

Oracle® Database

Database Concepts



18c
E84295-04
August 2018



Petitioner has added romanettes to pages (i) - (ii). Otherwise, it leaves the original page numbering.

SAP
EXHIBIT 1013

(i)

Oracle Database Database Concepts, 18c

E84295-04

Copyright © 1993, 2018, Oracle and/or its affiliates. All rights reserved.

Primary Authors: Lance Ashdown, Tom Kyte, Joe McCormack

Contributors: Drew Adams, Ashish Agrawal, Troy Anthony, Vikas Arora, Jagan Athraya, David Austin, Thomas Baby, Vladimir Barriere, Hermann Baer, Srinagesh Battula, Nigel Bayliss, Tammy Bednar, Virginia Beecher, Bjorn Bolltoft, Ashmita Bose, David Brower, Larry Carpenter, Lakshminaray Chidambaran, Deba Chatterjee, Shasank Chavan, Tim Chien, Gregg Christman, Bernard Clouse, Maria Colgan, Carol Colrain, Nelson Corcoran, Michael Coulter, Jonathan Creighton, Judith D'Addieco, Mark Dilman, Kurt Engeleiter, Bjørn Engsig, Marcus Fallon, Steve Fogel, Jonathan Giloni, Naveen Gopal, Bill Habeck, Min-Hank Ho, Lijie Heng, Bill Hodak, Yong Hu, Pat Huey, Praveen Kumar Tupati Jaganath, Sanket Jain, Prakash Jashnani, Caroline Johnston, Shantanu Joshi, Jesse Kamp, Vikram Kapoor, Feroz Khan, Jonathan Klein, Andre Kruglikov, Sachin Kulkarni, Surinder Kumar, Paul Lane, Adam Lee, Allison Lee, Jaebock Lee, Sue Lee, Teck Hua Lee, Yunrui Li, Ilya Listvinski, Bryn Llewellyn, Rich Long, Barb Lundhild, Neil Macnaughton, Vineet Marwah, Susan Mavris, Bob McGuirk, Joseph Meeks, Mughees Minhas, Sheila Moore, Valarie Moore, Gopal Mulagund, Charles Murray, Kevin Neel, Sue Pelski, Raymond Pfau, Gregory Pongracz, Vivek Raja, Ashish Ray, Bert Rich, Kathy Rich, Andy Rivenes, Scott Rotondo, Vivian Schupmann, Venkat Senaptai, Shrikanth Shankar, Prashanth Shanthaveerappa, Cathy Shea, Susan Shepard, Kam Shergill, Mike Skarpelos, Sachin Sonawane, James Spiller, Suresh Sridharan, Jim Stenoish, Janet Stern, Rich Strohm, Roy Swonger, Kamal Tbeileh, Juan Tellez, Ravi Thammaiah, Lawrence To, Tomohiro Ueda, Randy Urbano, Badhri Varanasi, Nick Wagner, Steve Wertheimer, Patrick Wheeler, Doug Williams, James Williams, Andrew Witkowski, Daniel Wong, Hailing Yu

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

Preface

Audience	xxi
Documentation Accessibility	xxi
Related Documentation	xxii
Conventions	xxii

Changes in This Release for Oracle Database Concepts

Changes in Oracle Database Release 18c, Version 18.1	xxiii
Changes in Oracle Database 12c Release 2 (12.2.0.1)	xxiv
Changes in Oracle Database 12c Release 1 (12.1.0.2)	xxvi
Changes in Oracle Database 12c Release 1 (12.1.0.1)	xxvii

1 Introduction to Oracle Database

About Relational Databases	1-1
Database Management System (DBMS)	1-1
Relational Model	1-2
Relational Database Management System (RDBMS)	1-2
Brief History of Oracle Database	1-3
Schema Objects	1-4
Tables	1-4
Indexes	1-5
Data Access	1-5
Structured Query Language (SQL)	1-5
PL/SQL and Java	1-6
Transaction Management	1-7
Transactions	1-7
Data Concurrency	1-7
Data Consistency	1-8
Oracle Database Architecture	1-8
Database and Instance	1-9
Multitenant Architecture	1-11

Sharding Architecture	1-14
Database Storage Structures	1-16
Physical Storage Structures	1-16
Logical Storage Structures	1-17
Database Instance Structures	1-18
Oracle Database Processes	1-18
Instance Memory Structures	1-18
Application and Networking Architecture	1-19
Application Architecture	1-19
Oracle Net Services Architecture	1-20
Oracle Database Documentation Roadmap	1-21
Oracle Database Documentation: Basic Group	1-21
Oracle Database Documentation: Intermediate Group	1-21
Oracle Database Documentation: Advanced Group	1-22

Part I Oracle Relational Data Structures

2 Tables and Table Clusters

Introduction to Schema Objects	2-1
Schema Object Types	2-2
Schema Object Storage	2-4
Schema Object Dependencies	2-5
SYS and SYSTEM Schemas	2-7
Sample Schemas	2-7
Overview of Tables	2-8
Columns	2-9
Virtual Columns	2-9
Invisible Columns	2-10
Rows	2-10
Example: CREATE TABLE and ALTER TABLE Statements	2-10
Oracle Data Types	2-12
Character Data Types	2-13
Numeric Data Types	2-15
Datetime Data Types	2-16
Rowid Data Types	2-17
Format Models and Data Types	2-18
Integrity Constraints	2-19
Table Storage	2-19
Table Organization	2-20
Row Storage	2-21

Rowids of Row Pieces	2-21
Storage of Null Values	2-21
Table Compression	2-22
Basic Table Compression and Advanced Row Compression	2-22
Hybrid Columnar Compression	2-23
Overview of Table Clusters	2-27
Overview of Indexed Clusters	2-28
Overview of Hash Clusters	2-30
Hash Cluster Creation	2-30
Hash Cluster Queries	2-31
Hash Cluster Variations	2-32
Hash Cluster Storage	2-32
Overview of Attribute-Clustered Tables	2-34
Advantages of Attribute-Clustered Tables	2-34
Join Attribute Clustered Tables	2-35
I/O Reduction Using Zones	2-35
Zone Maps	2-35
Zone Maps: Analogy	2-36
Zone Maps: Example	2-36
Attribute-Clustered Tables with Linear Ordering	2-37
Attribute-Clustered Tables with Interleaved Ordering	2-38
Overview of Temporary Tables	2-40
Purpose of Temporary Tables	2-40
Segment Allocation in Temporary Tables	2-41
Temporary Table Creation	2-41
Overview of External Tables	2-42
Purpose of External Tables	2-42
External Table Access Drivers	2-42
External Table Creation	2-43
Overview of Object Tables	2-44

3 Indexes and Index-Organized Tables

Introduction to Indexes	3-1
Benefits of Indexes	3-2
Index Usability and Visibility	3-2
Keys and Columns	3-3
Composite Indexes	3-3
Unique and Nonunique Indexes	3-5
Types of Indexes	3-5
How the Database Maintains Indexes	3-6

Index Storage	3-6
Overview of B-Tree Indexes	3-7
Branch Blocks and Leaf Blocks	3-8
Index Scans	3-8
Full Index Scan	3-9
Fast Full Index Scan	3-9
Index Range Scan	3-10
Index Unique Scan	3-11
Index Skip Scan	3-11
Index Clustering Factor	3-12
Reverse Key Indexes	3-14
Ascending and Descending Indexes	3-14
Index Compression	3-15
Prefix Compression	3-15
Advanced Index Compression	3-17
Overview of Bitmap Indexes	3-19
Example: Bitmap Indexes on a Single Table	3-19
Bitmap Join Indexes	3-21
Bitmap Storage Structure	3-23
Overview of Function-Based Indexes	3-24
Uses of Function-Based Indexes	3-24
Optimization with Function-Based Indexes	3-25
Overview of Application Domain Indexes	3-26
Overview of Index-Organized Tables	3-26
Index-Organized Table Characteristics	3-27
Index-Organized Tables with Row Overflow Area	3-30
Secondary Indexes on Index-Organized Tables	3-30
Logical Rowids and Physical Guesses	3-31
Bitmap Indexes on Index-Organized Tables	3-32

4 Partitions, Views, and Other Schema Objects

Overview of Partitions	4-1
Partition Characteristics	4-2
Partition Key	4-2
Partitioning Strategies	4-2
Partitioned Tables	4-12
Partitioned Indexes	4-13
Local Partitioned Indexes	4-13
Global Partitioned Indexes	4-16
Partial Indexes for Partitioned Tables	4-18

Partitioned Index-Organized Tables	4-19
Sharded Tables	4-20
Overview of Views	4-21
Characteristics of Views	4-23
Data Manipulation in Views	4-23
How Data Is Accessed in Views	4-24
Updatable Join Views	4-25
Object Views	4-26
Overview of Materialized Views	4-27
Characteristics of Materialized Views	4-28
Refresh Methods for Materialized Views	4-29
Complete Refresh	4-29
Incremental Refresh	4-29
In-Place and Out-of-Place Refresh	4-30
Query Rewrite	4-31
Overview of Sequences	4-32
Sequence Characteristics	4-32
Concurrent Access to Sequences	4-32
Overview of Dimensions	4-33
Hierarchical Structure of a Dimension	4-34
Creation of Dimensions	4-34
Overview of Synonyms	4-35

5 Data Integrity

Introduction to Data Integrity	5-1
Techniques for Guaranteeing Data Integrity	5-1
Advantages of Integrity Constraints	5-2
Types of Integrity Constraints	5-2
NOT NULL Integrity Constraints	5-3
Unique Constraints	5-4
Primary Key Constraints	5-6
Foreign Key Constraints	5-7
Self-Referential Integrity Constraints	5-9
Nulls and Foreign Keys	5-9
Parent Key Modifications and Foreign Keys	5-9
Indexes and Foreign Keys	5-11
Check Constraints	5-11
States of Integrity Constraints	5-12
Checks for Modified and Existing Data	5-12
When the Database Checks Constraints for Validity	5-13

Nondeferrable Constraints	5-13
Deferrable Constraints	5-14
Examples of Constraint Checking	5-14
Example: Insertion of a Value in a Foreign Key Column When No Parent Key Value Exists	5-15
Example: Update of All Foreign Key and Parent Key Values	5-15

6 Data Dictionary and Dynamic Performance Views

Overview of the Data Dictionary	6-1
Contents of the Data Dictionary	6-2
Views with the Prefix DBA_	6-3
Views with the Prefix ALL_	6-3
Views with the Prefix USER_	6-4
The DUAL Table	6-4
Storage of the Data Dictionary	6-5
How Oracle Database Uses the Data Dictionary	6-5
Public Synonyms for Data Dictionary Views	6-6
Data Dictionary Cache	6-6
Other Programs and the Data Dictionary	6-6
Overview of the Dynamic Performance Views	6-6
Contents of the Dynamic Performance Views	6-7
Storage of the Dynamic Performance Views	6-8
Database Object Metadata	6-8

Part II Oracle Data Access

7 SQL

Introduction to SQL	7-1
SQL Data Access	7-2
SQL Standards	7-2
Overview of SQL Statements	7-3
Data Definition Language (DDL) Statements	7-3
Data Manipulation Language (DML) Statements	7-5
SELECT Statements	7-6
Joins	7-7
Subqueries	7-8
Transaction Control Statements	7-9
Session Control Statements	7-10
System Control Statement	7-10

Embedded SQL Statements	7-11
Overview of the Optimizer	7-11
Use of the Optimizer	7-11
Optimizer Components	7-13
Query Transformer	7-14
Estimator	7-14
Plan Generator	7-14
Access Paths	7-15
Optimizer Statistics	7-16
Optimizer Hints	7-17
Overview of SQL Processing	7-18
Stages of SQL Processing	7-18
SQL Parsing	7-19
SQL Optimization	7-20
SQL Row Source Generation	7-20
SQL Execution	7-20
Differences Between DML and DDL Processing	7-21

8 Server-Side Programming: PL/SQL and Java

Introduction to Server-Side Programming	8-1
Overview of PL/SQL	8-2
PL/SQL Subprograms	8-3
Advantages of PL/SQL Subprograms	8-3
Creation of PL/SQL Subprograms	8-5
Execution of PL/SQL Subprograms	8-5
PL/SQL Packages	8-7
Advantages of PL/SQL Packages	8-7
Creation of PL/SQL Packages	8-8
Execution of PL/SQL Package Subprograms	8-8
PL/SQL Anonymous Blocks	8-9
PL/SQL Language Constructs	8-10
PL/SQL Collections and Records	8-11
Collections	8-11
Records	8-11
How PL/SQL Runs	8-12
Overview of Java in Oracle Database	8-13
Overview of the Java Virtual Machine (JVM)	8-15
Overview of Oracle JVM	8-15
Main Components of Oracle JVM	8-16
Java Programming Environment	8-17

Java Stored Procedures	8-18
Java and PL/SQL Integration	8-18
Overview of Triggers	8-20
Advantages of Triggers	8-20
Types of Triggers	8-21
Timing for Triggers	8-22
Creation of Triggers	8-22
Example: CREATE TRIGGER Statement	8-23
Example: Invoking a Row-Level Trigger	8-24
Execution of Triggers	8-26
Storage of Triggers	8-27

Part III Oracle Transaction Management

9 Data Concurrency and Consistency

Introduction to Data Concurrency and Consistency	9-1
Multiversion Read Consistency	9-2
Statement-Level Read Consistency	9-2
Transaction-Level Read Consistency	9-3
Read Consistency and Undo Segments	9-3
Locking Mechanisms	9-5
ANSI/ISO Transaction Isolation Levels	9-6
Overview of Oracle Database Transaction Isolation Levels	9-7
Read Committed Isolation Level	9-7
Read Consistency in the Read Committed Isolation Level	9-7
Conflicting Writes in Read Committed Transactions	9-8
Serializable Isolation Level	9-10
Read-Only Isolation Level	9-14
Overview of the Oracle Database Locking Mechanism	9-15
Summary of Locking Behavior	9-15
Use of Locks	9-16
Lock Modes	9-19
Lock Conversion and Escalation	9-19
Lock Duration	9-20
Locks and Deadlocks	9-20
Overview of Automatic Locks	9-22
DML Locks	9-22
Row Locks (TX)	9-23
Table Locks (TM)	9-26
Locks and Foreign Keys	9-27

DDL Locks	9-31
Exclusive DDL Locks	9-31
Share DDL Locks	9-31
Breakable Parse Locks	9-31
System Locks	9-32
Latches	9-32
Mutexes	9-33
Internal Locks	9-33
Overview of Manual Data Locks	9-34
Overview of User-Defined Locks	9-35

10 Transactions

Introduction to Transactions	10-1
Sample Transaction: Account Debit and Credit	10-2
Structure of a Transaction	10-3
Beginning of a Transaction	10-3
End of a Transaction	10-4
Statement-Level Atomicity	10-5
System Change Numbers (SCNs)	10-6
Overview of Transaction Control	10-6
Transaction Names	10-8
Active Transactions	10-8
Savepoints	10-9
Rollback to Savepoint	10-9
Enqueued Transactions	10-10
Rollback of Transactions	10-11
Commits of Transactions	10-12
Overview of Transaction Guard	10-13
Benefits of Transaction Guard	10-14
How Transaction Guard Works	10-15
Lost Commit Messages	10-15
Logical Transaction ID	10-15
Transaction Guard: Example	10-16
Overview of Application Continuity	10-17
Benefits of Application Continuity	10-18
Use Case for Application Continuity	10-18
Application Continuity for Planned Maintenance	10-18
Application Continuity Architecture	10-19
Overview of Autonomous Transactions	10-20
Overview of Distributed Transactions	10-21

Two-Phase Commit	10-22
In-Doubt Transactions	10-22

Part IV Oracle Database Storage Structures

11 Physical Storage Structures

Introduction to Physical Storage Structures	11-1
Mechanisms for Storing Database Files	11-2
Oracle Automatic Storage Management (Oracle ASM)	11-3
Oracle ASM Storage Components	11-3
Oracle ASM Instances	11-5
Oracle Managed Files and User-Managed Files	11-6
Overview of Data Files	11-7
Use of Data Files	11-7
Permanent and Temporary Data Files	11-8
Online and Offline Data Files	11-9
Data File Structure	11-10
Overview of Control Files	11-11
Use of Control Files	11-11
Multiple Control Files	11-12
Control File Structure	11-12
Overview of the Online Redo Log	11-13
Use of the Online Redo Log	11-13
How Oracle Database Writes to the Online Redo Log	11-14
Online Redo Log Switches	11-14
Multiple Copies of Online Redo Log Files	11-16
Archived Redo Log Files	11-17
Structure of the Online Redo Log	11-18

12 Logical Storage Structures

Introduction to Logical Storage Structures	12-1
Logical Storage Hierarchy	12-2
Logical Space Management	12-3
Locally Managed Tablespaces	12-4
Dictionary-Managed Tablespaces	12-7
Overview of Data Blocks	12-7
Data Blocks and Operating System Blocks	12-7
Database Block Size	12-8
Tablespace Block Size	12-8

Data Block Format	12-9
Data Block Overhead	12-9
Row Format	12-10
Data Block Compression	12-13
Space Management in Data Blocks	12-14
Percentage of Free Space in Data Blocks	12-15
Optimization of Free Space in Data Blocks	12-16
Chained and Migrated Rows	12-18
Overview of Index Blocks	12-19
Types of Index Blocks	12-20
Storage of Index Entries	12-20
Reuse of Slots in an Index Block	12-20
Coalescing an Index Block	12-21
Overview of Extents	12-22
Allocation of Extents	12-22
Deallocation of Extents	12-24
Storage Parameters for Extents	12-25
Overview of Segments	12-26
User Segments	12-26
User Segment Creation	12-27
Temporary Segments	12-28
Allocation of Temporary Segments for Queries	12-29
Allocation of Segments for Temporary Tables and Indexes	12-29
Undo Segments	12-30
Undo Segments and Transactions	12-31
Transaction Rollback	12-33
Temporary Undo Segments	12-33
Segment Space and the High Water Mark	12-34
Overview of Tablespaces	12-37
Permanent Tablespaces	12-37
The SYSTEM Tablespace	12-38
The SYSAUX Tablespace	12-39
Undo Tablespaces	12-39
Shadow Tablespaces	12-41
Temporary Tablespaces	12-44
Shared and Local Temporary Tablespaces	12-45
Default Temporary Tablespaces	12-46
Tablespace Modes	12-48
Read/Write and Read-Only Tablespaces	12-48
Online and Offline Tablespaces	12-49

Part V Oracle Instance Architecture

13 Oracle Database Instance

Introduction to the Oracle Database Instance	13-1
Database Instance Structure	13-1
Database Instance Configurations	13-2
Read/Write and Read-Only Instances	13-3
Duration of a Database Instance	13-4
Identification of a Database Instance	13-5
Oracle Base Directory	13-6
Oracle Home Directory	13-6
Oracle System Identifier (SID)	13-7
Overview of Database Instance Startup and Shutdown	13-7
Overview of Instance and Database Startup	13-8
Connection with Administrator Privileges	13-9
How an Instance Is Started	13-10
How a Database Is Mounted	13-10
How a Database Is Opened	13-11
Overview of Database and Instance Shutdown	13-13
Shutdown Modes	13-14
How a Database Is Closed	13-15
How a Database Is Unmounted	13-15
How an Instance Is Shut Down	13-16
Overview of Checkpoints	13-16
Purpose of Checkpoints	13-17
When Oracle Database Initiates Checkpoints	13-17
Overview of Instance Recovery	13-18
Purpose of Instance Recovery	13-18
When Oracle Database Performs Instance Recovery	13-19
Importance of Checkpoints for Instance Recovery	13-19
Instance Recovery Phases	13-20
Overview of Parameter Files	13-21
Initialization Parameters	13-21
Functional Groups of Initialization Parameters	13-22
Basic and Advanced Initialization Parameters	13-22
Server Parameter Files	13-22
Text Initialization Parameter Files	13-23
Modification of Initialization Parameter Values	13-24

Overview of Diagnostic Files	13-26
Automatic Diagnostic Repository	13-26
Problems and Incidents	13-26
ADR Structure	13-27
Alert Log	13-28
DDL Log	13-29
Trace Files	13-29
Types of Trace Files	13-30
Locations of Trace Files	13-30
Segmentation of Trace Files	13-31
Diagnostic Dumps	13-31
Trace Dumps and Incidents	13-31

14 Memory Architecture

Introduction to Oracle Database Memory Structures	14-1
Basic Memory Structures	14-1
Oracle Database Memory Management	14-3
Overview of the User Global Area	14-4
Overview of the Program Global Area (PGA)	14-5
Contents of the PGA	14-6
Private SQL Area	14-6
SQL Work Areas	14-7
PGA Usage in Dedicated and Shared Server Modes	14-8
Overview of the System Global Area (SGA)	14-9
Database Buffer Cache	14-10
Purpose of the Database Buffer Cache	14-10
Buffer States	14-11
Buffer Modes	14-11
Buffer I/O	14-12
Buffer Pools	14-16
Buffers and Full Table Scans	14-18
Redo Log Buffer	14-20
Shared Pool	14-21
Library Cache	14-22
Data Dictionary Cache	14-25
Server Result Cache	14-26
Reserved Pool	14-28
Large Pool	14-28
Java Pool	14-29
Fixed SGA	14-30

Optional Performance-Related SGA Subareas	14-30
In-Memory Area	14-30
Memoptimize Pool	14-31
Overview of Software Code Areas	14-32

15 Process Architecture

Introduction to Processes	15-1
Types of Processes	15-1
Multiprocess and Multithreaded Oracle Database Systems	15-3
Overview of Client Processes	15-5
Client and Server Processes	15-5
Connections and Sessions	15-6
Database Operations	15-8
Overview of Server Processes	15-8
Dedicated Server Processes	15-9
Shared Server Processes	15-9
How Oracle Database Creates Server Processes	15-10
Overview of Background Processes	15-11
Mandatory Background Processes	15-11
Process Monitor Process (PMON) Group	15-12
Process Manager (PMAN)	15-13
Listener Registration Process (LREG)	15-14
System Monitor Process (SMON)	15-14
Database Writer Process (DBW)	15-14
Log Writer Process (LGWR)	15-15
Checkpoint Process (CKPT)	15-17
Manageability Monitor Processes (MMON and MMNL)	15-18
Recoverer Process (RECO)	15-18
Optional Background Processes	15-18
Archiver Processes (ARCn)	15-19
Job Queue Processes (CJQ0 and Jnnn)	15-19
Flashback Data Archive Process (FBDA)	15-20
Space Management Coordinator Process (SMCO)	15-20
Slave Processes	15-21
I/O Slave Processes	15-21
Parallel Execution (PX) Server Processes	15-22

16 Application and Oracle Net Services Architecture

Overview of Oracle Application Architecture	16-1
---	------

Overview of Client/Server Architecture	16-1
Distributed Processing	16-1
Advantages of a Client/Server Architecture	16-3
Overview of Multitier Architecture	16-4
Clients	16-4
Application Servers	16-4
Database Servers	16-5
Service-Oriented Architecture (SOA)	16-5
Overview of Grid Architecture	16-6
Overview of Oracle Net Services Architecture	16-6
How Oracle Net Services Works	16-7
The Oracle Net Listener	16-7
Service Names	16-9
Service Registration	16-10
Dedicated Server Architecture	16-11
Shared Server Architecture	16-13
Dispatcher Request and Response Queues	16-14
Restricted Operations of the Shared Server	16-16
Database Resident Connection Pooling	16-17
Overview of the Program Interface	16-18
Program Interface Structure	16-19
Program Interface Drivers	16-19
Communications Software for the Operating System	16-20

Part VI Oracle Database Administration and Application Development

17 Topics for Database Administrators and Developers

Overview of Database Security	17-1
User Accounts	17-1
Privileges	17-2
Roles	17-2
Privilege Analysis	17-3
User Profiles	17-4
Database Authentication	17-4
Encryption	17-5
Network Encryption	17-5
Transparent Data Encryption	17-5
Oracle Data Redaction	17-6
Orientation	17-7
Oracle Database Vault	17-7

Virtual Private Database (VPD)	17-8
Oracle Label Security (OLS)	17-8
Data Access Monitoring	17-9
Database Auditing	17-9
Unified Audit Trail	17-11
Enterprise Manager Auditing Support	17-12
Oracle Audit Vault and Database Firewall	17-12
Overview of High Availability	17-13
High Availability and Unplanned Downtime	17-13
Site Failures	17-13
Computer Failures	17-14
Storage Failures	17-15
Data Corruption	17-16
Human Errors	17-18
High Availability and Planned Downtime	17-18
System and Database Changes	17-19
Data Changes	17-20
Application Changes	17-20
Overview of Grid Computing	17-21
Database Server Grid	17-22
Scalability	17-23
Fault Tolerance	17-23
Services	17-24
Oracle Flex Clusters	17-24
Database Storage Grid	17-25
Overview of Data Warehousing and Business Intelligence	17-25
Data Warehousing and OLTP	17-25
Data Warehouse Architecture	17-27
Data Warehouse Architecture (Basic)	17-27
Data Warehouse Architecture (with a Staging Area)	17-28
Data Warehouse Architecture (with a Staging Area and Data Marts)	17-28
Overview of Extraction, Transformation, and Loading (ETL)	17-29
Business Intelligence	17-30
Analytic SQL	17-30
Analytic Views	17-31
OLAP	17-32
Oracle Advanced Analytics	17-32
Overview of Oracle Information Integration	17-33
Federated Access	17-34
Distributed SQL	17-34
Database Links	17-34

Information Sharing	17-35
Oracle GoldenGate	17-35
Oracle Database Advanced Queuing (AQ)	17-36

18 Concepts for Database Administrators

Duties of Database Administrators	18-1
Tools for Database Administrators	18-2
Oracle Enterprise Manager	18-2
Oracle Enterprise Manager Cloud Control	18-2
Oracle Enterprise Manager Database Express 12c	18-3
SQL*Plus	18-3
Tools for Database Installation and Configuration	18-4
Tools for Oracle Net Configuration and Administration	18-4
Tools for Data Movement and Analysis	18-5
SQL*Loader	18-6
Oracle Data Pump Export and Import	18-8
Oracle LogMiner	18-9
ADR Command Interpreter (ADRCI)	18-10
Topics for Database Administrators	18-10
Backup and Recovery	18-10
Backup and Recovery Techniques	18-11
Recovery Manager Architecture	18-12
Database Backups	18-13
Data Repair	18-15
Zero Data Loss Recovery Appliance	18-19
Memory Management	18-22
Automatic Memory Management	18-22
Shared Memory Management of the SGA	18-23
Memory Management of the Instance PGA	18-24
Summary of Memory Management Methods	18-25
Resource Management and Task Scheduling	18-27
Database Resource Manager	18-27
Oracle Scheduler	18-29
Performance and Tuning	18-30
Database Self-Monitoring	18-31
Automatic Workload Repository (AWR)	18-31
Automatic Database Monitor (ADDM)	18-33
Active Session History (ASH)	18-33
Application and SQL Tuning	18-34

19 Concepts for Database Developers

Duties of Database Developers	19-1
Tools for Database Developers	19-2
SQL Developer	19-2
Oracle Application Express	19-2
Oracle JDeveloper	19-3
Oracle Developer Tools for Visual Studio .NET	19-3
Topics for Database Developers	19-4
Principles of Application Design and Tuning	19-4
Client-Side Database Programming	19-5
Embedded SQL	19-6
Client-Side APIs	19-8
Globalization Support	19-10
Globalization Support Environment	19-11
Oracle Globalization Development Kit	19-13
Unstructured Data	19-14
Overview of XML in Oracle Database	19-14
Overview of JSON in Oracle Database	19-15
Overview of LOBs	19-18
Overview of Oracle Text	19-19
Overview of Oracle Multimedia	19-20
Overview of Oracle Spatial and Graph	19-21

Glossary

Index

5

Data Integrity

This chapter explains how integrity constraints enforce the business rules associated with a database and prevent the entry of invalid information into tables.

This chapter contains the following sections:

- [Introduction to Data Integrity](#)
- [Types of Integrity Constraints](#)
- [States of Integrity Constraints](#)



See Also:

"[Overview of Tables](#)" for background on columns and the need for integrity constraints

Introduction to Data Integrity

It is important that data maintain **data integrity**, which is adherence to business rules determined by the database administrator or application developer.

Business rules specify conditions and relationships that must always be true or must always be false. For example, each company defines its own policies about salaries, employee numbers, inventory tracking, and so on.

Techniques for Guaranteeing Data Integrity

When designing a database application, developers have several options for guaranteeing the integrity of data stored in the database.

These options include:

- Enforcing business rules with triggered stored database procedures
- Using stored procedures to completely control access to data
- Enforcing business rules in the code of a database application
- Using Oracle Database integrity constraints, which are rules defined at the column or object level that restrict values in the database

**See Also:**

- ["Overview of Triggers"](#) explains the purpose and types of triggers
- ["Introduction to Server-Side Programming"](#) explains the purpose and characteristics of stored procedures

Advantages of Integrity Constraints

An integrity constraint is a schema object that is created and dropped using SQL. To enforce data integrity, use integrity constraints whenever possible.

Advantages of integrity constraints over alternatives for enforcing data integrity include:

- Declarative ease

Because you define integrity constraints using SQL statements, no additional programming is required when you define or alter a table. The SQL statements are easy to write and eliminate programming errors.

- Centralized rules

Integrity constraints are defined for tables and are stored in the [data dictionary](#). Thus, data entered by all applications must adhere to the same integrity constraints. If the rules change at the table level, then applications need not change. Also, applications can use metadata in the data dictionary to immediately inform users of violations, even before the database checks the SQL statement.

- Flexibility when loading data

You can disable integrity constraints temporarily to avoid performance overhead when loading large amounts of data. When the data load is complete, you can re-enable the integrity constraints.

**See Also:**

- ["Overview of the Data Dictionary"](#)
- *Oracle Database 2 Day Developer's Guide* and *Oracle Database Development Guide* to learn how to maintain data integrity
- *Oracle Database Administrator's Guide* to learn how to manage integrity constraints

Types of Integrity Constraints

Oracle Database enables you to apply constraints both at the table and column level.

A constraint specified as part of the definition of a column or attribute is an inline specification. A constraint specified as part of the table definition is an out-of-line specification.

A **key** is the column or set of columns included in the definition of certain types of integrity constraints. Keys describe the relationships between the tables and columns of a relational database. Individual values in a key are called **key values**.

The following table describes the types of constraints. Each can be specified either inline or out-of-line, except for NOT NULL, which must be inline.

Table 5-1 Types of Integrity Constraints

Constraint Type	Description	See Also
NOT NULL	Allows or disallows inserts or updates of rows containing a null in a specified column.	" NOT NULL Integrity Constraints "
Unique key	Prohibits multiple rows from having the same value in the same column or combination of columns but allows some values to be null.	" Unique Constraints "
Primary key	Combines a NOT NULL constraint and a unique constraint. It prohibits multiple rows from having the same value in the same column or combination of columns and prohibits values from being null.	" Primary Key Constraints "
Foreign key	Designates a column as the foreign key and establishes a relationship between the foreign key and a primary or unique key, called the referenced key .	" Foreign Key Constraints "
Check	Requires a database value to obey a specified condition.	" Check Constraints "
REF	Dictates types of data manipulation allowed on values in a REF column and how these actions affect dependent values. In an object-relational database, a built-in data type called a REF encapsulates a reference to a row object of a specified object type. Referential integrity constraints on REF columns ensure that there is a row object for the REF.	<i>Oracle Database Object-Relational Developer's Guide</i> to learn about REF constraints

 See Also:

- "[Overview of Tables](#)"
- *Oracle Database SQL Language Reference* to learn more about the types of constraints

NOT NULL Integrity Constraints

A NOT NULL constraint requires that a column of a table contain no null values. A **null** is the absence of a value. By default, all columns in a table allow nulls.

NOT NULL constraints are intended for columns that must not lack values. For example, the `hr.employees` table requires a value in the `email` column. An attempt to insert an employee row without an email address generates an error:

```
SQL> INSERT INTO hr.employees (employee_id, last_name) values (999, 'Smith');
.
.
.
ERROR at line 1:
ORA-01400: cannot insert NULL into ("HR"."EMPLOYEES"."EMAIL")
```

You can only add a column with a NOT NULL constraint if the table does not contain any rows or if you specify a default value.

See Also:

- *Oracle Database 2 Day Developer's Guide* for examples of adding NOT NULL constraints to a table
- *Oracle Database SQL Language Reference* for restrictions on using NOT NULL constraints
- *Oracle Database Development Guide* to learn when to use the NOT NULL constraint

Unique Constraints

A **unique key constraint** requires that every value in a column or set of columns be unique. No rows of a table may have duplicate values in a single column (the **unique key**) or set of columns (the **composite unique key**) with a unique key constraint.

Note:

The term *key* refers only to the columns defined in the integrity constraint. Because the database enforces a unique constraint by implicitly creating or reusing an [index](#) on the key columns, the term *unique key* is sometimes incorrectly used as a synonym for *unique key constraint* or *unique index*.

Unique key constraints are appropriate for any column where duplicate values are not allowed. Unique constraints differ from primary key constraints, whose purpose is to identify each table row uniquely, and typically contain values that have no significance other than being unique. Examples of unique keys include:

- A customer phone number, where the primary key is the customer number
- A department name, where the primary key is the department number

As shown in [Example 2-1](#), a unique key constraint exists on the `email` column of the `hr.employees` table. The relevant part of the statement is as follows:

```
CREATE TABLE employees ( ...
, email          VARCHAR2(25)
```

```

        CONSTRAINT emp_email_nn NOT NULL ...
, CONSTRAINT emp_email_uk UNIQUE (email) ... );

```

The emp_email_uk constraint ensures that no two employees have the same email address, as shown in the following example:

```
SQL> SELECT employee_id, last_name, email FROM employees WHERE email = 'PFAY';
```

EMPLOYEE_ID	LAST_NAME	EMAIL
202	Fay	PFAY

```
SQL> INSERT INTO employees (employee_id, last_name, email, hire_date, job_id)
1 VALUES (999, 'Fay', 'PFAY', SYSDATE, 'ST_CLERK');
```

```

.
.
.

```

ERROR at line 1:

ORA-00001: unique constraint (HR.EMP_EMAIL_UK) violated

Unless a NOT NULL constraint is also defined, a null always satisfies a unique key constraint. Thus, columns with both unique key constraints and NOT NULL constraints are typical. This combination forces the user to enter values in the unique key and eliminates the possibility that new row data conflicts with existing row data.

Note:

Because of the search mechanism for unique key constraints on multiple columns, you cannot have identical values in the non-null columns of a partially null composite unique key constraint.

Example 5-1 Unique Constraint

```
SQL> SELECT employee_id, last_name, email FROM employees WHERE email = 'PFAY';
```

EMPLOYEE_ID	LAST_NAME	EMAIL
202	Fay	PFAY

```
SQL> INSERT INTO employees (employee_id, last_name, email, hire_date, job_id)
1 VALUES (999, 'Fay', 'PFAY', SYSDATE, 'ST_CLERK');
```

```

.
.
.

```

ERROR at line 1:

ORA-00001: unique constraint (HR.EMP_EMAIL_UK) violated

See Also:

- ["Unique and Nonunique Indexes"](#)
- *Oracle Database 2 Day Developer's Guide* for examples of adding UNIQUE constraints to a table

Primary Key Constraints

In a **primary key constraint**, the values in the group of one or more columns subject to the constraint uniquely identify the row. Each table can have one **primary key**, which in effect names the row and ensures that no duplicate rows exist.

A primary key can be natural or a surrogate. A [natural key](#) is a meaningful identifier made of existing attributes in a table. For example, a natural key could be a postal code in a lookup table. In contrast, a [surrogate key](#) is a system-generated incrementing identifier that ensures uniqueness within a table. Typically, a [sequence](#) generates surrogate keys.

The Oracle Database implementation of the primary key constraint guarantees that the following statements are true:

- No two rows have duplicate values in the specified column or set of columns.
- The primary key columns do not allow nulls.

A typical situation calling for a primary key is the numeric identifier for an employee. Each employee must have a unique ID. An employee must be described by one and only one row in the employees table.

The example in [Unique Constraints](#) indicates that an existing employee has the employee ID of 202, where the employee ID is the primary key. The following example shows an attempt to add an employee with the same employee ID and an employee with no ID:

```
SQL> INSERT INTO employees (employee_id, last_name, email, hire_date, job_id)
1 VALUES (202, 'Chan', 'JCHAN', SYSDATE, 'ST_CLERK');
```

```
.
.
.
```

ERROR at line 1:

ORA-00001: unique constraint (HR.EMP_EMP_ID_PK) violated

```
SQL> INSERT INTO employees (last_name) VALUES ('Chan');
```

```
.
.
.
```

ERROR at line 1:

ORA-01400: cannot insert NULL into ("HR"."EMPLOYEES"."EMPLOYEE_ID")

The database enforces primary key constraints with an [index](#). Usually, a primary key constraint created for a column implicitly creates a unique index and a NOT NULL constraint. Note the following exceptions to this rule:

- In some cases, as when you create a primary key with a [deferrable constraint](#), the generated index is not unique.

 Note:

You can explicitly create a unique index with the CREATE UNIQUE INDEX statement.

- If a usable index exists when a primary key constraint is created, then the constraint reuses this index and does not implicitly create one.

By default the name of the implicitly created index is the name of the primary key constraint. You can also specify a user-defined name for an index. You can specify storage options for the index by including the `ENABLE` clause in the `CREATE TABLE` or `ALTER TABLE` statement used to create the constraint.



See Also:

Oracle Database 2 Day Developer's Guide and *Oracle Database Development Guide* to learn how to add primary key constraints to a table

Foreign Key Constraints

Whenever two tables contain one or more common columns, Oracle Database can enforce the relationship between the two tables through a **foreign key constraint**, also called a *referential integrity constraint*.

A foreign key constraint requires that for each value in the column on which the constraint is defined, the value in the other specified other table and column must match. An example of a referential integrity rule is an employee can work for only an existing department.

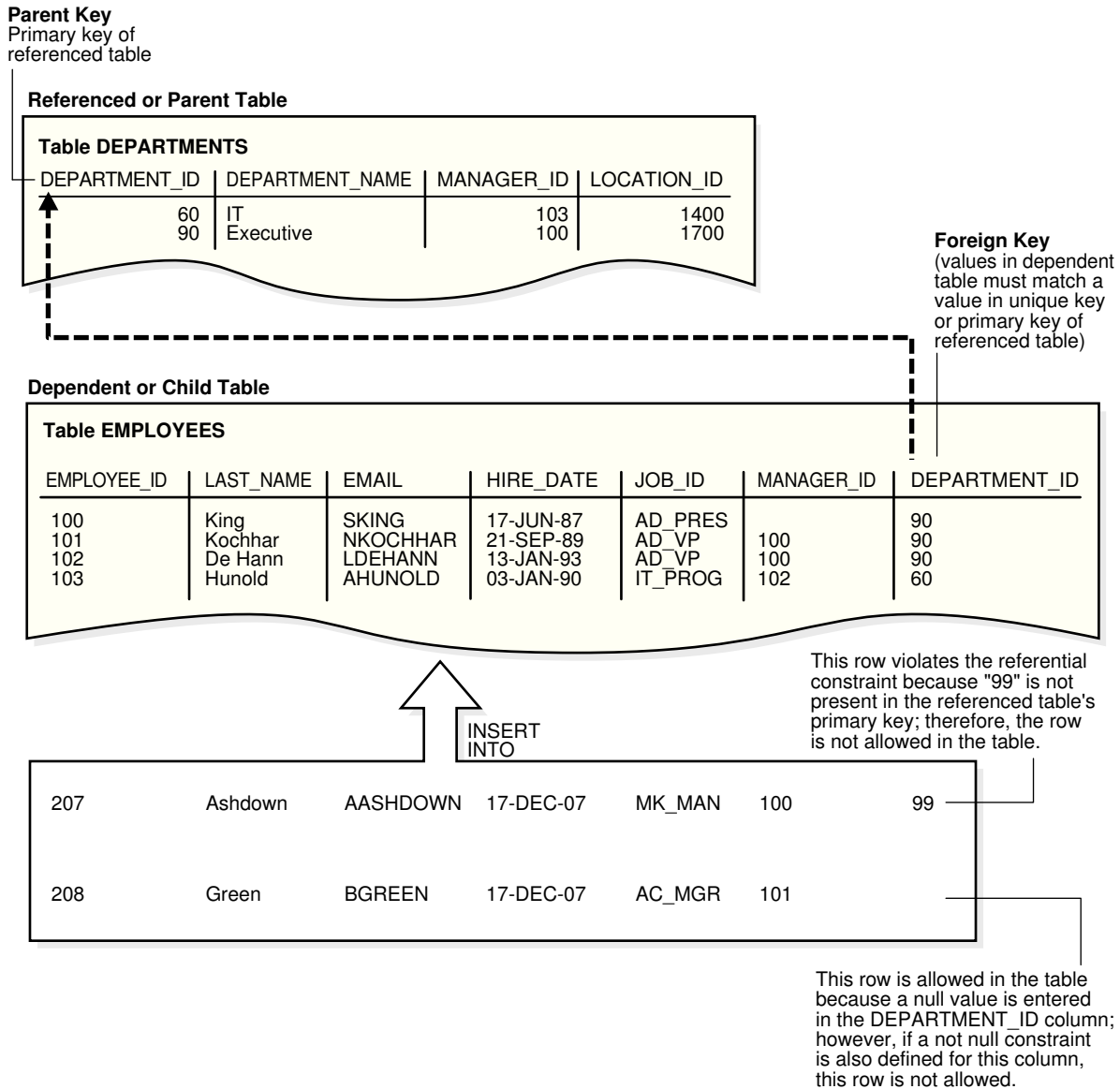
The following table lists terms associated with referential integrity constraints.

Table 5-2 Referential Integrity Constraint Terms

Term	Definition
Foreign key	The column or set of columns included in the definition of the constraint that reference a referenced key. For example, the <code>department_id</code> column in <code>employees</code> is a foreign key that references the <code>department_id</code> column in <code>departments</code>. Foreign keys may be defined as multiple columns. However, a composite foreign key must reference a composite primary or unique key with the same number of columns and the same data types. The value of foreign keys can match either the referenced primary or unique key value, or be null. If any column of a composite foreign key is null, then the non-null portions of the key do not have to match any corresponding portion of a parent key.
Referenced key	The unique key or primary key of the table referenced by a foreign key. For example, the <code>department_id</code> column in <code>departments</code> is the referenced key for the <code>department_id</code> column in <code>employees</code>.
Dependent or child table	The table that includes the foreign key. This table depends on the values present in the referenced unique or primary key. For example, the <code>employees</code> table is a child of <code>departments</code>.
Referenced or parent table	The table that is referenced by the foreign key of the child table. It is this table's referenced key that determines whether specific inserts or updates are allowed in the child table. For example, the <code>departments</code> table is a parent of <code>employees</code>.

Figure 5-1 shows a foreign key on the `employees.department_id` column. It guarantees that every value in this column must match a value in the `departments.department_id` column. Thus, no erroneous department numbers can exist in the `employees.department_id` column.

Figure 5-1 Referential Integrity Constraints



 See Also:

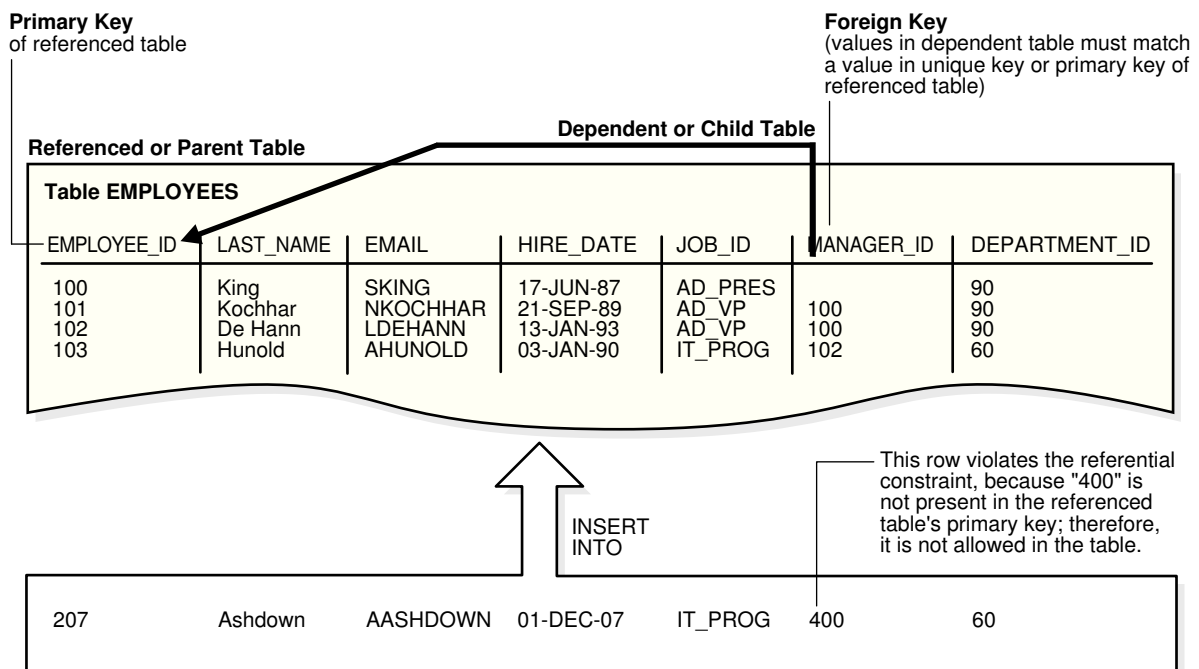
Oracle Database 2 Day Developer's Guide and *Oracle Database Development Guide* to learn how to add foreign key constraints to a table

Self-Referential Integrity Constraints

A **self-referential integrity constraint** is a foreign key that references a parent key in the same table.

In the following figure, a self-referential constraint ensures that every value in the `employees.manager_id` column corresponds to an existing value in the `employees.employee_id` column. For example, the manager for employee 102 must exist in the `employees` table. This constraint eliminates the possibility of erroneous employee numbers in the `manager_id` column.

Figure 5-2 Single Table Referential Constraints



Nulls and Foreign Keys

The relational model permits the value of foreign keys to match either the referenced primary or unique key value, or be null. For example, a row in `hr.employees` might not specify a department ID.

If any column of a composite foreign key is null, then the non-null portions of the key do not have to match any corresponding portion of a parent key. For example, a `reservations` table might contain a composite foreign key on the `table_id` and `date` columns, but `table_id` is null.

Parent Key Modifications and Foreign Keys

The relationship between foreign key and parent key has implications for deletion of parent keys. For example, if a user attempts to delete the record for this department, then what happens to the records for employees in this department?

When a parent key is modified, referential integrity constraints can specify the following actions to be performed on dependent rows in a child table:

- **No action on deletion or update**
In the normal case, users cannot modify referenced key values if the results would violate referential integrity. For example, if `employees.department_id` is a foreign key to `departments`, and if employees belong to a particular department, then an attempt to delete the row for this department violates the constraint.
- **Cascading deletions**
A deletion cascades (`DELETE CASCADE`) when rows containing referenced key values are deleted, causing all rows in child tables with dependent foreign key values to also be deleted. For example, the deletion of a row in `departments` causes rows for all employees in this department to be deleted.
- **Deletions that set null**
A deletion sets null (`DELETE SET NULL`) when rows containing referenced key values are deleted, causing all rows in child tables with dependent foreign key values to set those values to null. For example, the deletion of a department row sets the `department_id` column value to null for employees in this department.

[Table 5-3](#) outlines the DML statements allowed by the different referential actions on the key values in the parent table, and the foreign key values in the child table.

Table 5-3 DML Statements Allowed by Update and Delete No Action

DML Statement	Issued Against Parent Table	Issued Against Child Table
INSERT	Always OK if the parent key value is unique	OK only if the foreign key value exists in the parent key or is partially or all null
UPDATE NO ACTION	Allowed if the statement does not leave any rows in the child table without a referenced parent key value	Allowed if the new foreign key value still references a referenced key value
DELETE NO ACTION	Allowed if no rows in the child table reference the parent key value	Always OK
DELETE CASCADE	Always OK	Always OK
DELETE SET NULL	Always OK	Always OK



Note:

Other referential actions not supported by `FOREIGN KEY` integrity constraints of Oracle Database can be enforced using database triggers. See "[Overview of Triggers](#)".

**See Also:**

Oracle Database SQL Language Reference to learn about the `ON DELETE` clause

Indexes and Foreign Keys

As a rule, foreign keys should be indexed. The only exception is when the matching unique or primary key is never updated or deleted.

Indexing the foreign keys in child tables provides the following benefits:

- Prevents a full table lock on the child table. Instead, the database acquires a row lock on the index.
- Removes the need for a [full table scan](#) of the child table. As an illustration, assume that a user removes the record for department 10 from the departments table. If `employees.department_id` is not indexed, then the database must scan `employees` to see if any employees exist in department 10.

**See Also:**

- ["Locks and Foreign Keys"](#) explains the locking behavior for indexed and unindexed foreign key columns
- ["Introduction to Indexes"](#) explains the purpose and characteristics of indexes

Check Constraints

A **check constraint** on a column or set of columns requires that a specified **condition** be true or unknown for every row.

If DML results in the condition of the constraint evaluating to false, then the SQL statement is rolled back. The chief benefit of check constraints is the ability to enforce very specific integrity rules. For example, you could use check constraints to enforce the following rules in the `hr.employees` table:

- The salary column must not have a value greater than 10000.
- The commission column must have a value that is not greater than the salary.

The following example creates a maximum salary constraint on `employees` and demonstrates what happens when a statement attempts to insert a row containing a salary that exceeds the maximum:

```
SQL> ALTER TABLE employees ADD CONSTRAINT max_emp_sal CHECK (salary < 10001);
SQL> INSERT INTO employees (employee_id, last_name, email, hire_date, job_id, salary)
1  VALUES (999, 'Green', 'BGREEN', SYSDATE, 'ST_CLERK', 20000);
.
.
.
```

```
ERROR at line 1:  
ORA-02290: check constraint (HR.MAX_EMP_SAL) violated
```

A single column can have multiple check constraints that reference the column in its definition. For example, the `salary` column could have one constraint that prevents values over 10000 and a separate constraint that prevents values less than 500.

If multiple check constraints exist for a column, then they must be designed so their purposes do not conflict. No order of evaluation of the conditions can be assumed. The database does not verify that check conditions are not mutually exclusive.



See Also:

Oracle Database SQL Language Reference to learn about restrictions for check constraints

States of Integrity Constraints

As part of constraint definition, you can specify how and when Oracle Database should enforce the constraint, thereby determining the constraint state.

Checks for Modified and Existing Data

The database enables you to specify whether a constraint applies to existing data or future data. If a constraint is enabled, then the database checks new data as it is entered or updated. Data that does not conform to the constraint cannot enter the database.

For example, enabling a `NOT NULL` constraint on `employees.department_id` guarantees that every future row has a department ID. If a constraint is disabled, then the table can contain rows that violate the constraint.

You can set constraints to either of the following validation modes:

- `VALIDATE`

Existing data must conform to the constraint. For example, enabling a `NOT NULL` constraint on `employees.department_id` and setting it to `VALIDATE` checks that every existing row has a department ID.

- `NOVALIDATE`

Existing data need not conform to the constraint. In effect, this is a “trust me” mode. For example, if you are certain that every sale that you loaded into a table has a date, then you can create a `NOT NULL` constraint on the date column and set the constraint to `NOVALIDATE`. Unenforced constraints are typically useful only with materialized views and query rewrite.

For a constraint in `NOVALIDATE` mode, the `RELY` parameter indicates that the optimizer can use the constraint to determine join information. Even though the constraint is not used for validating data, it enables more sophisticated query rewrites for materialized views, and enables data warehousing tools to retrieve constraint information from the data dictionary. The default is `NORELY`, which means that the optimizer is effectively unaware of the constraint.

The behavior of `VALIDATE` and `NOVALIDATE` always depends on whether the constraint is enabled or disabled. The following table summarizes the relationships.

Table 5-4 Checks on Modified and Existing Data

Modified Data	Existing Data	Summary
ENABLE	VALIDATE	Existing and future data must obey the constraint. An attempt to apply a new constraint to a populated table results in an error if existing rows violate the constraint.
ENABLE	NOVALIDATE	The database checks the constraint, but it need not be true for all rows. Therefore, existing rows can violate the constraint, but new or modified rows must conform to the rules. This mode is often used in data warehouses that contain existing data whose integrity has already been verified.
DISABLE	VALIDATE	The database disables the constraint, drops its index, and prevents modification of the constrained columns.
DISABLE	NOVALIDATE	The constraint is not checked and is not necessarily true.



See Also:

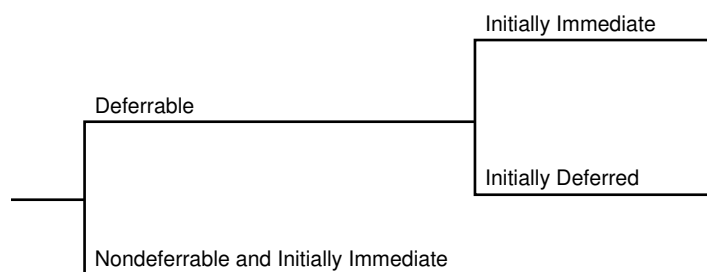
Oracle Database SQL Language Reference to learn about constraint states

When the Database Checks Constraints for Validity

Every constraint is either in a not deferrable (default) or deferrable state. This state determines when Oracle Database checks the constraint for validity.

The following graphic shows the options for deferrable constraints.

Figure 5-3 Options for Deferrable Constraints



Nondeferrable Constraints

In a **nondeferrable constraint**, Oracle Database never defers the validity check of the constraint to the end of the transaction. Instead, the database checks the constraint at the end of each statement. If the constraint is violated, then the statement rolls back.

For example, a nondeferrable `NOT NULL` constraint exists for the `employees.last_name` column. If a session attempts to insert a row with no last name, then the database immediately rolls back the statement because the `NOT NULL` constraint is violated. No row is inserted.

Deferrable Constraints

A **deferrable constraint** permits a transaction to use the `SET CONSTRAINT` clause to defer checking of this constraint until a `COMMIT` statement is issued. If you make changes to the database that might violate the constraint, then this setting effectively enables you to disable the constraint until all changes are complete.

You can set the default behavior for when the database checks the deferrable constraint. You can specify either of the following attributes:

- `INITIALLY IMMEDIATE`

The database checks the constraint immediately after each statement executes. If the constraint is violated, then the database rolls back the statement.

- `INITIALLY DEFERRED`

The database checks the constraint when a `COMMIT` is issued. If the constraint is violated, then the database rolls back the transaction.

Assume that a deferrable `NOT NULL` constraint on `employees.last_name` is set to `INITIALLY DEFERRED`. A user creates a transaction with 100 `INSERT` statements, some of which have null values for `last_name`. When the user attempts to commit, the database rolls back all 100 statements. However, if this constraint were set to `INITIALLY IMMEDIATE`, then the database would not roll back the transaction.

If a constraint causes an action, then the database considers this action as part of the statement that caused it, whether the constraint is deferred or immediate. For example, deleting a row in `departments` causes the deletion of all rows in `employees` that reference the deleted department row. In this case, the deletion from `employees` is considered part of the `DELETE` statement executed against `departments`.



See Also:

Oracle Database SQL Language Reference for information about constraint attributes and their default values

Examples of Constraint Checking

The following examples help illustrate when Oracle Database performs the checking of constraints.

Assume the following:

- The `employees` table has the structure shown in "[Self-Referential Integrity Constraints](#)".
- The self-referential constraint makes entries in the `manager_id` column dependent on the values of the `employee_id` column.

Example: Insertion of a Value in a Foreign Key Column When No Parent Key Value Exists

This example concerns the insertion of the first row into the `employees` table. No rows currently exist, so how can a row be entered if the value in the `manager_id` column cannot reference an existing value in the `employee_id` column?

Some possibilities are:

- If the `manager_id` column does not have a `NOT NULL` constraint defined on it, then you can enter a null for the `manager_id` column of the first row.

Because nulls are allowed in foreign keys, Oracle Database inserts this row into the table.

- You can enter the same value in the `employee_id` and `manager_id` columns, specifying that the employee is his or her own manager.

This case reveals that Oracle Database performs its constraint checking *after* the statement executes. To allow a row to be entered with the same values in the parent key and the foreign key, the database must first insert the new row, and then determine whether any row in the table has an `employee_id` that corresponds to the `manager_id` of the new row.

- A multiple row `INSERT` statement, such as an `INSERT` statement with nested `SELECT` statements, can insert rows that reference one another.

For example, the first row might have 200 for employee ID and 300 for manager ID, while the second row has 300 for employee ID and 200 for manager. Constraint checking is deferred until the complete execution of the `INSERT` statement. The database inserts all rows, and then checks all rows for constraint violations.

Default values are included as part of an `INSERT` statement before the statement is parsed. Thus, default column values are subject to all integrity constraint checking.

Example: Update of All Foreign Key and Parent Key Values

In this example, a self-referential constraint makes entries in the `manager_id` column of `employees` dependent on the values of the `employee_id` column.

The company has been sold. Because of this sale, all employee numbers must be updated to be the current value plus 5000 to coordinate with the employee numbers of the new company. As shown in the following graphic, some employees are also managers:

Figure 5-4 The employees Table Before Updates

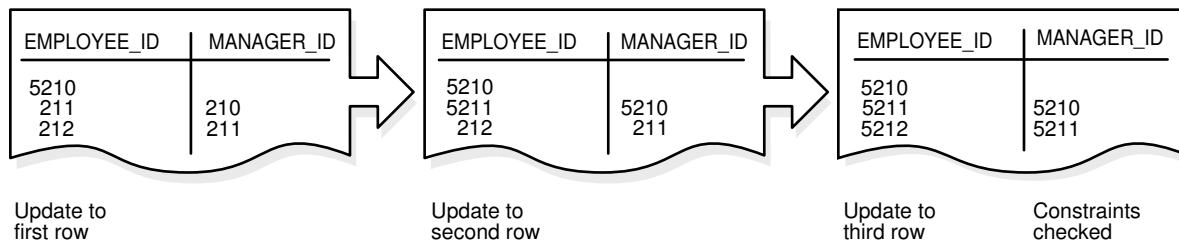
EMPLOYEE_ID	MANAGER_ID
210	
211	210
212	211

Because manager numbers are also employee numbers, the manager numbers must also increase by 5000. You could execute the following SQL statement to update the values:

```
UPDATE employees SET employee_id = employee_id + 5000,  
manager_id = manager_id + 5000;
```

Although a constraint is defined to verify that each `manager_id` value matches an `employee_id` value, the preceding statement is valid because the database effectively checks constraints after the statement completes. Figure 5-5 shows that the database performs the actions of the entire SQL statement before checking constraints.

Figure 5-5 Constraint Checking



The examples in this section illustrate the constraint checking mechanism during INSERT and UPDATE statements, but the database uses the same mechanism for all types of DML statements. The database uses the same mechanism for all types of constraints, not just self-referential constraints.



Note:

Operations on a [view](#) or [synonym](#) are subject to the integrity constraints defined on the base tables.