\]

**ROW_NUMBER**

\[
\text{ROW_NUMBER}() \quad \text{OVER} \quad (\text{PARTITION BY} \quad \text{column_reference}) \quad \text{ORDER BY} \quad \text{value_expression}^\text{ASC/DESC} \quad \text{RESET WHEN} \quad \text{condition}
\]
Period Functions and Operators

BEGIN

--- BEGIN(period_value_expression) ---
   1101A595

END

--- END(period_value_expression) ---
   1101A596

LAST

--- LAST(period_value_expression) ---
   1101A597

INTERVAL

--- INTERVAL(period_expression) ---- interval_qualifier ---
   1101A577

PRIOR

--- PRIOR(datetime_expression) ---
   1101A578

NEXT

--- NEXT(datetime_expression) ---
   1101A579
P_INTERSECT

\[ \text{period_expression} \text{ P_INTERSECT } \text{ period_expression} \]

LDIFF

\[ \text{period_expression} \text{ LDIFF } \text{ period_expression} \]

RDIFF

\[ \text{period_expression} \text{ RDIFF } \text{ period_expression} \]

P_NORMALIZE

\[ \text{period_expression} \text{ P_NORMALIZE } \text{ period_expression} \]

**Period Value Constructor**

<table>
<thead>
<tr>
<th>PERIOD (datetime_expression)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD (datetime_expression, datetime_expression)</td>
</tr>
<tr>
<td>PERIOD (datetime_expression, UNTIL_CHANGED)</td>
</tr>
</tbody>
</table>

**Arithmetic Operators**

\[ \text{period_expression} \quad + \quad \text{interval_expression} \]

\[ \text{interval_expression} \quad + \quad \text{period_expression} \]
Set Operators

Syntax for `query_term`

```
SELECT statement (query_expression)
```

Syntax for `query_factor`

```
query_term
query_factor INTERSECT ALL query_term
```

Syntax for `query_expression`

```
query_factor
query_expression UNION query_factor
MINUS ALL (query_expression) ORDER BY expression ASC DESC
```

INTERSECT Operator

```
query_expression_1 INTERSECT ALL query_expression_2
```

MINUS/EXCEPT Operator

```
query_expression_1 MINUS ALL query_expression_2
```

UNION Operator

```
query_expression_1 UNION ALL query_expression_2
```
String Operator and Functions

CHAR2HEXINT

\[
\text{CHAR2HEXINT} \left( \text{character\_string\_expression} \right)
\]

INDEX

\[
\text{INDEX} \left( \text{string\_expression\_1, string\_expression\_2} \right)
\]

LOWER

\[
\text{LOWER} \left( \text{character\_string\_expression} \right)
\]

POSITION

\[
\text{POSITION} \left( \text{string\_expression\_1 IN string\_expression\_2} \right)
\]

SOUNDEX

\[
\text{SOUNDEX} \left( \text{string\_expression} \right)
\]

STRING_CS

\[
\text{STRING\_CS} \left( \text{string\_expression} \right)
\]
SUBSTRING/SUBSTR (ANSI Syntax)

\[
\text{SUBSTRING} \ (\text{string_expression} \text{ FROM } n1 \text{ FOR } n2)
\]

SUBSTRING/SUBSTR (Teradata Syntax)

\[
\text{STRUBSTR} \ (\text{string_expression}, n1, n2)
\]

TRANSLATE

\[
\text{TRANSLATE} \ (\text{character_string_expression} \text{ USING } \text{source_repertoire_name} \text{ TO } \text{target_repertoire_name} \text{ WITH ERROR })
\]

TRANSLATE_CHK

\[
\text{TRANSLATE_CHK} \ (\text{character_string_expression} \text{ USING } \text{source_repertoire_name} \text{ TO } \text{target_repertoire_name})
\]

TRIM

\[
\text{TRIM} \ (\text{string_expression} \text{ FROM } \text{trim_expression} \text{ BOTH TRAILING LEADING})
\]

UPPER

\[
\text{UPPER} \ (\text{character_string_expression})
\]
Chapter 2: SQL Functions and Expressions

UDF Expressions

**VARGRAPHIC**

\[ \text{VARGRAPHIC} \ ( \text{character_string_expression} ) \]

**UDF Expressions**

**Scalar UDF Expression**

\[ \text{udf_name} \ ( \text{argument} ) \]

**Aggregate UDF Expression**

\[ \text{udf_name} \ ( \text{argument} ) \]
Chapter 2: SQL Functions and Expressions

UDT Expressions and Methods

**UDT Expression**

```
<table>
<thead>
<tr>
<th>database_name.</th>
<th>column_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>udf_name</td>
<td>(</td>
</tr>
<tr>
<td></td>
<td>argument</td>
</tr>
<tr>
<td>CAST</td>
<td>( expression AS udt_name )</td>
</tr>
<tr>
<td></td>
<td>constructor_name ( argument )</td>
</tr>
<tr>
<td></td>
<td>SYSUDTLIB.</td>
</tr>
<tr>
<td>method_name</td>
<td>(</td>
</tr>
<tr>
<td></td>
<td>argument</td>
</tr>
</tbody>
</table>
```

**NEW**

```
<table>
<thead>
<tr>
<th>NEW</th>
<th>constructor_name ( argument )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SYSUDTLIB.</td>
</tr>
</tbody>
</table>
```

**NEW VARIANT_TYPE**

```
<table>
<thead>
<tr>
<th>NEW VARIANT_TYPE</th>
<th>( expression AS alias_name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>table_name.column_name AS alias_name</td>
</tr>
</tbody>
</table>
```
Method Invocation

\[
\text{method_name} \rightarrow \text{column_name}, ( \text{argument} )
\]

\[
\text{CAST} \rightarrow ( \text{expression} \rightarrow \text{AS} \rightarrow \text{udt_name} )
\]

\[
\text{NEW} \rightarrow \text{constructor_name}, ( \text{argument} )
\]

\[
\text{database_name}. \text{table_name}. \text{udt_name} \rightarrow ( \text{argument} )
\]

\[
\text{SYSUDTLIB}. \text{udt_name} \rightarrow ( \text{argument} )
\]
ALTER FUNCTION

```
ALTER FUNCTION database_name.function_name
    (data_type UDT_name)
    SPECIFIC FUNCTION specific_function_name
    [EXECUTE PROTECTED]
    [EXECUTE ONLY]
    [COMPILE ONLY]
    SYSUDTLIB.
```

ALTER METHOD

```
ALTER METHOD database_name.function_name
    ([INSTANCE CONSTRUCTOR]
        [METHOD]
        [SPECIFIC METHOD]
    )
    SPECIFIC METHOD FOR UDT_name
    [EXECUTE PROTECTED]
    [EXECUTE ONLY]
    [COMPILE ONLY]
    SYSUDTLIB.
```
Chapter 3: SQL Data Definition Language

ALTER PROCEDURE (External Form)

ALTER PROCEDURE (SQL Form)

ALTER REPLICATION GROUP
**ALTER TABLE**

**Basic Table Parameters Modification Syntax**

```
ALTER TABLE database_name.table_name

  FALLBACK

  PROTECTION

  WITH JOURNAL TABLE = database_name.table_name

  NO DUAL

  BEFORE

  ON COMMIT DELETE ROWS

  PRESERVE

  LOG

  AFTER JOURNAL

  NO DUAL LOCAL NOT LOCAL

  CHECKSUM = integrity_checking_level

  DEFAULT FREESPACEx

  FREESPACEx = integer

  PERCENT

  DATABLOCKSIZE = data_block_size

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE

  IMMEDIATE
```
Data Type

- INTEGER
- SMALLINT
- BIGINT
- BYTEINT
- DATE
- TIME
- TIMESTAMP
- INTERVAL YEAR (precision) TO MONTH
- INTERVAL MONTH (precision)
- INTERVAL DAY (precision) TO HOUR MINUTE SECOND (fractional_seconds_precision)
- INTERVAL HOUR (precision) TO MINUTE SECOND (fractional_seconds_precision)
- INTERVAL MINUTE (precision) TO SECOND (fractional_seconds_precision)
- INTERVAL SECOND (precision)
- PERIOD(DATE)
- PERIOD(TIME)
- PERIOD(TIMESTAMP) (precision) WITH TIMEZONE
- REAL
- DOUBLE PRECISION
- FLOAT (integer)
- DECIMAL (integer, integer)
- NUMERIC (integer, integer)
Chapter 3: SQL Data Definition Language

ALTER TABLE

- CHAR
- BYTE
- GRAPHIC
- VARCHAR
- CHAR VARYING
- VARBYTE
- VARGRAPHIC
- LONG VARCHAR
- LONG VARGRAPHIC
- BINARY LARGE OBJECT
- BLOB
- CHARACTER LARGE OBJECT
- CLOB
- SYSUDTLIB.
- UDT_name
  - ST_Geometry
  - MBR
- (integer)
- (integer)
Primary Index Modification Syntax

```
ALTER TABLE  
  database_name  
  table_name  
  MODIFY  
    NOT UNIQUE  
  PRIMARY INDEX  
    index_name  
    NOT NAMED  
    (  
      column_name  
    )  

  NOT PARTITIONED  

  PARTITION BY  
    partitioning_expression  
      WITH DELETE  
      INSERT INTO  
        save_table  

  DROP RANGE  
    #Ln  
    WHERE  
      conditional_expression  
      BETWEEN  
        start_expression  
      AND  
        end_expression  

  ADD RANGE  
    #Ln  
    BETWEEN  
      start_expression  
    AND  
      end_expression  

  WITH  
    DELETE  
    INSERT INTO  
      save_table  
```

SQL Quick Reference
Chapter 3: SQL Data Definition Language

ALTER TRIGGER

Partitioned Primary Index Revalidation Syntax

```
ALTER TABLE table_name
    REVALIDATE PRIMARY INDEX
    WITH DELETE
    INTO save_table
```

Set Down/Reset Down Syntax

```
ALTER TABLE table_name
    SET DOWN
```

ALTER TRIGGER

```
ALTER TRIGGER trigger_name
    ENABLED
    DISABLED
    TIMESTAMP
```
ALTER TYPE

ALTER TYPE UDT_name

ADD DROP ATTRIBUTE attribute_name data_type

ADD DROP METHOD method_name SYSUDTLIB

ADD DROP INSTANCE CONSTRUCTOR

ADD DROP RETURNS data_type UDT_name SYSUDTLIB

ADD DROP LANGUAGE C CPP

ADD DROP PARAMETER STYLE SQL TD_GENERAL

ADD DROP DETERMINISTIC

ADD DROP NOT NO SQL

ADD DROP SPECIFIC METHOD SYSUDTLIB

ADD DROP specific_method_name

ADD DROP FOR UDT_name SYSUDTLIB

ADD DROP COMPILE

Chapter 3: SQL Data Definition Language
ALTER TYPE
BEGIN LOGGING

BEGIN LOGGING
  DENIALS
  WITH TEXT
  ON
    FIRST
    LAST
    FIRST AND LAST
    EACH
  A

A
  ALL
  ,
  operation
  GRANT
  BY
    database_name
    user_name
  B

B
  ON
    AUTHORIZATION
    authorization_name
    DATABASE
    database_name
    USER
    user_name
    TABLE
    object_name
    VIEW
    database_name
    user_name.
    MACRO
    user_name.
    PROCEDURE
    FUNCTION
    TYPE

1101W044
BEGIN QUERY LOGGING

BEGIN QUERY LOGGING
  WITH logging_option
  LIMIT limit_option
  AND limit_option
ON ALL
  user_name
    ACCOUNT = 'account_name'
      ( 'account_name' )
  user_name
    APPLNAME = 'application_name'
      ( 'application_name' )

SQL Quick Reference 105
COLLECT STATISTICS (Optimizer Form)

```
COLLECT STATISTICS
  USING SAMPLE
    INDEX
      UNIQUE
        index_name
          ALL
    COLUMN
      column_name
    PARTITION
      ( column_name_1 )
      ( column_name_2 )
      ORDER BY
        VALUES
          HASH
    ON
      table_name_1
        TEMPORARY
          database_name.
            user_name.
        VOLATILE
    FROM
      table_name_2
        TEMPORARY
          database_name.
            user_name.
        VOLATILE
    COLUMN
      column_name_3
      PARTITION
        ( column_name )
        ( column_name )
```

1101C322
COLLECT STATISTICS (Alternate Optimizer Form)

COLLECT STATISTICS
ON table_name
USING SAMPLE
(temporary VOLATILE)
(COLUMN column_name_1
PARTITION)
(COLUMN column_name_1
PARTITION)
(INDEX column_name_1
index_name)
(INDEX column_name_1
index_name)
FROM table_name
(temporary VOLATILE)
(COLUMN column_name_2)

COLLECT STATISTICS (Recollect Statistics)

COLLECT STATISTICS
ON table_name
;
Chapter 3: SQL Data Definition Language

COMMENT (Comment Placing Form)

CREATE AUTHORIZATION/
REPLACE AUTHORIZATION

CREATE

AUTHORIZATION

REPLACE

DOMAIN

'domain_name'

USER

'user_name'

PASSWORD

'password'

DEFINER

DEFAULT

INVOKER

DEFAULT

1101C227
CREATE CAST / REPLACE CAST

CREATE CAST ( source_predefined_data_type source_UDT_name ) AS source_predefined_data_type WITH SYSUDTLIB.

REPLACE CAST ( source_predefined_data_type source_UDT_name ) AS source_predefined_data_type WITH SYSUDTLIB.

SPECIFIC METHOD specific_method_name

SPECIFIC FUNCTION specific_function_name

FUNCTION function_name ( data_type UDT_name )

AS ASSIGNMENT ;
CREATE DATABASE

CREATE DATABASE name AS
FROM database_name

PERM = n
BYTES

SPOOL = n
BYTES

TEMPORARY = n
bytes

ACCOUNT = 'account_ID'

FALLBACK
PROTECTION

NO DUAL
BEFORE JOURNAL

AFTER JOURNAL

NO DUAL LOCAL NOT LOCAL

DEFAULT JOURNAL TABLE = database_name.table_name

CREATE ERROR TABLE

CREATE ERROR TABLE error_table_name
FOR database_name.
user_name.
data_table_name

;
CREATE FUNCTION /
REPLACE FUNCTION

CREATE  FUNCTION  database_name.
          user_name.

            function_name

                      (  parameter_name
                          data type )

  RETURNS  data type
         CAST FROM  data type

language_clause  —  SQL_data_access
SQL_data_access  —  language_clause

SPECIFIC  specific_function_name

CLASS  AGGREGATE
         AG
                   (  interim_size  — )

PARAMETER STYLE  SQL
             TD_GENERAL
                   JAVA

DETERMINISTIC

CALLED ON NULL INPUT

RETURNS NULL ON NULL INPUT

language_clause  —  SQL_data_access
SQL_data_access  —  language_clause

SPECIFIC  database_name.
          user_name.

CLASS  AGGREGATE
         AG
                   (  interim_size  — )

PARAMETER STYLE  SQL
             TD_GENERAL
                   JAVA

DETERMINISTIC

CALLED ON NULL INPUT

RETURNS NULL ON NULL INPUT
CREATE FUNCTION/ REPLACE FUNCTION

USING GLOP SET — GLOP_set_name

EXTERNAL

NAME

eexternal_function_name

---

DELIMITER

function_entry_name

---

EXTERNAL NAME

external_function_name

---

EXTERNAL SECURITY

DEFINER

authorization_name

---

INVOKER

1101A643
Chapter 3: SQL Data Definition Language

CREATE FUNCTION/ REPLACE FUNCTION

- CHAR
- BYTE
- GRAPHIC
- VARCHAR
- CHAR VARYING
- VARBYTE
- VARGRAPHIC
- LONG VARCHAR
- LONG VARGRAPHIC
- BINARY LARGE OBJECT
- BLOB
- CHARACTER LARGE OBJECT
- CLOB
- SYSUDTLIB
- UDT_name
- ST_Geometry
- MBR

(integer)
(G)
(K)
(M)

1101A536
CREATE FUNCTION (Table Form)

CREATE FUNCTION
  `database_name.`
  `function_name`

REPLACE

A ( parameter_name data_type )

B RETURNS TABLE
  ( column_name data_type )

VARYING COLUMNS ( maximum_output_columns )

C SQL_data_accesslanguage_clause

D SPECIFIC `database_name.`
  `specific_function_name`

PARAMETER STYLE
  SQL
  JAVA

DETERMINISTIC
  NOT
  CALLED ON NULL INPUT
CREATE FUNCTION (Table Form)

- MEMBER OF GLOP SET: GLOP_set_name
- EXTERNAL NAME: external_function_name
  - delimiter
  - F: delimiter - function_entry_name
  - D: delimiter
  - S: delimiter - name_on_server - delimiter - include_name
  - C: delimiter
  - L: delimiter - library_name
  - O: delimiter - name_on_server - delimiter - object_name
  - P: delimiter - package_name
  - S: delimiter - name_on_server - delimiter - source_name
  - JAR_ID: java_class_name.method_name
  - (primitive - [ ] - [ ] - [ ] - [ ])
  - object
  - [ ]

- PARAMETER STYLE: SQL
- EXTERNAL SECURITY: DEFINER
  - authorization_name
- INVOKE
Chapter 3: SQL Data Definition Language

CREATE FUNCTION (Table Form)

<table>
<thead>
<tr>
<th>Data Type</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>SMALLINT</td>
<td>BIGINT</td>
<td>BYTEINT</td>
</tr>
<tr>
<td>TIME</td>
<td>TIMESTAMP</td>
<td>WITH TIMEZONE</td>
<td></td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>(precision)</td>
<td>TO MONTH</td>
<td></td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>(precision)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>(precision)</td>
<td>TO HOUR</td>
<td>MINUTE</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>(precision)</td>
<td>TO MINUTE</td>
<td>SECOND</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>(precision)</td>
<td>TO SECOND</td>
<td></td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>(precision)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td></td>
<td>(precision)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td></td>
<td>(precision)</td>
<td>WITH TIMEZONE</td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
<td>(precision)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>REAL</td>
<td>DOUBLE PRECISION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>(integer)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>NUMERIC</td>
<td>(integer)</td>
<td></td>
</tr>
</tbody>
</table>
CREATE GLOBAL TEMPORARY TRACE TABLE

CREATE GLOBAL TEMPORARY TRACE TABLE
  
  | database_name. | table_name |
  | user_name.     |           |

  ( proc_ID BYTE(2) , sequence INTEGER

  | column_name | data type | data type attributes |

  ON COMMIT DELETE PRESERVE ROWS ;
Chapter 3: SQL Data Definition Language

CREATE GLOBAL TEMPORARY TRACE TABLE

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
</tr>
<tr>
<td>BYTEINT</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>(fractional_seconds_precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>TO MONTH</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>TO MONTH</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>TO HOUR MINUTE SECOND</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>TO MINUTE SECOND</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>TO SECOND</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td></td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
<td>WITH TIMEZONE</td>
</tr>
<tr>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>(integer)</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>(integer, integer)</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>(integer, integer)</td>
</tr>
</tbody>
</table>
Chapter 3: SQL Data Definition Language

CREATE GLOBAL TEMPORARY TRACE TABLE

Data Type Attributes

NOT NULL

UPPERCASE

CASESPECIFIC

NOT

FORMAT quotestring

TITLE quotestring

NAMED name

CHARACTER SET server_character_set
**CREATE GLOP SET**

```
CREATE GLOP SET
  database_name.
  user_name.
  GLOP_set_name
;  
```

**CREATE HASH INDEX**

```
CREATE HASH INDEX
  hash_index_name
  database_name.
  user_name.
;  
```

```
  ,

  FALLBACK
  PROTECTION
  CHECKSUM = integrity_checking_level

  ,

  column_name_1
  ON
  table_name

  BY
  (column_name_2)

  ORDER BY
  VALUES
  (column_name_3)
  HASH
  (column_name_3)
  (column_name_3)
;  
```
CREATE INDEX

CREATE INDEX

CREATE UNIQUE INDEX index_name ON table_name [INDEX ALL (order_column_name)]

ORDER BY

VALUES (order_column_name)

HASH

ON database_name.

user_name.

database_name.

user_name.

join_index_name

;
CREATE JOIN INDEX

CREATE JOIN INDEX

database_name.
user_name.

JOIN_INDEX_NAME

FALLBACK
PROTECTION
CHECKSUM = integrity_checking_level

AS SELECT

database_name.
table_name.
ROWID

EXTRACT – ( YEAR FROM date_expression ) AS expression_alias

MONTH

64

(a)

64

(a)

database_name.
table_name.
ROWID

column_1_name

column_2_name

database_name.
table_name.
ROWID

column_1_name

column_2_name

SUM – ( numeric_expression ) AS expression_alias

COUNT – ( column_expression )

EXTRACT – ( YEAR FROM date_expression )

MONTH

FROM

database_name.
table_name.

WHERE

search_condition

GROUP BY

column_name

column_position

ORDER BY

column_name

column_position

;
Chapter 3: SQL Data Definition Language

CREATE JOIN INDEX

joined_table

joined_table (joined_table)

joined_table INNER JOIN joined_table

joined_table LEFT JOIN joined_table

joined_table RIGHT OUTER JOIN joined_table

table_name

correlation_name

AS

indexes

PRIMARY INDEX index_name (primary_index_column)

PARTITION BY partitioning_expression

UNIQUE INDEX index_name (index_column_name)

INDEX index_name ALL (index_column_name)

ORDER BY VALUES HASH

VALUES

ORDER BY

PARTITION BY partitioning_expression

1101E050

1101I051
CREATE MACRO /  REPLACE MACRO

CREATE MACRO
CM
REPLACE MACRO

macro_name

database_name.

parameter_name

parameter_name - type declaration

A

LOCKING modifier

USING modifier

SQL_statement ;

2048

type attribute

A

1101G172
CREATE METHOD

CREATE METHOD method_name

(A) (B)

parameter_name data_type AS LOCATOR

RETURNS data_type FOR UDT_name

CAST FROM data_type

USING GLOP SET GLOP_set_name

(D) EXTERNAL

NAME external_function_name
delimiter method_entry_name
delimiter method_entry_name

delimiter name_on_server.delimiter include_name
delimiter library_name
delimiter name_on_server.delimiter object_name
delimiter name_on_server.delimiter source_name

EXTERNAL SECURITY DEFINER authorization_name

EXTERNAL SECURITY INVOKER authorization_name

;
<table>
<thead>
<tr>
<th>Data Type</th>
<th>Precision</th>
<th>Specialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BYTEINT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td></td>
<td>WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>(precision)</td>
<td>TO MONTH</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>(precision)</td>
<td>TO</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>(precision)</td>
<td>TO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HOUR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MINUTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECOND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>(precision)</td>
<td>TO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MINUTE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECOND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>(precision)</td>
<td>TO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECOND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>(precision)</td>
<td>TO</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SECOND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td>(precision)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fractional_seconds_precision</td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td>(precision)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
<td>(precision)</td>
<td>WITH TIMEZONE</td>
</tr>
<tr>
<td>REAL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>(integer)</td>
<td></td>
</tr>
<tr>
<td>DECIMAL</td>
<td>(integer)</td>
<td></td>
</tr>
<tr>
<td>NUMERIC</td>
<td>(integer)</td>
<td></td>
</tr>
</tbody>
</table>
CREATE ORDERING/ REPLACE ORDERING

CREATE ORDERING  FOR  UDT_name  ORDER FULL BY  MAP WITH

SPECIFIC METHOD
FOR UDT_name

FUNCTION

CREATE PROCEDURE (External Form) / REPLACE PROCEDURE (External Form)

CREATE PROCEDURE (External Form) / REPLACE PROCEDURE (External Form)

CREATE PROCEDURE

REPLACE

(database_name.
user_name.)

(procedure_name)

parameter_name

data type

IN
OUT
INOUT

DYNAMIC RESULT SETS - number_of_sets

language_clause

SQL data access

PARAMETER STYLE

SQL
td_general
java

PARAMETER STYLE

language_clause

SQL data access

PARAMETER STYLE

td_general
java

language_clause

SQL data access

PARAMETER STYLE

td_general
java

USING GLOP SET

GLOP_set_name

SQL SECURITY

privilege_option

EXTERNAL

NAME

external_function_name

delimiter

FUNCTION ENTRY NAME

function_entry_name

delimiter

(DELIMITER)

package_name

jar_name

java_data_type

java_method_name

java_class_name

package_name.

java_data_type

java_class_name

EXTERNAL SECURITY

DEFINER

authorization_name

INVOKER
### Data Type

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>Integer must be an integer number</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>Small Integer</td>
</tr>
<tr>
<td>BIGINT</td>
<td>Big Integer</td>
</tr>
<tr>
<td>BYTEINT</td>
<td>Byte Integer</td>
</tr>
<tr>
<td>DATE</td>
<td>Date</td>
</tr>
<tr>
<td>TIME</td>
<td>Time</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>Timestamp</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>Interval Year with precision and with time zone</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>Interval Month with precision and with time zone</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>Interval Day with precision and time zone</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>Interval Hour with precision and time zone</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>Interval Minute with precision and time zone</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>Interval Second with precision and time zone</td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td>Period of Date with precision and timezone</td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td>Period of Time with precision and timezone</td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
<td>Period of Timestamp with precision and timezone</td>
</tr>
<tr>
<td>REAL</td>
<td>Real</td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td>Double Precision</td>
</tr>
<tr>
<td>FLOAT</td>
<td>Float</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>Decimal</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>Numeric</td>
</tr>
</tbody>
</table>

**Example:**

```sql
CREATE PROCEDURE (External Form) / REPLACE PROCEDURE (External Form)
```
CREATE PROCEDURE (SQL Form) / REPLACE PROCEDURE

CREATE PROCEDURE (SQL Form) / REPLACE PROCEDURE

CREATE PROCEDURE procedure_name
    database_name.
    user_name.
    ( integer )
    SQL_data_access
    OUT
    parameter_name
    data_type
    INOUT
    number_of_sets
    SQL_SECURITY privilege_option
    statement

CHAR
BYTE
GRAPHIC
VARCHAR
CHAR VARYING
VARBYTE
VARGRAPHIC
LONG VARCHAR
LONG VARGRAPHIC
BINARY LARGE OBJECT
BLOB
CHARACTER LARGE OBJECT
CLOB
UDT_name
SYSUDTLIB.
ST_Geometry
MBR

condition_handler

DECLARE  CONTINUE  HANDLER
  EXIT
  condition_name  CONDITION
  FOR  SQLSTATE
    VALUE
      SQLSTATE
      VALUE
      SQLSTATE
      VALUE
      condition_name
      SQLSTATE
      VALUE
    SQLSTATE
    VALUE
  SQLSTATE
  SQLSTATE

open statement

OPEN  cursor_name
  USING
    SQL_identifier
    SQL_parameter

fetch statement

FETCH  cursor_name  INTO
  NEXT
  FIRST
  local_variable_name
  parameter_reference

assignment statement

SET  assignment_target =  assignment_source
Chapter 3: SQL Data Definition Language

CREATE PROCEDURE (SQL Form) / REPLACE PROCEDURE

condition statement

CASE
  operand_1
  WHEN operand_2 THEN statement
J
  WHEN conditional_expression THEN statement
  ELSE statement END CASE

IF conditional_expression THEN statement
G
ELSEIF conditional_expression THEN statement
H
ELSE statement END IF

iteration statement

WHILE conditional_expression DO statement END WHILE

LOOP statement END LOOP

FOR for_loop_variable AS cursor_name CURSOR FOR cursor_specification DO statement END FOR

REPEAT statement UNTIL conditional_expression END REPEAT

diagnostic statement

SIGNAL condition_name

RESIGNAL condition_name

GET DIAGNOSTICS parameter_name = statement_information_item

EXCEPTION condition_number parameter_name = condition_information_item

SQL Quick Reference 135
CREATE PROFILE

CREATE PROFILE profile_name AS

ACCOUNT = account_id,

DEFAULT DATABASE = database_name,

SPOOL = n

TEMPORARY = n

PASSWORD = EXPIRE = n

ATTRIBUTES

MINCHAR = n

MAXCHAR = n

DIGITS = c

SPECCHAR = c

MAXLOGONATTEMPTS = n

LOCKEDUSEREXPIRE = n

REUSE = n

RESTRICTWORDS = c

COST PROFILE = cost_profile_name

NULL
CREATE RECURSIVE VIEW /
REPLACE RECURSIVE VIEW

CREATE [REPLACE] RECURSIVE VIEW
  database_name.
  user_name.
  view_name
  (column_name)
  AS

LOCKING
  LOCK
  DATABASE
  database_name.
  user_name.
  table_name
  VIEW
  database_name.
  user_name.
  view_name
  ROW

SELECT
  DISTINCT
  expression
  table_name.*
  database_name.
  user_name.

FROM
  table_name
  joined_table
  correlation_name
  AS
  joined_table
  ON
  search_condition
  INNER
  LEFT
  RIGHT
  FULL
  CROSS JOIN
  database_name.
  user_name.
  table_name
  (subquery)
  derived_table_name
  (column_name)

WHERE
  search_condition

GROUP BY
  ordinary_grouping_set
  empty_grouping_set
  rollup_list
  cube_list
  grouping_sets_specification

HAVING
  condition

QUALIFY
  condition

UNION ALL
  seed_statement
  recursive_statement

UNION ALL
  database_name.
  user_name.
  table_name
  VIEW
  database_name.
  user_name.
  view_name
  ROW

ACCESS
  EXCLUSIVE
  MODE
  NOWAIT

LOCKING
  LOCK
  DATABASE
  database_name.
  user_name.
  table_name
  VIEW
  database_name.
  user_name.
  view_name
  ROW

SELECT
  DISTINCT
  expression
  table_name.*
  database_name.
  user_name.

FROM
  table_name
  joined_table
  correlation_name
  AS
  joined_table
  ON
  search_condition
  INNER
  LEFT
  RIGHT
  FULL
  CROSS JOIN
  database_name.
  user_name.
  table_name
  (subquery)
  derived_table_name
  (column_name)

WHERE
  search_condition

GROUP BY
  ordinary_grouping_set
  empty_grouping_set
  rollup_list
  cube_list
  grouping_sets_specification

HAVING
  condition

QUALIFY
  condition

UNION ALL
  seed_statement
  recursive_statement

UNION ALL
  database_name.
  user_name.
  table_name
  VIEW
  database_name.
  user_name.
  view_name
  ROW

ACCESS
  EXCLUSIVE
  MODE
  NOWAIT
CREATE REPLICATION GROUP

CREATE REPLICATION GROUP  
replication_group_name

A

A

( 

, 

table_name

)
CREATE REPLICATION RULESET/
REPLACE REPLICATION RULESET

CREATE ROLE
CREATE TABLE

Create Table Syntax

CREATE

set

MULTISET

GLOBAL TEMPORARY

VOLATILE

table_name

database_name.

user_name.

CT

<table>
<thead>
<tr>
<th>CREATE</th>
<th>TABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_name</td>
<td></td>
</tr>
</tbody>
</table>

A

FALLBACK

PROTECTION

WITH JOURNAL TABLE =

database_name.

user_name.

LOG

NO

DUAL

LOCAL

NOT LOCAL

AFTER JOURNAL

CHECKSUM = integrity_checking_level

FREESPACE = integer

PERCENT

DATABLOCKSIZE = data_block_size

BYTE

KBYTE

KILOBYTE

MINIMUM

MAXIMUM

REPLICATION GROUP

replication_group_name

B

Create Table Options
Chapter 3: SQL Data Definition Language

CREATE TABLE

column_name

data type attributes

NOT NULL

COMPRESS

constant

NULL

255 constan

CONSTRAINT name

UNIQUE

PRIMARY KEY

CHECK ( boolean condition )

REFERENCES

WITH CHECK OPTION

NO table_name
column_name

CONSTRAINT FOREIGN KEY

CONSTRAINT name

CHECK ( boolean_condition )

column_name

REFERENCES

WITH CHECK OPTION

table_name
column_name

GENERATED ALWAYS AS IDENTITY

START WITH

INCREMENT BY

MINVALUE

NO

MAXVALUE

NO

CYCLE

NO

Unique Definition

CONSTRAINT name

UNIQUE ( column_name )

Primary Key Definition

CONSTRAINT name

PRIMARY KEY

Foreign Key Definition

CONSTRAINT name

FOREIGN KEY ( column_name ) REFERENCES

REFERENCES

WITH CHECK OPTION

table_name
column_name

Check Definition

CONSTRAINT name

CHECK ( boolean_condition )
Chapter 3: SQL Data Definition Language

CREATE TABLE

Index Definition

INDEX

UNIQUE

INDEX

PRIMARY INDEX

UNIQUE

PARTITION BY

ON COMMIT

DELETE ROWS

PRESERVE

ORDER BY

VALUES

HASH

NO PRIMARY INDEX

INDEX

ORDER BY

VALUES

HASH

PARTITION BY

(i expression)

(i expression)

PRIMARY INDEX

UNIQUE

INDEX

UNIQUE

ON COMMIT DELETE ROWS

PRESERVE

;
Data Type

- INTEGER
- SMALLINT
- BIGINT
- BYTEINT
- DATE
- TIME
- TIMESTAMP

- INTERVAL YEAR
- INTERVAL MONTH
- INTERVAL DAY

- INTERVAL HOUR
- INTERVAL MINUTE
- INTERVAL SECOND

- PERIOD(DATE)
- PERIOD(TIME)
- PERIOD(TIMESTAMP)

- REAL
- DOUBLE PRECISION
- FLOAT
- DECIMAL
- NUMERIC
Chapter 3: SQL Data Definition Language

CREATE TABLE

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td></td>
</tr>
<tr>
<td>BYTE</td>
<td></td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td></td>
</tr>
<tr>
<td>CHAR VARYING</td>
<td></td>
</tr>
<tr>
<td>VARBYTE</td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
</tr>
<tr>
<td>LONG VARCHAR</td>
<td></td>
</tr>
<tr>
<td>LONG VARGRAPHIC</td>
<td></td>
</tr>
<tr>
<td>BINARY LARGE OBJECT</td>
<td></td>
</tr>
<tr>
<td>BLOB</td>
<td></td>
</tr>
<tr>
<td>CHARACTER LARGE OBJECT</td>
<td></td>
</tr>
<tr>
<td>CLOB</td>
<td></td>
</tr>
<tr>
<td>UDT_name</td>
<td></td>
</tr>
<tr>
<td>SYSUDTLIB</td>
<td></td>
</tr>
<tr>
<td>ST_Geometry</td>
<td></td>
</tr>
<tr>
<td>MBR</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>CHARACTER SET</td>
<td>server_character_set</td>
</tr>
<tr>
<td>UC</td>
<td></td>
</tr>
<tr>
<td>CASESPECIFIC</td>
<td></td>
</tr>
<tr>
<td>NOT</td>
<td></td>
</tr>
<tr>
<td>FORMAT</td>
<td>quotestring</td>
</tr>
<tr>
<td>TITLE</td>
<td>quotestring</td>
</tr>
<tr>
<td>NAMED</td>
<td>name</td>
</tr>
<tr>
<td>DEFAULT</td>
<td>number</td>
</tr>
<tr>
<td>USER</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
</tr>
<tr>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>WITH DEFAULT</td>
<td></td>
</tr>
</tbody>
</table>

1101A536

1101G205
### Copy Table Syntax

```sql
CREATE TABLE table_name
    SET GLOBAL TEMPORARY VOLATILE
    VOLATILE
    FALLBACK
    NO PROTECTION
    MULTISET
    NOT LOCAL
    LOCAL
    NOT LOCAL
    CHECKSUM = integrity_checking_level
    FREESPACE = integer
    PERCENT
    DATABLOCKSIZE = data_block_size
    BYTE
    KBYTE
    KILOBYTE
    MINIMUM
    MAXIMUM
```
CREATE TABLE

---

**Column Definition**
- `column_name` — `data type`
- `data type attributes`
- `COMRESS` { `constant`, NULL } [255]
- `NOT` { `constant`, NULL }
- `UNIQUE` [primary_index_column]
- `PRIMARY KEY`
- `CONSTRAINT` `name`

**Table Level Definition**
- `AS` { `database_name`, `user_name`, `query_expression` }
- `WITH` { `DATA` { `NO`, `AND` { `STATISTICS`, `NO` }, `STATS` } }

**Index Definition**
- `UNIQUE` [index_name]
- `INDEX` { index_column_name }
- `PARTITION BY` partitioning_expression
- `NO PRIMARY INDEX`

**Column Storage Attributes**
- `NULL`
- `NOT NULL`
- `COMPRESS`
- `255`

**Unique Constraint**
- ` constraint { `boolean condition` }
- `CHECK`

**Check Constraint**
- `constraint { `boolean condition` }
- `CHECK`

**Statistics**
- `NO`
- `STATS`

**Order by**
- `VALUES`
- `HASH`

**Preserve on Commit**
- `DELETE ROWS`
### Data Type

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Precision Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
</tr>
<tr>
<td>BYTEINT</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>(precision)</td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td>(precision) ,fractional_seconds_precision</td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
<td>(precision)</td>
</tr>
<tr>
<td>REAL</td>
<td>WITH TIMEZONE</td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>( integer )</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>( integer , integer )</td>
</tr>
<tr>
<td>NUMERIC</td>
<td></td>
</tr>
</tbody>
</table>
CREATE TABLE (Queue Table Form)

CREATE TABLE

- SET
- MULTISET

database_name.
user_name.

A

Queue

B

CREATE Table Options

- NO JOURNAL
- CHECKSUM = integrity_checking_level
- FREESPACEx = integer
- DATABLOCKSIZE = integer
  - BYTE
  - KBYTE
  - KILOBYTE

- MINIMUM
- MAXIMUM
- REPLICATION GROUP = replication_group_name

- NO
- PROTECTION
- FALLBACK

CREATE TABLE (Queue Table Form)
Chapter 3: SQL Data Definition Language

CREATE TABLE (Queue Table Form)

CREATE TABLE (Queue Table Form)

B) \( \text{QITS}_{\text{column_name}} \) TIMESTAMP (6) NOT NULL DEFAULT CURRENT_TIMESTAMP

C) data type

D) CHECK (boolean condition)

E) CONSTRAINT name

Column Definition

Column Storage Attributes

Column Constraint Attributes

Unique Definition

Check Definition

Table Level Definition

CONSTRAINT name

UNIQUE

CONSTRAINT name

PRIMARY KEY

CHECK (boolean condition)

CONSTRAINT name

GENERATED ALWAYS AS IDENTITY

START WITH

INCREMENT BY

MINVALUE

MAXVALUE

NO CYCLE

NOT NULL DEFAULT CURRENT_TIMESTAMP (6)

column_name data type

NULL

COMPRESS constant

255

CONSTRAINT name

UNIQUE

CONSTRAINT name

PRIMARY KEY

CHECK (boolean condition)

CONSTRAINT name

GENERATED ALWAYS AS IDENTITY

START WITH

INCREMENT BY

MINVALUE

MAXVALUE

NO CYCLE

NOT NULL DEFAULT CURRENT_TIMESTAMP (6)
CREATE TABLE (Queue Table Form)

INDEX
  (index_column_name)

UNIQUE
  index_name

PRIMARY INDEX
  (primary_index_column)

INDEX
  (index_column_name)

ORDER BY
  (order_column_name)

VALUES

HASH
### Data Type

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>Integer</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td>Small Integer</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td>Big Integer</td>
<td></td>
</tr>
<tr>
<td>BYTEINT</td>
<td>Byte Integer</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>Date</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>Time</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>Timestamp</td>
<td>WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>Interval Year</td>
<td>TO MONTH</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>Interval Month</td>
<td>TO MONTH</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>Interval Day</td>
<td>TO HOUR, MINUTE, SECOND</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>Interval Hour</td>
<td>TO MINUTE, SECOND</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>Interval Minute</td>
<td>TO SECOND</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>Interval Second</td>
<td></td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td>Period(Date)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td>Period(Time)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
<td>Period(TIMESTAMP)</td>
<td>WITH TIMEZONE</td>
</tr>
<tr>
<td>REAL</td>
<td>Real</td>
<td></td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td>Double Precision</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>Float</td>
<td>( fractional_seconds_precision )</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>Decimal</td>
<td>( integer, integer )</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>Numeric</td>
<td>( integer, integer )</td>
</tr>
</tbody>
</table>
Chapter 3: SQL Data Definition Language

CREATE TABLE (Queue Table Form)

---

SQL Quick Reference 155

---

BINARY LARGE OBJECT (integer)

CHAR

BYTE

GRAPHIC

VARCHAR (integer)

CHAR VARYING

VARBYTE

VARGRAPHIC

LONG VARCHAR

LONG VARGRAPHIC

BINARY LARGE OBJECT (integer)

CHARACTER LARGE OBJECT

CLOB

UDT_name

SYSUDTLIB:

ST_Geometry

MBR
CREATE TRANSFORM/ REPLACE TRANSFORM

CREATE TRANSFORM FOR SYSUDTLIB. UDT_name transform_group_name

REPLACE TRANSFORM

TO SQL WITH SPECIFIC FUNCTION specific_function_name

SPECIFIC METHOD method_name (method_name data_type)

SPECIFIC FUNCTION FUNCTION function_name (function_name data_type)

FROM SQL WITH SPECIFIC FUNCTION specific_function_name

SPECIFIC METHOD method_name (method_name data_type)

SPECIFIC FUNCTION FUNCTION function_name (function_name data_type)
CREATE TRIGGER/ REPLACE TRIGGER

CREATE TRIGGER/ REPLACE TRIGGER

A. INSERT
   DELETE
   UPDATE
       OF
           column_name
           (column_name)

B. ON
       database_name.
       table_name
       ORDER - integer

C. REFERRING
       OLD
           old_transition_variable_name
           ROW AS
           NEW
           new_transition_variable_name
           ROW AS
           OLD_TABLE
           old_transition_table_name
           AS
           NEW_TABLE
           new_transition_table_name
           AS
           OLD_NEW_TABLE
           old_new_table_name (old_value, new_value)

D. FOR EACH ROW
   WHEN (search_condition)

   BEGIN ATOMIC
   SQL_procedure_statement ;
   ( SQL_procedure_statement ; )
   END

   SQL_procedure_statement ;
   ( SQL_procedure_statement ; )
CREATE TYPE (Distinct Form)

CREATE TYPE
  UDT_name AS predefined_data_type
  CHARACTER SET server_character_set

FINAL

METHOD
  instance
  method_name
  parameter_name
  predefined_data_type
  CHARACTER SET server_character_set
  AS LOCATOR

RETURNS
  predefined_data_type
  CHARACTER SET server_character_set
  AS LOCATOR
  CAST FROM predefined_data_type
  AS LOCATOR

SPECIFIC
  specific_method_name
  SELF AS RESULT
  language_clause
  SQL_data_access
  language_clause

SPECIFIC
  specific_method_name
  PARAMETER STYLE SQL
  TD_GENERAL
  NOT
  DETERMINISTIC
  CALLED ON NULL INPUT
  RETURNS NULL ON NULL INPUT
Data Type Declaration

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
</tr>
<tr>
<td>BYTEINT</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>(fractional_seconds_precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>(fractional_seconds_precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>(precision) TO MONTH</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>(precision) TO HOUR</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>(precision) TO MINUTE</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>(precision) TO SECOND</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>(precision)</td>
</tr>
<tr>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>(integer)</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>(integer, integer)</td>
</tr>
<tr>
<td>NUMERIC</td>
<td></td>
</tr>
<tr>
<td>CHAR</td>
<td></td>
</tr>
<tr>
<td>BYTE</td>
<td>(integer)</td>
</tr>
<tr>
<td>GRAPHIC</td>
<td></td>
</tr>
<tr>
<td>VARCHAR</td>
<td>(integer)</td>
</tr>
<tr>
<td>CHAR VARYING</td>
<td></td>
</tr>
<tr>
<td>VARBYTE</td>
<td></td>
</tr>
<tr>
<td>VARGRAPHIC</td>
<td></td>
</tr>
<tr>
<td>LONG VARCHAR</td>
<td></td>
</tr>
<tr>
<td>LONG VARGRAPHIC</td>
<td></td>
</tr>
<tr>
<td>BINARY LARGE OBJECT</td>
<td>(-integer                  )</td>
</tr>
<tr>
<td>BLOB</td>
<td>G, K, M</td>
</tr>
<tr>
<td>CHARACTER LARGE OBJECT</td>
<td>(-integer                  )</td>
</tr>
<tr>
<td>CLOB</td>
<td>G, K, M</td>
</tr>
</tbody>
</table>
CREATE TYPE (Structured Form)

CREATE TYPE `UDT_name` AS SYSUDTLIB.

(`attribute_name` `predefined_data_type`
  CHARACTER SET `server_character_set`)

`UDT_name`
Chapter 3: SQL Data Definition Language
CREATE TYPE (Structured Form)
CREATE USER — user_name

FROM database_name

AS

PERMANENT = n

PERM = n

BYTES

PASSWORD = password

PERMANENT

PASSWORD = password

BEFORE

NO

DUAL

LOCAL

NOT LOCAL

TEMPORARY = n

bytes

SPOOL = n

BYTES

DEFAULT DATABASE = database_name

COLLATION = collation_sequence

ACCOUNT = 'account_ID'

( 'account_ID'

FALLBACK

NO

PROTECTION

JOURNAL

AFTER JOURNAL

NO

DUAL

LOCAL

NOT LOCAL

DEFAULT JOURNAL TABLE =

table_name

database_name.

TIME ZONE = LOCAL

quotestring

sign

NULL

DATEFORM = INTEGERDATE

ANSIDATE

NULL

DEFAULT CHARACTER SET — server_character_set —

DEFAULT ROLE = role_name

NONE

NULL

ALL

PROFILE = profile_name

NULL
CREATE VIEW/
REPLACE VIEW

CREATE VIEW
CV
REPLACE VIEW

LOCKING
LOCK
DATABASE
ACCESS
FOR
MODE
NOWAIT

TABLE
ACCESS
IN
EXCL
SHARE
READ
WRITE

VIEW
ROW
ACCESS
IN
EXCL
SHARE
READ
WRITE

SELECT
DISTINCT
ALL
TOP
n
m
PERCENT
WITH TIES

expression
expression_alias
AS
table_name.

FROM
TABLE
VIEW
LOCK
ROW

WHERE
GROUP BY
ORDINARY GROUPING SET
EMPTY GROUPING SET
ROLLUP LIST
CUBE LIST
GROUPING SETS SPECIFICATION

HAVING
QUALIFY
WITH CHECK OPTION

ORDER BY
expression
column_name
ASC
DESC

1101A621
Chapter 3: SQL Data Definition Language

DATABASE

DATABASE database_name

DELETE DATABASE

DELETE DATABASE name

DELETE USER

DELETE USER ALL

DROP AUTHORIZATION

DROP AUTHORIZATION authorization_name database_name

DROP CAST

DROP CAST (source_data_type AS target_data_type) database_name
<table>
<thead>
<tr>
<th>Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
</tr>
<tr>
<td>SMALLINT</td>
</tr>
<tr>
<td>BIGINT</td>
</tr>
<tr>
<td>BYTEINT</td>
</tr>
<tr>
<td>DATE</td>
</tr>
<tr>
<td>TIME</td>
</tr>
<tr>
<td>TIMESTAMP</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
</tr>
<tr>
<td>REAL</td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
</tr>
<tr>
<td>FLOAT</td>
</tr>
<tr>
<td>DECIMAL</td>
</tr>
<tr>
<td>NUMERIC</td>
</tr>
</tbody>
</table>
Chapter 3: SQL Data Definition Language

DROP DATABASE

DROP DATABASE database_name
;

DROP ERROR TABLE

DROP ERROR TABLE FOR data_table_name
;

1101A438
 DROP FUNCTION

```
DROP FUNCTION database_name.function_name
  specific_function_name

(data_type)

;  
```

<table>
<thead>
<tr>
<th>Data Type</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
</tr>
<tr>
<td>BYTEINT</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td></td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>(fractional_seconds_precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>(precision) TO MONTH</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>(precision) TO HOUR</td>
</tr>
<tr>
<td></td>
<td>MINUTE</td>
</tr>
<tr>
<td></td>
<td>SECOND</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>(precision) TO MINUTE</td>
</tr>
<tr>
<td></td>
<td>SECOND</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>(precision) TO SECOND (fractional_seconds_precision)</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>(precision)</td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td>(precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
<td>(precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>(integer)</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>(integer, integer)</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>(integer, integer)</td>
</tr>
</tbody>
</table>
DROP GLOP SET

DROP GLOP SET
database_name.
user_name.
GLOP_set_name
;

DROP HASH INDEX

DROP HASH INDEX
database_name.
hash_index_name
;
Chapter 3: SQL Data Definition Language

**DROP INDEX**

**DROP index_name Syntax**

\[
\text{DROP INDEX} \quad \text{database_name.} \quad \text{index_name} \quad \text{ON} \quad \text{table_name}
\]

**DROP index_definition Syntax**

\[
\text{DROP INDEX} \quad \text{INDEX} \quad \text{(column_name)} \quad \text{index_name} \quad \text{ORDER BY} \quad \text{VALUES} \quad \text{HASH} \quad \text{ON} \quad \text{table_name} \quad \text{TEMPORARY} \quad \text{database_name.} \quad \text{join_index_name} \quad ;
\]

**DROP JOIN INDEX**

\[
\text{DROP JOIN INDEX} \quad \text{database_name.} \quad \text{join_index_name} \quad ;
\]

**DROP MACRO/ DROP PROCEDURE/ DROP TABLE/
### DROP TRIGGER/
### DROP VIEW

- **DROP**
- **MACRO** `database_name. macro_name`
- **TABLE** `database_name. table_name`
- **TEMPORARY**
- **TRIGGER** `database_name. trigger_name`
- **VIEW** `database_name. view_name`
- **PROCEDURE** `database_name. procedure_name`

### DROP ORDERING

- **DROP ORDERING**
- **FOR** `database_name. user_defined_type_name`

### DROP PROFILE

- **DROP PROFILE**
- `database_name. profile_name`
**DROP REPLICATION GROUP**

```
DROP REPLICATION GROUP replication_group_name;
```

**DROP REPLICATION RULESET**

```
DROP REPLICATION RULESET rule_set_name FOR replication_group_name;
```

**DROP ROLE**

```
DROP ROLE role_name;
```

```
DROP ROLE database_name.role_name;
```
DROP STATISTICS (Optimizer Form)

Syntax (Alternate)
Chapter 3: SQL Data Definition Language

DROP TRANSFORM

DROP TRANSFORM
database_name.

user_defined_type_name

DROP TYPE

DROP TYPE

database_name.

user_defined_type_name

DROP USER

DROP USER

user_name

1101A327

1101A328

1101A071
GRANT operation ON database_name
BY user_name;

ON AUTHORIZATION authorization_name
DATA database_name
USER user_name
TABLE object_name
VIEW database_name.
USER user_name.
MODIFY DATABASE

MODIFY DATABASE - database_name - AS

- PERMANENT = number
- TEMPORARY = number
- SPOOL = number
- ACCOUNT = 'account_ID'
- FALLBACK = PROTECTION
- NO BEFORE JOURNAL
- NO AFTER JOURNAL
- LOCAL
- NOT LOCAL
- DEFAULT JOURNAL TABLE = journal_table_name
- DROP DEFAULT JOURNAL TABLE = journal_table_name

MODIFY DATABASE database_name;
MODIFY PROFILE

MODIFY PROFILE  profile_name AS database_name

ACCOUNT = 'account_id'

DEFAULT DATABASE = database_name

SPOOL = n

TEMPORARY = n

PASSWORD = EXPIRE = n

ATTRIBUTES = MINCHAR = n

MAXCHAR = n

DIGITS = c

SPECCHAR = c

MAXLOGONATTEMPTS = n

LOCKEDUSEREXPIRE = n

REUSE = n

RESTRICWORDS = c

COST PROFILE = cost_profile_name

1101B467
MODIFY USER

MODIFY USER-username-AS

PERMANENT = number

PERM = number

BYTES

STARTUP = NULL

quotestring

PASSWORD = password

FOR USER

RELEASE PASSWORD LOCK

TEMPORARY = number

BYTES

SPOOL = number

BYTES

ACCOUNT = 'account_ID'

( 'account_ID', 'account_ID' )

DEFAULT DATABASE = database_name

COLLATION = collation_sequence

FALLBACK

PROTECTION

NO

BEFORE

JOURNAL

NO

DUAL

LOCAL

NOT LOCAL

DEFAULT JOURNAL TABLE = table_name

DROP DEFAULT JOURNAL TABLE = database_name

drop_table_name

TIME ZONE = LOCAL

quotestring

sign

NULL

DATEFORM = INTEGERDATE

ANSIDATE

NULL

DEFAULT CHARACTER SET = server_character_set

DEFAULT ROLE = role_name

NONE

NULL

ALL

PROFILE = profile_name

NULL
RENAME FUNCTION

```
RENAME
  SPECIFIC FUNCTION
    database_name.
  function_name
    ( data_type )
  TO
  AS
  new_function_name
    new_specific_function_name;
```

Chapter 3: SQL Data Definition Language
RENAME FUNCTION
### Data Type

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
</tr>
<tr>
<td>BYTEINT</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>(integer)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>(precision)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>(precision)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td>(precision)</td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
<td>(precision)</td>
</tr>
<tr>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td>(precision)</td>
</tr>
<tr>
<td>FLOAT</td>
<td>(integer)</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>(integer)</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>(integer)</td>
</tr>
</tbody>
</table>
RENAME MACRO/
RENAME PROCEDURE/
RENAME TABLE/
RENAME TRIGGER/
RENAME VIEW

RENAME MACRO/
RENAME PROCEDURE/
RENAME TABLE/
RENAME TRIGGER/
RENAME VIEW
REPLACE METHOD

REPLACE SPECIFIC METHOD SYSUDTLIB METHOD SYSUDTLIB.

INSTANCE CONSTRUCTOR

A specific_method_name

B method_name parameter_name data_type UDT_name AS LOCATOR SYSUDTLIB.

EXTERNAL NAME external_method_name delimiter name_on_server delimiter include_name delimiter library_name delimiter object_name delimiter object_name delimiter source_name ;
### Data Type

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td></td>
</tr>
<tr>
<td>SMALLINT</td>
<td></td>
</tr>
<tr>
<td>BIGINT</td>
<td></td>
</tr>
<tr>
<td>BYTEINT</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td></td>
</tr>
<tr>
<td>TIME</td>
<td>(integer)</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>(precision)</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>(precision)</td>
</tr>
<tr>
<td>PERIOD(DATE)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td></td>
</tr>
<tr>
<td>PERIOD(TIMESTAMP)</td>
<td>(fractional_seconds_precision)</td>
</tr>
<tr>
<td>REAL</td>
<td></td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td></td>
</tr>
<tr>
<td>FLOAT</td>
<td>(integer)</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>(integer, integer)</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>(integer, integer)</td>
</tr>
</tbody>
</table>

A

B

1101A535

Chapter 3: SQL Data Definition Language

REPLACE METHOD
SET QUERY_BAND

SET QUERY_BAND = 'pair_name = pair_value;' FOR SESSION TRANSACTION UPDATE NONE;
Chapter 3: SQL Data Definition Language

SET ROLE

SET ROLE

SET ROLE

SET SESSION

SET SESSION

SET SESSION

SET SESSION ACCOUNT

SET SESSION ACCOUNT

SET SESSION ACCOUNT
Chapter 3: SQL Data Definition Language

SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL

SET SESSION CHARACTERISTICS AS TRANSACTION ISOLATION LEVEL – isolation_level

SET SESSION COLLATION

SET SESSION COLLATION — collation_sequence

SET SESSION DATABASE

SET SESSION DATABASE — database_name

SET SESSION DATEFORM

SET SESSION DATEFORM= ANSIDATE INTEGRERDATE

SQL Quick Reference 189
Chapter 3: SQL Data Definition Language

**SET SESSION FUNCTION TRACE**

```sql
SET SESSION FUNCTION TRACE US[

ging ] table_name

FOR OFF

TABLE database_name.

USING mask_string

; ;
```

**SET SESSION OVERRIDE REPLICATION**

```sql
SET SESSION OVERRIDE REPLICATION ON

OFF

; ;
```

**SET SESSION SUBSCRIBER**

```sql
SET SESSION SUBSCRIBER ON

OFF

; ;
```

**SET TIME ZONE**

```sql
SET TIME ZONE LOCAL

INTERVAL 'time_zone_displacement' HOUR TO MINUTE

USER sign

; ;
```
HELP CAST

HELP CAST  
SYSUDTLIB.  
UDT_name  
SOURCE  
TARGET  
;  
1101B331

HELP COLUMN

Syntax 1

HELP COLUMN  
column_name  
FROM  
database_name.  
user_name.  
join_index_name  
hash_index_name  
;  
1101B473

Syntax 2

HELP COLUMN  
*  
FROM  
database_name.  
user_name.  
join_index_name  
hash_index_name  
;  
1101G245

Syntax 3

HELP COLUMN  
*  
table_name  
*  
column_name  
;  
1101G246

Syntax 4

HELP COLUMN  
database_name.  
user_name.  
join_index_name  
hash_index_name  
;  
1101H247
Chapter 3: SQL Data Definition Language
HELP CONSTRAINT

Syntax 5

```
HELP COLUMN expression ;
```

Syntax 6

```
HELP COLUMN expression ,
  database_name.
  user_name.
  join_index_name.
  hash_index_name.
  table_name.
  expression ;
```

Syntax 7

```
HELP COLUMN column_name FROM ERROR TABLE FOR
data_table_name ;
```

Syntax 8

```
HELP COLUMN column_name FROM error_table_name ;
```

**HELP CONSTRAINT**

```
HELP CONSTRAINT database_name.
  user_name.
  table_name.
  constraint_name ;
```
**HELP DATABASE/HELP USER**

HELP DATABASE database_name user_name

HELP USER user_name

**HELP ERROR TABLE**

Syntax 1

HELP ERROR TABLE FOR database_name data_table_name user_name

Syntax 2

HELP TABLE database_name error_table_name user_name

**HELP FUNCTION**

HELP SPECIFIC FUNCTION database_name specific_function_name user_name

FUNCTION database_name function_name (data_type)
**Data Type**

- INTEGER
- SMALLINT
- BIGINT
- BYTEINT
- DATE
- TIME
- TIMESTAMP\[(`fractional_seconds_precision`)]\[ WITH TIMEZONE\]
- INTERVAL YEAR\[(`precision`)\] \[ TO MONTH\]
- INTERVAL MONTH\[(`precision`)\]
- INTERVAL DAY\[(`precision`)\] \[ TO HOUR\]
- \[ MINUTE\]
- \[ SECOND\]
- (\[ `fractional_seconds_precision`\]
- INTERVAL HOUR\[(`precision`)\] \[ TO MINUTE\]
- \[ SECOND\]
- (\[ `fractional_seconds_precision`\]
- INTERVAL MINUTE\[(`precision`)\] \[ TO SECOND\]
- (\[ `fractional_seconds_precision`\]
- INTERVAL SECOND\[(`precision`)\]
- PERIOD(DATE)
- PERIOD(TIME)
- PERIOD(TIMESTAMP\[(`precision`)\] \[ WITH TIMEZONE\]
- REAL
- DOUBLE PRECISION
- FLOAT\[(`integer`)\]
- DECIMAL\[(`integer`, `integer`)\]
- NUMERIC\[(`integer`, `integer`)\]
Chapter 3: SQL Data Definition Language
HELP JOIN INDEX

HELP JOIN INDEX

1101A559

HELP MACRO/
HELP TABLE/
HELP VIEW

HELP MACRO

1101G241

HELP METHOD

1101A330
HELP PROCEDURE

```
HELP PROCEDURE  procedure_name
                 database_name;
```

HELP REPLICATION GROUP

```
HELP REPLICATION GROUP  replication_group_name
```

HELP SESSION

```
HELP SESSION
```

HELP STATISTICS (Optimizer Form)

```
HELP
  STATISTICS
    TEMPORARY
    STATS
    STAT
     database_name.
user_name.
     table_name
     database_name.
user_name.
     join_index_name
   database_name.
user_name.
   hash_index_name
   view_name
```
HELP STATISTICS (QCD Form)

HELP STATISTICS
  STATS
    STAT
      database_name.
        user_name.
    table_name
      view_name
  FROM QCD_name

HELP TRANSFORM

HELP TRANSFORM
  database_name.
    user_name.
  UDT_name

HELP TRIGGER

HELP TRIGGER
  database_name.
    user_name.
  trigger_name
    table_name
**HELP TYPE**

```
HELP TYPE
  UDT_name
    SYSDTTLIB.
    ATTRIBUTE
    METHOD
```

**HELP VOLATILE TABLE**

```
HELP VOLATILE TABLE
  volatile_table_name
```

**HELP (Online Form)**

```
HELP
  SQL
    commandname
  ARCHIVE
    commandname
  DUMP
    commandname
  FASTEXPORT
    commandname
  FASTLOAD
    commandname
  MULTILOAD
    commandname
  PMPC
    commandname
  TPCCONS
    commandname
  SPL
    commandname
```
SHOW

SHOW CAST/
SHOW ERROR TABLE/
SHOW FUNCTION/
SHOW HASH INDEX/
SHOW JOIN INDEX/
SHOW MACRO/
SHOW METHOD/
SHOW PROCEDURE/
SHOW REPLICATION GROUP/
SHOW TABLE/
SHOW TRIGGER/
SHOW TYPE/
SHOW VIEW
General Syntax

SHOW HASH INDEX hash_index_name
SHOW JOIN INDEX join_index_name
SHOW MACRO macro_name
SHOW TABLE table_name
SHOW TRIGGER trigger_name
SHOW TEMPORARY VIEW view_name
SHOW ERROR TABLE FOR data_table_name
SHOW TABLE error_table_name
SHOW TRIGGER trigger_name
SHOW VIEW view_name
SHOW PROCEDURE procedure_name
SHOW REPLICATION GROUP replication_group_name
SHOW SPECIFIC FUNCTION specific_function_name
SHOW FUNCTION function_name (data_type UDT_name)
SHOW SPECIFIC METHOD specific_method_name
SHOW SYSTEM UDTLIB.
SHOW METHOD method_name (data_type UDT_name)
SHOW CONSTRUCTOR FOR UDT_name
SHOW TYPE UDT_name
BINARY LARGE OBJECT (integer)

CHAR
  BYTE
  GRAPHIC

VARCHAR (integer)
  CHAR VARYING
  VARCHAR
  VARGRAPHIC
  LONG VARCHAR
  LONG VARGRAPHIC

BINARY LARGE OBJECT (integer)
  BLOB
  CHARACTER LARGE OBJECT
    CLOB

SYSUDTLIB.
  UDT_name
    ST_Geometry
    MBR
Chapter 3: SQL Data Definition Language
SHOW CAST/ SHOW ERROR TABLE/ SHOW FUNCTION/ SHOW HASH INDEX/ SHOW JOIN INDEX/ SHOW MACRO/ SHOW METHOD/

Embedded SQL Syntax
Data Type

- INTEGER
- SMALLINT
- BIGINT
- BYTEINT
- DATE

- TIME
- TIMESTAMP [(fractional_seconds_precision)] WITH TIMEZONE

- INTERVAL YEAR [(precision)] TO MONTH
- INTERVAL MONTH [(precision)]
- INTERVAL DAY [(precision)] TO HOUR MINUTE SECOND [(fractional_seconds_precision)]
- INTERVAL HOUR [(precision)] TO MINUTE SECOND [(fractional_seconds_precision)]
- INTERVAL MINUTE [(precision)] TO SECOND [(fractional_seconds_precision)]
- INTERVAL SECOND [(fractional_seconds_precision)]

- PERIOD(DATE) [(precision), (fractional_seconds_precision)]
- PERIOD(TIME) [(precision), (fractional_seconds_precision)]
- PERIOD(TIMESTAMP [(precision)] WITH TIMEZONE)

- REAL
- DOUBLE PRECISION
- FLOAT [(integer)]
- DECIMAL [(integer), (integer)]
- NUMERIC [(integer), (integer)]
SHOW QUERY LOGGING

SHOW QUERY LOGGING ON

ALL

ACCOUNT = 'account_name'

APPLNAME = 'application_name'

1101A536

CHAR

BYTE

GRAPHIC

VARCHAR

CHAR VARYING

VARBYTE

VARGRAPHIC

LONG VARCHAR

LONG VARGRAPHIC

BINARY LARGE OBJECT

BLOB

CHARACTER LARGE OBJECT

CLOB

SYSDTLIB.

UDT_name

ST_Geometry

MBR

1101A547
CHAPTER 4 SQL Data Control Language

GIVE

\[
\text{GIVE} \quad \text{database}\_\text{name} \quad \text{TO} \quad \text{recipient}\_\text{name} \quad \text{user}\_\text{name} \quad ;
\]

GRANT

Monitor Form

\[
\text{GRANT} \quad \text{MONITOR} \quad \text{PRIVILEGES} \quad \text{BUT NOT} \quad \text{monitor}\_\text{privilege} \quad \text{monitor}\_\text{privilege} \quad \text{TO} \quad \text{A} \quad \text{user}\_\text{name} \quad \text{PUBLIC} \quad \text{WITH GRANT OPTION} \quad ;
\]

Role Form

\[
\text{GRANT} \quad \text{role}\_\text{name} \quad \text{TO} \quad \text{user}\_\text{name} \quad \text{role}\_\text{name} \quad \text{WITH ADMIN OPTION} \quad ;
\]
Chapter 4: SQL Data Control Language

**GRANT**

**SQL Form**

```
GRANT
  PRIVILEGES
    ALL,
    privilege,
    ALL BUT privilege
  OR
  ALL
    ON
      database_name
      user_name
      role_name
    PUBLIC

  database_name
    object_name
    user_name.
    procedure_name

  database_name.
    procedure_name

  database_name.
    specific_function_name

  database_name.
    function_name
    ( )
    data_type
    parameter_name

  database_name.
    function_name
    ( )

  database_name.
    function_name
    ( )
    data_type
    parameter_name

  database_name.
    UDT_name

  role_privilege
  profile_privilege

TO
  user_name
    ALL
    WITH GRANT OPTION
  ALL
  PUBLIC
  role_name

;```

1101W055
### Data Type

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>SMALLINT</td>
</tr>
<tr>
<td>BIGINT</td>
<td>BIGINT</td>
</tr>
<tr>
<td>BYTEINT</td>
<td>BYTEINT</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>TIME</td>
<td>TIME</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>TIMESTAMP WITH TIMEZONE (fractional_seconds_precision)</td>
</tr>
<tr>
<td>INTERVAL YEAR</td>
<td>INTERVAL YEAR WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL MONTH</td>
<td>INTERVAL MONTH WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL DAY</td>
<td>INTERVAL DAY (precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL HOUR</td>
<td>INTERVAL HOUR (precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL MINUTE</td>
<td>INTERVAL MINUTE (precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>INTERVAL SECOND</td>
<td>INTERVAL SECOND (precision) WITH TIMEZONE</td>
</tr>
<tr>
<td>PERIOD.DATE</td>
<td>PERIOD.DATE (precision)</td>
</tr>
<tr>
<td>PERIOD.TIME</td>
<td>PERIOD.TIME (precision)</td>
</tr>
<tr>
<td>PERIOD(TIME)</td>
<td>PERIOD(TIME (precision)) WITH TIMEZONE</td>
</tr>
<tr>
<td>REAL</td>
<td>REAL</td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td>DOUBLE PRECISION</td>
</tr>
<tr>
<td>FLOAT</td>
<td>FLOAT (integer)</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>DECIMAL (integer, integer)</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>NUMERIC (integer, integer)</td>
</tr>
</tbody>
</table>
Chapter 4: SQL Data Control Language

GRANT CONNECT THROUGH

GRANT CONNECT THROUGH — trusted_user_name — TO

A

application_user_name WITH ROLE

PERMANENT

permanent_user_name WITH ROLE

WITHOUT ROLE

A

role_name

role_name

WITH ROLE

1101A536

WITH ROLE

1101A541
GRANT LOGON

```
GRANT LOGON ON host_id, ALL AS DEFAULT TO user_name
FROM user_name
WITH NULL PASSWORD;
```

REVOKE

Monitor Form

```
REVOKE GRANT OPTION FOR MONITOR PRIVILEGES
BUT NOT monitor_privilege,
monitor_privilege,

TO user_name
FROM ALL PUBLIC
```

Role Form

```
REVOKE ADMIN OPTION FOR role_name
TO user_name
FROM role_name
```

SQL Quick Reference 213
Chapter 4: SQL Data Control Language

REVOKE

SQL Form

REVOKE [GRANT OPTION FOR [ALL PRIVILEGES | ALL BUT privilege [, role_privilege [, profile_privilege]]]]

ON [database_name.] [user_name. | role_name | PUBLIC]

PROCEDURE procedure_name

SPECIFIC FUNCTION specific_function_name

FUNCTION function_name

TYPE UDT_name

SYSUDTLIB.

TO [ALL | PUBLIC | role_name [, role_privilege [, profile_privilege]]]

FROM [ALL | PUBLIC | role_name [, role_privilege [, profile_privilege]]]
### Data Type

- **INTEGER**
- **SMALLINT**
- **BIGINT**
- **BYTEINT**
- **DATE**
- **TIME**
- **TIMESTAMP**\( (\text{fractional\_seconds\_precision}) \text{ WITH TIMEZONE} \)
- **INTERVAL YEAR**\( (\text{precision}) \text{ TO MONTH} \)
- **INTERVAL MONTH**\( (\text{precision}) \)
- **INTERVAL DAY**\( (\text{precision}) \text{ TO HOUR MINUTE SECOND} \)\( (\text{fractional\_seconds\_precision}) \)
- **INTERVAL HOUR**\( (\text{precision}) \text{ TO MINUTE SECOND} \)\( (\text{fractional\_seconds\_precision}) \)
- **INTERVAL MINUTE**\( (\text{precision}) \text{ TO SECOND} \)\( (\text{fractional\_seconds\_precision}) \)
- **INTERVAL SECOND**\( (\text{precision} .\text{fractional\_seconds\_precision}) \)
- **PERIOD\( (\text{DATE}) \)**
- **PERIOD\( (\text{TIME}) \)**
- **PERIOD\( (\text{TIMESTAMP}) \)\( (\text{precision}) \text{ WITH TIMEZONE} \)**
- **REAL**
- **DOUBLE PRECISION**
- **FLOAT**\( (\text{integer}) \)
- **DECIMAL**\( (\text{integer} , \text{integer}) \)
- **NUMERIC**\( (\text{integer} , \text{integer}) \)
REVOKE CONNECT THROUGH

REVOKE CONNECT THROUGH — trusted_user_name TO application_user_name
WITH ROLE role_name
FROM permanent_user_name
PERMANENT

CHAR
BYTE
GRAPHIC

VARCHAR
CHAR VARYING
VARBYTE
VARGRAPHIC

LONG VARCHAR
LONG VARGRAPHIC

BINARY LARGE OBJECT
BLOB
CHARACTER LARGE OBJECT
CLOB

UDT_name
ST_Geometry
MBR
REVOKE LOGON

REVOKE LOGON ON host_id, ALL AS DEFAULT, TO user_name FROM

1101B036
Chapter 4: SQL Data Control Language
REVOKE LOGON
SELECT

Seed Statement

A - SELECT
- DISTINCT
- ALL
- expression
- expression_alias_name
- table_name.*

A - FROM
- table_name
- correlation_name
- AS
- join_table_name
- INNER
- LEFT
- RIGHT
- OUTER
- FULL
- CROSS JOIN
- (subquery)
- derived_table_name
- AS
- column_name

A - WHERE
- search_condition

A - GROUP BY
- ordinary_group_set
- empty_grouping_set
- rollup_list
- cube_list
- grouping_sets_specification

A - HAVING
- conditional_expression

A - QUALIFY
- search_condition

A - ORDER BY
- expression
- column_name
- column_name_alias
- column_position
- ASC
- DESC
SELECT AND CONSUME

SELECT AND CONSUME TOP 1 — select_list — FROM — queue_table_name

;
WITH [RECURSIVE] Request Modifier

WITH query_name AS ( select_expression )

recursive

Non-recursive

WITH RECURSIVE query_name AS ( select_expression )

Recursive

AS ( select_expression )

A

SEED STATEMENT

UNION ALL

Recursive Statement

A

AS ( Seed Statement )

1101A291

224 SQL Quick Reference
Seed Statement

- **SELECT**
  - **DISTINCT**
  - **ALL**
  - **expression**
  - **expression_alias_name**
  - **table_name.***

- **FROM**
  - **table_name**
  - **correlation_name**
  - **join_table_name**
  - **ON**
  - **search_condition**
  - **JOIN**
  - **derived_table_name**
  - **AS**
  - **(subquery)**

- **WHERE**
  - **search_condition**
  - **GROUP BY**
  - **empty_grouping_set**
  - **ordinary_group_set**
  - **rollup_list**
  - **cube_list**
  - **grouping_sets_specification**

- **HAVING**
  - **conditional_expression**

- **ORDER BY**
  - **expression**
  - **column_name**
  - **column_name_alias**
  - **column_position**
  - **ASC**
  - **DESC**
DISTINCT, ALL, and .ALL Options

Recursive Statement

```
SELECT expression AS expression_alias_name 
FROM query_name
ON joined_table
WHERE search_condition
```

Implicit Join

Explicit Join

DISTINCT, ALL, and .ALL Options

```
DISTINCT
ALL
.ALL
```

Table and column options

```
table_name.
column_name
```

SQL Quick Reference
TOP n Operator

FROM Clause
Chapter 5: SQL Data Manipulation Language

HASH BY Clause

```sql
HASH BY column_name
```

LOCAL ORDER BY Clause

```sql
LOCAL ORDER BY column_name
```

WHERE Clause

```sql
WHERE search_condition
```

Subqueries in Search Conditions

Syntax 1

```sql
expression comparison_operator (query)
```

Syntax 2: Logical Expressions

```sql
EXISTS (query)
```
GROUP BY Clause

CUBE Option

GROUPING SETS Option

ROLLUP Option

HAVING Clause
QUALIFY Clause

--- QUALIFY --- search_condition ---

SAMPLE Clause

--- SAMPLE --- WITH REPLACEMENT — RANDOMIZED ALLOCATION ---

--- fraction_description --- count_description ---

--- WHEN condition THEN --- fraction_description --- count_description ---

--- ELSE --- fraction_description --- count_description ---

--- END ---

SAMPLEID Expression

--- SAMPLEID ---

--- FF07D087 ---

--- FF07D180 ---
ORDER BY Clause

WITH Clause

Outer Join

Null
**ABORT**

```
ABORT
  abort_message [ FROM option ] [ WHERE abort_condition ]
```

**BEGIN TRANSACTION**

```
BEGIN TRANSACTION
  BT
```

**CALL**

```
CALL
  database_name .
  user_name .
  procedure_name
    ( [ IN argument ] [ INOUT argument ] [ OUT argument ] )

IN argument
  value_expression
    [ ? ]

INOUT argument
  value_expression
    [ ? ]

OUT argument
  out_call_variable
    OUT call placeholder

OUT call placeholder
  parameter_name
    [ CAST — ( — OUT call placeholder — AS — data_type — ) ]
```
**CHECKPOINT**

**Interactive Syntax**

```
CHECKPOINT table_name, NAMED checkpoint_name;
```

**Embedded SQL and Stored Procedure Syntax**

```
CHECKPOINT database_name.table_name, NAMED checkpoint_label,: label_host_variable INTO host_variable_name:
INDICATOR host_indicator_variable_name
```

**COMMENT (Comment-Retrieving Form)**

```
COMMENT ON object_kind.object_name database_name.user_name;
```

**COMMIT**

```
COMMIT WORK RELEASE
```

---

SQL Quick Reference 233
DELETE

Basic/Searched Form

```sql
DELETE
  FROM table_name
  WHERE condition
  ALL
```

Join Condition Form

```sql
DELETE
  FROM table_name
  WHERE condition
  ALL
```

```sql
DELETE
  FROM table_name
  WHERE condition
  ALL
```

```sql
DELETE
  FROM table_name
  WHERE condition
  ALL
```
**ECHO**

```
ECHO
  ' — string — '
  ' — command — '
```

**END TRANSACTION**

```
END TRANSACTION
ET
```

**EXECUTE**

**Macro Form**

```
EXECUTE macro_name
  (parameter_name = constant_expression)
  (constant_expression)
  ;
```

**INSERT/INSERT . . . SELECT**

```
INSERT INTO table_name
  VALUES
  (expression)
  (column_name) — VALUES — (expression)
  subquery
  logging errors
  DEFAULT VALUES
```
LOCKING Request Modifier

logging errors

| LOGGING | ERRORS | WITH | NO LIMIT | LIMIT OF error_limit |

LOCKING DATABASE database_name

| TABLE table_name |
| VIEW view_name |
| ROW |

LOCKING LOCK ACCESS

| FOR IN EXCLUSIVE |
| SHARE |
| READ |
| WRITE |
| OVERRIDE |
| CHECKSUM |

MODE NOWAIT

SQL_request

;
**ROLLBACK**

```
ROLLBACK          WORK         'abort_message'

FROM_clause      WHERE_clause

1101S032
```

**UPDATE**

**Basic Form, No FROM Clause Syntax**

```
UPDATE  table_name
UPD     AS
correlation_name

SET      column_name
column_name.mutator_method_name

WHERE    condition
ALL

1101B210
```

**Basic Form, FROM Clause Syntax**

```
UPDATE  table_name_1
UPD     FROM table_name_2
AS      correlation_name

SET      column_name
column_name.mutator_method_name

WHERE    condition
ALL

1101B209
```
Chapter 5: SQL Data Manipulation Language
USING Request Modifier

**Joined Tables Syntax**

```
UPDATE table_name_1
UPD FROM table_name_2
AS correlation_name
A
SET column_name = expression
WHERE condition
ALL
```

**Upsert Form**

```
UPD table_name_1
FROM correlation_name
A
SET column_name = expression
WHERE condition
ELSE
A
(VALUES expression)
B
VALUES (column_name, expression)
DEFAULT VALUES
```

**USING Request Modifier**

```
USING (using_variable_name - data_type AS DEFERRED LOCATOR BY NAME)
A
SQL_request
```

SQL Quick Reference 239
COLLECT DEMOGRAPHICS

\[
\text{COLLECT DEMOGRAPHICS FOR } \quad \text{database_name.} \quad \text{table_name} \quad ( \text{database_name.} \quad \text{table_name} ) \quad \text{database_name.} \quad \text{table_name} \\
\text{INTO } \quad \text{QCD_name} \quad \text{ALL} \quad \text{WITH NO INDEX} \quad \text{database_name.} \quad \text{table_name} ;
\]

COLLECT STATISTICS (QCD Form)

\[
\text{COLLECT STATISTICS FOR SAMPLE } \quad \text{percentage} \quad \text{PERCENT} \quad \text{database_name.} \quad \text{table_name} \\
\text{INTO } \quad \text{QCD_name} \quad \text{SET QUERY } \quad \text{query_ID} \quad \text{SAMPLEID } \quad \text{statistics_ID} \quad \text{database_name.} \quad \text{table_name} \\
\text{UPDATE MODIFIED ON } \quad \text{database_name.} \quad \text{table_name} \\
\text{COLUMN column_name PARTITION} \\
\text{COLUMN ( column_name PARTITION) } \\
\text{INDEX index_name} \\
\text{INDEX ( column_name )} \\
\]

\[
1101A472
\]

\[
1101D004
\]
Chapter 5: SQL Data Manipulation Language

DROP STATISTICS (QCD Form)

```
DROP STATISTICS FROM-QCD_name ON database_name.
    table_name -A
    database_name.
    user_name.

COLUMN column_name PARTITION
COLUMN (column_name) PARTITION
INDEX index_name
INDEX (column_name)

1101C373
```

DUMP EXPLAIN

```
DUMP EXPLAIN INTO-QCD_name AS-query_plan_name

LIMIT SQL = n

CHECK STATISTICS

SQL_request ;

1101H321
```

EXPLAIN Request Modifier

```
EXPLAIN IN XML

COMPRESS
NODDLTEXT
NODDLTEXT

SQL_request ;

1101C468
```
INITIATE INDEX ANALYSIS

INITIATE INDEX ANALYSIS
ON database_name.table_name FOR workload_name
ON database_name.table_name
IN QCD_name AS index_name_tag
SET boundary_option = value
KEEP_INDEX
USE MODIFIED
STATISTICS
STAT
WITH INDEX_TYPE number
CHECKPOINT checkpoint_trigger
TIME LIMIT = elapsed_time

INITIATE PARTITION ANALYSIS

INITIATE PARTITION ANALYSIS
ON database_name.table_name FOR workload_name
ON database_name.table_name
IN QCD_name AS result_name_tag
TIME LIMIT = elapsed_time

1101F013
1101B443
**INSERT EXPLAIN**

```
INSERT EXPLAIN
  -- WITH
  -- STATISTICS
  -- USING SAMPLE
  -- PERCENT
  -- AND DEMOGRAPHICS
  -- FOR
  -- table_name
  -- INTO QCD_name
  AS query_plan_name
  -- LIMIT
  -- SQL = n
  ;

WITH STATISTICS USING SAMPLE PERCENT AND DEMOGRAPHICS
FOR table_name INTO QCD_name AS query_plan_name
LIMIT SQL = n;
```

**RESTART INDEX ANALYSIS**

```
RESTART INDEX ANALYSIS
  -- FOR
  -- workload_name
  -- IN QCD_name
  AS index_name_tag
  -- TIME LIMIT
  ;

FOR workload_name IN QCD_name AS index_name_tag
  -- CHECKPOINT
  -- checkpoint_trigger
  ;
```

TIME LIMIT = elapsed_time;
Chapter 5: SQL Data Manipulation Language

DIAGNOSTIC COSTPRINT

DIAGNOSTIC DUMP COSTS

DIAGNOSTIC HELP COSTS
Chapter 5: SQL Data Manipulation Language

DIAGNOSTIC SET COSTS

Syntax (Full)

```
DIAGNOSTIC SET COSTS (target_system_name) profile_name,
ON FOR REQUEST IFP SYSTEM
```

Syntax (Restricted)

```
DIAGNOSTIC SET COSTS target_system_name,
ON FOR REQUEST IFP SYSTEM
```

DIAGNOSTIC HELP PROFILE

```
DIAGNOSTIC HELP PROFILE scope_level
( report_option lower_boundary upper_boundary )
```

DIAGNOSTIC SET PROFILE

```
DIAGNOSTIC SET PROFILE profile_name
ON FOR scope_level
```

SQL Quick Reference 245
Chapter 5: SQL Data Manipulation Language

DIAGNOSTIC DUMP SAMPLES

\[
\text{DIAGNOSTIC DUMP SAMPLES} \quad \text{target_system_name}
\]

\[
\text{TABLE} \quad \text{database_name.} \quad \text{table_name}
\]

GO01A001

DIAGNOSTIC HELP SAMPLES

\[
\text{DIAGNOSTIC HELP SAMPLES} \quad \text{target_system_name}
\]

GO01A002

DIAGNOSTIC SET SAMPLES

General Syntax

\[
\text{DIAGNOSTIC SET SAMPLES} \quad \text{target_system_name} \quad \text{NOT ON FOR}
\]

\[
\text{SESSION} \quad \text{TABLE} \quad \text{database_name.} \quad \text{table_name}
\]

GO01A004

Disable All Samples Syntax

\[
\text{DIAGNOSTIC SET ALL SAMPLES NOT ON FOR}
\]

\[
\text{SESSION} \quad \text{SYSTEM}
\]

GO01A003
DIAGNOSTIC "Validate Index"

```sql
DIAGNOSTIC "validate index" NOT ON FOR SESSION ;
```
Chapter 5: SQL Data Manipulation Language
DIAGNOSTIC “Validate Index”
**CLOSE**

CLOSE — cursor_name

**DECLARE CURSOR**

**Dynamic SQL Form**

DECLARE — cursor_name CURSOR FOR — statement_name

**Macro Form**

DECLARE — cursor_name — CURSOR FOR EXEC database_name.

**Request Form**

DECLARE — cursor_name CURSOR FOR — 'request_specification'
Chapter 6: SQL Cursor Control

Selection Form

```
DECLARE `cursor_name` CURSOR FOR
```

- SCROLL
- COMMENT
- EXPLAIN
- HELP
- SHOW
- SELECT
- SELECT AND CONSUME

Stored Procedures Form

```
DECLARE `cursor_name` CURSOR
```

- SCROLL
- NO SCROLL
- WITHOUT RETURN
- WITH RETURN
- ONLY
- TO
- CALLER
- CLIENT
- FOR `cursor_specification`
- FOR
- READ ONLY
- UPDATE
- `statement_name`

DELETE

Positioned Form

```
DELETE FROM `table_name` WHERE CURRENT OF `cursor_name`
```
Chapter 6: SQL Cursor Control

FETCH

Embedded SQL Form

```
FETCH cursor_name
    NEXT
    PRIOR
    FIRST
    LAST
    ABSOLUTE n
    RELATIVE n
    INTO host_variable_name
    USING DESCRIPTOR descriptor_area
    :host_indicator_name
```

STORED PROCEDURES FORM

```
FETCH FROM cursor_name INTO local_variable_name
    NEXT
    FIRST
    local_variable_name
    parameter_reference
```
Chapter 6: SQL Cursor Control

OPEN

Embedded SQL Form

```
OPEN cursor_name
    USING host_variable_name:
        host_indicator_name
    USING DESCRIPTOR descriptor_area
```

GW01A027

Stored Procedures Form

```
OPEN cursor_name
```

```
USING SQL_identifier SQL_parameter
```

1101B073

POSITION

```
POSITION cursor_name
    TO NEXT
    TO STATEMENT statement_number numeric_variable
```

1101A312

PREPARE

```
PREPARE statement_name FROM 'statement_string' statement_string_variable
```

1101A450
REWIND

SELECT ... INTO

Stored Procedures Only

```
with_[recursive]_modifier

SELECT

ALL

DISTINCT

local_variable_name

parameter_name

from_clause

where_clause

INTO A
```

Embedded SQL Only

```
with_[recursive]_modifier

SELECT

select_list

INTO

A

host_variable_name

:indicators

host_indicator_name

from_clause

where_clause
```
SELECT AND CONSUME ... INTO

Stored Procedures Only

```
SELECT AND CONSUME TOP 1 select_list INTO A 
, local_variable_name FROM queue_table_name 
A : parameter_name 
```

Embedded SQL Only

```
SELECT AND CONSUME TOP 1 select_list INTO A 
, host_variable_name INDICATOR host_indicator_name 
B FROM queue_table_name 
```

UPDATE (Positioned Form)

```
UPDATE table_name alias_name SET column_name = expression A 
WHERE CURRENT OF cursor_name A 
```

CHAPTER 7 SQL Stored Procedures: Control Statements and Condition Handling

BEGIN - END Statement

CASE

Syntax 1

Syntax 2
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

CASE
local_declaration

DECLARE variable_name data_type DEFAULT literal NULL FOR sqlstate_code;

condition_name CONDITION

cursor_declaration

DECLARE cursor_name CURSOR SCROLL NO SCROLL WITH RETURN;

WITH RETURN

ONLY TO CALLER CLIENT

FOR cursor_specification

statement_name

PREPARE statement_name FROM 'statement_string'

statement_string_variable

cursor_specification

SELECT column_name alias_name AS

expression AS

FROM table_name

INNER JOIN OUTER LEFT RIGHT FULL

WHERE clause other SELECT clauses

SELECT

column_name alias_name AS

expression AS

FROM table_name

WHERE clause other SELECT clauses

PREPARE statement_name FROM 'statement_string'

statement_string_variable
CASE

### condition_handler

```
DECLARE
  CONTINUE
  HANDLER
    EXIT
  condition_name
    CONDITION
```

```
FOR SQLSTATEsqlstate_code
  VALUE
    SQLWARNING
    NOT FOUND
    condition_name
```

```
SQLSTATEsqlstate_code
```

### open statement

```
OPEN cursor_name
  USING SQL_identifier
```

```
FOR NEXT FIRST
  local_variable_name
    parameter_reference
```

### fetch statement

```
FETCH cursor_name INTO
```

### assignment statement

```
SET assignment_target = assignment_source
```
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

CASE

condition statement

- CASE
  - operand_1
    - WHEN conditional_expression THEN statement
    - ELSE statement END CASE

- WHEN conditional_expression THEN statement
- ELSE statement END CASE

- IF conditional_expression THEN statement
- ELSEIF conditional_expression THEN statement
- ELSE statement END IF

iteration statement

- WHILE conditional_expression DO statement END WHILE
- LOOP statement END LOOP
- FOR for_loop_variable AS cursor_specification DO statement END FOR
- REPEAT statement UNTIL conditional_expression END REPEAT

diagnostic statement

- SIGNAL condition_name
  - SQLSTATE VALUE SQLSTATE_code
  - SET condition_information_item=value
- RESIGNAL condition_name
  - SQLSTATE VALUE SQLSTATE_code
  - SET condition_information_item=value
- GET DIAGNOSTICS parameter_name = statement_information_item
- exception-condition_number parameter_name = condition_information_item

SQL Quick Reference 259
DECLARE

DECLARE variable_name predefined_data_type UDT_name attribute A

DEFAULT literal A FOR for_loop_variable AS DO statement END FOR cur or curs specification CURSOR FOR cursor_name label_name : A

cursor_specification

SELECT column_name AS correlation_name AS expression AS correlation_name A

FROM table_name B

WHERE clause other SELECT clauses B

OUTER LEFT RIGHT FULL

INNER JOIN ON OUTER LEFT RIGHT FULL

A

B

1101A372

A

1148A006

1101B384
local_declaration

DECLARE 
variable_name 
data_type 

condition_name 
CONDITION 
FOR sqlstate_code 
DEFAULT 
literal 
NULL 

DEALLOCATE

cursor_declaration

DECLARE 
cursor_name 
CURSOR 

SCROLL 
NO SCROLL 

WITH RETURN 
ONLY 
TO 
CALLER 
CLIENT 

FOR 
cursor_specification 

statement_name 

PREPARE 
statement_name 
FROM 
statement_string 
statement_string_variable 

SELECT 
column_name 

AS 
alias_name 

expression 
AS 
alias_name 

FROM 
table_name 

INNER 
JOIN 

LEFT 
RIGHT 
FULL 

WHERE clause 
other SELECT clauses 

1101C448 

1101B384
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

**SQL Quick Reference**

**Condition Statement**

- **CASE**
  - **operand_1**
  - **WHEN** **operand_2** **THEN** **statement**
  - **ELSE** **statement**
  - **END CASE**

- **IF** **conditional_expression** **THEN** **statement**
  - **ELSE** **statement**
  - **END IF**

- **ELSEIF** **conditional_expression** **THEN** **statement**

- **WHEN** **conditional_expression** **THEN** **statement**

**Iteration Statement**

- **WHILE** **conditional_expression** **DO** **statement**
  - **END WHILE**

- **LOOP** **statement**
  - **END LOOP**

- **FOR** **for_loop_variable** **AS**
  - **cursor_name** **CURSOR FOR** **cursor_specification**
  - **DO** **statement**
  - **END FOR**

- **REPEAT** **statement** **UNTIL** **conditional_expression** **END REPEAT**

**Diagnostic Statement**

- **SIGNAL** **condition_name**
  - **SQLSTATE** **SQLSTATE_code** **SET** **condition_information_item=value**

- **RESIGNAL** **condition_name**
  - **SQLSTATE** **SQLSTATE_code** **SET** **condition_information_item=value**

- **GET DIAGNOSTICS**
  - **parameter_name** **=** **statement_information_item**
  - **variable_name**

- **EXCEPTION** **condition_number**
  - **parameter_name** **=** **condition_information_item**
  - **variable_name**
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

IF

```
IF  conditional_expression  THEN  statement  A

ELSEIF  conditional_expression  THEN  statement  B

ELSE  statement

END IF
```

```
SQL_statement

BEGIN REQUEST  SQL_multistatement_request  END REQUEST

| compound statement |
| open statement |
| fetch statement |
| assignment statement |
| condition statement |
| iteration statement |
| diagnostic statement |

ITERATE  label_name

LEAVE  label_name
```

```
compound statement

label_name : BEGIN local_declaration cursor_declaration END label_name

condition_handler  statement  label_name
```

```

1148A007

1101C234

1101A383

SQL Quick Reference 265
**Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling**

**local_declaration**

```
DECLARE variable_name data_type
DEFAULT literal
NOT NULL
CONDITION FOR sqlstate_code
```

**cursor_declaration**

```
DECLARE cursor_name CURSOR FOR statement_name
```

**cursor_specification**

```
DECLARE WITH RETURN ONL Y CURSOR FOR statement_name
```

**cursor_specification**

```
SELECT column_name AS alias_name
FROM table_name
```

```
WHERE clause
```

```
other SELECT clauses
```
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

**Condition Statement**

```plaintext
CASE
  operand_1 WHEN operand_2 THEN [statement];
  WHEN conditional_expression THEN [statement];
  ELSE [statement];
END CASE
```

```plaintext
IF conditional_expression THEN [statement];
ELSE [statement];
END IF
```

```plaintext
ELSEIF conditional_expression THEN [statement];
END IF
```

**Iteration Statement**

```plaintext
WHILE conditional_expression DO [statement]; END WHILE
```

```plaintext
LOOP [statement]; END LOOP
```

```plaintext
FOR for_loop_variable AS cursor_name CURSOR FOR cursor_specification DO [statement]; END FOR
```

```plaintext
REPEAT [statement]; UNTIL conditional_expression END REPEAT
```

**Diagnostic Statement**

```plaintext
SIGNAL condition_name 
  SQLSTATE [VALUE] SQLSTATE_code
  SET condition_information_item=value;
RESIGNAL condition_name 
  SQLSTATE [VALUE] SQLSTATE_code
  SET condition_information_item=value;
GET DIAGNOSTICS [parameter_name]=statement_information_item [variable_name];
EXCEPTION condition_number [parameter_name]=condition_information_item [variable_name];
```
compound statement

```
label_name : BEGIN
  local_declaration
  cursor_declaration
  condition_handler
  statement
END
```

local_declaration

```
DECLARE
  variable_name
  data_type
  DEFAULT
  literal
  NULL
  condition_name
  CONDITION
  FOR sqlstate_code
;```

cursor_declaration

```
DECLARE
  cursor_name
  SCROLL
  NO SCROLL
  CURSOR
WITH RETURN
  ONLY
  TO
  CALLER
  CLIENT
FOR
  cursor_specification
  FOR
  READ ONLY
  UPDATE
  statement_name
;```

PREPARE
  statement_name
  FROM
  statement_string
  statement_string_variable
;```
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

**cursor_specification**

```sql
SELECT column_name AS alias_name, expression AS alias_name
FROM table_name
INNER JOIN table_name ON condition
OUTER LEFT RIGHT FULL
WHERE clause
```

**condition_handler**

```sql
DECLARE HANDLER condition_name FOR SQLSTATE 'sqlstate_code'
```

**open statement**

```sql
OPEN cursor_name USING SQL_identifier SQL_parameter
```
fetch statement

```
FETCH cursor_name INTO A
FROM NEXT FIRST
A local_variable_name parameter_reference ;
```

assignment statement

```
SET assignment_target = assignment_source
```

condition statement

```
CASE operand_1 WHEN operand_2 THEN [statement]
WHEN conditional_expression THEN [statement]
ELSE [statement] END CASE

IF conditional_expression THEN [statement]
ELSEIF conditional_expression THEN [statement]
ELSE [statement] END IF
```
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

iteration statement

WHILE conditional_expression DO statement ; END WHILE

LOOP statement ; END LOOP

FOR for_loop_variable — AS cursor_name — CURSOR FOR

END FOR

REPEAT statement ; UNTIL conditional_expression END REPEAT

diagnostic statement

SIGNAL condition_name

SQLSTATE SQLSTATE_code SET condition_information_item=value

RESIGNAL condition_name

SQLSTATE SQLSTATE_code SET condition_information_item=value

GET DIAGNOSTICS parameter_name = statement_information_item

variable_name = condition_information_item

EXCEPTION condition_number parameter_name = condition_information_item

variable_name

REPEAT

label_name:

statement

UNTIL A

conditional_expression

END REPEAT

label_name

YSERPT01
**SET**

\[ \text{SET} \quad \text{assignment_target} = \quad \text{assignment_source} \quad ; \]

**WHILE**

\[ \text{WHILE} \quad \text{conditional_expression} \quad \text{DO} \quad \text{statement} \quad \text{END WHILE} \]

\[ \text{statement} \]

\[ \text{BEGIN REQUEST} \quad \text{SQL\_multistatement\_request} \quad \text{END REQUEST} \]

- compound statement
- open statement
- fetch statement
- assignment statement
- condition statement
- iteration statement
- diagnostic statement
- ITERATE \[ \text{label_name} \]
- LEAVE \[ \text{label_name} \]
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

WHILE

cursor_specification

```sql
SELECT column_name
  AS alias_name
FROM table_name
  INNER JOIN table_name ON condition
  LEFT OUTER RIGHT FULL
WHERE clause
```

condition_handler

```sql
DECLARE CONTINUE HANDLER FOR SQLSTATE sqlstate_code
  , SQLWARNING NOT FOUND SQLSTATE sqlstate_code
  , OTHER SQLSTATE sqlstate_code
```

open statement

```sql
OPEN cursor_name USING SQL_identifier SQL_parameter
```
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

**fetch statement**

```
FETCH cursor_name INTO A
FROM NEXT FIRST
local_variable_name parameter_reference ;
```

**assignment statement**

```
SET assignment_target = assignment_source
```

**condition statement**

```
CASE operand_1
  WHEN operand_2 THEN statement;
  WHEN conditional_expression THEN statement;
  ELSE statement;
END CASE

IF conditional_expression THEN statement;
ELSEIF conditional_expression THEN statement;
ELSE statement;
END IF
```
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

DECLARE CONDITION

iteration statement

WHILE  conditional_expression  DO  \_statement  \  END WHILE

LOOP  \_statement  \  END LOOP

FOR  for_loop_variable  AS  cursor_name  CURSOR FOR  cursor_specification  DO  \_statement  \  END FOR

REPEAT  \_statement  \  UNTIL  conditional_expression  END REPEAT

---

diagnostic statement

SIGNAL  condition_name  SQLSTATE  SQLSTATE_code  SET  condition_information_item=value

RESIGNAL  condition_name  SQLSTATE  SQLSTATE_code  SET  condition_information_item=value

GET DIAGNOSTICS  parameter_name  =  statement_information_item

EXCEPTION  condition_number  parameter_name  =  condition_information_item

---

DECLARE CONDITION

DECLARE  condition_name  CONDITION  FOR SQLSTATE  VALUE  sqlstate_code

---
Chapter 7: SQL Stored Procedures: Control Statements and Condition Handling

DECLARE HANDLER (Basic Syntax)

```plaintext
DECLARE HANDLER FOR 
EXCEPTION 
SQLSTATE sqlstate_code 
VALUE condition_name ; 
; 
; 
; 
; 

SIGNAL 
condition_name 
SQLSTATE SQLSTATE_code 
VALUE SET condition_information_item=value ; 

RESIGNAL 
condition_name 
SQLSTATE SQLSTATE_code 
VALUE SET condition_information_item=value ; 
```

SQL Quick Reference 279
GET DIAGNOSTICS

GET DIAGNOSTICS
parameter_name = statement_information_item
variable_name

EXCEPTION-condition_number
parameter_name = condition_information_item
variable_name
BEGIN DECLARE SECTION

COMMENT

Returning Form

DATABASE

SQL Quick Reference  281
Chapter 8: Static Embedded SQL Statements

DECLARE STATEMENT

DECLARE table_name TABLE A

DECLARE view_name (column_name data_type null_attribute)

END DECLARE SECTION

END-EXEC Statement Terminator
EXEC

EXEC — macro_name — (parameter_list)

EXEC SQL Statement Prefix

EXEC SQL
FOR count_value — embedded_sql_statement — sql_statement_terminator

INCLUDE

INCLUDE — include_file_name

INCLUDE SQLCA

INCLUDE SQLCA
Chapter 8: Static Embedded SQL Statements

INCLUDE SQLDA

INCLUDE SQLDA

WHENEVER

WHENEVER condition action

GW01A022

GW01R035
CHAPTER 9 Dynamic Embedded SQL Statements

DESCRIBE

```sql
DESCRIBE statement_name INTO : descriptor_area
  USING NAMES | ANY | BOTH LABELS
  FOR STATEMENT statement_number
    statement_number_variable
```

EXECUTE

Dynamic SQL Form

```sql
EXECUTE statement_name
  USING host_variable_name
    indicator
  USING DESCRIPTOR descriptor_area
```

1101B017

GW01A017
EXECUTE IMMEDIATE

PREPARE
CHAPTER 10 SQL Client-Server Connectivity

**CONNECT**

CONNECT user_id_variable - IDENTIFIED BY password_variable AS connection_name

**GET CRASH**

GET CRASH — WAIT, TELL — INTO wait_variable, tell_variable,

**LOGOFF**

LOGOFF CURRENT ALL connection_name

:connection_name_variable
Chapter 10: SQL Client-Server Connectivity Statements

LOGON

LOGON

SET BUFFERSIZE

SET CHARSET

SET CONNECTION

SET CRASH
Chapter 10: SQL Client-Server Connectivity Statements

SET ENCRYPTION
ASYNC Statement Modifier

ASYNC (async_statement_identifier async_SQL_statement :async_statement_identifier_variable_name)

TEST

TEST async_statement_identifier COMPLETION :async_statement_identifier_variable_name

WAIT

WAIT async_statement_identifier COMPLETION ALL ANY COMPLETION INTO statement_variable session_variable
This appendix describes the conventions that apply to reading the syntax diagrams used in this book.

## Syntax Diagram Convention

### Notation Conventions

<table>
<thead>
<tr>
<th>Item</th>
<th>Definition / Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Letter</td>
<td>An uppercase or lowercase alphabetic character ranging from A through Z.</td>
</tr>
<tr>
<td>Number</td>
<td>A digit ranging from 0 through 9.</td>
</tr>
<tr>
<td></td>
<td>Do not use commas when typing a number with more than 3 digits.</td>
</tr>
<tr>
<td>Word</td>
<td>Keywords and variables.</td>
</tr>
<tr>
<td></td>
<td>- UPPERCASE LETTERS represent a keyword.</td>
</tr>
<tr>
<td></td>
<td>Syntax diagrams show all keywords in uppercase, unless operating system restrictions</td>
</tr>
<tr>
<td></td>
<td>require them to be in lowercase.</td>
</tr>
<tr>
<td></td>
<td>- lowercase letters represent a keyword that you must type in lowercase, such as a</td>
</tr>
<tr>
<td></td>
<td>UNIX command.</td>
</tr>
<tr>
<td></td>
<td>- <strong>lowercase italic letters</strong> represent a variable such as a column or table name.</td>
</tr>
<tr>
<td></td>
<td>Substitute the variable with a proper value.</td>
</tr>
<tr>
<td></td>
<td>- <strong>lowercase bold letters</strong> represent an excerpt from the diagram. The excerpt is</td>
</tr>
<tr>
<td></td>
<td>defined immediately following the diagram that contains it.</td>
</tr>
<tr>
<td></td>
<td>- <strong>UNDERLINED LETTERS</strong> represent the default value.</td>
</tr>
<tr>
<td></td>
<td>This applies to both uppercase and lowercase words.</td>
</tr>
<tr>
<td>Spaces</td>
<td>Use one space between items such as keywords or variables.</td>
</tr>
<tr>
<td>Punctuation</td>
<td>Type all punctuation exactly as it appears in the diagram.</td>
</tr>
</tbody>
</table>

### Paths

The main path along the syntax diagram begins at the left with a keyword, and proceeds, left to right, to the vertical bar, which marks the end of the diagram. Paths that do not have an arrow or a vertical bar only show portions of the syntax.

The only part of a path that reads from right to left is a loop.
Continuation Links
Paths that are too long for one line use continuation links. Continuation links are circled letters indicating the beginning and end of a link:

When you see a circled letter in a syntax diagram, go to the corresponding circled letter and continue reading.

Required Entries
Required entries appear on the main path:

If you can choose from more than one entry, the choices appear vertically, in a stack. The first entry appears on the main path:

Optional Entries
You may choose to include or disregard optional entries. Optional entries appear below the main path:

If you can optionally choose from more than one entry, all the choices appear below the main path:
Some commands and statements treat one of the optional choices as a default value. This value is **UNDERLINED**. It is presumed to be selected if you type the command or statement without specifying one of the options.

**Strings**

String literals appear in single quotes:

```
'msgtext'
```

**Abbreviations**

If a keyword or a reserved word has a valid abbreviation, the unabbreviated form always appears on the main path. The shortest valid abbreviation appears beneath.

In the above syntax, the following formats are valid:

- SHOW CONTROLS
- SHOW CONTROL

**Loops**

A loop is an entry or a group of entries that you can repeat one or more times. Syntax diagrams show loops as a return path above the main path, over the item or items that you can repeat:

Read loops from right to left.

The following conventions apply to loops:
Appendix A: How to Read Syntax Diagrams
Syntax Diagram Conventions

**Excerpts**

Sometimes a piece of a syntax phrase is too large to fit into the diagram. Such a phrase is indicated by a break in the path, marked by (|) terminators on each side of the break. The name for the excerpted piece appears between the terminators in boldface type. The boldface excerpt name and the excerpted phrase appears immediately after the main diagram. The excerpted phrase starts and ends with a plain horizontal line:

```
LOCKING | excerpt | A
        A -------HAVING con------
```

Multiple Legitimate Phrases

In a syntax diagram, it is possible for any number of phrases to be legitimate:

```
IF... | THEN...
--------- | --------------
there is a maximum number of entries allowed | the number appears in a circle on the return path. In the example, you may type `cname` a maximum of 4 times.
there is a minimum number of entries required | the number appears in a square on the return path. In the example, you must type at least three groups of column names.
a separator character is required between entries | the character appears on the return path. If the diagram does not show a separator character, use one blank space. In the example, the separator character is a comma.
a delimiter character is required around entries | the beginning and end characters appear outside the return path. Generally, a space is not needed between delimiter characters and entries. In the example, the delimiter characters are the left and right parentheses.
```
In this example, any of the following phrases are legitimate:

- `dbname`
- `DATABASE dbname`
- `tname`
- `TABLE tname`
- `vname`
- `VIEW vname`

**Sample Syntax Diagram**

```
CREATE VIEW viewname AS
  cname AS LOCKING LOCK
  CV

DATABASE dbname
  TABLE tname
  VIEW vname

FOR IN

ACCESS
  LOCKING
  LOCK

MODE
  SHARE
  READ
  WRITE
  EXCLUSIVE
  EXCL

SEL expr FROM tname
  aname
  qual_cond

HAVING cond

qual_cond

WHERE cond
  GROUP BY cnames
  col_pos
```
Diagram Identifier

The alphanumeric string that appears in the lower right corner of every diagram is an internal identifier used to catalog the diagram. The text never refers to this string.