Oracle Tables for DBAs and Developers

Daniel A. Morgan
email: dmorgan@forsythe.com
mobile: +1 206-669-2949
skype: damorgan11g

Wednesday: 09 August, 2017

Unsafe Harbor

- This room is an unsafe harbor
- You can rely on the information in this presentation to help you protect your data, your databases, your organization, and your career
- No one from Oracle has previewed this presentation
- No one from Oracle knows what I'm going to say
- No one from Oracle has supplied any of my materials
- Everything I will present is existing, proven, functionality

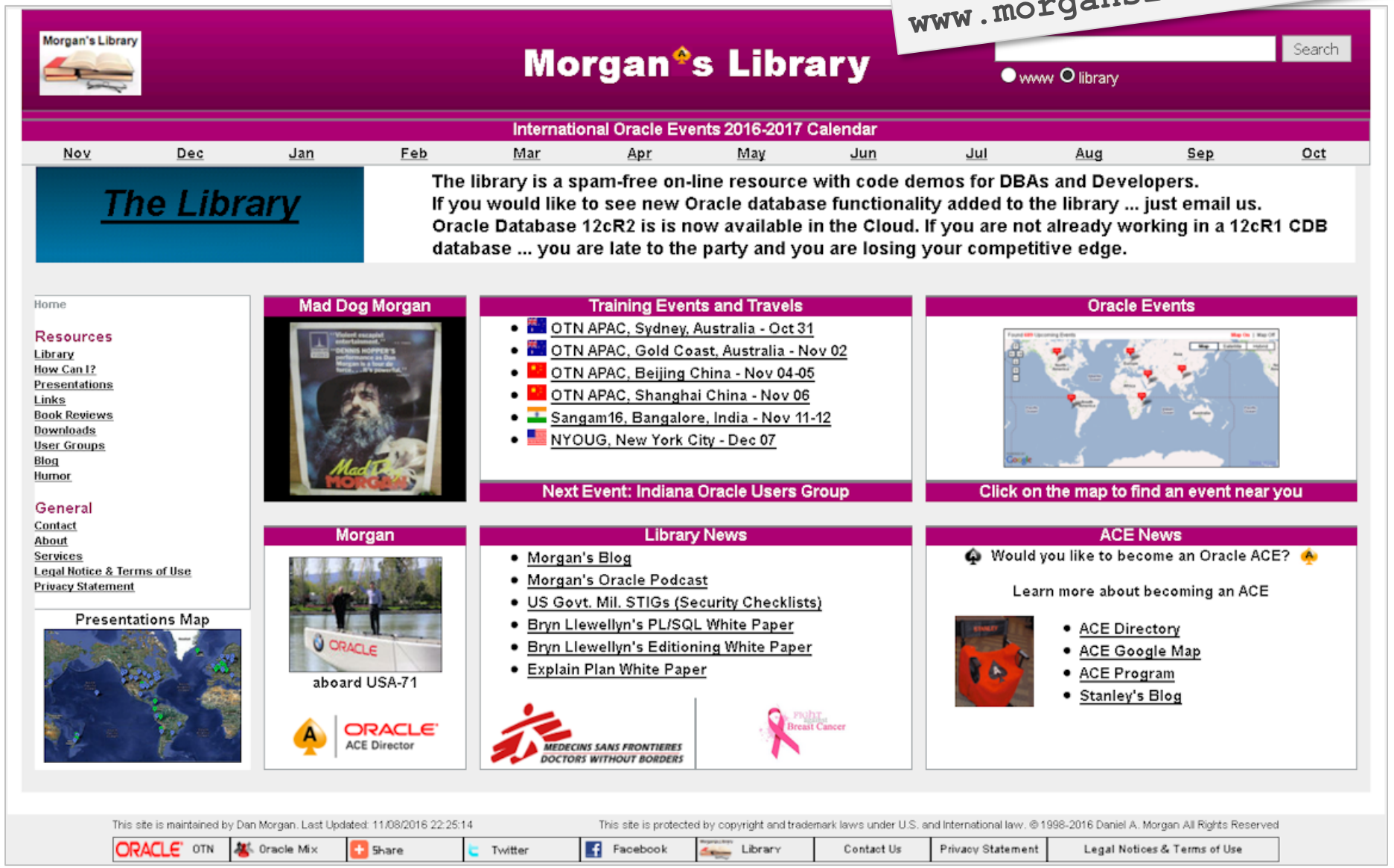
Introduction

Daniel Morgan

- Oracle ACE Director Alumni
- Oracle Educator
  - Curriculum author and primary program instructor at University of Washington
  - Consultant: Harvard University
  - University Guest Lecturers
    - APAC: University of Canterbury (NZ)
    - EMEA: University of Oslo (Norway)
    - Latin America: Universidad Cenfotec, Universidad Latina de Panama, Technologico de Costa Rica
- IT Professional
  - First computer: IBM 360/40 in 1969: Fortran IV
  - Oracle Database since 1988-9 and Oracle Beta tester
  - The Morgan behind www.morganslibrary.org
  - Member Oracle Data Integration Solutions Partner Advisory Council
  - Vice President Twin Cities Oracle Users Group (Minneapolis-St. Paul)
  - Co-Founder International GoldenGate Oracle Users Group
- Principal Adviser: Forsythe **Meta7**

System/370-145 system console

My Websites: Morgan's Library



www.morganslibrary.org

Forsythe (1:2)

- In business 46 years
- $1.2B in 2016
- Partner with more than 200 technology OEMs

- Second largest security integrator in North America

A10 Networks	DataCableTech	Liquidware Labs Logitech	Riverbed Technology
AccessData	Dataram	LockPath	RSA Security
Accutech	Dell EMC	LogLogic	SafeNet
Acronis	Dialogic Dovetailed Technologies	LogRhythm	Sanbolic
ADVA	Digital Guardian	Loop1 Systems	Seagate
Aerohive	Dynatrace	LSI Corporation	Securonix
AirMagnet	Eaton Powerware	Luminex	Server Technology
AirTight Networks	EDGE Memory	Maxell	Service Now
AirWatch	Emulex	McAfee	Silver Peak
AlgoSec	EndRun Technologies	Mellanox Technologies	Software Diversified Services
Amazon	Entrust	Microsoft	Solarflare Communications
APC	Equinix	MobileIron	SolarWinds
AppDynamics	ExtraHop	MRV	Sophos
AppSense	F5 Networks	Multi-Tech Systems	Spectra Logic
Apptio	Fidelis Cybersecurity	nCircle Network Security	Splunk
APTARE	Finisar	Net Optics	STEALTHbits Technologies
Arbor Networks	FireEye	NetApp	SUSE
Arista	FireMon	NetBrain	Symantec
Aruba Networks	Fluke Networks	NetScout	Symmetricom
Avago Technologies	ForeScout Technologies	Netskope	T5
Avant Communications	Fortinet	Network Executive Software	Tele-Communication, Inc.
Avocent Corporation	Fuji	Nimble Storage	Tenable Network Security
Axway	Fujifilm	Norman Data Defense Systems,	Texas Memory Systems
Barracuda Networks	Fujitsu	Inc.	The Written Word
BlueCat Networks	Fusion-io	Northern Software	TierPoint
BMC Software	Gemalto	Novell	Tintri
Boldon James	GIGABYTE	NTP Software	Titus
Box	Gigamon	Nutanix	TransVault
Bradford Networks	Google	NVIDIA	Trend Micro
Brocade	Guidance Software	OCZ Technology	Tripp Lite
CA Technologies	HBGary	Opengear	Tripwire
Cable-Comm Technologies	HDS	Oracle	Trustwave Holdings
Carbon Black	Hewlett Packard Enterprise	Palo Alto Networks	Tufin Software North America,
Catbird Networks	IBM	Panasonic North America	Inc.
CCX Corporation	Imation	Panduit	Variphy

Forsythe (2:2)

- In business 46 years
- $1.2B in 2016
- Partner with more than 200 technology OEMs

Centrify	Imperva	Panzura	Varonis
Cenzic	Index Engines	Peer Software	VCE
Chatsworth	Infoblox	Pivot3	Veeam
Check Point	Intel	PKWARE	Veracode
Ciena	IPsoft	Proofpoint	Veritas
Cisco	Ipswitch	Pure Storage	Vertiv
Citrix	ISI Telemanagement Solutions,	Qlogic	Viavi Solutions
Cloudgenix	Inc.	Qualys	Violin Memory
CommVault	Ixia	Quantum	Viptela
Cortelco	JadeLiquid Software	Radware	Virtual Instruments
Crossbeam Systems	JDSU	Rapid7	VMTurbo
CrowdStrike	Juniper	Raritan	VMware
CTERA Networks	Kingston	RecoveryPlanner	Voltage Security
CyberArk	Lancope	Red Hat	Vormetric
Cylance	Lantronix	RedSeal Systems	Websense
Damballa	Lenovo	Resilient, an IBM Company	Winchester Systems
	Liebert	Reveille Software	Zerto

- Focusing on solutions to business problems ... not products

What Meta7 Brings To The Party

- Oracle only division of Forsythe
- Platinum Partner
- Focuses on the entire Oracle technology stack
  - The entire line of Oracle infrastructure from x86 through the full stack of engineered systems and storage
  - Oracle Database
    - Design and Deployment
    - Stability
    - Security
    - Scalability
  - Data Integration (GoldenGate)
  - Oracle Cloud
    - DevOps
    - Infrastructure as Code
- Focusing on solutions to business problems ... not products

Stability: IT Fire Fighting


Oracle Stack Security

Scalability: VLDBs and Partitioning

Database Performance


Zero Downtime Migration

Just In Time IT Procurement

with the Oracle Bare Metal Cloud and Infrastructure as Code

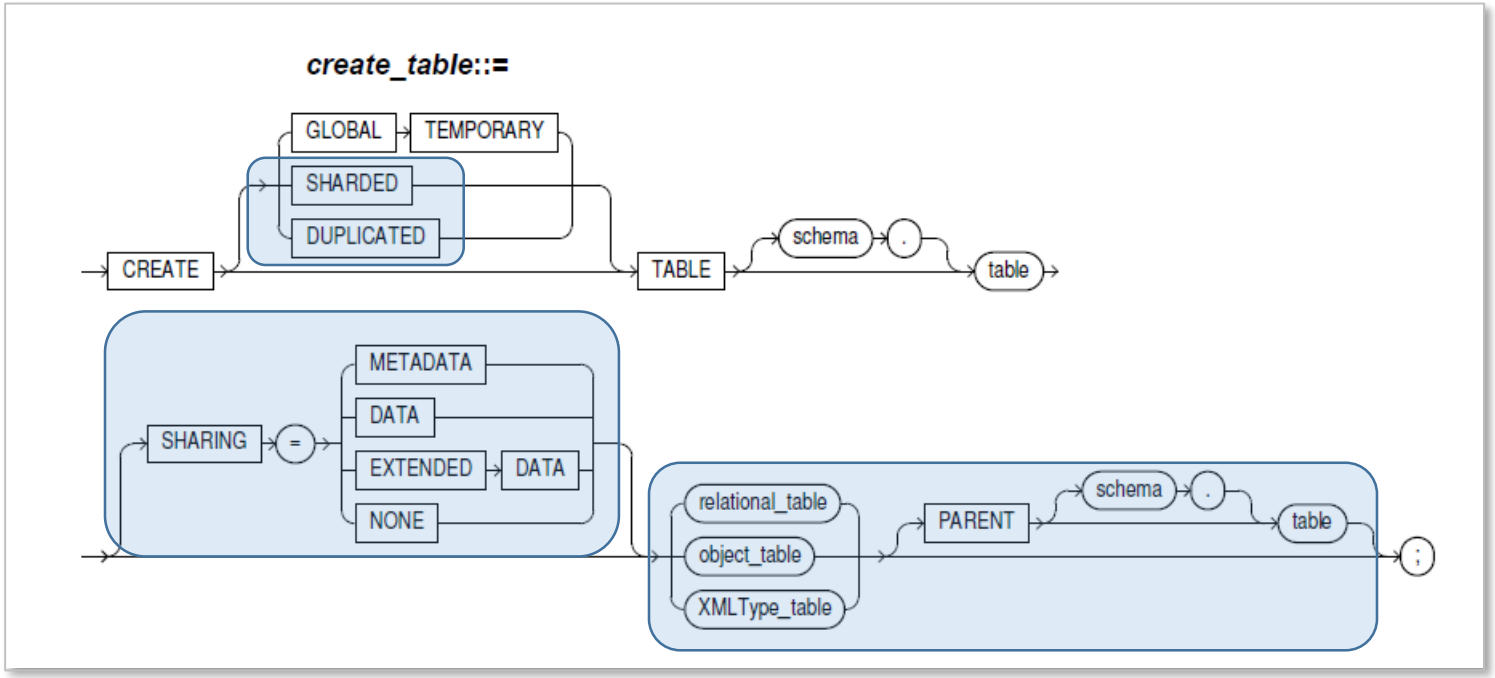
Content Density Warning

Introduction to Oracle Tables

Why Am I Focusing On Oracle Tables?

- Because no one else is
- Because Oracle University doesn't teach this material
- Because there are 119 pages in the 12cR2 docs under CREATE TABLE
- Because no one knows the full syntax for basic DDL statements
- Because we have now spent more than 30 years talking about performance tuning and yet the number one conference and training topic remains tuning which proves that we need to stop focusing on edge cases and focus, instead, on the basics
- Because explain plans, AWR Reports, and trace files will never fix a problem if you don't know the full range of options available
- Because the best way to achieve high performance is to choose techniques that reduce resource utilization

How Well Do You Know CREATE TABLE?



Legacy Architecture Tables

Legacy Data Dictionary: DBA_, ALL_, USER_

VIEW_NAME

- **DBA_ALL_TABLES**
- DBA_APPLY_TABLE_COLUMNS
- DBA_BASE_TABLE_MVIEWS
- DBA_CAPTURE_PREPARED_TABLES
- **DBA_CLUSTERING_TABLES**
- DBA_EVALUATION_CONTEXT_TABLES
- **DBA_EXTERNAL_TABLES**
- DBA_FILE_GROUP_TABLES
- DBA_FILE_GROUP_TABLESPACES
- DBA_FLASHBACK_ARCHIVE_TABLES
- DBA_HIST_DATABASE_INSTANCE
- DBA_LBAC_TABLE_POLICIES
- DBA_MINING_MODEL_TABLES
- DBA_NESTED_TABLES
- DBA_NESTED_TABLE_COLS
- DBA_OBJECT_TABLES
- DBA_PARTIAL_DROP_TABS
- **DBA_PART_TABLES**
- DBA_PENDING_CONV_TABLES
- DBA_QUEUE_TABLES
- DBA_SECUREFILE_LOG_TABLES
- DBA_SOURCE_TABLES
- DBA_SUBSCRIBED_TABLES

VIEW_NAME

- DBA_SUMMARY_DETAIL_TABLES
- DBA_SYNC_CAPTURE_PREPARED_TABS
- DBA_SYNC_CAPTURE_TABLES
- **DBA_TABLES**
- **DBA_TAB_COLS**
- DBA_TAB_COLS_V$
- **DBA_TAB_COLUMNS**
- **DBA_TAB_COL_STATISTICS**
- DBA_TAB_COMMENTS
- DBA_TAB_HISTGRM_PENDING_STATS
- DBA_TAB_HISTOGRAMS
- **DBA_TAB_IDENTITY_COLS**
- **DBA_TAB_MODIFICATIONS**
- **DBA_TAB_PARTITIONS**
- DBA_TAB_PENDING_STATS
- **DBA_TAB_PRIVS**
- DBA_TAB_STATISTICS
- DBA_TAB_STATS_HISTORY
- DBA_TAB_STAT_PREFS
- DBA_TAB_SUBPARTITIONS
- DBA_TSTZ_TABLES
- DBA_TSTZ_TAB_COLS
- DBA_UNUSED_COL_TABS

VIEW_NAME

- **DBA_UPDATABLE_COLUMNS**
- DBA_WM_VERSIONED_TABLES
- **DBA_XML_NESTED_TABLES**
- DBA_XML_OUT_OF_LINE_TABLES
- **DBA_XML_TABLES**
- **DBA_XML_TAB_COLS**

Heap Tables (1:2)

- Almost every Oracle table ever built by everyone except Oracle Corp. is a heap table stored in a tablespaces ... a description of "vanilla" as good as any
- Heap tables are optimized for persistent storage of data when you have little idea of what you are putting in and less of an idea of how that data, after storage will be used
- There isn't one good reason for making every table a heap table: Not one

- By default all Oracle tables are heap tables
  - A heap table is a single segment consisting of one or more extents: each extent consisting of one or more blocks
  - A heap table stores data as an unorganized pile of rows ... unordered
  - With a heap table each "8K" block can contain information from only a single table

Heap Tables (2:2)

```
SQL> CREATE TABLE state (state_abbrev VARCHAR2(2));

Table created.

SQL> desc state
 Name                           Null?    Type
 ------------------------------ -------- ----------------
 STATE_ABBREV                            VARCHAR2(2)


SQL> SELECT dbms_metadata.get_ddl('TABLE', 'STATE')
  2  FROM dual;

DBMS_METADATA.GET_DDL('TABLE','STATE')
--------------------------------------------------------------------------------
  CREATE TABLE "UWCLASS"."STATE"
   (    "STATE_ABBREV" VARCHAR2(2)
   ) SEGMENT CREATION DEFERRED
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
 TABLESPACE "UWDATA"
```

External Tables (1:3)

- An external tables is a physical file stored in a file system on disk
  - Columns can be defined as fixed length or delimited
  - External tables can be created by the database
  - DML is not allowed on external tables
  - SELECT statements interact with external tables precisely the same way they interact with internal heap tables
  - External tables are a replacement for the legacy SQL*Loader tool

```
SQL> conn sys@pdbdev as sysdba
Enter password:
Connected.

SQL> CREATE OR REPLACE DIRECTORY ext AS 'c:\external';

Directory created.

SQL>
SQL> GRANT read, write ON DIRECTORY ext TO uwclass;

Grant succeeded.
```

External Tables (2:3)

```
-- create a file named
7369,KYTE,SME,20
7499,MILLSAP,SALESMAN,30
7521,NORGAARD,SALESMAN,30
7566,KOLK,MANAGER,20
7654,LEWIS,ANALYST,30

-- create a file named
1111,MORGAN,DIRECTOR,10
2222,HARDIE,MANAGER,30
3333,HAVEMEYER,CTO,10
4444,LOFSTROM,DEVELOPER,10
5555,TOWNSEND,MANAGER,30

-- create a file on the hard disk named cost.txt with the follow 4 lines:
YEAR PID CPU  GROSS REVENUE
2003 def 2.00 123.4567890
2004 ABC 1.00 39.7288841651344
2005 xyz 1.99 1107.5458517352
```

A comma delimited flat file is export from a UNIX database and transferred for loading into a Windows based Oracle database. The lines are terminated with 0x0A and not the 0x0A 0x0D pair for DOS/Windows. The fields are randomly double quoted, and some contain commas internal to the field's data like the following:

```
100001,"7123 HIGHLAND DR","03/28/1965"," Mercer Island, WA 98040","Morgan, Dan A", "","","63034630752",
14,1,"F","T","06/28/2009","",0,"","01/01/2010","N", "01/01/1999",""
100020,"5432 SOUTH 28TH ST","01/01/1951"," Mercer Island, WA 98040", "Burger,Tom",
"","2566400","ZPW345070938",64,1,"M","B","02/23/2000","",0,"", "12/31/1799","P","12/31/1979",""
-- save external file as c:\external\preprocess.dat.gz (if UNIX or LINUX use your home directory)
```

External Tables (3:3)

- Note the key words "ORGANIZATION EXTERNAL"

```
CREATE TABLE <table_name> (
<column_definitions>)
ORGANIZATION EXTERNAL
(TYPE oracle_loader
DEFAULT DIRECTORY <oracle_directory_object_name>
ACCESS PARAMETERS (
RECORDS DELIMITED BY newline
BADFILE <file_name>
DISCARDFILE <file_name>
LOGFILE <file_name>
[READSIZE <bytes>]
[SKIP <number_of_rows>]
FIELDS TERMINATED BY '<terminator>'
OPTIONALLY ENCLOSED BY '<character>'
MISSING FIELD VALUES ARE <VALUE | NULL>
REJECT ROWS WITH ALL NULL FIELDS
(<column_name_list>))\
LOCATION ('<file_name>'))
[PARALLEL]
REJECT LIMIT <UNLIMITED | integer>
MONITORING | NOMONITORING];
```

```
conn uwclass/uwclass@pdbdev

CREATE TABLE ext_tab1 (
empno  CHAR(4),
ename  CHAR(20),
job    CHAR(20),
deptno CHAR(3))
ORGANIZATION EXTERNAL (
  TYPE oracle_loader
  DEFAULT DIRECTORY ext
  ACCESS PARAMETERS (
    RECORDS DELIMITED BY NEWLINE
    BADFILE ext:'bad_%a_%p.bad'
    LOGFILE ext:'log_%a_%p.log'
    FIELDS TERMINATED BY ','
    OPTIONALLY ENCLOSED BY '"'
    MISSING FIELD VALUES ARE NULL
    REJECT ROWS WITH ALL NULL FIELDS
    (empno, ename, job, deptno))
    LOCATION ('demo1.dat'))
PARALLEL
REJECT LIMIT 0
NOMONITORING;

SELECT * FROM ext_tab;
```

Global Temporary Tables (GTT) (1:2)

- One of the most expensive things you can do in an Oracle database is "DROP TABLE"
- Global temporary tables address this issue by creating a permanent table where the data is temporary
- GTTs can have
  - Constraints
  - Indexes
  - Optimizer Statistics
  - Synonyms
  - Views
- GTTs are built in the temporary tablespace so the same METADATA is shared by all users while the data remains private

Global Temporary Tables (GTT) (2:2)

- There are two types of GTTs
  - ON COMMIT DELETE ROWS
    - Data is deleted when a COMMIT is issued
    - Use these primarily to preserve a temporary working data set or an intermediate result set
  - ON COMMIT PRESERVE ROWS
    - Data is deleted when the session ends
    - Use these primarily to preserve a working data set across multiple transactions or as a result table for a report writing/business intelligence tool

```
CREATE GLOBAL TEMPORARY TABLE gtt_ocdr (
zip_code     VARCHAR2(5),
by_user      VARCHAR2(30),
entry_date   DATE)
ON COMMIT DELETE ROWS;


CREATE GLOBAL TEMPORARY TABLE gtt_ocpr (
zip_code     VARCHAR2(5),
by_user      VARCHAR2(30),
entry_date   DATE)
ON COMMIT PRESERVE ROWS;
```

Index Organized Tables (IOT) (1:2)

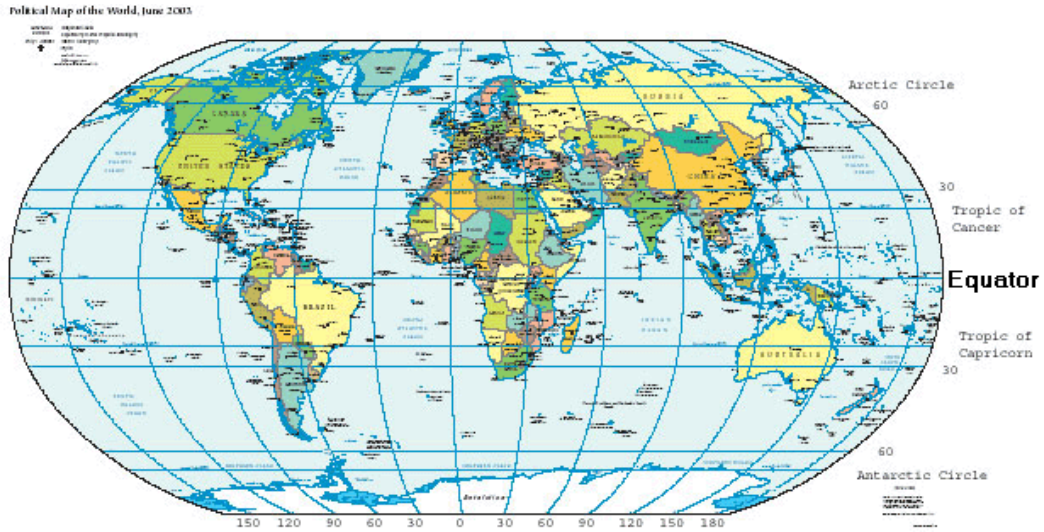
- IOTs are wholly underutilized by developers and DBAs
- An IOT is a table stored as a B*Tree index rather than as an unorganized heap of data
- Use IOTs where the size of the primary key index is a substantial fraction, or larger, than the size of the table
- Use IOTs when some of the columns are accessed frequently and others may be rarely or never accessed
- Look up tables, the children of foreign key referential constraints are often perfect candidates for storage as an IOT

Index Organized Tables (IOT) (2:2)

```
CREATE TABLE labor_hour (
WORK_DATE   DATE,
EMPLOYEE_NO VARCHAR2(8),
CONSTRAINT pk_labor_hour
PRIMARY KEY (work_date, employee_no))
ORGANIZATION INDEX;

CREATE TABLE compressed_iot (owner, object_type, object_name,
CONSTRAINT pk_compressed_iot
PRIMARY KEY(owner, object_type, object_name))
ORGANIZATION INDEX
COMPRESS 2 AS
SELECT owner, object_type, object_name
FROM all_objects;

CREATE TABLE labor_hour (
WORK_DATE            DATE,
EMPLOYEE_NO          VARCHAR2(8),
SUMMIT_WORK_ORDER_NO VARCHAR2(7),
DASH                 VARCHAR2(2),
CLASS_CODE           VARCHAR2(6),
PAYCODE              VARCHAR2(2),
ASSIGNED_CREW_NUMBER VARCHAR2(5),
TRANSFER_CREW_NUMBER VARCHAR2(5),
REFERENCE_TYPE       VARCHAR2(1),
REFERENCE_NUMBER     VARCHAR2(10),
OVERTIME_CODE        VARCHAR2(1),
SHIFT_DIFFERENTIAL   VARCHAR2(1)  NOT NULL,
HOURS                NUMBER(4,2)  NOT NULL,
MOD_USER_ID          VARCHAR2(30) DEFAULT USER,
MOD_USER_DATE        DATE DEFAULT SYSDATE,
CONSTRAINT pk_labor_hour
PRIMARY KEY (work_date, employee_no, summit_work_order_no, dash, class_code, paycode, assigned_crew_number,
transfer_crew_number, reference_type, reference_number, overtime_code, shift_differential))
ORGANIZATION INDEX
INCLUDING hours
PCTTHRESHOLD 10
OVERFLOW TABLESPACE uwdata;
```

Partitioned Tables (1:7)

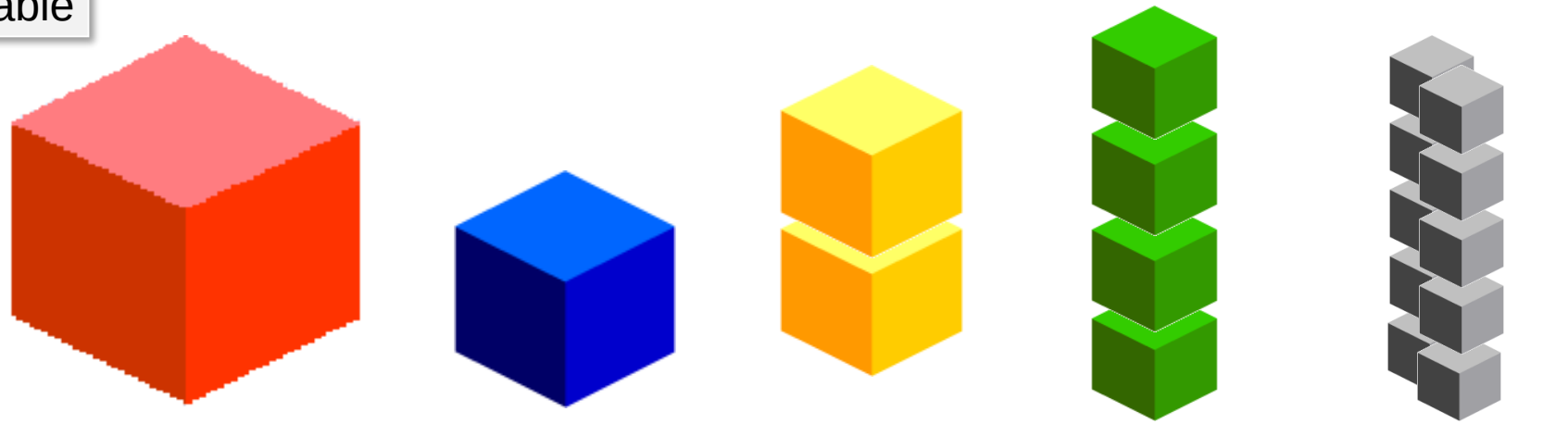
- Tables are a logical, not a physical, construct
- Partitions and Subpartition are maps
- Just as a *tablespace* maps many *datafile*
- Just as a regular *heap table* maps many *blocks*
- Partitioned tables map *segments* so that they appear to be a single object
- Partitions may optionally map many *subpartitions* for increased granularity



Partitioned Tables (2:7)

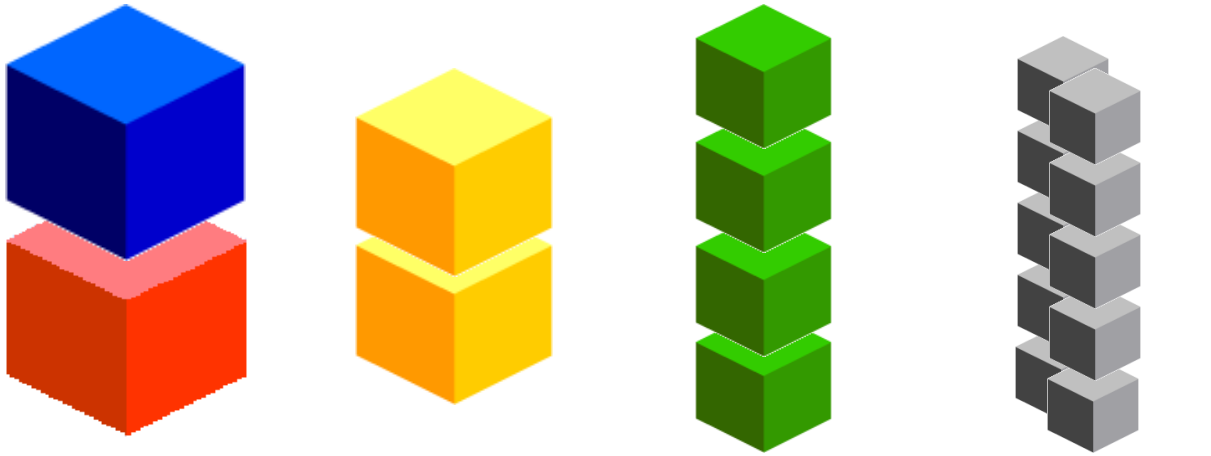
- Segments exist within a tablespace
- Segments are a collection of extents
- Extents are a collection of data blocks
- Data blocks are mapped to disk blocks

Basic Heap Table



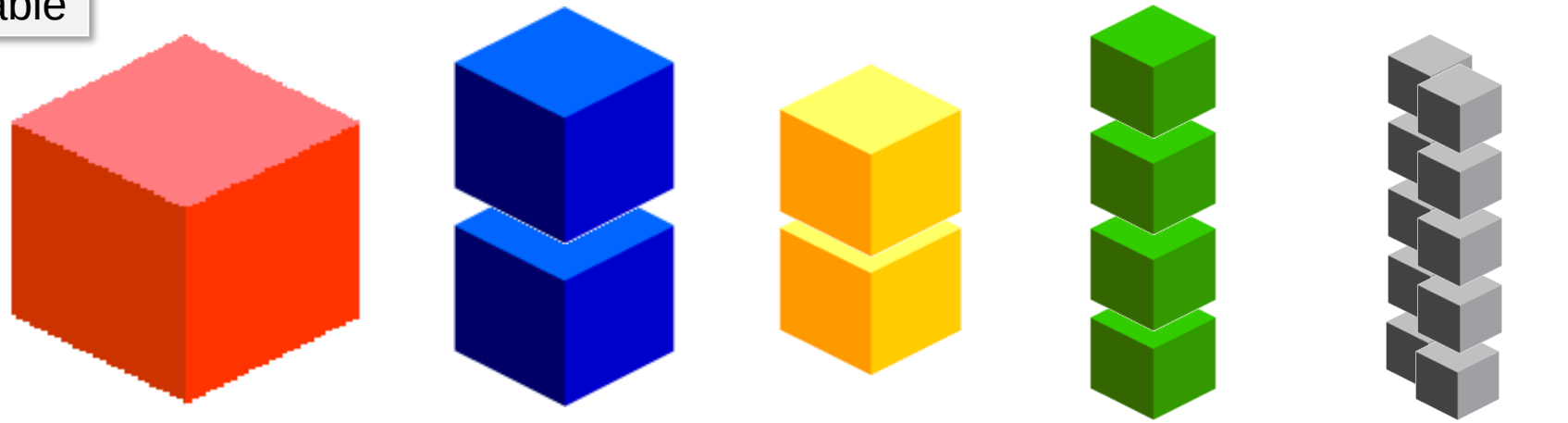
Partitioned Tables (3:7)

Basic Heap Table



Segment=Object Extents DB Blocks O/S Blocks

Partitioned Table



Object <> Segment Extents DB Blocks O/S Blocks

Partitioned Tables (4:7)

- Multiple types of partitioning
  - HASH (a database internally generated hashing algorithm) - least valuable and overused
  - LIST (for example North, South, East West)
  - RANGE (for example 2015, 2016, 2017, 2018) - most valuable and most commonly used
    - Multiple types of RANGE partitioning
      - Basic
      - Interval (date or numeric)
      - Reference
  - SYSTEM
- Data stored in partitions can be
  - Encrypted to protect sensitive information (for example credit cards)
  - Compressed to save on storage costs (for example older data)
  - Relocated to slower, less expensive, disk when it is less frequently accessed
  - Made READ ONLY so that it cannot be accidentally changed
  - Easily exported to flat files, spreadsheets, and other databases
  - Exported into spreadsheets with lessened system impact

Partitioned Tables (5:7)

- System Partitioning
  - Partitioned tables created by the Oracle Database and owned by one of the database's built-in schemas
- User Partitioning
  - Partitioned tables created by users or applications installed by the user
  - Only available for Enterprise Edition

Feature Name	Curr- ently Used	Detected Usages	Total Samples	Last Usage Time
Automatic SGA Tuning	TRUE	22	22	02/11/17 04:52
Automatic SQL Execution Memory	TRUE	22	22	02/11/17 04:52
Automatic Segment Space Management (system)	TRUE	22	22	02/11/17 04:52
Automatic Segment Space Management (user)	TRUE	21	22	02/11/17 04:52
Automatic Storage Management	TRUE	22	22	02/11/17 04:52
Automatic Undo Management	TRUE	22	22	02/11/17 04:52
Character Set	TRUE	22	22	02/11/17 04:52
Client Identifier	TRUE	6	22	02/11/17 04:52
Deferred Segment Creation	TRUE	22	22	02/11/17 04:52
Extensibility	TRUE	21	22	02/11/17 04:52
Job Scheduler	TRUE	22	22	02/11/17 04:52
LOB	TRUE	22	22	02/11/17 04:52
Locally Managed Tablespaces (system)	TRUE	22	22	02/11/17 04:52
Locally Managed Tablespaces (user)	TRUE	22	22	02/11/17 04:52
Logfile Multiplexing	TRUE	22	22	02/11/17 04:52
Object	TRUE	21	22	02/11/17 04:52
Oracle Java Virtual Machine (system)	TRUE	22	22	02/11/17 04:52
Parallel SQL Query Execution	TRUE	22	22	02/11/17 04:52
Partitioning (system)	TRUE	22	22	02/11/17 04:52
RMAN - Disk Backup	TRUE	13	22	02/11/17 04:52

Parallel SQL DML Execution	FALSE	0	22	
Partitioning (user)	FALSE	0	22	
Pillar Storage	FALSE	0	22	

Partitioned Tables (6:7)

- Speed with Very Large Databases
  - Partition Pruning (Developer)

    Oracle optimizes SQL statements to mark the partitions or subpartitions that need to be accessed and eliminates (prunes) unnecessary partitions or subpartitions from access. Partition pruning is the skipping of unnecessary index and data partitions or subpartitions by a query

  - Partition Elimination (DBA)
- Ease of Management
- The following built-in objects support partition tables
  - DATAOBJ_TO_PARTITION function
  - DBMS_PART package
  - DBMS_PCLXUTIL package
- Partitions can be based on normal, hidden, and virtual columns
- It is always preferable, whenever possible, to use local indexes

Partitioned Tables (7:7)

- The following are examples of special use cases for table partitioning

```
CREATE TABLE orders OF XMLType
XMLTYPE STORE AS BINARY XML
VIRTUAL COLUMNS (site_id AS (XMLCast(XMLQuery('/Order/@SiteId' PASSING OBJECT_VALUE RETURNING CONTENT) AS NUMBER)))
PARTITION BY RANGE (site_id) (
PARTITION p1 VALUES LESS THAN (10),
PARTITION p2 VALUES LESS THAN (20),
PARTITION pm VALUES LESS THAN (MAXVALUE));
```

```
CREATE TABLE json_orders (
tx_id    NUMBER(5),
tx_date  DATE,
jsondata VARCHAR2(4000),
site_id  AS (JSON_VALUE(jsondata, '$.siteId' RETURNING NUMBER))
PARTITION BY RANGE (site_id) (
PARTITION p1 VALUES LESS THAN (10),
PARTITION p2 VALUES LESS THAN (20),
PARTITION pm VALUES LESS THAN (MAXVALUE));
```

```
CREATE TABLE ref_child (
table_name VARCHAR2(30) NOT NULL,
index_name VARCHAR2(30) NOT NULL,
CONSTRAINT fk_ref_child_parent
FOREIGN KEY(table_name) REFERENCES ref_parent(table_name))
PARTITION BY REFERENCE(fk_ref_child_parent);
```

XML Tables

- Use the optimized storage of XML tables when persisting storage of XML documents

```
CREATE TABLE xml_lob_tab OF XMLTYPE;

CREATE TABLE uwclass$schema OF SYS.XMLTYPE
XMLSCHEMA "http://xmlns.oracle.com/xdb/XDBSchema.xsd"
ELEMENT "schema" ID 81
TABLESPACE uwdata;

CREATE TABLE orders OF XMLType
XMLTYPE STORE AS BINARY XML;
```

Object-Relational Tables

Object/Nested Tables with a NESTED TABLE

- The best advice I can provide you is don't build these ... you will sacrifice data integrity, space, and performance
- That said ... here's how you do it

```
CREATE OR REPLACE NONEDITIONABLE TYPE CourseList AS TABLE OF VARCHAR2(64);
/

CREATE TABLE department (
name     VARCHAR2(20),
director VARCHAR2(20),
office   VARCHAR2(20),
courses  CourseList)
NESTED TABLE courses STORE AS courses_tab;

INSERT INTO department
(name, director, office, courses)
VALUES
('English', 'Lynn Saunders', 'Breakstone Hall 205', CourseList(
'Expository Writing',
'Film and Literature',
'Modern Science Fiction',
'Discursive Writing',
'Modern English Grammar',
'Introduction to Shakespeare',
'Modern Drama',
'The Short Story',
'The American Novel'));
```

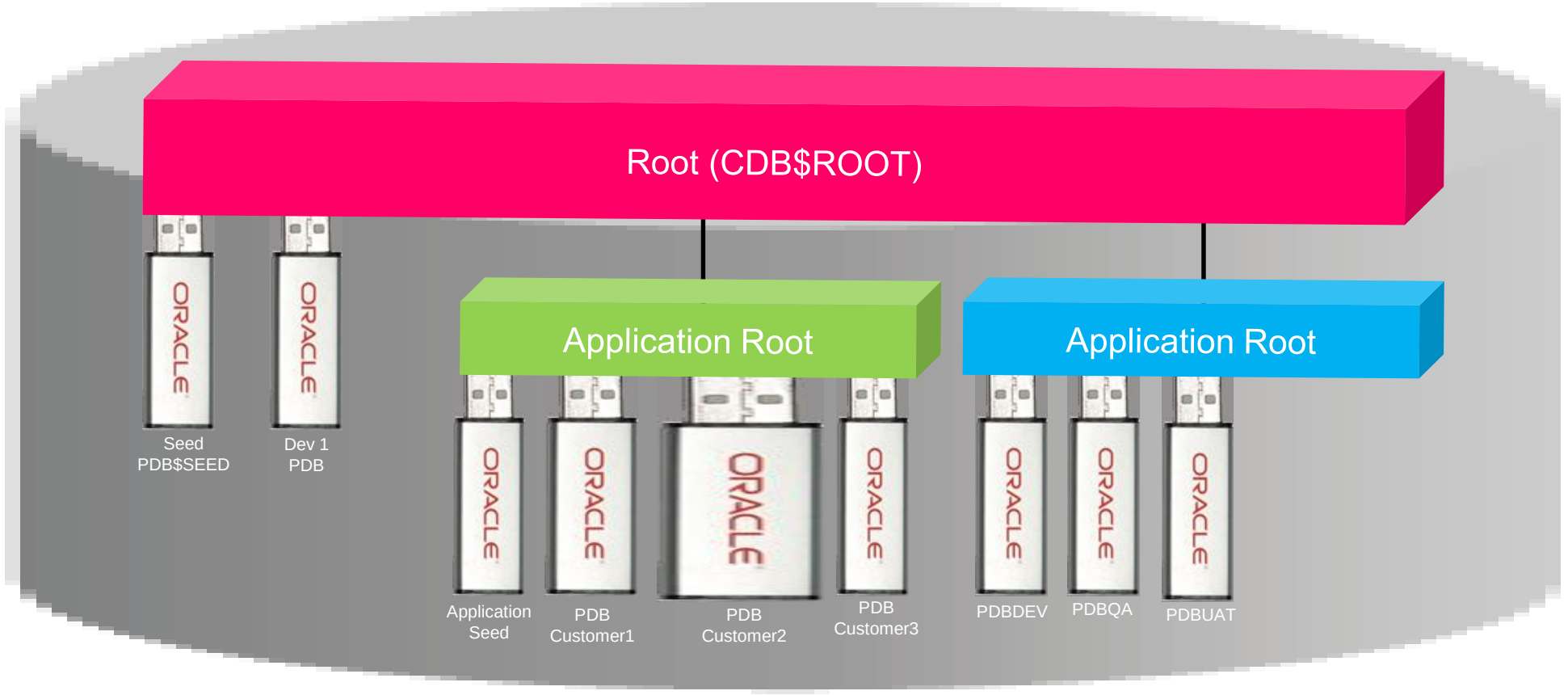
Object/Nested Table Based with a VARRAY

- The best advice I can provide you is don't build these ... you will sacrifice data integrity, space, and performance
- That said ... here's how you do it

```
CREATE OR REPLACE TYPE Project AUTHID DEFINER AS OBJECT (
project_no NUMBER(2),
title      VARCHAR2(35),
cost       NUMBER(7,2));
/

CREATE OR REPLACE TYPE ProjectList AS VARRAY(50) OF Project;
/

CREATE TABLE department (
dept_id  NUMBER(2),
dname    VARCHAR2(15),
budget   NUMBER(11,2),
projects ProjectList);

INSERT INTO department
(dept_id, dname, budget, projects)
VALUES
(42, 'Quantum Mechanics', '$424242.42',
ProjectList(Project(1, 'Standard Model', 2121'),
            Project(2, 'Uncertainty Principal', 21.21)));
```

12c Container Architecture Tables

New 12cR2 Container Database Architecture



Common Tables (1:2)

- As of Oracle Database version 12.2 tables in an Application Root container can be optionally created with shared metadata or shared data
- When **metadata**, the DDL, is shared into PDBs as pointers from the Application Root data dictionary

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION uw_app
  2  BEGIN UPGRADE '1.0' TO '2.0'
  3  COMMENT 'Adding New Table With Sharing';

SQL> CREATE TABLE serv_inst
  2  SHARING=METADATA (
  3  siid NUMBER(10),
  4  si_status VARCHAR2(15),
  5  type VARCHAR2(5),
  6  installstatus VARCHAR2(1),
  7  location_code NUMBER(10),
  8  custacct_id VARCHAR2(10),
  9  srvr_id NUMBER(10),
 10* ws_id NUMBER(10));

Table created.

SQL> ALTER PLUGGABLE DATABASE APPLICATION uw_app END UPGRADE;
```

Common Tables (2:2)

- As of Oracle Database version 12.2 tables in an Application Root container can be optionally created with shared metadata or shared data
- When **data** is shared the DDL and the table rows are shared into PDBs under the application root

```
SQL> ALTER PLUGGABLE DATABASE APPLICATION uw_app
  2  BEGIN UPGRADE '1.0' TO '2.0'
  3  COMMENT 'Adding New Table With Sharing';

SQL> CREATE TABLE serv_inst
  2  SHARING=DATA (
  3  siid NUMBER(10),
  4  si_status VARCHAR2(15),
  5  type VARCHAR2(5),
  6  installstatus VARCHAR2(1),
  7  location_code NUMBER(10),
  8  custacct_id VARCHAR2(10),
  9  srvr_id NUMBER(10),
 10* ws_id NUMBER(10));

Table created.

SQL> ALTER PLUGGABLE DATABASE APPLICATION uw_app END UPGRADE;
```

Table Options

Tables in the 12.2 Data Dictionary (1:2)

- By default all Oracle tables are heap tables

```
SQL> desc dba_tables
 Name                          Null?    Type
 ----------------------------- -------- ---------------
 OWNER                         NOT NULL VARCHAR2(128)
 TABLE_NAME                    NOT NULL VARCHAR2(128)
 TABLESPACE_NAME                        VARCHAR2(30)
 CLUSTER_NAME                           VARCHAR2(128)
 IOT_NAME                               VARCHAR2(128)
 STATUS                                 VARCHAR2(8)
 PCT_FREE                               NUMBER
 PCT_USED                               NUMBER
 INI_TRANS                              NUMBER
 MAX_TRANS                              NUMBER
 INITIAL_EXTENT                         NUMBER
 NEXT_EXTENT                            NUMBER
 MIN_EXTENTS                            NUMBER
 MAX_EXTENTS                            NUMBER
 PCT_INCREASE                           NUMBER
 FREELISTS                              NUMBER
 FREELIST_GROUPS                        NUMBER
 LOGGING                                VARCHAR2(3)
 BACKED_UP                              VARCHAR2(1)
 NUM_ROWS                               NUMBER
 BLOCKS                                 NUMBER
 EMPTY_BLOCKS                           NUMBER
 AVG_SPACE                              NUMBER
 CHAIN_CNT                              NUMBER
 AVG_ROW_LEN                            NUMBER
 AVG_SPACE_FREELIST_BLOCKS              NUMBER
 NUM_FREELIST_BLOCKS                    NUMBER
 DEGREE                                 VARCHAR2(10)
 INSTANCES                              VARCHAR2(10)
 CACHE                                  VARCHAR2(5)
 TABLE_LOCK                             VARCHAR2(8)
```

```
 SAMPLE_SIZE                    NUMBER
 LAST_ANALYZED                  DATE
 PARTITIONED                    VARCHAR2(3)
 IOT_TYPE                       VARCHAR2(12)
 TEMPORARY                      VARCHAR2(1)
 SECONDARY                      VARCHAR2(1)
 NESTED                         VARCHAR2(3)
 BUFFER_POOL                    VARCHAR2(7)
 FLASH_CACHE                    VARCHAR2(7)
 CELL_FLASH_CACHE               VARCHAR2(7)
 ROW_MOVEMENT                   VARCHAR2(8)
 GLOBAL_STATS                   VARCHAR2(3)
 USER_STATS                     VARCHAR2(3)
 DURATION                       VARCHAR2(15)
 SKIP_CORRUPT                   VARCHAR2(8)
 MONITORING                     VARCHAR2(3)
 CLUSTER_OWNER                  VARCHAR2(128)
 DEPENDENCIES                   VARCHAR2(8)
 COMPRESSION                    VARCHAR2(8)
 COMPRESS_FOR                   VARCHAR2(30)
 DROPPED                        VARCHAR2(3)
 READ_ONLY                      VARCHAR2(3)
 SEGMENT_CREATED                VARCHAR2(3)
 RESULT_CACHE                   VARCHAR2(7)
 CLUSTERING                     VARCHAR2(3)
 ACTIVITY_TRACKING              VARCHAR2(23)
 DML_TIMESTAMP                  VARCHAR2(25)
 HAS_IDENTITY                   VARCHAR2(3)
 CONTAINER_DATA                 VARCHAR2(3)
 INMEMORY                       VARCHAR2(8)
 INMEMORY_PRIORITY              VARCHAR2(8)
 INMEMORY_DISTRIBUTE            VARCHAR2(15)
 INMEMORY_COMPRESSION           VARCHAR2(17)
 INMEMORY_DUPLICATE             VARCHAR2(13)
```

Tables in the 12.2 Data Dictionary (2:2)

- By default all Oracle tables are heap tables

```
SQL> desc dba_tables
 Name                          Null?    Type
 ----------------------------- -------- ---------------
 OWNER                         NOT NULL VARCHAR2(128)
 TABLE_NAME                    NOT NULL VARCHAR2(128)
 TABLESPACE_NAME                        VARCHAR2(30)
 CLUSTER_NAME                           VARCHAR2(128)
 IOT_NAME                               VARCHAR2(128)
 STATUS                                 VARCHAR2(8)
 PCT_FREE                               NUMBER
 PCT_USED                               NUMBER
 INI_TRANS                              NUMBER
 MAX_TRANS                              NUMBER
 INITIAL_EXTENT                         NUMBER
 NEXT_EXTENT                            NUMBER
 MIN_EXTENTS                            NUMBER
 MAX_EXTENTS                            NUMBER
 PCT_INCREASE                           NUMBER
 FREELISTS                              NUMBER
 FREELIST_GROUPS                        NUMBER
 LOGGING                                VARCHAR2(3)
 BACKED_UP                              VARCHAR2(1)
 NUM_ROWS                               NUMBER
 BLOCKS                                 NUMBER
 EMPTY_BLOCKS                           NUMBER
 AVG_SPACE                              NUMBER
 CHAIN_CNT                              NUMBER
 AVG_ROW_LEN                            NUMBER
 AVG_SPACE_FREELIST_BLOCKS              NUMBER
 NUM_FREELIST_BLOCKS                    NUMBER
 DEGREE                                 VARCHAR2(10)
 INSTANCES                              VARCHAR2(10)
 CACHE                                  VARCHAR2(5)
 TABLE_LOCK                             VARCHAR2(8)
 SAMPLE_SIZE                            NUMBER
 LAST_ANALYZED                          DATE
 PARTITIONED                            VARCHAR2(3)
 IOT_TYPE                               VARCHAR2(12)
 TEMPORARY                              VARCHAR2(1)
 SECONDARY                              VARCHAR2(1)
 NESTED                                 VARCHAR2(3)
 BUFFER_POOL                            VARCHAR2(7)
 FLASH_CACHE                            VARCHAR2(7)
 CELL_FLASH_CACHE                       VARCHAR2(7)
 ROW_MOVEMENT                           VARCHAR2(8)
 GLOBAL_STATS                           VARCHAR2(3)
 USER_STATS                             VARCHAR2(3)
 DURATION                               VARCHAR2(15)
 SKIP_CORRUPT                           VARCHAR2(8)
 MONITORING                             VARCHAR2(3)
 CLUSTER_OWNER                          VARCHAR2(128)
 DEPENDENCIES                           VARCHAR2(8)
 COMPRESSION                            VARCHAR2(8)
 COMPRESS_FOR                           VARCHAR2(30)
 DROPPED                                VARCHAR2(3)
 READ_ONLY                              VARCHAR2(3)
 SEGMENT_CREATED                        VARCHAR2(3)
 RESULT_CACHE                           VARCHAR2(7)
 CLUSTERING                             VARCHAR2(3)
 ACTIVITY_TRACKING                      VARCHAR2(23)
 DML_TIMESTAMP                          VARCHAR2(25)
 HAS_IDENTITY                           VARCHAR2(3)
 CONTAINER_DATA                         VARCHAR2(3)
 INMEMORY                               VARCHAR2(8)
 INMEMORY_PRIORITY                      VARCHAR2(8)
 INMEMORY_DISTRIBUTE                    VARCHAR2(15)
 INMEMORY_COMPRESSION                   VARCHAR2(17)
 INMEMORY_DUPLICATE                     VARCHAR2(13)
```

Table Storage Types (1:2)

- Tablespace
  - A tablespace is the default table storage definition and is a logical mapping to space in one or more physical datafiles
  - Tablespaces are limited to storing one segment per block
  - Tablespaces can be read-write or read-only
  - The default Oracle tablespace definition wastes 10% of most tables
- Clusters
  - A cluster is a storage definition that can be mapped onto a tablespace that suspends the default rules and allows for capabilities that can be used to greatly enhance performance
  - The Oracle data dictionary's incredible performance speed is based in large part on the use of Clusters
- SecureFiles
  - Like Clusters a SecureFile is a mapped portion of a tablespace that suspends the default rules and enhances the ability to store LOBs (they have nothing to do with security)
- File System
  - Used by external tables

Table Storage Types (2:2)

- Here's how Oracle makes use of Clusters to enhance data dictionary performance

```
SQL> SELECT cluster_name, tablespace_name TBS_NAME,
cluster_type CTYPE
  2  FROM dba_clusters
  3  ORDER BY 1;

CLUSTER_NAME           TBS_NAME  CTYPE
---------------------- --------- -----
C_COBJ#                SYSTEM    INDEX
C_FILE#_BLOCK#         SYSTEM    INDEX
C_MLOG#                SYSTEM    INDEX
C_OBJ#                 SYSTEM    INDEX
C_OBJ#_INTCOL#         SYSTEM    INDEX
C_RG#                  SYSTEM    INDEX
C_TOID_VERSION#        SYSTEM    INDEX
C_TS#                  SYSTEM    INDEX
C_USER#                SYSTEM    INDEX
SMON_SCN_TO_TIME_AUX   SYSAUX    INDEX
```

- C_OBJ# stores information from 17 separate but related tables within a single block minimizing I/O and join overhead

```
SQL> SELECT table_name
  2  FROM dba_tables
  3  WHERE cluster_name = 'C_OBJ#'
  4* ORDER BY 1;

TABLE_NAME
------------------------------
ASSEMBLY$    -- Assemblies
ATTRCOL$     -- Column attributes
CLU$         -- Clusters
COL$         -- Columns
COLTYPE$     -- Column types
ICOL$        -- Index columns
ICOLDEP$     -- Index column dependents
IND$         -- Indexes
LIBRARY$     -- Libraries
LOB$         -- LOBs
NTAB$        -- Nested tables
OPQTYPE$     -- Opaque Types
REFCON$      -- Referential constraints
SUBCOLTYPE$ -- Subcolumn types
TAB$         -- Tables
TYPE_MISC$   -- Audits
VIEWTRCOL$   -- View Column Attributes
```

Cluster Types

- Single Table Cluster By Hash
- Multi-Table Cluster By Hash
- Multi-Table Cluster By Index
- Sorted Hash Clusters
  - Can reduce the overhead of an ORDER BY clause to zero

```
CREATE CLUSTER sorted_hc (
program_id  NUMBER(3),
line_id     NUMBER(10) SORT,
delivery_dt DATE SORT)
TABLESPACE uwdata
HASHKEYS 9
SIZE 750
HASH IS program_id;

CREATE TABLE shc_airplane (
program_id  NUMBER(3),
line_id     NUMBER(10) SORT,
delivery_dt DATE       SORT,
customer_id VARCHAR2(3),
order_dt    DATE)
CLUSTER sorted_hc (program_id, line_id, delivery_dt);
```

```
CREATE CLUSTER sc_srvr_id (
srvr_id NUMBER(10))
SIZE 1024;

CREATE INDEX idx_sc_srvr_id ON CLUSTER sc_srvr_id;

CREATE TABLE cservers (
srvr_id    NUMBER(10),
network_id NUMBER(10),
status     VARCHAR2(1),
latitude   FLOAT(20),
longitude  FLOAT(20),
netaddress VARCHAR2(15))
CLUSTER sc_srvr_id (srvr_id);

CREATE TABLE cserv_inst (
siid          NUMBER(10),
si_status     VARCHAR2(15),
type          VARCHAR2(5),
installstatus VARCHAR2(1),
location_code NUMBER(10),
custacct_id   VARCHAR2(10),
srvr_id       NUMBER(10),
ws_id         NUMBER(10))
CLUSTER sc_srvr_id (srvr_id);
```

SecureFiles

- SecureFiles LOB storage is one of two storage types used with Oracle Database 12c the other type is BasicFiles LOB storage
- The SECUREFILE LOB parameter enables advanced features, including compression and deduplication (part of the Advanced Compression Option), and encryption (part of the Advanced Security Option)
- Starting with Oracle Database 12c, SecureFiles is the default storage mechanism for LOBs

```
CREATE TABLE sec_tab_dd (
rid  NUMBER(5),
bcol BLOB)
LOB (bcol)
STORE AS SECUREFILE bcol2 (
TABLESPACE securefiletbs
RETENTION MIN 3600
COMPRESS ENCRYPT CACHE READS)
TABLESPACE uwdata;
```

ASM

- Not all ASM DiskGroups should be created equal
- For diskgroups not on SSD you can get more I/Os from the outside of the platter than the inside

```
ALTER DISKGROUP data
ADD TEMPLATE datafile_hot ATTRIBUTE (HOT MIRRORHOT);

ALTER DISKGROUP archives
ADD TEMPLATE datafile_cold ATTRIBUTE (COLD MIRRORCOLD);

ALTER DISKGROUP data
MODIFY FILE '+data/orcl/datafile/users.259.679156903' ATTRIBUTE (HOT MIRRORHOT);
```

Column Data Types (1:2)

- Consider your options before choosing a data type

```
CREATE TABLE tstring (
charcol                      CHAR(2000),
charvaryingcol               CHAR VARYING(4000),
charactercol                 CHARACTER(2000),
charactervaryingcol          CHARACTER VARYING(4000),
nationalcharvarying          NATIONAL CHAR VARYING(2000),
nationalcharactervaryingcol NATIONAL CHARACTER VARYING(2000),
ncharcol                     NCHAR(1000),
ncharvaryingcol              NCHAR VARYING(2000),
nvarchar2col                 NVARCHAR2(2000),
varcharcol                   VARCHAR(4000),
varchar2col                  VARCHAR2(4000));

SQL> desc tstring
Name                         Type
---------------------------- --------------
CHARCOL                      CHAR(2000)
CHARVARYINGCOL               VARCHAR2(4000)
CHARACTERCOL                 CHAR(2000)
CHARACTERVARYINGCOL          VARCHAR2(4000)
NATIONALCHARVARYING          NVARCHAR2(2000)
NATIONALCHARACTERVARYINGCOL NVARCHAR2(2000)
NCHARCOL                     NCHAR(1000)
NCHARVARYINGCOL              NVARCHAR2(2000)
NVARCHAR2COL                 NVARCHAR2(2000)
VARCHARCOL                   VARCHAR2(4000)
VARCHAR2COL                  VARCHAR2(4000)
```

```
CREATE TABLE tnumeric (
deccol               DEC(38),
decimalcol           DECIMAL(38),
doubleprecisioncol   DOUBLE PRECISION,
floatcol             FLOAT(126),
intcol               INT,
integercol           INTEGER,
numbercol            NUMBER(38),
numberfcol           NUMBER,
numericcol           NUMERIC(38),
numericfcol          NUMERIC,
realcol              REAL,
smallintcol          SMALLINT);

SQL> desc tnumeric
Name                 Type
-------------------- --------------
DECCOL               NUMBER(38)
DECIMALCOL           NUMBER(38)
DOUBLEPRECISIONCOL   FLOAT(126)
FLOATCOL             FLOAT(126)
INTCOL               NUMBER(38)
INTEGERCOL           NUMBER(38)
NUMBERCOL            NUMBER(38)
NUMBERFCOL           NUMBER
NUMERICCOL           NUMBER(38)
NUMERICFCOL          NUMBER(38)
REALCOL              FLOAT(63)
SMALLINTCOL          NUMBER(38)
```

Column Data Types (2:2)

- Consider your options before choosing a data type

```
CREATE TABLE ttemporal (
date_col    DATE,
int_d2s_col INTERVAL DAY TO SECOND,
int_y2m_col INTERVAL YEAR TO MONTH,
ts_col      TIMESTAMP,
tswtz_col   TIMESTAMP WITH TIME ZONE,
tswltz_col  TIMESTAMP WITH LOCAL TIME ZONE);

SQL> desc ttemporal
Name                 Type
-------------------- --------------
DATE_COL             DATE
INT_D2S_COL          INTERVAL DAY(2) TO SECOND(6)
INT_Y2M_COL          INTERVAL YEAR(2) TO MONTH
TS_COL               TIMESTAMP(6)
TSWTZ_COL            TIMESTAMP(6) WITH TIME ZONE
TSWLTZ_COL           TIMESTAMP(6) WITH LOCAL TIME
ZONE
```

```
CREATE TABLE tmisc (
rowid_col            ROWID,
urowid_col           UROWID,
anytype_col          ANYTYPE,
anydata_col          ANYDATA,
anydataset_col       ANYDATASET);

SQL> desc tmisc
Name                 Type
-------------------- --------------
ROWID_COL            ROWID
UROWID_COL           ROWID
ANYTYPE_COL          ANYTYPE
ANYDATA_COL          ANYDATA
ANYDATASET_COL       ANYDATASET
```

Column Defaults

- Provides a default value during DML if the SQL statement would leave the column NULL
- In 12c this can include a sequence generated surrogate key known as an identity column

```
CREATE TABLE customers (
cust_id NUMBER GENERATED ALWAYS AS IDENTITY
 INCREMENT BY 2
 START WITH 100
 MAXVALUE 110
 MINVALUE 100
 CYCLE
 CACHE 5
 NOORDER,
first_name VARCHAR2(25)
last_name  VARCHAR2(25));
```

```
SQL> desc dba_tab_cols
 Name                          Null?    Type
 ----------------------------- -------- --------------
 OWNER                         NOT NULL VARCHAR2(128)
 TABLE_NAME                    NOT NULL VARCHAR2(128)
 COLUMN_NAME                   NOT NULL VARCHAR2(128)
 DATA_TYPE                              VARCHAR2(128)
 DATA_TYPE_MOD                          VARCHAR2(3)
 DATA_TYPE_OWNER                        VARCHAR2(128)
 DATA_LENGTH                   NOT NULL NUMBER
 DATA_PRECISION                         NUMBER
 DATA_SCALE                             NUMBER
 NULLABLE                               VARCHAR2(1)
 COLUMN_ID                              NUMBER
 DEFAULT_LENGTH                         NUMBER
 DATA_DEFAULT                           LONG
 NUM_DISTINCT                           NUMBER
 LOW_VALUE                              RAW(2000)
 HIGH_VALUE                             RAW(2000)
 DENSITY                                NUMBER
 NUM_NULLS                              NUMBER
 NUM_BUCKETS                            NUMBER
 ...
 HIDDEN_COLUMN                          VARCHAR2(3)
 VIRTUAL_COLUMN                         VARCHAR2(3)
 ...
 COLLATION                              VARCHAR2(100)
 COLLATED_COLUMN_ID                     NUMBER
```

Hidden Columns (1:2)

- Hidden, aka Invisible, columns are columns that are invisible unless specifically named in a command or statement
- The following are examples where a hidden column will not be visible
  - SQL*Plus describe command
  - SELECT *

```
SQL> desc dba_tab_cols
 Name                          Null?    Type
 ----------------------------- -------- ------------
 OWNER                         NOT NULL VARCHAR2(128)
 TABLE_NAME                    NOT NULL VARCHAR2(128)
 COLUMN_NAME                   NOT NULL VARCHAR2(128)
 DATA_TYPE                              VARCHAR2(128)
 DATA_TYPE_MOD                          VARCHAR2(3)
 DATA_TYPE_OWNER                        VARCHAR2(128)
 DATA_LENGTH                   NOT NULL NUMBER
 DATA_PRECISION                         NUMBER
 DATA_SCALE                             NUMBER
 NULLABLE                               VARCHAR2(1)
 COLUMN_ID                              NUMBER
 DEFAULT_LENGTH                         NUMBER
 DATA_DEFAULT                           LONG
 NUM_DISTINCT                           NUMBER
 LOW_VALUE                              RAW(2000)
 HIGH_VALUE                             RAW(2000)
 DENSITY                                NUMBER
 NUM_NULLS                              NUMBER
 NUM_BUCKETS                            NUMBER
...
 HIDDEN_COLUMN                          VARCHAR2(3)
 VIRTUAL_COLUMN                         VARCHAR2(3)
...
 COLLATION                              VARCHAR2(100)
 COLLATED_COLUMN_ID                     NUMBER
```

Hidden Columns (2:2)

```
SQL> CREATE TABLE invis (
rid         NUMBER,
acct_active NUMBER(1) INVISIBLE,
acct_name   VARCHAR2(20));

Table created.

SQL> desc invis
 Name          Type
 ------------ --------------
 RID           NUMBER
 ACCT_NAME     VARCHAR2(20)

SQL> INSERT INTO invis
  2  (rid, acct_active, acct_name)
  3  VALUES
  4  (1, 0, 'Morgan');

1 row created.
```

```
SQL> SELECT * FROM invis;
RID ACCT_NAME
---------- --------------------
         1 Morgan

SQL> SELECT rid, acct_active, acct_name
FROM invis;
  2
       RID ACCT_ACTIVE ACCT_NAME
---------- ----------- --------------------
         1           0 Morgan
```

Virtual Columns

- Virtual columns are columns that do not physically exist from the standpoint of providing persistent data storage but rather virtualize the result of a function or expression
- Virtual columns can be used for constraints, indexing, and partitioning

```
CREATE TABLE vcol (
salary      NUMBER(8),
bonus       NUMBER(3),
total_comp NUMBER(10) AS (salary+bonus));

desc vcol

SELECT column_id, column_name, virtual_column
FROM user_tab_cols
WHERE table_name = 'VCOL'

INSERT INTO vcol
(salary, bonus)
VALUES
(1,2);
```

```
SQL> desc dba_tab_cols
 Name                          Null?    Type
 ----------------------------- -------- ------------
 OWNER                         NOT NULL VARCHAR2(128)
 TABLE_NAME                    NOT NULL VARCHAR2(128)
 COLUMN_NAME                   NOT NULL VARCHAR2(128)
 DATA_TYPE                              VARCHAR2(128)
 DATA_TYPE_MOD                          VARCHAR2(3)
 DATA_TYPE_OWNER                        VARCHAR2(128)
 DATA_LENGTH                   NOT NULL NUMBER
 DATA_PRECISION                         NUMBER
 DATA_SCALE                             NUMBER
 NULLABLE                               VARCHAR2(1)
 COLUMN_ID                              NUMBER
 DEFAULT_LENGTH                         NUMBER
 DATA_DEFAULT                           LONG
 NUM_DISTINCT                           NUMBER
 LOW_VALUE                              RAW(2000)
 HIGH_VALUE                             RAW(2000)
 DENSITY                                NUMBER
 NUM_NULLS                              NUMBER
 NUM_BUCKETS                            NUMBER
 ...
 HIDDEN_COLUMN                          VARCHAR2(3)
 VIRTUAL_COLUMN                         VARCHAR2(3)
 ...
 COLLATION                              VARCHAR2(100)
 COLLATED_COLUMN_ID                     NUMBER
```

Performance Tuning Tables

Table Tuning Rules

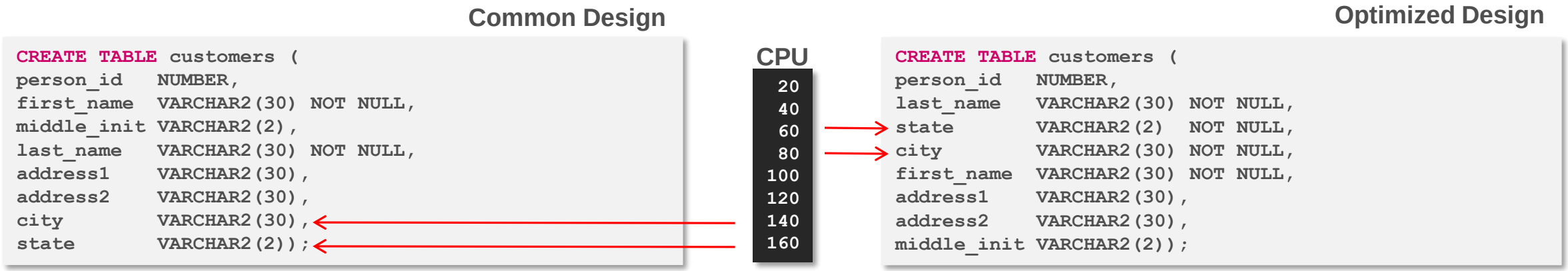
- Choose the optimal type of table and table storage options based on the way the table will be accessed
- It is almost always better to accept a small penalty during INSERTs and UPDATEs to gain an advantage for SELECT
- When designing tables ... thinking vertically ... not horizontally ... generally, the more columns the more issues
- Order columns in the table by the probability they will be accessed in a filter or join operation
- Do not make primary key columns NOT NULL
- Do not use the LONG or LONG RAW data types

Too Many Columns

- Oracle claims that a table can contain up to 1,000 columns: It is not true. No database can do 1,000 columns no matter what their marketing claims may be
- The maximum number of real table columns is 255
- Break the 255 barrier and optimizations such as advanced and hybrid columnar compression no longer work
- A 1,000 column table is actually four segments joined together behind the scenes just as a partitioned table appears to be a single segment but isn't
- Be suspicious of any table with more than 50 columns. At 100 columns it is time to take a break and re-read the Codd-Date rules on normalization
- Think vertically not horizontally
- Be very suspicious of any table with column names in the form "SPARE1", "SPARE2", "..."
- The more columns a table has the more cpu is required when accessing columns to the right (as the table is displayed in a SELECT * query ... or at the bottom if the table is displayed by a DESCribe)

Column Ordering

- Computers are not humans and tables are not paper forms
- CBO's column retrieval cost
  - Oracle stores columns in variable length format
  - Each row is parsed in order to retrieve one or more columns
  - Each subsequently parsed column introduces a cost of 20 cpu cycles regardless of whether it is of value or not
- These tables will be accessed by person_id or state: No one will ever put the address2 column into the WHERE clause as a filter ... they won't filter on middle initial either



Proof Column Ordering Matters in 12.1.0.2

```
CREATE TABLE read_test AS
SELECT *
FROM apex_040200.wwv_flow_page_plugs
WHERE rownum = 1;

SQL> explain plan for
  2  select * from read_test;

PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |     1 |   214K|     2   (0)| 00:00:01 |
|   1 |  TABLE ACCESS FULL| READ_TEST |     1 |   214K|     2   (0)| 00:00:01 |
------------------------------------------------------------------------------

-- fetch value from column 1
Final cost for query block SEL$1 (#0) - All Rows Plan:
  Best join order: 1
  Cost: 2.0002  Degree: 1  Card: 1.0000  Bytes: 13
  Resc: 2.0002  Resc_io: 2.0000  Resc_cpu: 7271
  Resp: 2.0002  Resp_io: 2.0000  Resc_cpu: 7271

-- fetch value from column 193
Final cost for query block SEL$1 (#0) - All Rows Plan:
  Best join order: 1
  Cost: 2.0003  Degree: 1  Card: 1.0000  Bytes: 2002
  Resc: 2.0003  Resc_io: 2.0000  Resc_cpu: 11111
  Resp: 2.0003  Resp_io: 2.0000  Resc_cpu: 11111
```

It Still Matters in 12.2.0.1

- Run the query while performing a 10053 Level 1 Trace

```
ALTER SESSION SET tracefile_identifier = 'test_plan2';

ALTER SESSION SET EVENTS '10053 trace name context forever, level 1';

SELECT * FROM  audsys.aud$unified WHERE inst_id = 2 AND ROWNUM = 1;

SELECT * FROM  audsys.aud$unified WHERE con_id = 9 AND ROWNUM = 1;

ALTER SESSION SET EVENTS '10053 trace name context OFF';
```

Column 1

```
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for AUD$UNIFIED[AUD$UNIFIED]
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = TABLE

 kkecdn: Single Table Predicate:"AUD$UNIFIED"."INST_ID"=2
 Column (#1): INST_ID(NUMBER)
   AvgLen: 22 NDV: 0 Nulls: 81 Density: 0.000000
 Estimated selectivity: 0.010000 , col: #1
 Table: AUD$UNIFIED  Alias: AUD$UNIFIED
   Card: Original: 81.000000  Rounded: 1  Computed: 0.000000  Non Adjusted: 0.000000
 Scan IO  Cost (Disk) =   4.000000
 Scan CPU Cost (Disk) =   85364.400000
 Cost of predicates:
   io = NOCOST, cpu = 50.000000, sel = 0.010000 flag = 2048  ("AUD$UNIFIED"."INST_ID"=2)
 Total Scan IO  Cost  =   4.000000 (scan (Disk))
                       + 0.000000 (io filter eval) (= 0.000000 (per row) * 81.000000 (#rows))
                      =   4.000000
  Total Scan CPU  Cost =   85364.400000 (scan (Disk))
                       + 4050.000000 (cpu filter eval) (= 50.000000 (per row) * 81.000000 (#rows))
                              =   89414.400000
 Access Path: TableScan
   Cost:  4.002375  Resp: 4.002375  Degree: 0
     Cost_io: 4.000000  Cost_cpu: 89414
     Resp_io: 4.000000  Resp_cpu: 89414
 Best:: AccessPath: TableScan
        Cost: 4.002375  Degree: 1  Resp: 4.002375  Card: 0.000000  Bytes: 0.000000
```

Column 101

```
SINGLE TABLE ACCESS PATH
  Single Table Cardinality Estimation for AUD$UNIFIED[AUD$UNIFIED]
  SPD: Return code in qosdDSDirSetup: NOCTX, estType = TABLE

 kkecdn: Single Table Predicate:"AUD$UNIFIED"."CON_ID"=9
 Column (#101): CON_ID(NUMBER)
   AvgLen: 22 NDV: 0 Nulls: 81 Density: 0.000000
 Estimated selectivity: 0.010000 , col: #101
 Table: AUD$UNIFIED  Alias: AUD$UNIFIED
   Card: Original: 81.000000  Rounded: 1  Computed: 0.000000  Non Adjusted: 0.000000
 Scan IO  Cost (Disk) =   4.000000
 Scan CPU Cost (Disk) =   245364.400000
 Cost of predicates:
   io = NOCOST, cpu = 50.000000, sel = 0.010000 flag = 2048  ("AUD$UNIFIED"."CON_ID"=9)
 Total Scan IO  Cost  =   4.000000 (scan (Disk))
                       + 0.000000 (io filter eval) (= 0.000000 (per row) * 81.000000 (#rows))
                      =   4.000000
  Total Scan CPU  Cost =   245364.400000 (scan (Disk))
                       + 4050.000000 (cpu filter eval) (= 50.000000 (per row) * 81.000000 (#rows))
                              =   249414.400000
 Access Path: TableScan
   Cost:  4.006624  Resp: 4.006624  Degree: 0
     Cost_io: 4.000000  Cost_cpu: 249414
     Resp_io: 4.000000  Resp_cpu: 249414
 Best:: AccessPath: TableScan
        Cost: 4.006624  Degree: 1  Resp: 4.006624  Card: 0.000000  Bytes: 0.000000
```

Column 101

How much are you willing to pay for accessing column 101?

Stop Making Primary Keys NOT NULL

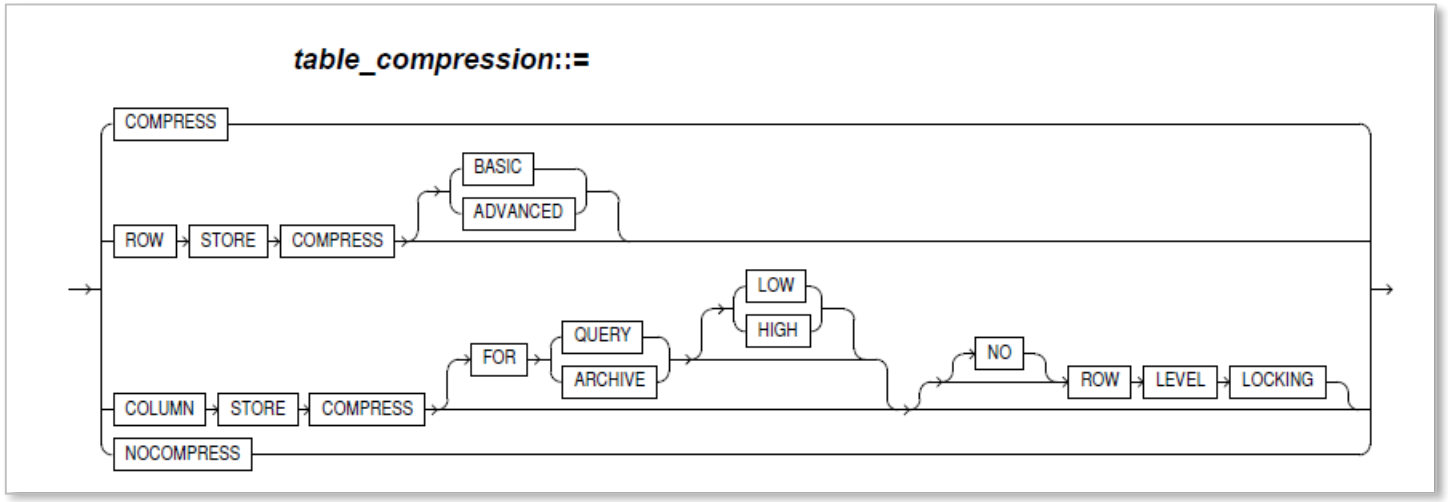
- the only value in creating a needless, and meaningless, check constraint?

```
SQL> CREATE TABLE t (
  2  recid NUMBER);

Table created.

SQL> desc t
 Name                          Null?    Type
 ----------------------------- -------- ----------------
 RECID                                  NUMBER

SQL> ALTER TABLE t
  2  ADD CONSTRAINT pk_t
  3  PRIMARY KEY (recid);

Table altered.

SQL> desc t
 Name                          Null?    Type
 ----------------------------- -------- ----------------
 RECID                         NOT NULL NUMBER
```

- Is that you get to waste a little more CPU than everyone else

Compressed Tables

- Table compression comes in three separate flavors
  - Basic ... no additional cost and including in the database license
  - Advanced ... requires the Advanced Compression Option license
  - Hybrid Columnar ... requires Exadata, ZFS, or Pillar Data Systems storage
- Compression does far more than save disk space ... it can greatly enhance performance by minimizing I/O
- Compression and decompression require cpu so carefully calibrate usage on systems that are cpu bound



DBMS_SPACE

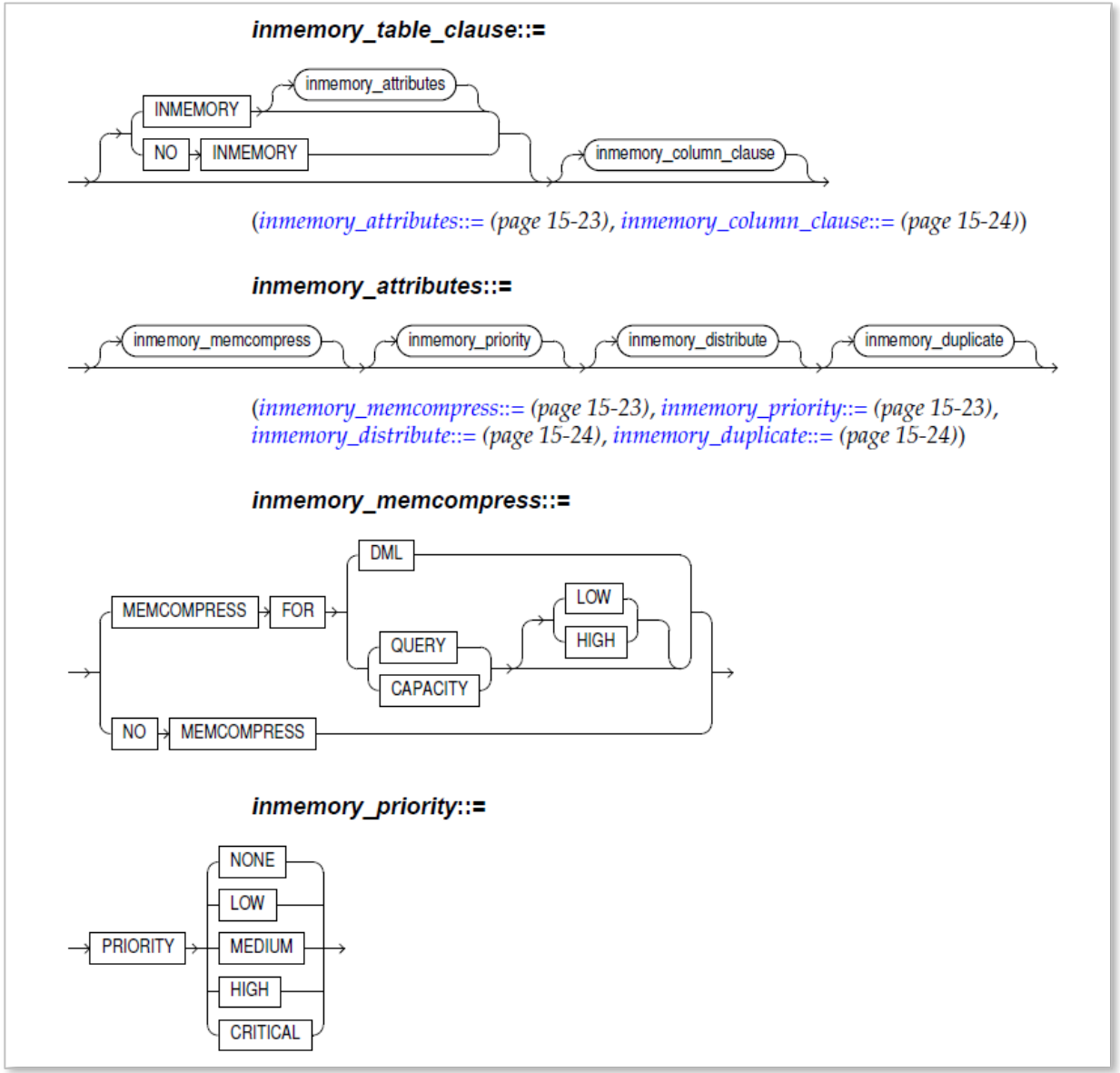
- This package provides segment space information not currently available through the standard views
- Use the CREATE_TABLE_COST and CREATE_INDEX_COST to determine the sizes of tables and indexes prior to their creation
- Use FREEBLOCKS and SPACE_USAGE to determine the free blocks consumed by tables, partitions, LOBs, clusters, and indexes
- Use OBJECT_SPACE_USAGE, SPACE_USAGE, and UNUSED_SPACE to learn how efficiently space allocated to an object is used
- Use VERIFY_SHRINK_CANDIDATE to determine whether an object can be shrunk to drop the high water ma…nance

```
dbms_space.create_table_cost(
tablespace_name IN  VARCHAR2,
avg_row_size    IN  NUMBER,
row_count       IN  NUMBER,
pct_free        IN  NUMBER,
used_bytes      OUT NUMBER,
alloc_bytes     OUT NUMBER);
```

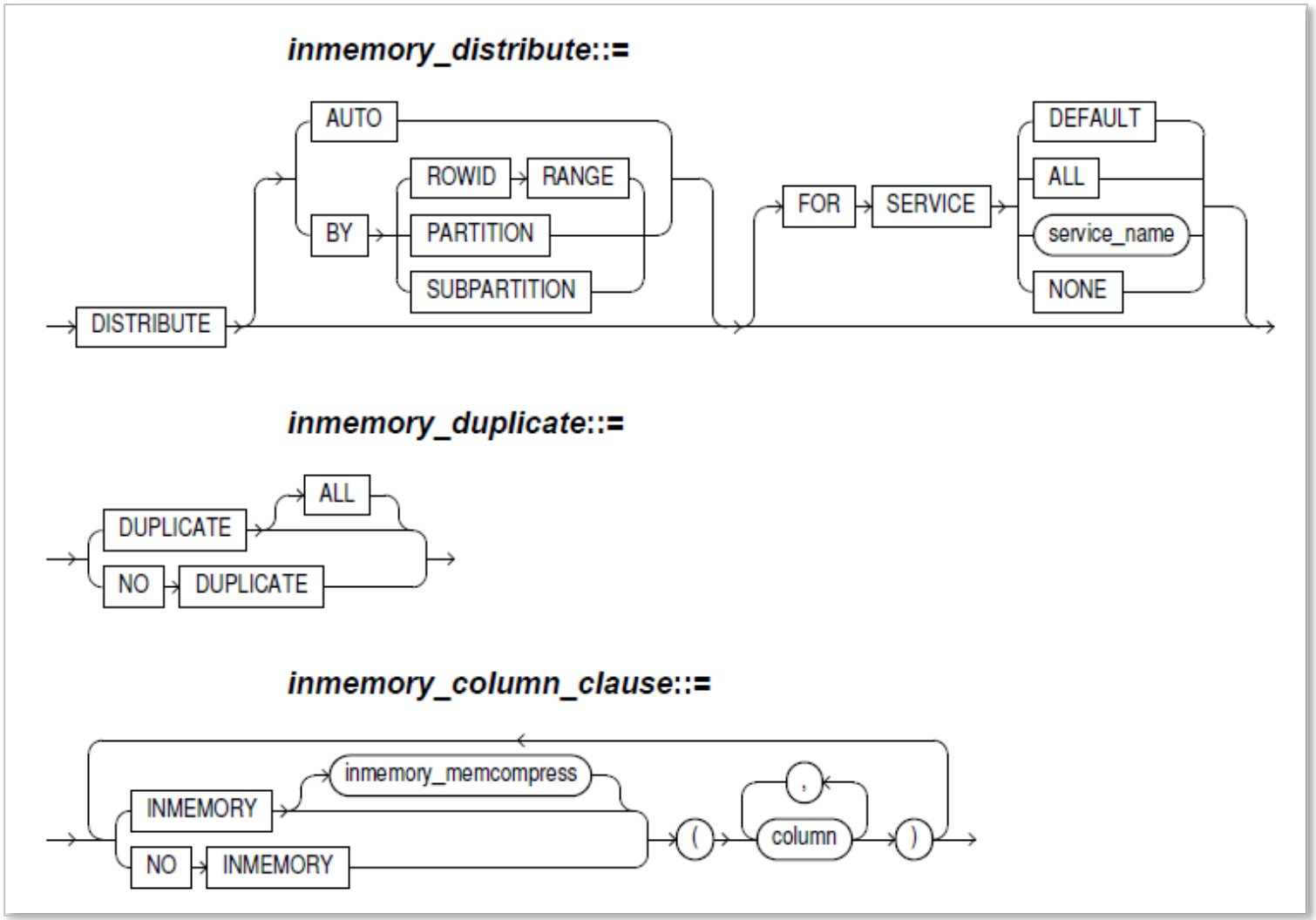
```
DECLARE
 ub NUMBER;
 ab NUMBER;
BEGIN
  dbms_space.create_table_cost('UWDATA',28,250000,0,ub,ab);

  dbms_output.put_line('Used Bytes: ' || TO_CHAR(ub));
  dbms_output.put_line('Alloc Bytes: ' || TO_CHAR(ab));
END;
/
Used Bytes: 7880704
Alloc Bytes: 8126464
```

In-Memory Clause (1:2)

- To greatly increase access speed for joins and filters consider the In-Memory option which covers
  - Full Database Caching
  - In-Memory Aggregation
  - In-Memory Column Store



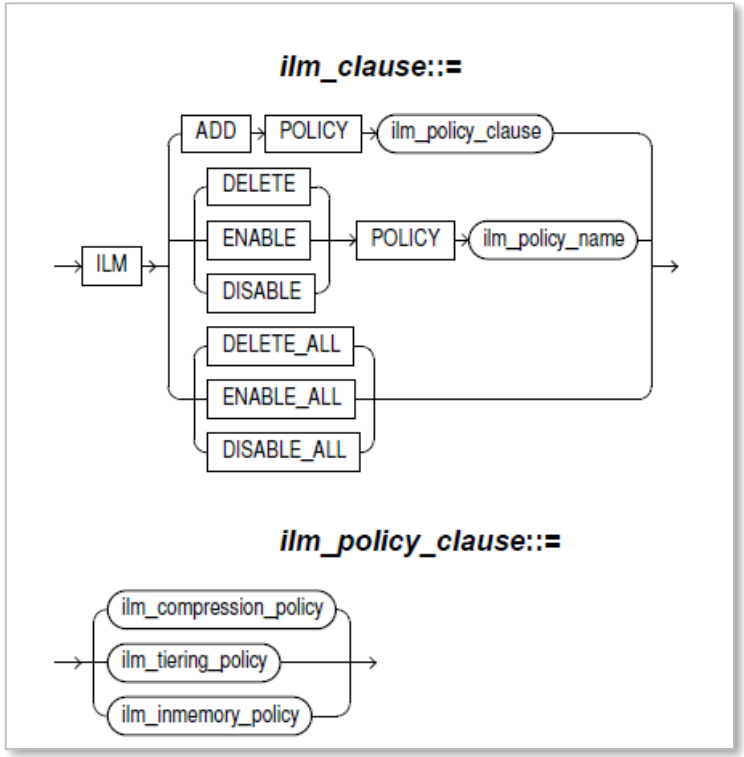
In-Memory Clause (2:2)



Automated Data Optimization (ADO) (1:2)

- ADO is implemented with two Oracle capabilities ... heat maps utilizing the DBMS_HEATMAP package and Integrated Lifecycle Management (ILM) utilizing the DBMS_ILM and DBMS_ILM_ADMIN packages
- ILM can be used to apply policies to table data that automatically apply management policies
  - Compression
  - Data Tiering
  - In-Memory Option

```
CREATE TABLE rowmove_test (
testcol  VARCHAR2(20))
ENABLE ROW MOVEMENT;

SELECT table_name, row_movement
FROM user_tables;

ALTER TABLE rowmove_test DISABLE ROW MOVEMENT;

SELECT table_name, row_movement
FROM user_tables;
```

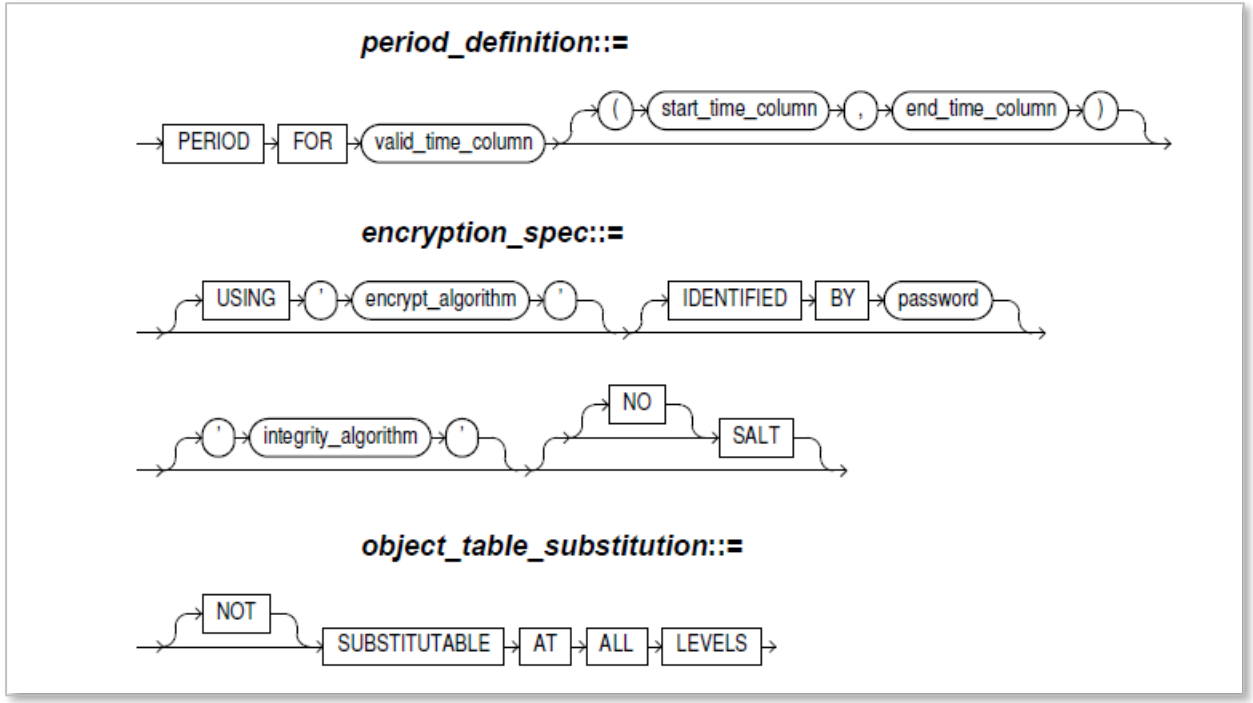


Automated Data Optimization (ADO) (2:2)

```
CREATE TABLESPACE tier1_ts DATAFILE '/u01/app/oracle/oradata/orcl12c/orcl/tier1.dbf' SIZE 10M AUTOEXTEND ON NEXT 5M;
CREATE TABLESPACE tier2_ts DATAFILE '/u01/app/oracle/oradata/orcl12c/orcl/tier2.dbf' SIZE 10M AUTOEXTEND ON NEXT 5M;
CREATE TABLESPACE tier3_ts DATAFILE '/u01/app/oracle/oradata/orcl12c/orcl/tier3.dbf' SIZE 50M AUTOEXTEND ON NEXT 5M;

CREATE TABLE order_hdr(
order_no NUMBER NOT NULL,
order_date DATE NOT NULL,
order_note VARCHAR2(500))
PARTITION BY RANGE (order_date)
(
PARTITION orders_2016_q3 VALUES LESS THAN (TO_DATE('01/10/2016', 'DD/MM/YYYY')) TABLESPACE tier3_ts,
PARTITION orders_2016_q4 VALUES LESS THAN (TO_DATE('01/01/2017', 'DD/MM/YYYY')) TABLESPACE tier3_ts,
PARTITION orders_2017_q1 VALUES LESS THAN (TO_DATE('01/04/2017', 'DD/MM/YYYY')) TABLESPACE tier2_ts
ILM ADD POLICY TIER TO tier3_ts READ ONLY SEGMENT AFTER 3 MONTHS OF NO ACCESS,
PARTITION orders_2017_q2 VALUES LESS THAN (TO_DATE('01/07/2017', 'DD/MM/YYYY')) TABLESPACE tier2_ts
ILM ADD POLICY TIER TO tier3_ts READ ONLY SEGMENT AFTER 3 MONTHS OF NO ACCESS,
PARTITION orders_2017_q3 VALUES LESS THAN (TO_DATE('01/10/2017', 'DD/MM/YYYY')) TABLESPACE tier1_ts
ILM ADD POLICY TIER TO tier2_ts READ ONLY SEGMENT AFTER 1 MONTHS OF NO ACCESS,
PARTITION orders_2017_q4 VALUES LESS THAN (TO_DATE('01/01/2018', 'DD/MM/YYYY')) TABLESPACE tier1_ts
ILM ADD POLICY TIER TO medium_storage_ts READ ONLY SEGMENT AFTER 2 MONTHS OF NO ACCESS)
ILM ADD POLICY ROW STORE COMPRESS BASIC SEGMENT AFTER 1 MONTHS OF NO ACCESS;
```

Temporal Validity

- Associates one or more valid time dimensions with a table and makes data visible depending on its time-based validity, as determined by the start and end dates or time stamps of the period for which a given record is considered valid



More You Should Know About Tables

Enabling Row Movement

- After a rowid is assigned to a row piece, the rowid can change in special
- circumstances if row movement is enabled
  - Partition key updates
  - Flashback Table operations
  - Shrink table operations

```
CREATE TABLE rowmove_test (
testcol  VARCHAR2(20))
ENABLE ROW MOVEMENT;

SELECT table_name, row_movement
FROM user_tables;

ALTER TABLE rowmove_test DISABLE ROW MOVEMENT;

SELECT table_name, row_movement
FROM user_tables;
```

Row Dependencies

- ORA_ROWSCN pseudocolumn
  - Returns, for each row, the conservative upper bound system change number (SCN) of the most recent change to the row
  - This is useful for determining approximately when a row was last updated
  - It is not absolutely precise, because Oracle tracks SCNs by transaction committed for the block in which the row resides

```
CREATE TABLE scnprec (
testcol VARCHAR2(20))
ROWDEPENDENCIES;

SELECT table_name, dependencies
FROM user_tables;

SELECT current_scn
FROM v$database;

INSERT INTO scnprec VALUES ('ABC');
COMMIT;

INSERT INTO scnprec VALUES ('DEF');
COMMIT;

INSERT INTO scnprec VALUES ('GHI');
COMMIT;

SELECT ORA_ROWSCN, ROWID, testcol
FROM scnprec;

ORA_ROWSCN ROWID              TESTCOL
---------- ------------------ --------
   6022658 AAATuZAABAAAZxhAAA ABC
   6022661 AAATuZAABAAAZxhAAB DEF
   6022670 AAATuZAABAAAZxhAAC GHI
```

DBMS_COMPARISON

- Use this fully documented and supported built-in package to compare data in tables, views and materialized views
- First available in 11.1.0.6
- Can converge (fix) data discrepancies automatically

```
dbms_comparison.create_comparison(
comparison_name     IN VARCHAR2, -- cannot contain spaces
schema_name         IN VARCHAR2,
object_name         IN VARCHAR2,
dblink_name         IN VARCHAR2,
index_schema_name   IN VARCHAR2 DEFAULT NULL,
index_name          IN VARCHAR2 DEFAULT NULL,
remote_schema_name  IN VARCHAR2 DEFAULT NULL,
remote_object_name  IN VARCHAR2 DEFAULT NULL,
comparison_mode     IN VARCHAR2 DEFAULT CMP_COMPARE_MODE_OBJECT,
column_list         IN VARCHAR2 DEFAULT '*',
scan_mode           IN VARCHAR2 DEFAULT CMP_SCAN_MODE_FULL,
scan_percent        IN NUMBER   DEFAULT NULL,
null_value          IN VARCHAR2 DEFAULT CMP_NULL_VALUE_DEF,
local_converge_tag  IN RAW      DEFAULT NULL,
remote_converge_tag IN RAW      DEFAULT NULL,
max_num_buckets     IN NUMBER   DEFAULT CMP_MAX_NUM_BUCKETS,
min_rows_in_bucket  IN NUMBER   DEFAULT CMP_MIN_ROWS_IN_BUCKET);
```

```
exec dbms_comparison.create_comparison(comparison_name=>'UWCompare',
                                       schema_name=>'SCOTT',
                                       object_name=>'DEPT',
                                       dblink_name=>NULL,
                                       remote_schema_name=>'ABC',
                                       remote_object_name=>'DEPT',
                                       scan_percent=>90);

DECLARE
 ct dbms_comparison.comparison_type;
BEGIN
  dbms_comparison.converge('UWCOMPARE', 2, ct,
  dbms_comparison.CMP_CONVERGE_LOCAL_WINS, TRUE);

  dbms_output.put_line(ct.scan_id);
  dbms_output.put_line(ct.loc_rows_merged);
  dbms_output.put_line(ct.rmt_rows_merged);
  dbms_output.put_line(ct.loc_rows_deleted);
  dbms_output.put_line(ct.rmt_rows_merged);
END;
/
```

DBMS_DBCOMP

- Assumes that a primary database and one or more Data Guard physical standby databases are deployed. The databases should be at least mounted or open before block comparison is run
- Logical standby databases, Far Sync instances, and cascaded standbys cannot be the target database
- First available in 12.2.0.1

```
dbms_dbcomp.dbcomp(
datafile   IN VARCHAR2,
outputfile IN VARCHAR2,
block_dump IN BOOLEAN := FALSE);
```

```
exec dbms_dbcomp.dbcomp('ALL', '/home/oracle/lost_write_check.txt', TRUE);

-- in a separate SQL*Plus session
SELECT target_desc, sofar, totalwork
FROM v$session_longops
WHERE opname = 'BlockCompare';

TARGET_DESC                    SO FAR  TOTALWORK
------------------------------ --------- ----------
Compared Blocks                367104     403142
Lost Writes                         0          0
```

DBMS_REDEFINITION

- Used to online redefine
  - Table columns
  - Table column names
  - Table column data types
  - Constraints
  - Indexes
  - Triggers

```
BEGIN
  dbms_redefinition.can_redef_table('PM', 'PRINT_ADS', dbms_redefinition.cons_use_pk);
  dbms_redefinition.redef_table(
    uname => 'PM',
    tname => 'PRINT_ADS',
    table_compression_type => 'ROW STORE COMPRESS ADVANCED',
    table_part_tablespace => 'NEWTBS',
    index_key_compression_type => 'COMPRESS 1',
    index_tablespace => 'NEWIDXTBS',
    lob_compression_type => 'COMPRESS HIGH',
    lob_tablespace => 'NEWLOBTBS',
    lob_store_as => 'SECUREFILE');
  dbms_redefinition.sync_interim_table('PM', 'PRINT_ADS', 'INT_PRINT_ADS');
  dbms_redefinition.finish_redef_table('PM', 'PRINT_ADS', 'INT_PRINT_ADS');
END;
/
```

Wrap Up

Conclusion

- How comfortable are you with your knowledge of CREATE TABLE?
- Probably feel pretty good about DROP TABLE
- But how about all of the ALTER TABLE syntax?
- The most important principle in creating optimal tables is the same as everything else in Oracle is "do the least work"
  - Minimize CPU utilization
  - Minimize I/O
    - Take the load off the storage array
    - Off the HBA cards
    - Off the SAN switch
  - Minimize network utilization
    - Bandwidth
    - Round Trips
  - Minimize serialization
- Stop using 7.3.4 best practices ... and start using the best practices for your current version

```
*
ERROR at line 1:
ORA-00028: your session has been killed
```

Thank You

Daniel A. Morgan
email: dmorgan@forsythe.com
mobile: +1 206-669-2949