The Oracle Database for SQL, Java, and .NET Developers

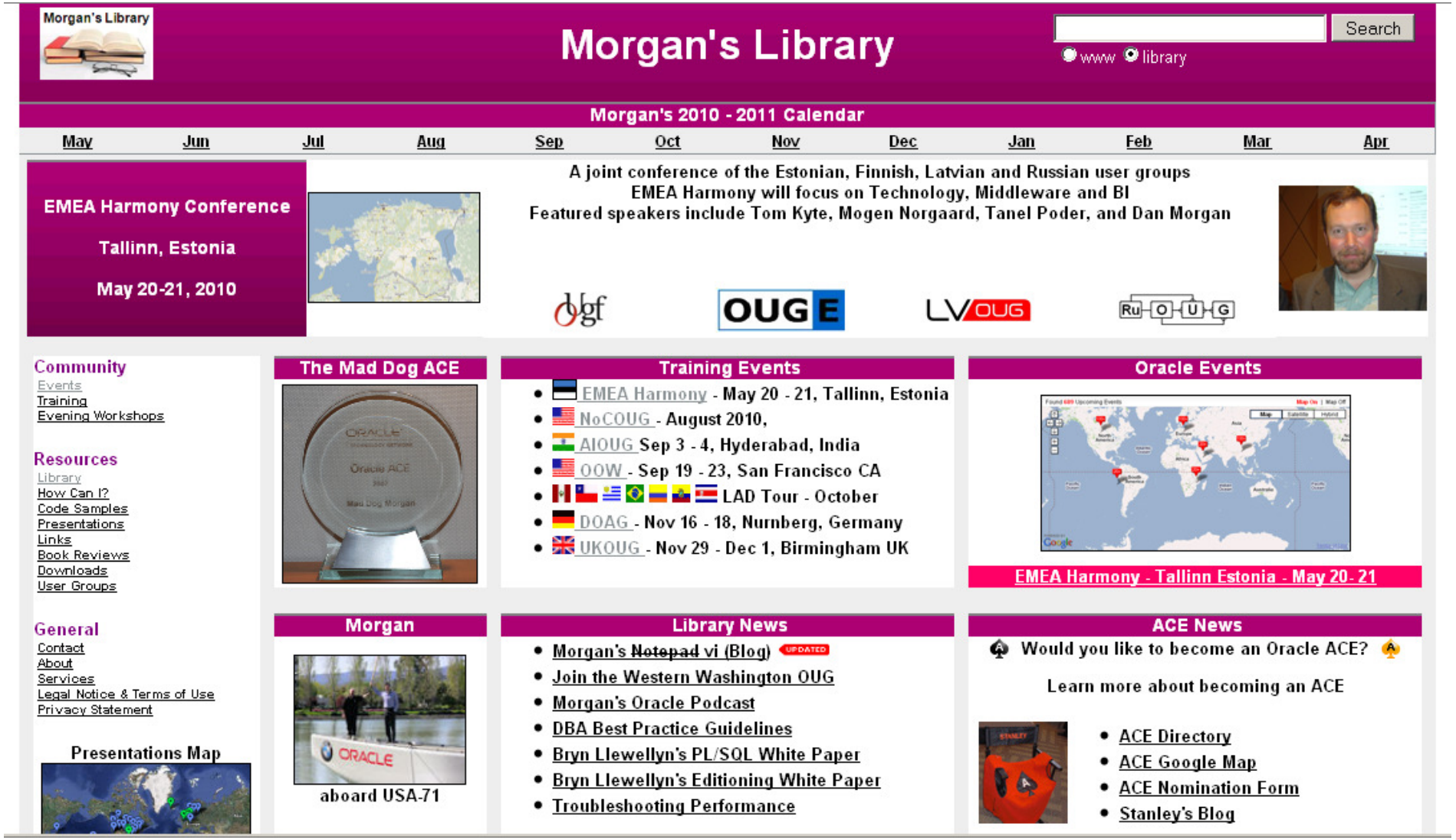
Daniel A. Morgan

Official Beta Site
ORACLE DATABASE 12c

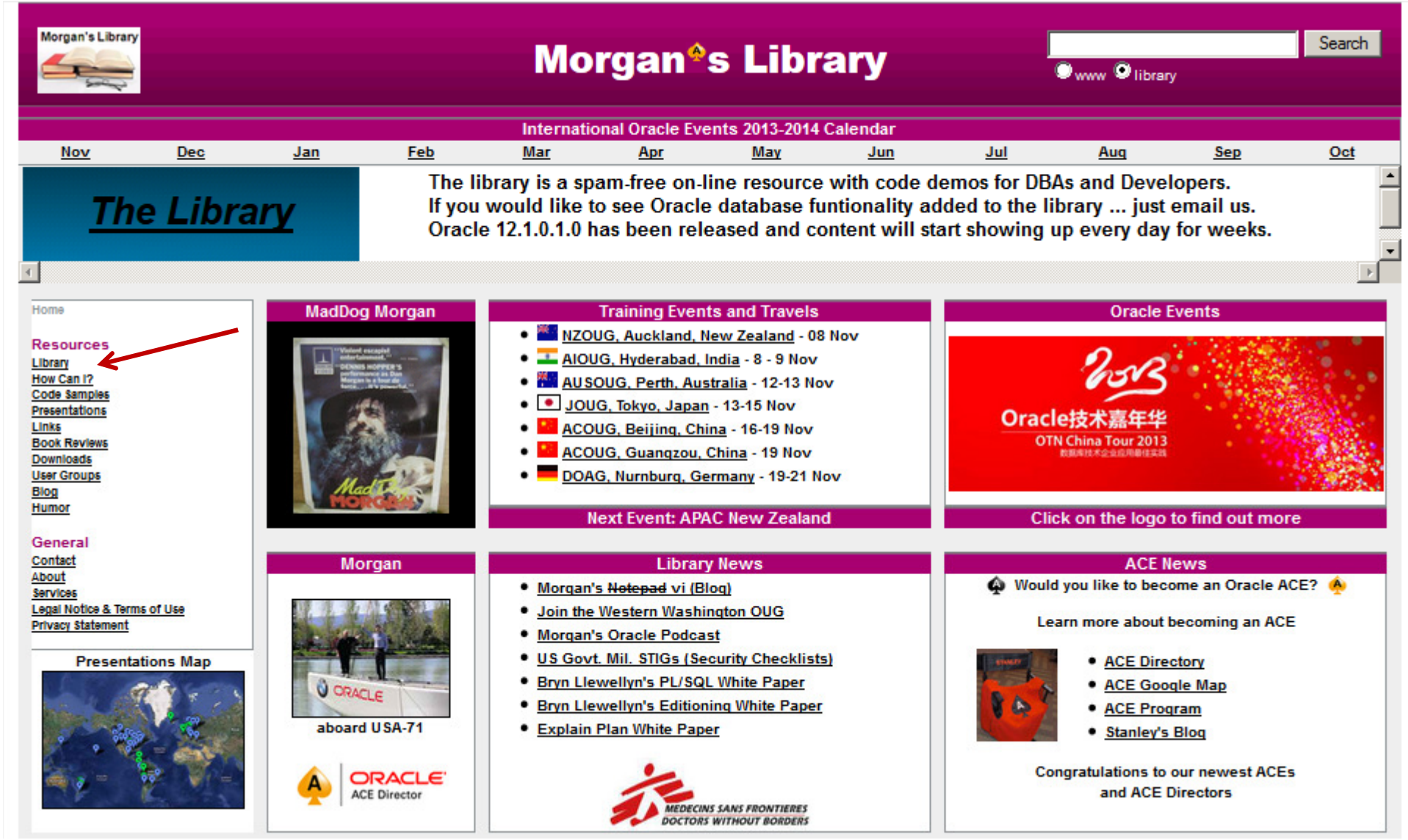
- Oracle ACE Director
- Consultant to Harvard University
- University of Washington Oracle Instructor, ret.
- The Morgan of Morgan's Library on the web
  - Executive Board Member: Vancouver OUG
  - Upcoming Presentations & Events
    - August: LAD Tour:
      Panama, Costa Rica, Peru, Ecuador
    - September: Oracle OpenWorld
    - October: Azerbaijan Oracle Users Group
    - October: Bulgarian Oracle Users Group
    - November APAC Tour:
      Thailand, New Zealand
  - 10g, 11g, 12c Beta Tester

Mad Dog Morgan

Morgan's Library: www.morganslibrary.org

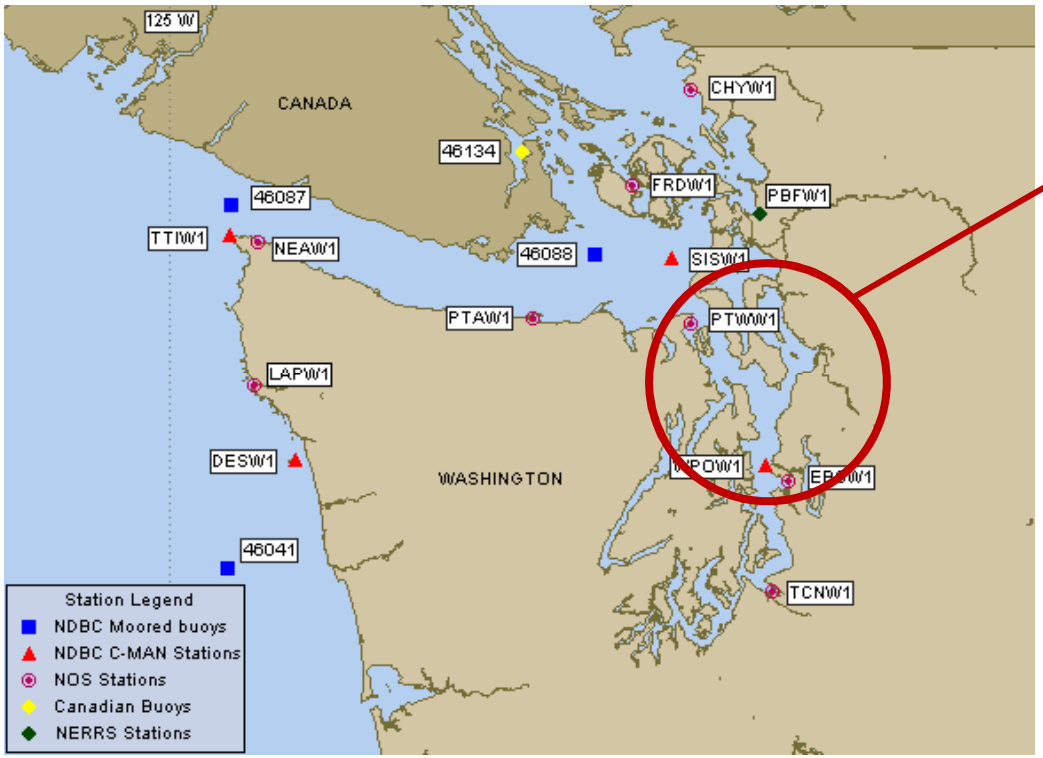


Morgan's Library: www.morganslibrary.org

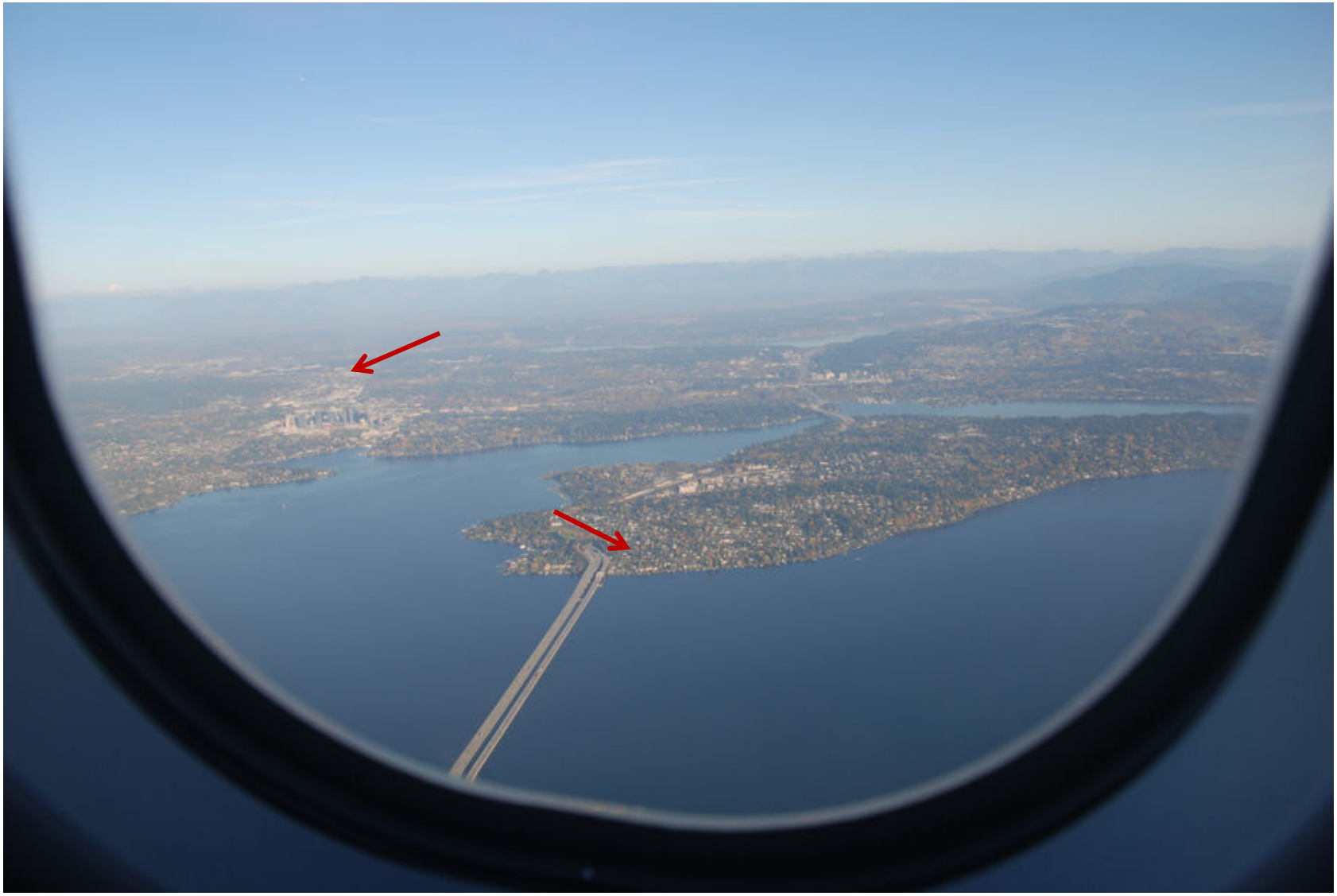


cd $MORGAN_HOME

cd $MORGAN_HOME



cd $MORGAN_HOME



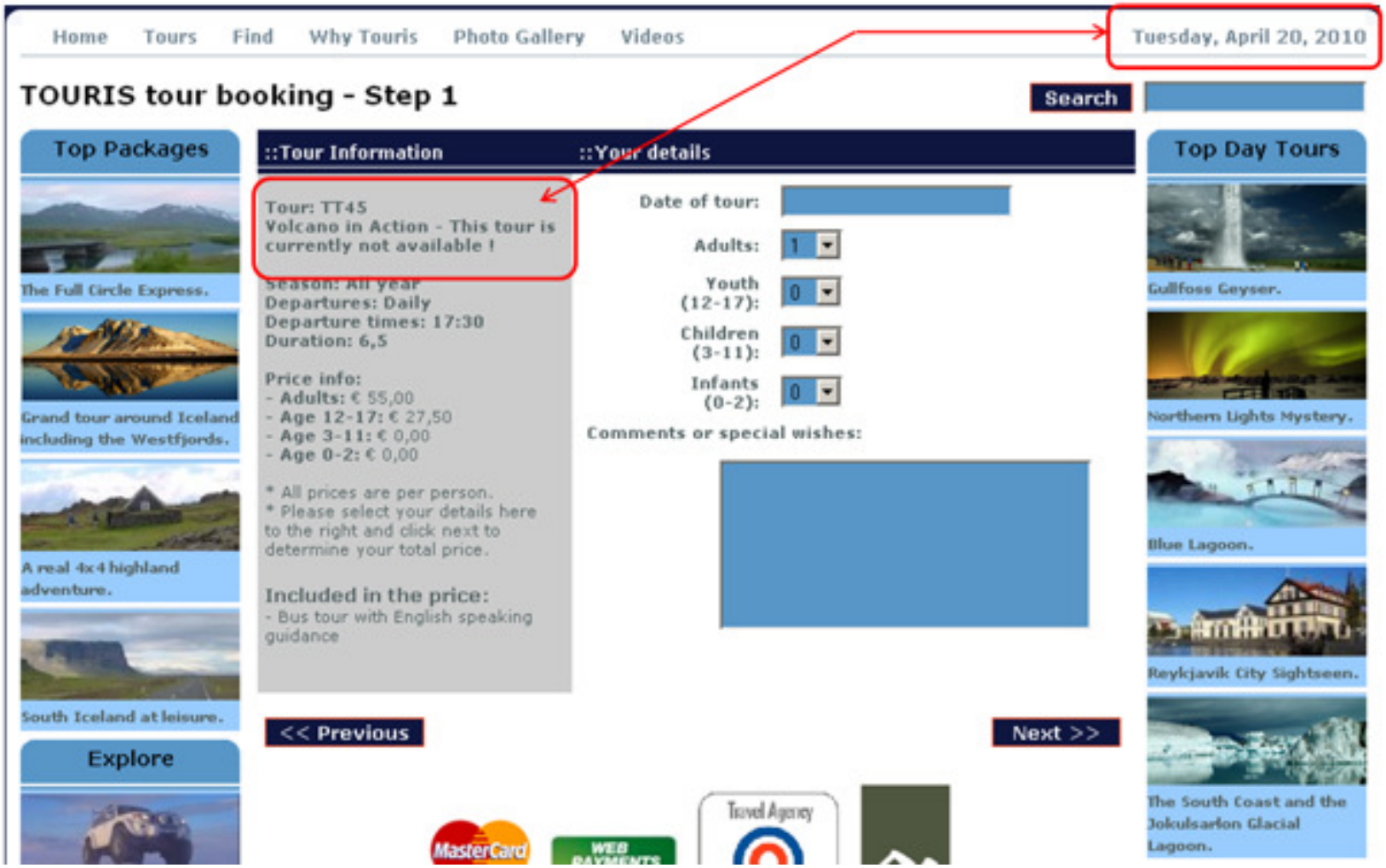
cd $MORGAN_BASE/San_Francisco

Travel Log: My Sled Meets Larry's

Travel Log: Amsterdam and Cuzco

Travel Log: Thank You SAS

Fun things to do in Iceland ... Take a tour



Greenland: April 2010

What Greenland is supposed to look like in April

Greenland: April 2010

Clearly global warming is a myth

Syllabus

- Introductions
- Your questions
- Business rules and data integrity
- What is the Oracle database?
  - Multi-Version Concurrency Control
- Basic Object Types
  - Partitioning
- Access Optimization
- Connection Pooling
- Common and Costly Mistakes
- RAC
- References

Introductions

- Your name
- What tools (Java, C# Dot Net, PHP, Ruby)?
- What databases have you worked with?
- What don't you like about databases the most?

The theory is that we break every hour

...

your job is to remind me if I forget

Approach new experiences with care

"Your Questions" from ACOUG (1 of 3)

- Will US spying on other countries affect government decisions on using the Oracle's database
  - The future of Oracle
- Education and Career Development
  - Effective learning methods for Oracle
  - How best to learn Java and Oracle
  - Professional skills required
  - DBA career planning
  - Career development
  - Recommended books for database learning?

"Your Questions" from ACOUG (2 of 3)

- Architecture
  - System Architecture
  - Product Architecture
  - Technical details of garbage collection
  - Product structure
- Performance
  - SQL tuning
  - How to design the structure of my database to optimize for speed
  - Thoughts about diagnosis of poor DB performance
  - Explain Plan and AWR Reports
  - Design and performance tuning

"Your Questions" from ACOUG (3 of 3)

- High Availability
  - High availability, data protection, and disaster recovery for enterprise data
  - Replication technology
  - Backup recover case
- "The Cloud"
  - Deployment of Cloud Computing
  - Large Data and the Cloud
- New Oracle 12c Features
- Oracle Database's development
- SQL*Plus
- Java's Future
- JDBC

The First Question

What should you do when your subjective opinion does not correspond to objective reality?

The purpose of IT is to support the core business of the organization

It's the economy stupid

~ US President Bill Clinton

Never base an argument about technology

When it can be about money

The Second Question

So ...
where should business rules be enforced?

and ...
where does code belong?

Business Rules: Are these countries?

468
?
??
AD - Andora
AE - United Arab Emirates
AF - Afghanistan
AG - Antigua
AI - Albania
AM - Armenia
AN - Netherland Antilles
AO - Angola
AR - Argentina
AS - American Samoa
AT - Austria
AU - Australia
AW - Aruba
BB - Barbados
BE - Belgium
BG - Bulgaria
BH - Bahrain
BI - Burundi
BM - Bermuda
BN - Brunei
BO - Bolivia
BR - Brazil
BS - Bahamas
BT - Bhutan
BZ - Belize
CA - Canada
CC - Cocos Islands
|CG - Congo
CH - Switzerland
CK - Cook Islands
CL - Chile
CN - China
CO - Colombia
CR - Costa Rica
CU - Cuba
CV - Cape Verde
CX - Christmas Island
CY - Cyprus
CZ - Czech Republic
DE - Germany
DK - Denmark
DM - Dominica
DO - Dominican Republic
EC - Ecuador
EG - Egypt
EH
ES - Spain
ET - Ethiopia
FI - Finland
FJ - Fiji
FK - Falkland Islands
FM - Micronesia
FO - Faroe Islands
FR - France
FX
GB - United Kingdom
GD - Grenada
GF - French Guiana
GH - Ghana
GI - Gibraltar
GL - Greenland
GP - Guadalupe
GR - Greece
GS - South Georgia
GT - Guatemala
GU - Guam
GY - Guyana
HK - Hong Kong
HN - Honduras
HR - Croatia
HT - Haiti
HU - Hungary
ID - Indonesia
IE - Ireland
IL - Israel
IN - India
IO - British Ocean Terr.
IQ - Iraq
IR - Iran
IS - Iceland
IT - Italy
JM - Jamaica
JO - Jordan
JP
KE - Kenya
KM - Comoros
KN - Saint Kitts
KP
KR - Korea
KW - Kuwait
KY - Cayman Islands
LA - Laos
LB - Lebanon
LC - Saint Lucia
LI - Liechtenstein
LK - Sri Lanka
LT - Lithuania
LU - Luxembourg
LV - Latvia
MA - Morocco
MC - Monaco
MH - Marshall Islands
MN - Mongolia
MO - Macau
MP - Marianas Islands
MQ - Martinique
MS - Montserrat
MT - Malta
MU - Mauritius
MV - Maldives
MX - Mexico
MY - Malaysia
NC - New Caledonia
NF - Norfolk Island
NG - Nigeria
NI - Nicaragua
NL - Netherlands
NO - Norway
NP - Nepal
NR - Nauru
NU - Niue
NZ - New Zealand
OM - Oman
PA - Panama
PE - Peru
PF - French Polynesia
PG - Papua New Guinea
PH - Philippines
PK - Pakistan
PL - Poland
PR - Puerto Rico
PT - Portugal
PW - Palau
PY - Paraguay
QA - Qatar
RE - Reunion Island
RO - Romania
RS - Serbia
RU - Russia
SA - Saudi Arabia
SB - Solomon Islands
SC - Seychelles
SE - Sweden
SG - Singapore
SH - St. Helena
SI - Slovenia
SK - Slovakia
SM - San Marino
SN - Senegal
SO - Somalia
SR - Suriname
SV - El Salvador
TC - Turks & Caicos Is.
TD - Chad
TG - Togo
TH - Thailand
TO - Tonga
TR - Turkey
TT - Trinidad
TV - Tuvalu
TW - Taiwan
TZ - Tanzania
UA - Ukraine
UG - Uganda
UK - United Kingdom
UM - US Minor Islands
US - United States
USA
UY - Uruguay
VA - Vatican City
VC - Saint Vincent
VE - Venezuela
VG - Virgin Islands (UK)
VI - Virgin Islands (US)
VN - Vietnam
VU- Vanuatu
WF - Wallis & Futuna
WS - Samoa
YE - Yemen
YT - Mayotte
ZA - South Africa
ZM - Zambia
ZW - Zimbabwe
okus

135 of 145
93% valid

Business Rules: Are these states in the U.S.?

Hamburg	*	....	16 Dov Fridman st., Ramat Gan	ACCRA 00233 GH
La.	*******	.....	1603	ACCRA-GH
am Main	*AE	/ Baba Gankyo Tokeiten	16200	ACHAIA
"string'Misquoted"'	*AP	0	166/38 Kavanagh st. soutbank	ACT
#5 jungang-dong jung gu	|*Utah	00	1704-4	ACT 2600
#Chuoukensashitsu	*Westvl	0000	1705	ADMIRALTY
(Ge)	,, Shizuoka	00000	17171 Solna	AE
(107) 266-0415	,Hampshire	000000	1730	AG
(1426) BUENOS AIRES	,Higasi-yodogawa-ku	00000000	1730004	AGHIA PARASKEVI
(Alicante)	,Koyama-chou-minam	0000000000	17456	AGIOS ATHANASIOS
(Almeria)	,Lancashire	000000000000000000000000	178 Bunsee Tr. Penal Rock Road	AGIOS TYXONAS
(BA)	-	00153	1780	AGUADA
(BO)	- - - -	00258 82 3024370	1800	AGUAS CLARAS
(BZ) South Tirol	- Minas Gerais	00354	1816	AHARNESAHOME
(Barcelona)	- Seleccione un estado / provi	0064	1816GT	AICHI
(CO)	- Select a -	00725	182-0006	AICHI-KEN
(Cadiz)	- Select a /province -	00750	189, Sinseol-dong, Dongdaemun	AISNE - PICARDIE
(Europe)	- Slectionnez un tat / provi	00802	193-3 Okutsubakidai, Yuwa	AJMAN
(Fi)	- Whlen Sie einen Staat / Pro	00803	2	AK
(GALICIA)	--	00821	2 Grange Hall, Mountmellick,	AKERSHUS
(Ge)	-- Select --	00987	2-2-1 Jinnan, Shibuya-ku	AL
(Gerona)	---	01	2018 Antwerpen	AL NAHDA1
(Girona) Catalunya	----	010-5263-5839	20200	AL QUOZ
(Guam is U.S. Territory)	-----	01051053719	2083385597	AL.
(Hamburg)	------	01254581344	21,307 flushcombe rd,	ALABAMA
(Le Grand-Sacconex)	--------	013-0501	21-307 flushcombe rd,blacktown	ALAVA
MI)	----------	01900	21018 Varese	ASDF
(MO) Modena	----------	03278816639300	2123	ASEAN
(Milano)	--Select a --	039731850	21307 flushcombe rd,blacktown	ASIA
(Non-US)	-A2-machiku-office	04	3	ASIA (65-8163-3902)
(Province de Lige)	-Bijutsukandoori-#707	0474	3-63sakaemachi toyota aichi	ASIA 886-932385379
(SO)	-Ohizumi, Nerima-ku, Tokyo	05	30 Pitt Street Sydney	ASIA PACIFIC
(Salerno)	-aomorihachinohesetsubi-CS	0524453936	301ho, 1631-23, In Heon Dong	ASIAN PACIFIC
(South Korea)	-gun, Hokkaido	06	304	ASIKRKS011GYEONGSAN
(South Pacific)	-kabushikikaisya	061 397549	3118	ASKIM
(TO)	-kitsuke Kaisei-tsushou	07	3137	ASKM
(TV)	-konsaruteingu-kabushikikaisha	08400	3175	ASTURIAS
(Tarragona) Provincia	-kougyou Chuukousou	09129 CAGLIARI (Sardinia)	32 Engr Regt	ASahikawa-shi.Hokkaido
(non-us	.	09366	32-400 Myslenice	ATEATHENS
(via DENMARK)	..	09593	3242354	ATHENS ATTIKI
(via Denmark)	...	0981	32618	
		099-4405	33	

2 of 210
0.1% valid

Business Rules: Are these cities in Canada?

CITY	STATE
*Cahewood	*Utah
APO	AA
116 street	AB
116 street, N.W.	AB
1700 XI Tech, 734	AB
2668 32 Bstreet	AB
AB	AB
Calgar	AB
Calgar7	AB
Calgaray	AB
Calgaru	AB
Calgary	AB
Calgary AB	AB
Calgary Alberta	AB
Calgary Canada	AB
Calgary NW	AB
Calgary Nw	AB
Calgary SW	AB
Calgary T3B5Y4	AB
Calgary,	AB
Calgary, AB	AB
Calgary, Alberta	AB
Calgary, Alta.	AB
Calgary, T2K 1B7	AB
Calgay	AB
Calgery	AB
Calgry	AB
Ed	AB
Edmonotn	AB
Edmontn	AB
Edmontno	AB
Edmonton	AB
Edmonton,	AB
Edmonton, AB	AB
Edmonton, Alberta	AB
Edmonton, Alberta	AB
Edmontont	AB
Edmotnon	AB
Edmoton	AB
Edomonton	AB

CITY	STATE
f	AB
fa	AB
fairview	AB
fhg	AB
foisy	AB
forl lkjlkdjf	AB
gfhjdjj	AB
ghgjh	AB
gibbons	AB
gjhgjhg	AB
gleichen	AB
goie	AB
x	AB
xdf	AB
Legal	ALTA
Calgary	ALberta
Kindersley	AP
Calgary	Ab
,Vancouver	
,.	BC
.	BC
0SOYOOS	BC
1	BC
100 mile House	BC
108 Mile Ranch	BC
108 Mile house	BC
12345 45st	BC
150 Mile House	BC
28 Ave	BC
73A Avenue	BC
C?hilliwack	BC
Castlegar B.C.	BC
Canada	BD
Ottawa	CA
San Diego	CA
San Jose	CA
Toronto	CA
montreal	CA
Edmonton	CANADA
Darien	CT

CITY	STATE
Seattle	Halifax
wpg	MB
Halifax	MD
Windsor	MI
Tour Du Lac	Mont-Tremblant
Berlington	ND
Pasadena	Newfoiundland
Bridgewater	NoveScotia
columbus	OH
138 Cove Crescent	ON
Mississauga	ON
Miss	ON
Missaissaug	ON
Missasauga	ON
Missassauga	ON
Missauga	ON
Missisauga	ON
Mississaga	ON
Mississagua	ON
Mississaua	ON
Mississauga	ON
Mississauga Ontario	ON
Mississauga,	ON
Mississauga, ON	ON
Mississauga, On	ON
Mississauga, Ontario	ON
Mississauga`	ON
Mississauge	ON
Mississaugua	ON
Mississayuga	ON
Mississigauga	ON
Mississigua	ON
Mississippi Stn	ON
Mississisauga	ON
Mississsauga	ON
Mississuaga	ON
Misssissauga	ON
Misssissuga	ON
Missuaga	ON
ASDASD	PE

CITY	STATE
Seattle	Halifax
Maniwaki	PQ
Maniwaki	QC
ontreal	Quebeck
montreal	Quebeck
Oak Hill	VA
Victoria, BC	Victoria, BC
Collingwood	WA
Point Roberts	WA
surrey	WA
zxcv	SK

3 of 131
2.3% valid

Business Rules: Are these valid dates?

TABLE_NAME	COLUMN_NAME	MIN_DATE
ALTERNATE_PAYMENTS	EXPIRE_DATE	01-JAN-0001 00:00:00
AUTO_REORDER	UPDATE_DATE	01-JAN-0001 00:00:00
PERSONAL_MSGS	DATE_EXPIRES	20-SEP-0001 12:00:00
EMEDALERT_INFO	END_DATE_PURCHASE	20-JUN-0002 12:00:00
PERSONAL_MSGS	DATE_START	20-JUN-0007 12:00:00
RELATIONINSTANCE	START_DATE	03-FEB-0008 00:01:00
GIFT_CERTIFICATES_TRANSACTIONS	WAG_TRANSACTION_DATE	10-DEC-0012 16:02:02
EMEDALERT_INFO	START_DATE_PURCHASE	20-APR-0020 10:00:00
PERSONAL_MSG_MEMBER_FLAGS	DATE_EMAIL_SENT	20-JUN-0020 12:00:00
CONTACT_LENS_PATIENT_INFO	BIRTH_DATE	23-APR-0088 00:00:00
LUX_CONTACT_LENS_PATIENT_INFO	BIRTH_DATE	23-FEB-1792 00:00:00
PHARMACY_PATIENTS	BIRTHDATE	01-JAN-1800 00:00:00
INSURANCE_CARDS	BIRTHDATE	09-JUN-1883 00:00:00
MEMBER_INFO	FIRST_ORDER_SHIPPED	30-DEC-1899 00:00:00
RX_LINEITEMS	HISTORY_DATE_FILLED	30-DEC-1899 00:00:00
MV_BOUNCED_EMAIL_LIST	LAST_UNSUB_DATE	30-DEC-1899 00:00:00
EMAIL_SUBSCRIPTIONS	NEVER_CREATED	30-DEC-1899 00:00:00
SUBORDERS	SHIPPED_DATE	30-DEC-1899 00:00:00
SUBORDERS	UPDATE_DATE	30-DEC-1899 00:00:00
SURVEYS	DATE_END	01-JAN-1900 00:00:00
SURVEYS	DATE_START	01-JAN-1900 00:00:00
CONTACT_LENS_INFO	EXPIRE_DATE	01-JAN-1900 00:00:00
CONTACT_LENS_INFO	REPLENISHMENT_DATE	01-JAN-1900 00:00:00
LUX_CONTACT_LENS_INFO	REPLENISHMENT_DATE	01-JAN-1900 00:00:00
RIGHTNOW_MEMBER_TRANSFER	ROW_SENT	01-JAN-1900 00:00:00
RIGHTNOW_ORDER_TRANSFER	ROW_SENT	01-JAN-1900 00:00:00
TD_MASTER	TD_EXPIRE_DATE	01-JAN-1900 00:00:00
VA_PROFILES	NEXT_ORDER_DATE	30-MAR-1900 00:00:00
ALTERNATE_PAYMENTS	CREATE_DATE	01-JAN-1909 00:00:00
ALTERNATE_PAYMENTS_11212009	CREATE_DATE	01-JAN-1909 00:00:00
RIGHTNOW_MEMBER_TRANSFER	DATE_LAST_VISITED	01-JAN-1909 00:00:00
MEMBERS	LAST_VISITED	01-JAN-1909 00:00:00
RA_PRESCRIPTIONS	LAST_DISPENSED_EXPIRES	01-FEB-1920 00:00:00
MS_NDCPRICE	EFFDATE	01-FEB-1979 00:00:00
MAS_LONG_DRUG	DEACTIVATE_DATE	01-JUL-1985 00:00:00
MAS_LONG_DRUG_03092012	DEACTIVATE_DATE	01-JUL-1985 00:00:00
MAS_LONG_DRUG_060109	DEACTIVATE_DATE	01-JUL-1985 00:00:00
MAS_LONG_DRUG_OLD	DEACTIVATE_DATE	01-JUL-1985 00:00:00
MAS_LONG_DRUG	REACTIVATE_DATE	16-SEP-1986 00:00:00

1 of 39
2.6% valid

Business Rules: Are These Valid Dates?

TABLE_NAME	COLUMN_NAME	MAX_DATE
RA_LINEITEMS	PICKUP_DATE	10-NOV-2013 11:00:00
CHECK_PAY_DETAIL	CHECK_DATE	26-AUG-2014 00:00:00
AUTO_REORDER_ITEM	START_DATE	07-OCT-2017 00:00:00
AUTO_REORDER_ITEM	LAST_REORDER_DATE	10-APR-2020 00:00:00
RA_PRESCRIPTIONS	LAST_PICKUP_DATE	05-FEB-2030 12:00:00
CONTACT_LENS_PATIENT_INFO	BIRTH_DATE	01-JAN-2035 00:00:00
SALON_LICENSES	LICENSE_EXPIRATION	29-DEC-2035 00:00:00
MAS_LONG_DRUG	DEACTIVATE_DATE	15-APR-2044 00:00:00
PERSONAL_MSGS	DATE_EXPIRES	01-JAN-2050 00:00:00
SUBORDERS	SHIPPED_DATE	20-DEC-2058 00:00:47
SURVEYS	DATE_END	11-APR-2099 00:00:00
RELATIONINSTANCE	START_DATE	07-AUG-2099 16:48:00
STORECATS	UPDATE_DATE	11-NOV-2099 00:00:00
TD_MASTER	TD_RELEASE_DATE	23-DEC-2099 00:00:00
TD_MASTER	TD_EXPIRE_DATE	31-DEC-2099 00:00:00
PROMOTIONS	DATE_EXPIRES	31-DEC-2099 23:59:00
INSURANCE_CARDS	BIRTHDATE	30-DEC-2100 00:00:00
GIFT_CERTIFICATES	DATE_EXPIRES	16-SEP-2111 00:00:00
ALTERNATE_PAYMENTS	EXPIRE_DATE	16-SEP-2111 00:00:00
PRODUCTS	AVAILABLE_DATE	20-AUG-2154 00:00:00
MEMBER_INFO	FIRST_ORDER_SHIPPED	15-JAN-2203 00:00:00
REFILL_TOO_SOON	READY_TO_FILL_DATE	01-MAR-3011 00:00:00
RELATIONINSTANCE	END_DATE	01-JUN-3011 00:00:00
LUX_CONTACT_LENS_INFO	EXPIRE_DATE	01-JUN-6470 00:00:00
VA_PROFILES	NEXT_ORDER_DATE	18-MAR-7483 00:00:00
RITEAID_RX	SCHEDULED_PICKUP_DATE	01-JAN-9865 14:00:00
CONTACT_LENS_INFO	EXPIRE_DATE	01-JUN-9865 00:00:00
PHARMACY_PATIENTS	BIRTHDATE	31-DEC-9999 00:00:00
RASCRUB_AGENCY	EXPN_DTE	31-DEC-9999 00:00:00
RASCRUB_CARDTEXT	EXPN_DTE	31-DEC-9999 00:00:00
RASCRUB_ENRLFORM	EXPN_DTE	31-DEC-9999 00:00:00
RA_PRESCRIPTIONS	LAST_DISPENSED_EXPIRES	31-DEC-9999 00:00:00

2 of 32
6.3% valid

Where business rules belong

- Everywhere
- Is it redundant?
  - Of course: you have to code some things twice
- Is it necessary?
  - Of course: not every database access will be through the application
  - Batch jobs will not see logic in the front-end application
  - Imports will not see logic in the front-end
  - DBAs will not be constrainted by logic in the front-end
- The most efficient SQL is SQL embedded in PL/SQL packages
  - Pass through parameters ... not code

What is the Oracle Database?

Origins

Ingres	Begun as a research project at UC Berkeley: 1970 Several companies used the Ingres source code to produce products. The most successful was a company named Relational Technology, Inc. (RTI) founded in 1980 by Michael Stonebraker and another Berkeley professor, Lawrence A. Rowe. RTI was renamed Ingres Corporation in the late 1980s.
Informix	Followed the 1995 purchase of Illustra, Informix concentrated on object-relational database (O-R) technology. Illustra was written by ex-Postgres team members and led by database pioneer Michael Stonebraker. Illustra included DataBlades that allowed new data types and features to be included in the basic server as options such as time series, spatial and multimedia data. Informix integrated Illustra's O-R mapping and DataBlades into the 7.x OnLine product, resulting in Informix Universal Server (IUS), or more generally, Version 9.
Sybase	Robert Epstein, the chief programmer on the Ingres project while he was at Berkeley, formed Britton-Lee along with other students from the Ingres Project, Paula Hawthorne and Mike Ubell; they were joined later by Eric Allman. Later, they founded Sybase.
SQL Server	Sybase's product line was licensed to Microsoft in 1992, who re-branded it as SQL Server. The product lines were separated in the late 1990s.

How Consistency Works

- System Change Numbers (SCN)
- Locking
  - Reads never block writes
  - Writes never block reads
  - Lock escalation is impossible
  - The number of possible row locks in Oracle is equal to the total number of possible rows in all tables in all schemas in any size database
- Undo Tablespace
- Databases are not stand-alone systems. Everything you do must assume tens, hundreds, or thousands of other processes are accessing the same objects and the same data at the same time

MultiVersion Concurrency Control (MVCC)

- Not present in:
  - DB2
  - Informix
  - Sybase
- Also present in:
  - Berkeley DB
  - Firebird
  - InnoDB engine (for mySQL)
  - InterBase
  - PostgreSQL
  - SQL Server 2005 (and above but not the default)

Data Dictionary

- X$
  - Linked lists in memory
- V$ and GV$
  - Views built on X$ objects
- CDB_
  - New in 12c ... everything in every container
- DBA_
  - In 11gR2 and before everything in the database
  - In 12cR1 and above everything in the current container
- ALL_
  - Everything your schema owns or has permission to access
- USER_
  - Everything your schema created and owns

What is Oracle Really?

- The most efficient mapping tool ever invented
- Maps physical locations
  - Hard disk blocks

```
SELECT file_name, user_blocks, user_bytes
FROM dba_data_files;
```

- Maps logical names to physical locations
  - Tablespaces

```
SELECT file_name, tablespace_name
FROM dba_data_files;
ORDER BY 1;

SELECT tablespace_name
FROM dba_tablespaces
ORDER BY 1;
```

What is Oracle Really?

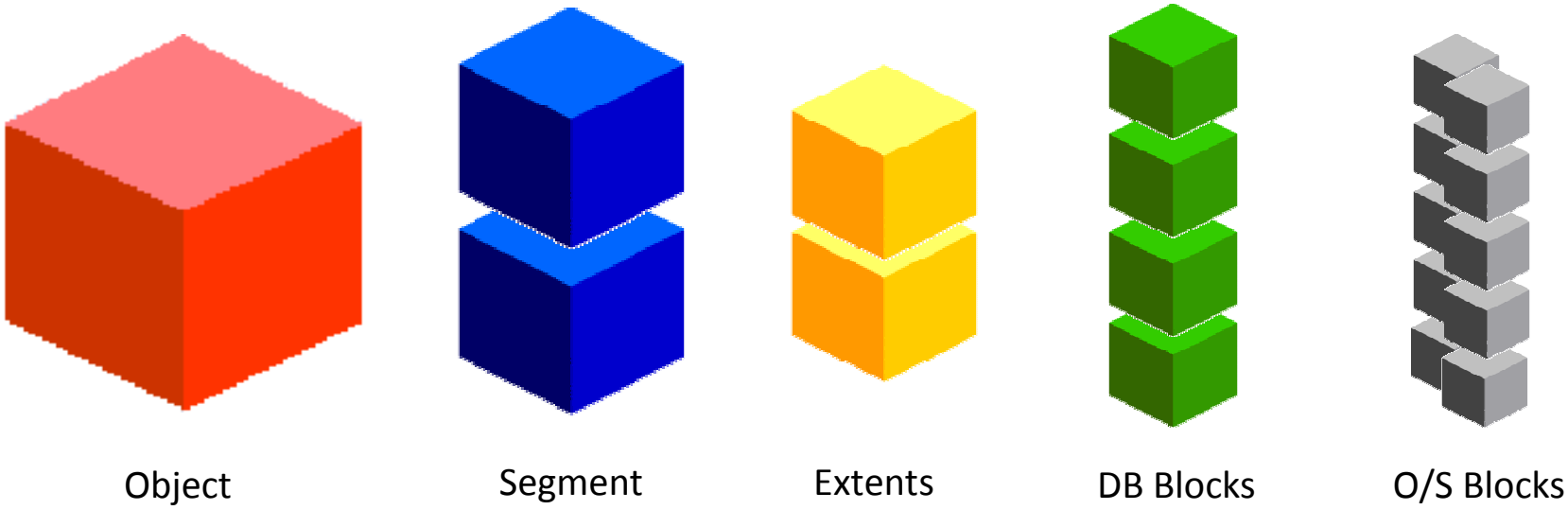
- Maps logical locations where data is stored
  - Segments
  - Tables
  - Indexes
  - Extents
  - Blocks

```
desc dba_extents

SELECT owner, table_name,
tablespace_name
FROM dba_tables
WHERE rownum < 21
ORDER BY 2;
```

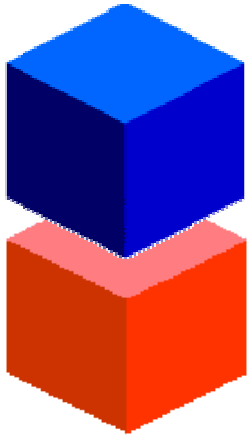
Segments, Extents, and Blocks

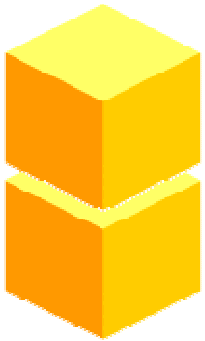
- Segments exist within a tablespace
- Segments are a collection of extents
- Extents are a collection of data blocks
- Data blocks are mapped to disk blocks

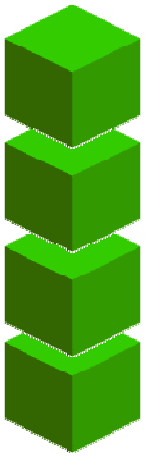


Segments, Extents, and Blocks

Normal Table

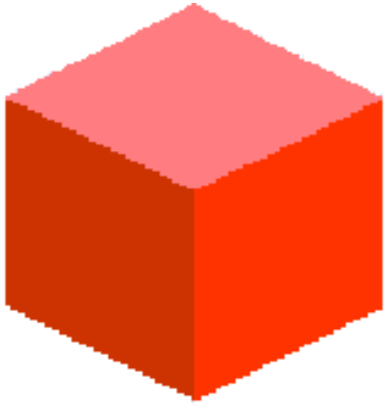


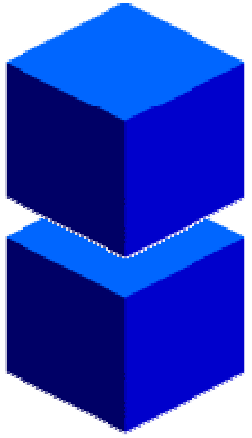


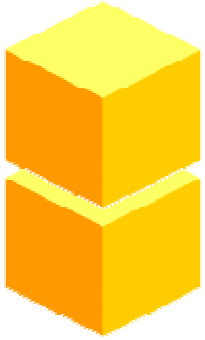


Segment=Object | Extents | DB Blocks | O/S Blocks

Partitioned Table







Object <> Segment | Extents | DB Blocks | O/S Blocks

What is it?

- Maps network connections
  - LISTENER.ORA
  - SQLNET.ORA
  - TNSNAMES.ORA
- Maps processes in memory

```
SELECT DISTINCT program
FROM v$session
ORDER BY 1;
```

What is it?

- Maps memory pools

```
desc v$rowcache

desc v$subcache

desc v$library_cache_memory

SELECT component, current_size
FROM
v$memory_dynamic_components;

SELECT category, allocated,
used, max_allocated
FROM v$process_memory;
```

What is it?

- Stores pointers to data locations (like GPS coordinates)
  - Indexes

- A robust application development framework accessible through multiple languages
  - SQL
  - PL/SQL and Dynamic SQL
  - Java
  - C and C++
  - Cobol
  - Fortran

How does it work?

How?

- Transactions
  - ACID
    - Atomicity: Each transaction is "ALL" or "NOTHING"
    - Consistency: Each transaction brings the database to a valid state
    - Isolation: Concurrent transactions are serially consistent
    - Durability: A committed transaction is permanent (crash protected)
  - System Change Numbers
  - Redo
  - Undo
  - Commit, Savepoints, and Rollback
  - Flashback
- Locking
  - Blocking
  - Deadlocks

Basic Objects

Object Types

```
SELECT DISTINCT object_type
FROM dba_objects
ORDER BY 1;

SELECT DISTINCT segment_type
FROM dba_segments
ORDER BY 1;
```

Segments

- Tables
  - Heap Tables
  - External Tables
  - Global Temporary Tables
  - Compressed Tables
  - XML Tables
  - Partitioned Tables
  - LOBs and LOB Partition
  - Clusters
- Materialized Views
- Clusters

Segments

- Indexes
  - B*Tree
  - Bitmap
  - Bitmap Join
  - Compressed
  - Descending
  - Function Based
  - Invisible
  - Reverse Key
  - XML
- Index Organized Tables

Partitioning

What is Partitioning?

- Tables are a logical, not a physical, construct
- A partition, or subpartition is a map
- Just as a tablespace maps many datafile
- Just as a regular table maps many blocks
- A partitioned table maps many partitions
- A partition maps many subpartitions

Why Partitioning?

- 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 (DBA)
- Licensing

Data Dictionary Views

- dba_tab_partitions
- dba_tab_subpartitions
- dba_tab_cols
- dba_lob_partitions
- dba_lob_subpartitions
- dba_subpartition_templates
- dba_subpart_col_statistics
- dba_subpart_histograms
- dba_subpart_key_columns
- dba_mview_detail_partitions
- dba_mview_detail_subpartitions
- dba_ind_partitions
- dba_ind_subpartitions

Partitioning in 8i

- Simple
  - Partition by Hash
  - Partition by Range
- Composite
  - Range - Hash

Partition by Hash

```
CREATE TABLE hash_part (
prof_history_id  NUMBER(10),
person_id        NUMBER(10) NOT NULL,
organization_id  NUMBER(10) NOT NULL,
record_date      DATE NOT NULL,
prof_hist_comments VARCHAR2(2000))
PARTITION BY HASH (prof_history_id)
PARTITIONS 3
STORE IN (part1, part2, part3);
```

Partition by Range - Alpha

```
CREATE TABLE students (
student_id NUMBER(6),
student_fn VARCHAR2(25),
student_ln VARCHAR2(25),
PRIMARY KEY (student_id))
PARTITION BY RANGE (student_ln) (
PARTITION student_ae VALUES LESS THAN ('F%')
TABLESPACE part1,
PARTITION student_fl VALUES LESS THAN ('M%')
TABLESPACE part2,
PARTITION student_mr VALUES LESS THAN ('S%')
TABLESPACE part3,
PARTITION student_sz VALUES LESS THAN (MAXVALUE)
TABLESPACE part4);
```

Querying Partitioned Tables

```
SELECT * FROM students;

SELECT * FROM students PARTITION(student_ae);
```

Explain Planning Partitioned Tables

```
SELECT * FROM students;

EXPLAIN PLAN FOR
SELECT * FROM students;

SELECT * FROM TABLE(dbms_xplan.display);

SELECT * FROM students
WHERE student_ln LIKE 'E%';

EXPLAIN PLAN FOR
SELECT * FROM students
WHERE student_ln LIKE 'E%';

SELECT * FROM TABLE(dbms_xplan.display);

SELECT * FROM students PARTITION(student_ae);

EXPLAIN PLAN FOR
SELECT * FROM students PARTITION(student_ae);

SELECT * FROM TABLE(dbms_xplan.display);
```

Partition by Range - Date

```
CREATE TABLE range_part (
prof_history_id NUMBER(10),
person_id       NUMBER(10) NOT NULL,
organization_id NUMBER(10) NOT NULL,
record_date     DATE NOT NULL,
ph_comments     VARCHAR2(200))
PARTITION BY RANGE (record_date) (
PARTITION yr0 VALUES LESS THAN (TO_DATE('01-JAN-2000','DD-MON-YYYY'))
TABLESPACE part1,
PARTITION yr1 VALUES LESS THAN (TO_DATE('01-JAN-2001','DD-MON-YYYY'))
TABLESPACE part2,
PARTITION yr2 VALUES LESS THAN (TO_DATE('01-JAN-2002','DD-MON-YYYY'))
TABLESPACE part3,
PARTITION yr9 VALUES LESS THAN (MAXVALUE) TABLESPACE part4);
```

Partition by Range - Numeric

```
CREATE TABLE students (
student_id NUMBER(6),
student_fn VARCHAR2(25),
student_ln VARCHAR2(25),
PRIMARY KEY (student_id))
PARTITION BY RANGE (student_id) (
PARTITION student_neg VALUES LESS THAN (0)
TABLESPACE part1,
PARTITION student_one VALUES LESS THAN (1000)
TABLESPACE part2,
PARTITION student_two VALUES LESS THAN (2000)
TABLESPACE part3,
PARTITION student_max VALUES LESS THAN (MAXVALUE)
TABLESPACE part4);
```

Composite Partitioning by Range-Hash

```
CREATE TABLE composite_rng_hash (
cust_id     NUMBER(10),
cust_name   VARCHAR2(25),
amount_sold NUMBER(10,2),
time_id     DATE)
PARTITION BY RANGE (time_id)
SUBPARTITION BY HASH (cust_id)
SUBPARTITION TEMPLATE(
SUBPARTITION sp1 TABLESPACE part1,
SUBPARTITION sp2 TABLESPACE part2,
SUBPARTITION sp3 TABLESPACE part3,
SUBPARTITION sp4 TABLESPACE part4) (
PARTITION sales_pre06
VALUES LESS THAN (TO_DATE('01/01/2006','DD/MM/YYYY')),
PARTITION sales_2006
VALUES LESS THAN (TO_DATE('01/01/2007','DD/MM/YYYY')),
PARTITION sales_2007
VALUES LESS THAN (TO_DATE('01/01/2008','DD/MM/YYYY')),
PARTITION sales_2008
VALUES LESS THAN (TO_DATE('01/01/2009','DD/MM/YYYY')),
PARTITION sales_2009
VALUES LESS THAN(MAXVALUE));
```

Querying Subpartitioned Tables

```
SELECT * FROM composite_rng_hash;

SELECT * FROM composite_rng_hash PARTITION(SALES_FUTURE);

SELECT * FROM composite_rng_hash SUBPARTITION(SALES_FUTURE);
```

Partitioning in 9i

- 9.0.#
  - Partition by List
- 9.2.0
  - Range – List Composite

Partition by List

```
CREATE TABLE list_part (
deptno          NUMBER(10),
deptname        VARCHAR2(20),
quarterly_sales NUMBER(10,2),
state           VARCHAR2(2))
PARTITION BY LIST (state) (
PARTITION q1_northwest VALUES ('OR', 'WA')
TABLESPACE part1,
PARTITION q1_southwest VALUES ('AZ', 'CA', 'NM')
TABLESPACE part1,
PARTITION q1_northeast VALUES ('NY', 'VT', 'NJ')
TABLESPACE part4,
PARTITION q1_southeast VALUES ('FL', 'GA')
TABLESPACE part2,
PARTITION q1_northcent VALUES ('MN', 'WI')
TABLESPACE part3,
PARTITION q1_southcent VALUES ('OK', 'TX')
TABLESPACE part3);
```

Composite Partition by Range-List

```
CREATE TABLE composite_rng_list (
cust_id     NUMBER(10),
cust_name   VARCHAR2(25),
cust_state  VARCHAR2(2),
time_id     DATE)
PARTITION BY RANGE (time_id)
SUBPARTITION BY LIST (cust_state)
SUBPARTITION TEMPLATE(
SUBPARTITION west VALUES ('OR', 'WA') TABLESPACE part1,
SUBPARTITION east VALUES ('NY', 'CT') TABLESPACE part2,
SUBPARTITION cent VALUES ('OK', 'TX') TABLESPACE part3) (
PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
PARTITION future VALUES LESS THAN(MAXVALUE));
```

Partitioning in 10g

- 10gR1
  - Nothing New
- 10gR2
  - Nothing New

New Partitioning Options in 11gR1

- Partition by Interval
- Partition by Reference
- Partition by System
- Partition by Virtual Column
- New Composite Partitioning Options
  - List – Hash
  - List – List
  - List – Range
  - Range – Range

Partitioning by Interval - Numeric

```
CREATE TABLE interval_part1 (
person_id   NUMBER(5) NOT NULL,
first_name  VARCHAR2(30),
last_name   VARCHAR2(30))
PARTITION BY RANGE (person_id)
INTERVAL (100) STORE IN (uwdata, users) (
PARTITION p1 VALUES LESS THAN (101));


CREATE TABLE interval_part2 (
person_id   NUMBER(5) NOT NULL,
first_name  VARCHAR2(30),
last_name   VARCHAR2(30))
PARTITION BY RANGE (person_id)
INTERVAL (100) (
PARTITION p1 VALUES LESS THAN (101))
TABLESPACE uwdata;
```

Partitioning by Interval - Date

```
CREATE TABLE interval_date (
person_id   NUMBER(5) NOT NULL,
last_name   VARCHAR2(30),
dob         DATE)
PARTITION BY RANGE (dob)
INTERVAL (NUMTOYMINTERVAL(1,'MONTH'))
STORE IN (uwdata) (
PARTITION p1 VALUES LESS THAN (TO_DATE('2008-03-15','YYYY-MM-DD')));
```

Partition by Reference

```
CREATE TABLE ref_parent (
table_name   VARCHAR2(30),
order_date   DATE,
num_rows     NUMBER)
PARTITION BY RANGE (num_rows) (
PARTITION num_rows1 VALUES LESS THAN (100) TABLESPACE part1,
PARTITION num_rows2 VALUES LESS THAN (1000) TABLESPACE part2,
PARTITION num_rows3 VALUES LESS THAN (10000) TABLESPACE part3,
PARTITION num_rows4 VALUES LESS THAN (MAXVALUE) TABLESPACE part4);


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);
```

Partitioning by System

```
CREATE TABLE syst_part (
tx_id   NUMBER(5),
begdate DATE)
PARTITION BY SYSTEM (
PARTITION p1 TABLESPACE part1,
PARTITION p2 TABLESPACE part2,
PARTITION p3 TABLESPACE part3);
```

Partition by Virtual Column

```
CREATE TABLE vcol_part (
tx_id   NUMBER(5),
begdate DATE,
enddate DATE,
staylen NUMBER(5) AS (enddate-begdate))
PARTITION BY RANGE (staylen)
INTERVAL (10) STORE IN (uwdata) (
PARTITION p1 VALUES LESS THAN (11))
TABLESPACE uwdata;
```

Composite Partitioning by List-Hash

```
CREATE TABLE composite_list_hash (
cust_id     NUMBER(10),
cust_name   VARCHAR2(25),
cust_state  VARCHAR2(2),
time_id     DATE)
PARTITION BY LIST (cust_state)
SUBPARTITION BY HASH (cust_id)
SUBPARTITION TEMPLATE(
SUBPARTITION sp1 TABLESPACE part1,
SUBPARTITION sp2 TABLESPACE part2,
SUBPARTITION sp3 TABLESPACE part3,
SUBPARTITION sp4 TABLESPACE part4) (
PARTITION west VALUES ('OR', 'WA'),
PARTITION east VALUES ('NY', 'CT'),
PARTITION cent VALUES ('IL', 'MN'));
```

Composite Partitioning by List-List

```
CREATE TABLE composite_list_list (
cust_id     NUMBER(10),
cust_name   VARCHAR2(25),
cust_state  VARCHAR2(2),
time_id     DATE)
PARTITION BY LIST (cust_state)
SUBPARTITION BY LIST (cust_id)
SUBPARTITION TEMPLATE(
SUBPARTITION beg VALUES (1,3,5) TABLESPACE part1,
SUBPARTITION mid VALUES (2,4,6) TABLESPACE part2,
SUBPARTITION end VALUES (7,8,9,0) TABLESPACE part3) (
PARTITION west VALUES ('OR', 'WA'),
PARTITION east VALUES ('NY', 'CT'),
PARTITION cent VALUES ('IL', 'MN'));
```

Composite Partitioning by List-Range

```
CREATE TABLE composite_list_rng (
cust_id     NUMBER(10),
cust_name   VARCHAR2(25),
cust_state  VARCHAR2(2),
time_id     DATE)
PARTITION BY LIST (cust_state)
SUBPARTITION BY RANGE (cust_id)
SUBPARTITION TEMPLATE(
SUBPARTITION original VALUES LESS THAN (1001) TABLESPACE part1,
SUBPARTITION acquired VALUES LESS THAN (8001) TABLESPACE part2,
SUBPARTITION recent VALUES LESS THAN (MAXVALUE) TABLESPACE part3) (
PARTITION west VALUES ('OR', 'WA'),
PARTITION east VALUES ('NY', 'CT'),
PARTITION cent VALUES ('IL', 'MN'));
```

Composite Partitioning by Range-Range

```
CREATE TABLE composite_rng_rng (
cust_id     NUMBER(10),
cust_name   VARCHAR2(25),
cust_state  VARCHAR2(2),
time_id     DATE)
PARTITION BY RANGE (time_id)
SUBPARTITION BY RANGE (cust_id)
SUBPARTITION TEMPLATE(
SUBPARTITION original VALUES LESS THAN (1001) TABLESPACE part1,
SUBPARTITION acquired VALUES LESS THAN (8001) TABLESPACE part2,
SUBPARTITION recent VALUES LESS THAN (MAXVALUE) TABLESPACE part3) (
PARTITION per1 VALUES LESS THAN (TO_DATE('01/01/2000','DD/MM/YYYY')),
PARTITION per2 VALUES LESS THAN (TO_DATE('01/01/2005','DD/MM/YYYY')),
PARTITION per3 VALUES LESS THAN (TO_DATE('01/01/2010','DD/MM/YYYY')),
PARTITION future VALUES LESS THAN (MAXVALUE));
```

Production Code

```
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));

 DECLARE
  x XMLTYPE;
 BEGIN
   x := XMLTYPE('<?xml version="1.0" encoding="utf-8"?>
          <Order orderId="1" orderRevision="1" orderTimeStamp="01-JAN-2012">
            <OrderHeader>
              <AlternateIds>
                <AlternateId altIdType="SiteId">12</AlternateId>
                <AlternateId altIdType="MerchantOrderNumber">Merch</AlternateId>
                <AlternateId altIdType="MarketplaceOrderNumber">Place</AlternateId>
                <AlternateId altIdType="CustomerReferenceId">Ref</AlternateId>
                <AlternateId altIdType="CartId">Cart</AlternateId>
                <AlternateId altIdType="SessionId">1</AlternateId>
              </AlternateIds>
            </OrderHeader>
          </Order>');
   INSERT INTO orders VALUES (x);
 END;
 /

...
```

Altering Partitioned/Subpartitioned Tables

- Convert (to a stand-alone table)
- Drop
- Merge
- Modify Partition List
- Modify Subpartition Template
- Move
- Rename
- Split
- Truncate

Code Objects (1:3)

- Views
  - Any SQL statement permanently stored, as an object, in the database
  - Oracle allows you to use views for all SELECT statements and, with UPDATABLE views to use them for INSERT, UPDATE, and DELETE
- Triggers
  - Table triggers
  - Instead-Of triggers on Views
  - DDL Event Triggers
  - System Event Triggers
  - Advanced Queuing events can also be used to trigger Scheduler Jobs
  - Materialized View objects can be set up to refresh based upon internally created triggers

Code Objects (2:3)

- Types and Type Bodies
  - Encapsulation
  - Inheritance
  - Methods
  - Open Recursion
  - Polymorphism
- Functions
  - Normal
  - Pipelined
- Operators
  - Make function overloading possible without packages
- Procedures (PL/SQL and Java)
  - Ok to write these during development but it is best that they then be put into packages for testing and production deployment

Code Objects (3:3)

- Packages
  - The optimal way to deploy functions and procedures into production
  - Simplify permission grants
  - Simplify change management
  - All related code loaded into memory together
  - Higher likelihood of code not aging out of memory
- Native Compilation
  - PL/SQL objects such as functions, procedures and packages can be compiled as C from their PL/SQL source code
  - Native compilation enhances cpu intensive operations but has no value in I/O intensive operations
  - Can be a barrier to export and import across operating systems

Other Objects (1:2)

- Sequences
  - Number generating object
- Database Link
  - Connection to other databases: Both Oracle and others
- Synonyms
  - Alternative names for clarity and simplicity
- Directories
  - Pointers to operating system file locations
- Queues
  - JMS (Java Messaging Service and others)
- Contexts
  - User defined namespaces

Other Objects (2:2)

- Rules and Rule Sets
  - Rules and sets of rules applicable to processing such as replication
- Schedules
  - Scheduled jobs and events
- Windows
  - Dates and times during which scheduled jobs can be run
- .... many more object types are available

Access Optimization

Optimization

- The Optimizers
  - Rule Based
  - Cost Based
- Parsing
  - Hard Parse
  - Soft Parse

Optimizers

- Rule Based Optimizer (RBO)
  - 14 rules that only apply to version 7 object types
- Cost Based Optimizer (CBO)
  - Introduced in Oracle 7
  - It was horrible: that began changing significantly in 8*i*
  - The CBO has been the best choice since 9.2.0.4
  - The better the information provided the optimizer the better the choices it makes (almost always)
- What is true in one version of the optimizer may not be true in another

RBO: Rule Based Optimizer

Access paths and their ranking	
Path 1	Single Row by Rowid
Path 2	Single Row by Cluster Join
Path 3	Single Row by Hash Cluster key with Unique or Primary Key
Path 4	Single Row by Unique or Primary Key
Path 5	Clustered Join
Path 6	Hash Cluster Key
Path 7	Indexed Cluster Key
Path 8	Composite Index
Path 9	Single-Column Index
Path 10	Bounded Range Search on Indexed Columns
Path 11	Unbounded Range Search on Indexed Columns
Path 12	Sort-Merge Join
Path 13	MAX or MIN of Indexed Column
Path 14	ORDER BY on Indexed Column
Path 15	Full Table Scan

CBO: Cost Based Optimizer

- Artificial intelligence uses information about the data to calculate the best access path
- Statistics must be collected using DBMS_STATS
- How do you determine if statistics are current?

```
CREATE TABLE demo AS SELECT * FROM dba_tables;

SELECT COUNT(*) FROM demo;

SELECT table_name, num_rows, last_analyzed
FROM user_tables
ORDER BY 1;

exec dbms_stats.gather_table_stats(USER, 'DEMO');

SELECT table_name, num_rows, last_analyzed
FROM user_tables
ORDER BY 1;
```

Parsing

- Parse Steps
  - Syntax Check
    - Is the statement a valid one? Does it make sense given the SQL grammar documented in the SQL Reference Manual? Does it follow all of the rules for SQL?
  - Semantic Analysis
    - Do the tables and columns referenced exist? Does the user running the query have valid permissions on the objects? Are there ambiguities in the statement?
  - Determine of the query has been previously processed in a different session (if yes skip the next two steps which are very CPU intensive and a point of contention due to serialization)
  - Optimization
  - Row Source Generation

Performance

- Your code will run fast if
  - The optimizer can avoid a hard parse
    - Develop formatting and coding standards
    - Use bind variables
  - You minimize the amount of work by
    - Not including tables and rows you don't need
    - Creating efficient joins
    - Not joining to tables unnecessarily (use EXISTS subqueries where possible)
    - Writing your code to favor efficient access
    - Not forcing unnecessary sorts with DISTINCT, ORDER BY and UNION clauses

CBO: Cost Based Optimizer Revisited

- Plan Caching
  - Hashing and ORA_HASH
- Cursor Sharing
  - Fully qualified object names
  - V$SQL_SHARED_CURSOR
  - AUTH_PLAN_MISMATCH
- Explain Plan
- Efficient SQL
- Efficient PL/SQL

Connection Pooling

What is Connection Pooling

- Opening and maintaining a database connection for each user
  - Especially when dynamically creating sessions for requests made to a dynamic database-driven web application are costly and waste resources
- A cache with multiple persistent database connections
- After a connection is created, it is placed in the pool, and it is reused so that a new connection does not have to be established for each new session or inquiry
- Each persistent connection can be reused when a future request to the database is made
- Used to enhance the performance of executing commands in a database

Inactive Sessions

Attachments: killed-inactive-sessions.txt (23 KB)

Friday, April 12, 2013 1:15 PM

Hi Bob,

we have killed 250 inactive sessions only that are blocking other sessions.

1. What session was causing the blocking lock? The client would normally like to know what SCHEMA and WHAT WAS THE LAST SQL STATEMNET ISSUED ...
Ans :blocking session
id's:(25,153,889,967,1098,1356,522,554,620,740,949,1172,2,224,410,596,610,635,814,817,937,1089,1177,1195,1307,1316,1458,24,53,68,308,467,509,556,825,970,1035,1119,1325,38,686,830,832,1414,1511,333,506,792,932,942,115,1008,1014,19,313,1198,1202,1224,1376);
Please Note : Not killed any active sessions for these blocking sessions. Killed only INACTIVE sessions.

Ignoring, for the moment, issues of grammar and spelling ...
clearly there is a technology problem

Common Mistakes

Costly Mistakes

My Short List

- Not learning how it works
- Not leveraging its optimizations
- Reinventing the wheel
  - DBMS_CHANGE_NOTIFICATION
  - DBMS_APPLICATION_INFO
    - SET_CLIENT
    - SET_ACTION
    - SET_MODULE
    - SESSION_LONGOPS
  - DBMS_SCHEDULER and File Watchers
  - Advanced Queuing
  - Real Application Testing
  - Oracle Media and DICOM
- Map Reduce

Just Because You Can Doesn't Mean You Should

- Multiple Identical Schemas
- Wide Tables
- Embedded SQL vs. Database APIs
- Commits are expensive
  - Make a transaction as large as logically consistent
  - Always avoid incremental commits if you can avoid them
- Large Queries

I am not afraid to show mine in public

This Is Page One Of A Single Query

Complete List of SQL Text

SQL Id	SQL Text
0cjpvvqy854kh	SELECT 'ERICSSON_CSCORE.EOS' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from ERICSSON_CSCORE.EOS where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(s ysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_CSCORE.EOS_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_CSCORE.MSCSTATS' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from ERICSSON_CSCORE.MSCSTATS where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(dist inct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_CSCORE.MSCSTATS_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_CSCORE.MTRAFTYPE' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count , count (*) num_of_rows from ERICSSON_CSCORE.MTRAFTYPE where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_CSCORE.MTRAFTYPE_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_GERAN.CELLGPRS' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(d atetime_ins, 'mi') > trunc(sysdate)+3/24 then BSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then BSC end) ontime_count, count (*) num_of_rows from ERICSSON_GERAN.CELLGPRS where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then BSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then BSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_GERAN.CELLGPRS_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_GERAN.CELLSTATS' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_ count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then BSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then BSC end) ontime_count, count (*) num_of_rows from ERICSSON_GERAN.CELLSTATS where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then BSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then BSC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_GERAN.CELLSTATS_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'ERICSSON_PSCORE.MM' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count,

Page 2

```
(ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count,
2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count curr ent_raw_elements,
summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff,
percent(num_of_sumrows, num_of_rows, 2) accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then SGSNID end) late_count, count(distinct case when trunc(datetime_ins, 'mi') <
trunc(sysdate)+3/24 then SGSNID end) ontime_count, count (*) num_of_rows from ERICSSON_PSCORE.MM where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sumday, count(distinct case
when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then SGSNID end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then SGSNID end) summarised_ontime_count, sum(entries)
num_of_sumrows from ERICSSON_PSCORE.MM_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT 'ERICSSON_PSCORE.SGSNSTAT' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count,
(ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0
else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+la te_count, 2)percentage_of_total_by5am,
percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count
current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2)
accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then SGSNID end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then SGSNID end) ontime_count,
count (*) num_of_rows from ERICSSON_PSCORE.SGSNSTAT where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') >
trunc(sysdate)+7/24 then SGSNID end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then SGSNID end) summarised_ontime_count, sum(entries) num_of_sumrows from
ERICSSON_PSCORE.SGSNSTAT_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT 'ERICSSON_UTRAN.RBS_CARRIER' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count),
2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count- late_count), 2) < 0 then 0 else
100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am,
percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count
current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2)
accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then NODEB end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then NODEB end) ontime_count,
count (*) num_of_rows from ERICSSON_UTRAN.RBS_CARRIER where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') >
trunc(sysdate)+7/24 then NODEB end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then NODEB end) summarised_ontime_count, sum(entries) num_of_sumrows from
ERICSSON_UTRAN.RBS_CARRIER_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT 'ERICSSON_UTRAN.RBS_EDCHRES' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-l
ate_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else
100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am,
percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count
current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2)
accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then NODEB end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then NODEB end) ontime_count,
count (*) num_of_rows from ERICSSON_UTRAN.RBS_EDCHRES where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') >
trunc(sysdate)+7/24 then NODEB end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then NODEB end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_UT
RAN.RBS_EDCHRES_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT 'ERICSSON_UTRAN.RBS_HSDSCHRES' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100
THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else
100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am,
percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count
current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2)
accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then NODEB end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then NODEB end) ontime_count,
count (*) num_of_rows from ERICSSON_UTRAN.RBS_HSDSCHRES where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') >
trunc(sysdate)+7/24 then NODEB e nd) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then NODEB end) summarised_ontime_count, sum(entries) num_of_sumrows from
ERICSSON_UTRAN.RBS_HSDSCHRES_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT 'ERICSSON_UTRAN.RNC_IURLINK' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-
late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else
100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am,
```

Page 3

```
percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count
current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2)
accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then RNC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then RNC end) ontime_count, count (*)
num_of_rows from ERICSSON_UTRAN.RNC_IURLINK where dateti me between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24
then RNC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then RNC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_UTRAN.RNC_IURLINK_DY
where datetime = trunc(sysdate)-1 )sumt union all SELECT 'ERICSSON_UTRAN.RNC_RNCFUNC' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE
percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count,
(ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count,
ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count
SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') >
trunc(sysdate)+3/24 then RNC end) late_c ount, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then RNC end) ontime_count, count (*) num_of_rows from ERICSSON_UTRAN.RNC_RNCFUNC where datetime
between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then RNC end) summarised_late_count, count(distinct case when
trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then RNC end) summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_UTRAN.RNC_RNCFUNC_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT
'ERICSSON_UTRAN.RNC_UCELL' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END
percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed,
percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count
current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-
late_count availabl e_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then RNC end) late_count, count(distinct case when
trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then RNC end) ontime_count, count (*) num_of_rows from ERICSSON_UTRAN.RNC_UCELL where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select
max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then RNC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then RNC end)
summarised_ontime_count, sum(entries) num_of_sumrows from ERICSSON_UTRAN.RNC_UCELL_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT 'NORTEL_CSCORE.BASE_OM' AS TABLENAME, sunday, CASE WHEN
percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count,
(ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count,
2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_
late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows,
num_of_rows, 2) accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end)
ontime_count, count (*) num_of_rows from NORTEL_CSCORE.BASE_OM where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') >
trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from
NORTEL_CSCORE.BASE_OM_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT 'NORTEL_CSCORE.C7SP' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN
100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else
100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_c
ount+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements,
summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from ( select
count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from
NORTEL_CSCORE.C7SP where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end)
summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_CSCORE.C7SP_DY where datetime =
trunc(sysdate)-1 )sumt union all SELECT 'NORTEL_CSCORE.CCS7' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count,
(ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end
percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2)
current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM,
late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end)
```

Page 4

late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.CCS7 where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_CSCORE.CCS7_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.LOCATIONAREACODE' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cut off, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.LOCATIONAREACODE where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_CSCORE.LOCATIONAREACODE_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.MSCCP_CP' AS TABLENAME, sumd ay, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.MSCCP_CP where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_c ount, sum(entries) num_of_sumrows from NORTEL_CSCORE.MSCCP_CP_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.MSCCP' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.MSCCP where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(s ysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_CSCORE.MSCCP_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.TRKGRP' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.TRKGRP where dateti me between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_CSCORE.TRKGRP_DY where datetime = trunc(sysdate)-1)sumt union all SELECT 'NORTEL_CSCORE.VLR6' AS TABLENAME, sunday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from (select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then MSC end) late_count, count(d istinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then MSC end) ontime_count, count (*) num_of_rows from NORTEL_CSCORE.VLR6 where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60)rawt, (select max(datetime)sunday, count(distinct case

Page 5

```
when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then MSC end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then MSC end) summarised_ontime_count, sum(entries)
num_of_sumrows from NORTEL_CSCORE.VLR6_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT 'NORTEL_PSCORE.GSCGMM' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-
late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else
100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am,
percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count
current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumro ws, num_of_rows, 2)
accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then SGSN end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then SGSN end) ontime_count, count
(*) num_of_rows from NORTEL_PSCORE.GSCGMM where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24
then SGSN end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then SGSN end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_PSCORE.GSCGMM_DY where
datetime = trunc(sysdate)-1 )sumt union all SELECT 'NORTEL_PSCORE.GSCSM_ACT' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE
percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count,
(ontime_count-late_count), 2) end percent_of_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count,
ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count
SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') >
trunc(sysdate)+3/24 then SGSN end) late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then SGSN end) ontime_count, count (*) num_of_rows from NORTEL_PSCORE.GSCSM_ACT where datetime
between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then SGSN end) summarised_late_count, count(distinct case when
trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then SGSN end) summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_PSCORE.GSCSM_ACT_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT
'NORTEL_PSCORE.GSCSM' AS TABLENAME, sumday, CASE WHEN percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END
percent_of_cutoff, case when 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_of_cutoff_not_summed,
percent(summarised_ontime_count, ontime_count+late_count, 2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count
current_raw_elements, summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-
late_count available_at_cutoff, percent(num_of_sumrows, num_of_rows, 2) accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then SGSN end) late_count, count(distinct case when
trunc(datetime_ins, 'mi') < trunc(sysdate)+3/24 then SGSN end) ontime_count, count (*) num_of_rows from NORTEL_PSCORE.GSCSM where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select
max(datetime)sumday, count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then SGSN end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then SGSN end)
summarised_ontime_count, sum(entries) num_of_sumrows from NORTEL_PSCORE.GSCSM_DY where datetime = trunc(sysdate)-1 )sumt union all SELECT 'NORTEL_PSCORE.GSDSTATS' AS TABLENAME, sumday, CASE WHEN
percent(summarised_ontime_count, (ontime_count-late_count), 2) > 100 THEN 100 ELSE percent(summarised_ontime_count, (ontime_count-late_count), 2) END percent_of_cutoff, case when 100-percent(summarised_ontime_count,
(ontime_count-late_count), 2) < 0 then 0 else 100-percent(summarised_ontime_count, (ontime_count-late_count), 2) end percent_o f_cutoff_not_summed, percent(summarised_ontime_count, ontime_count+late_count,
2)percentage_of_total_by5am, percent(summarised_ontime_count+summarised_late_count, ontime_count+late_count, 2) current_percentage, ontime_count+late_count current_raw_elements,
summarised_ontime_count+summarised_late_count current_summary_elements, summarised_ontime_count-summarised_late_count SUM_ELEMENTS_AT5AM, late_count late_raw_elements, ontime_count-late_count available_at_cutoff,
percent(num_of_sumrows, num_of_rows, 2) accuracy from ( select count(distinct case when trunc(datetime_ins, 'mi') > trunc(sysdate)+3/24 then SGSN end) late_count, count(distinct case when trunc(datetime_ins, 'mi') <
trunc(sysdate)+3/24 then SGSN end) ontime_count, count (*) num_of_rows from NORTEL_PSCORE.GSDSTATS where datetime between trunc(sysdate)-1 and trunc(sysdate)-1/24/60 )rawt, ( select max(datetime)sumday, count(distinct
case when trunc(datetime_ins, 'mi') > trunc(sysdate)+7/24 then SGSN end) summarised_late_count, count(distinct case when trunc(datetime_ins, 'mi') < trunc(sysdate)+7/24 then SGSN end) summarised_ontime_count, sum(entries)
num_of_sumrows from NORTEL_PSCORE.GSDSTATS_DY where datetime = trunc(sysdate)-1 )sumt
```

Optimizer Plans (the way it was)

```
SELECT DISTINCT E1_2.OBJECT_ID
   FROM PMCM.ELEMENT_DETAIL E1_1, PMCM.ELEMENT_DETAIL E1_2, PMCM.MARK_NETW_HIERARCHY H1,
        PMCM.ELEMENT_DETAIL E2_1, PMCM.ELEMENT_DETAIL E2_2, PMCM.MARK_NETW_HIERARCHY H2
   WHERE E1_1.OBJECT_ID = H1.PARENT_ID
     AND E1_2.OBJECT_ID = H1.OBJECT_ID
     AND E2_1.OBJECT_ID = H2.PARENT_ID
     AND E2_2.OBJECT_ID = H2.OBJECT_ID
     AND E1_1.CURRENT_IND = 'Y' AND E2_1.CURRENT_IND = 'Y'
     AND E2_1.CURRENT_IND = 'Y' AND E2_2.CURRENT_IND = 'Y'
     AND H1.CURRENT_IND = 'Y' AND H2.CURRENT_IND = 'Y'
     AND H1.HIERARCHY_TYPE = 'NETWORK' AND H2.HIERARCHY_TYPE = 'NETWORK'
     AND H1.PARENT_TYPE IN ('BSC','RNC')  AND H2.PARENT_TYPE IN ('BSC','RNC')
     AND E2_2.ELEMENT_TYPE = 'CELL' AND E1_2.ELEMENT_TYPE = 'CELL'
     AND H1.PARENT_TYPE IN ('BSC','RNC')
     AND E1_1.ELEMENT_NAME = E2_1.ELEMENT_NAME
     AND E1_1.ELEMENT_ID = E2_1.ELEMENT_ID
     AND E1_2.ELEMENT_NAME = E2_2.ELEMENT_NAME
     AND E1_2.ELEMENT_ID = E2_2.ELEMENT_ID
     AND E1_2.USEID LIKE '*%' AND E2_2.USEID NOT LIKE '*%';

| Id  | Operation                    | Name                          | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
|   0 | SELECT STATEMENT             |                               |     1 |    78 |       |    74M (40)| 50:54:42 |       |       |
|   1 |  TEMP TABLE TRANSFORMATION   |                               |       |       |       |             |          |       |       |
|   2 |   LOAD AS SELECT             |                               |       |       |       |             |          |       |       |
|   3 |    PARTITION RANGE ALL       |                               |   22M|  1111M|       | 38153   (11)| 00:01:34 |     1 |    29 |
|*  4 |     TABLE ACCESS FULL        | ELEMENT_DETAIL                |   22M|  1111M|       | 38153   (11)| 00:01:34 |       |       |
|   5 |   LOAD AS SELECT             |                               |       |       |       |             |          |       |       |
|   6 |    PARTITION HASH ALL        |                               |  337K|  9231K|       |  3514   (15)| 00:00:09 |     1 |    16 |
|*  7 |     TABLE ACCESS FULL        | MARK_NETW_HIERARCHY           |  337K|  9231K|       |  3514   (15)| 00:00:09 |       |       |
|   8 |   SORT AGGREGATE             |                               |     1 |    78 |       |             |          |       |       |
|*  9 |    HASH JOIN                 |                               |  927G|   65T|   534M|    74M (40)| 50:53:00 |       |       |
|  10 |     VIEW                     |                               |   22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  11 |      TABLE ACCESS FULL       | SYS_TEMP_0FDA7485F_6A66C42E   |   22M|  1111M|       | 16808   (12)| 00       |       |       |
|* 12 |     HASH JOIN                |                               |   21G|  1272G|   534M|  1616K (43)| 01:06:04 |       |       |
|  13 |      VIEW                    |                               |   22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  14 |       TABLE ACCESS FULL      | SYS_TEMP_0FDA7485F_6A66C42E   |   22M|  1111M|       | 16808   (12)| 0        |       |       |
|* 15 |      HASH JOIN               |                               |  476M|    23G|   524M| 97327   (22)| 00:03:59 |       |       |
|* 16 |       HASH JOIN              |                               |   10M|   401M|  8704K| 34520   (10)| 00:01:25 |       |       |
|* 17 |        HASH JOIN             |                               |  234K|  5948K|  8256K|   783   (10)| 00:00:02 |       |       |
|  18 |         VIEW                 |                               |  337K|  4286K|       |   142   (14)| 00:00:01 |       |       |
|  19 |          TABLE ACCESS FULL   | SYS_TEMP_0FDA74860_6A66C42E   |  337K|  3956K|       |   142   (14)| 00:00:01 |       |       |
|  20 |         VIEW                 |                               |  337K|  4286K|       |   142   (14)| 00:00:01 |       |       |
|  21 |          TABLE ACCESS FULL   | SYS_TEMP_0FDA74860_6A66C42E   |  337K|  3956K|       |   142   (14)| 00:00:01 |       |       |
|  22 |        VIEW                  |                               |   22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  23 |         TABLE ACCESS FULL    | SYS_TEMP_0FDA7485F_6A66C42E   |   22M|  1111M|       | 16808   (12)| 00:00:42 |       |       |
|  24 |       VIEW                   |                               |   22M|   277M|       | 16808   (12)| 00:00:42 |       |       |
|  25 |        TABLE ACCESS FULL     | SYS_TEMP_0FDA7485F_6A66C42E   |   22M|  1111M|       | 16808   (12)| 0        |       |       |
```

Optimizer Plans (tuning gone terribly wrong)

```
-----------------------------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                          | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     | Pstart| Pstop |
-----------------------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                               |     1 |    78 |       |    14T(100)|999:59:59 |       |       |
|   1 |  TEMP TABLE TRANSFORMATION   |                               |       |       |       |            |          |       |       |
|   2 |   LOAD AS SELECT             |                               |       |       |       |            |          |       |       |
|   3 |    PARTITION RANGE ALL       |                               |    22M|  1111M|       | 38153  (11)| 00:01:34 |     1 |    29 |
|*  4 |     TABLE ACCESS FULL        | ELEMENT_DETAIL                |    22M|  1111M|       | 38153  (11)| 00:01:34 |       |       |
|   5 |   LOAD AS SELECT             |                               |       |       |       |            |          |       |       |
|   6 |    PARTITION HASH ALL        |                               |   337K|  9231K|       |  3514  (15)| 00:00:09 |     1 |    16 |
|*  7 |     TABLE ACCESS FULL        | MARK_NETW_HIERARCHY           |   337K|  9231K|       |  3514  (15)| 00:00:09 |       |       |
|   8 |   SORT AGGREGATE             |                               |     1 |    78 |       |            |          |       |       |
|   9 |    MERGE JOIN                |                               |  471P|    15E|       |    14T(100)|999:59:59 |       |       |
|  10 |     MERGE JOIN               |                               |   10P|   616P|       |   694G (81)|999:59:59 |       |       |
|  11 |      MERGE JOIN              |                               |  231T|    10P|       |   377G (64)|999:59:59 |       |       |
|  12 |       SORT JOIN              |                               |  334T|    11P|   28P|   377G (64)|999:59:59 |       |       |
|  13 |        MERGE JOIN CARTESIAN  |                               |  334T|    11P|       |   140G (14)|999:59:59 |       |       |
|* 14 |         HASH JOIN            |                               |   989M|    23G|   534M| 96010  (38)| 00:03:56 |       |       |
|  15 |          VIEW                |                               |    22M|   277M|       | 16808  (12)| 00:00:42 |       |       |
|  16 |           TABLE ACCESS FULL  | SYS_TEMP_0FDA7485B_6A66C42E   |    22M|  1111M|       | 16808  (12)| 00:00:42 |       |       |
|  17 |          VIEW                |                               |    22M|   277M|       | 16808  (12)| 00:00:42 |       |       |
|  18 |           TABLE ACCESS FULL  | SYS_TEMP_0FDA7485B_6A66C42E   |    22M|  1111M|       | 16808  (12)| 00:00:42 |       |       |
|  19 |         BUFFER SORT          |                               |   337K|  4286K|       |   140G (14)|999:59:59 |       |       |
|  20 |          VIEW                |                               |   337K|  4286K|       |   142  (14)| 00:00:01 |       |       |
|  21 |           TABLE ACCESS FULL  | SYS_TEMP_0FDA7485C_6A66C42E   |   337K|  3956K|       |   142  (14)| 00:00:01 |       |       |
|* 22 |       SORT JOIN              |                               |   337K|  4286K|    12M|   844  (14)| 00:00:03 |       |       |
|  23 |        VIEW                  |                               |   337K|  4286K|       |   142  (14)| 00:00:01 |       |       |
|  24 |         TABLE ACCESS FULL    | SYS_TEMP_0FDA7485C_6A66C42E   |   337K|  3956K|       |   142  (14)| 00:00:01 |       |       |
|* 25 |      SORT JOIN               |                               |    22M|   277M|   855M| 65084  (16)| 00:02:40 |       |       |
|  26 |       VIEW                   |                               |    22M|   277M|       | 16808  (12)| 00:00:42 |       |       |
|  27 |        TABLE ACCESS FULL     | SYS_TEMP_0FDA7485B_6A66C42E   |    22M|  1111M|       | 16808  (12)| 0        |       |       |
|* 28 |     SORT JOIN                |                               |    22M|   277M|   855M| 65084  (16)| 00:02:40 |       |       |
|  29 |      VIEW                    |                               |    22M|   277M|       | 16808  (12)| 00:00:42 |       |       |
|  30 |       TABLE ACCESS FULL      | SYS_TEMP_0FDA7485B_6A66C42E   |    22M|  1111M|       | 16808  (12)| 0        |       |       |
-----------------------------------------------------------------------------------------------------------------------------------
```

Optimizer Plans (making it better)

```
WITH ed AS (SELECT object_id, element_id, element_name, element_type, useid
            FROM pmcm.element_detail
            WHERE element_type = 'CELL'
            AND current_ind = 'Y'),
     mnh AS (SELECT parent_id, object_id
             FROM pmcm.mark_netw_hierarchy
             WHERE current_ind = 'Y'
             AND hierarchy_type = 'NETWORK'
             AND parent_type IN ('BSC', 'RNC'))
SELECT COUNT(*)
FROM ed e1_1, ed e1_2, ed e2_1, ed e2_2, mnh h1, mnh h2
WHERE e1_1.object_id = h1.parent_id AND   e1_2.object_id = h1.object_id
AND   e2_1.object_id = h2.parent_id AND   e2_2.object_id = h2.object_id
AND   e1_1.element_name = e2_1.element_name
AND   e1_1.element_id = e2_1.element_id
AND   e1_2.element_name = e2_2.element_name
AND   e1_2.element_id = e2_2.element_id
AND   e1_2.useid LIKE '*%'
AND   e2_2.useid NOT LIKE '*%';
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		1	214		100K (6)	00:04:08
1	HASH UNIQUE		1	214		100K (6)	00:04:08
* 2	HASH JOIN		1	214	12M	100K (6)	00:04:08
3	PARTITION HASH ALL		337K	9231K		3514 (15)	00:00:09
* 4	TABLE ACCESS FULL	MARK_NETW_HIERARCHY	337K	9231K		3514 (15)	00:00:00
* 5	HASH JOIN		207K	36M	22M	95860 (6)	00:03:56
6	PARTITION RANGE ALL		586K	15M		16233 (2)	00:00:40
7	TABLE ACCESS BY LOCAL INDEX ROWID	ELEMENT_DETAIL	586K	15M		16233	??:??:??
* 8	INDEX SKIP SCAN	ED_ET_TECH_CI	586K			12791 (1)	00:00:3?
* 9	HASH JOIN		207K	31M	22M	77982 (7)	00:03:12
10	PARTITION RANGE ALL		586K	15M		16233 (2)	00:00:40
11	TABLE ACCESS BY LOCAL INDEX ROWID	ELEMENT_DETAIL	586K	15M		16233	??:??:??
* 12	INDEX SKIP SCAN	ED_ET_TECH_CI	586K			12791 (1)	00:00:??
* 13	HASH JOIN		179K	22M	12M	60372 (8)	00:02:29
14	PARTITION HASH ALL		337K	9231K		3514 (15)	00:00:09
* 15	TABLE ACCESS FULL	MARK_NETW_HIERARCHY	337K	9231K		3514 (15)	00:00:??
* 16	HASH JOIN		184K	17M	10M	55886 (8)	00:02:18
17	PARTITION RANGE ALL		184K	9008K		37137 (8)	00:01:32
* 18	TABLE ACCESS FULL	ELEMENT_DETAIL	184K	9008K		37137 (8)	00:01:32
19	PARTITION RANGE ALL		576K	28M		17383 (8)	00:00:43
* 20	TABLE ACCESS BY LOCAL INDEX ROWID	ELEMENT_DETAIL	576K	28M		17383 (8)	??:??:??
* 21	INDEX SKIP SCAN	ED_ET_TECH_CI	583K			13939 (9)	00:00:35

Result Cache

SQL ordered by Executions

- Total Executions: 29,717,627
- Captured SQL account for 77.4% of Total

Executions	Rows Processed	Rows per Exec	CPU per Exec (s)	Elap per Exec (s)	SQL Id	SQL Module	SQL Text
10,128,178	2,506,529	0.25	0.00	0.00	932srzg1krc33	ASN_07B_DIP(004110016)	SELECT NE_TIMEZONE FROM CMPM.E...
7,576,759	7,579,197	1.00	0.00	0.00	1h698sb62un99	asci_56_RANAPProtocolStats(016110008)	SELECT DISTINCT NE_TIMEZONE FR...
3,914,621	3,848,268	0.98	0.00	0.00	5tbzddgguu8cc	asci_56_RANAPProtocolStats(016110008)	SELECT SYS_VERSION FROM CMPM.T...
311,645	311,604	1.00	0.00	0.00	7qtztzv329wg0		select c.name, u.name from co...
301,428	301,325	1.00	0.00	0.00	36s446f9cnwhw		SELECT C.NAME FROM COL$ C WHER...
200,692	200,669	1.00	0.00	0.00	4vs91dcv7u1p6	OMS	insert into sys.aud$(sessioni...
65,044	65,035	1.00	0.00	0.00	fz9xwpt2cvt0k		SELECT par_type, param_clob, ...
64,949	3,945,482	60.75	0.00	0.00	f5ra7dru5fk5n	XML_P7R_RNC_RCS(003110008)	SELECT NAME, PATH, READ, WR...
64,801	64,807	1.00	0.00	0.00	fhzj09a7fnrnb	XML_V7I_IN_LP_DC(00811000V)	SELECT DBTIMEZONE, LENGTH(DBT...
64,632	64,542	1.00	0.00	0.00	15jnrrb6016nd	XML_V7I_IN_LP_DC(00811000V)	SELECT SESSIONTIMEZONE, LENGT...

```
CREATE OR REPLACE FUNCTION rcache(p_srvr_id IN servers.srvr_id%TYPE) RETURN BOOLEAN
RESULT_CACHE RELIES_ON(servers) IS
  srvrow servers%ROWTYPE;
 BEGIN
   SELECT *
   INTO srvrow
   FROM servers
   WHERE srvr_id = p_srvr_id;
   RETURN TRUE;
 EXCEPTION
   WHEN OTHERS THEN
     RETURN FALSE;
 END rcache;
 /
```

```
SELECT /*+ RESULT_CACHE */ srvr_id
FROM (
  SELECT srvr_id, SUM(cnt) SUMCNT
  FROM (
    SELECT DISTINCT srvr_id, 1 AS CNT
    FROM servers
    UNION ALL
    SELECT DISTINCT srvr_id, 1
    FROM serv_inst)
  GROUP BY srvr_id)
WHERE sumcnt = 2;
```

PL/SQL Warnings

- 11g new warnings
- Invaluable for finding suboptimal code, use of reserved words, inappropriate usages, and orphansEmbedded SQL vs. Database APIs
  - Severe
    - 5018 - omitted optional AUTHID clause
    - 5019 - deprecated language element
    - 5020 - parameter name must be identified
  - Informative
    - 6016 - native code generation turned off (size/time)
    - 6017 - operation will raise an exception
    - 6018 - an infinity or NaN value computed or used
  - Performance
    - None

Objects Named with Keywords

- How to test a word to determine whether it can be used

```
SELECT keyword
FROM v$reserved_words
WHERE keyword LIKE 'ID%';

KEYWORD
------------------------------
ID
IDENTIFIED
IDGENERATORS
IDLE_TIME
IDENTITY
IDENTIFIER

6 rows selected.
```

Columns should never be named "ID." Best practice is to identify the nature of the ID, for example, MEMBER_ID or ORDER_ID. This holds true for columns such as COMMENTS and DATES.

Implicit Casts

- Code that does not correctly define data types will either fail to run or run very inefficiently

  The following example shows both the correct way and the incorrect way to work with dates. The correct way is to perform an explicit cast

```
orabase> create table t (
      2  datecol date);

Table created.

orabase> insert into t values ('01-JAN-2012');

1 row created.

orabase> insert into t values (TO_DATE('01-JAN-2012'));

1 row created.
```

In Oracle dates are dates ... not strings. Similarly numbers should either be explicitly cast with TO_NUMBER or only processed using numeric variables.

Statement Labeling

- When SQL statements are labeled it makes it very easy for the DBA team to trace activity and help with performance tuning and debugging

  The following example shows how easy this is. In this example "mypackage.myprocedure" is the name of the PL/SQL package and procedure into which the statements are embedded

```
SELECT /* mypackage.myprocedure */ COUNT(*)
FROM all_tables;

INSERT /* mypackage.myprocedure */ INTO servers
(srvr_id)
VALUES
(11111);

UPDATE /* mypackage.myprocedure */ servers
SET srvr_id = 11111
WHERE srvr_id = 1;

DELETE /* mypackage.myprocedure */ FROM servers
WHERE srvr_id = 11111;
```

Explicit Commits and Rollbacks

- An explicit commit must be present at the end of any transaction
- You can not rely on an ODBC or JDBC driver configuration to commit your changes in a reliable and consistent way
- Similarly ... you must explicitly rollback in exception handlers as the ODBC driver may perform an unintended commit

```
CREATE OR REPLACE PROCEDURE test AUTHID DEFINER IS
BEGIN
  INSERT INTO t
  (testcolumn)
  VALUES
  (1);

  COMMIT;
EXCEPTION
  WHEN dup_val_on_index THEN
    ROLLBACK;
END;
/
```

Ignoring Optimizations

- Writing generic code is an attempt to write code that will run anywhere ... very very poorly
- Cursor Sharing
- Fully qualified object names
  - Multiple Identical Schemas
  - V$SQL_SHARED_CURSOR
  - AUTH_PLAN_MISMATCH

Cached Cursors

```
SQL> SELECT * FROM v$system_cursor_cache;

     OPENS       HITS  HIT_RATIO     CON_ID
---------- ---------- ---------- ----------
      2968       2968          1          0



SQL> SELECT * FROM v$session_cursor_cache;

   MAXIMUM      COUNT      OPENS       HITS  HIT_RATIO     CON_ID
---------- ---------- ---------- ---------- ---------- ----------
        50         50        468        343 .732905983          0
```

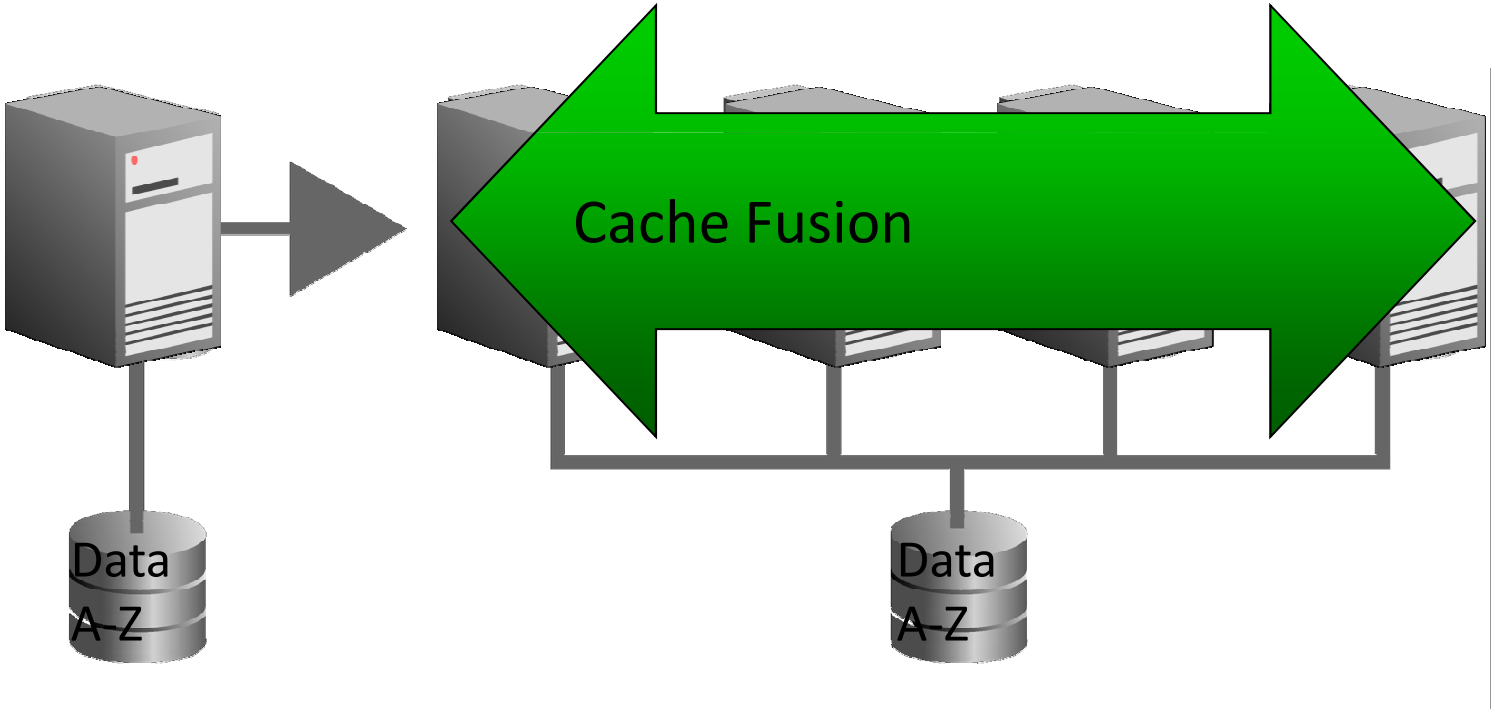
RAC: Real Application Clusters

Benefits of RAC

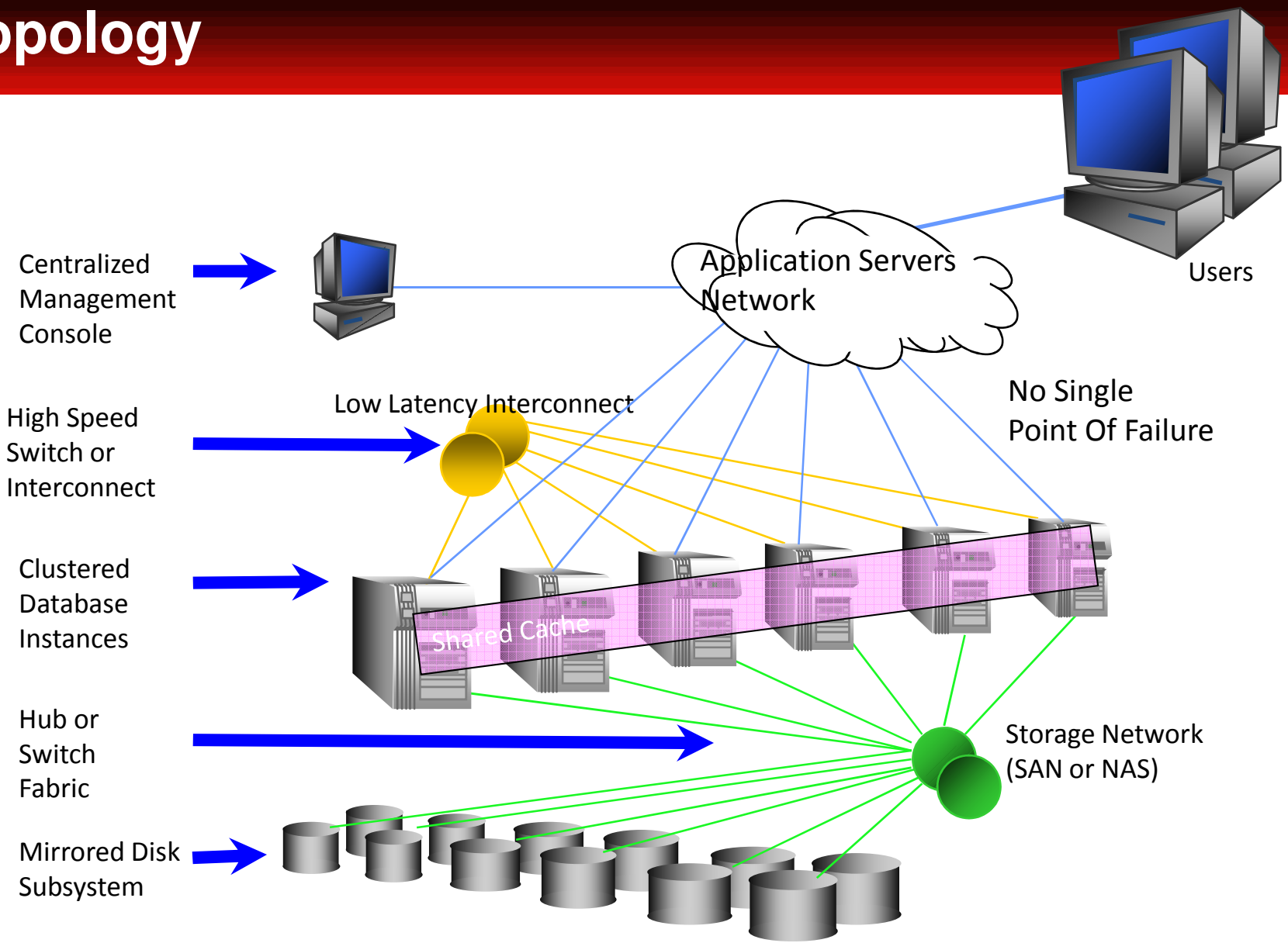
- Reliability
  - Eliminate the Instance (and server) as a single point of failure
  - Service failover
- Availability
  - Less downtime
  - Load Balancing
  - Service relocation
    - Application Failover (TAF and FCF)
  - Rolling upgrades
- Scalability
  - Exploit low cost commodity hardware for incremental scaling
  - Add servers as necessary

Shared Cache Databases

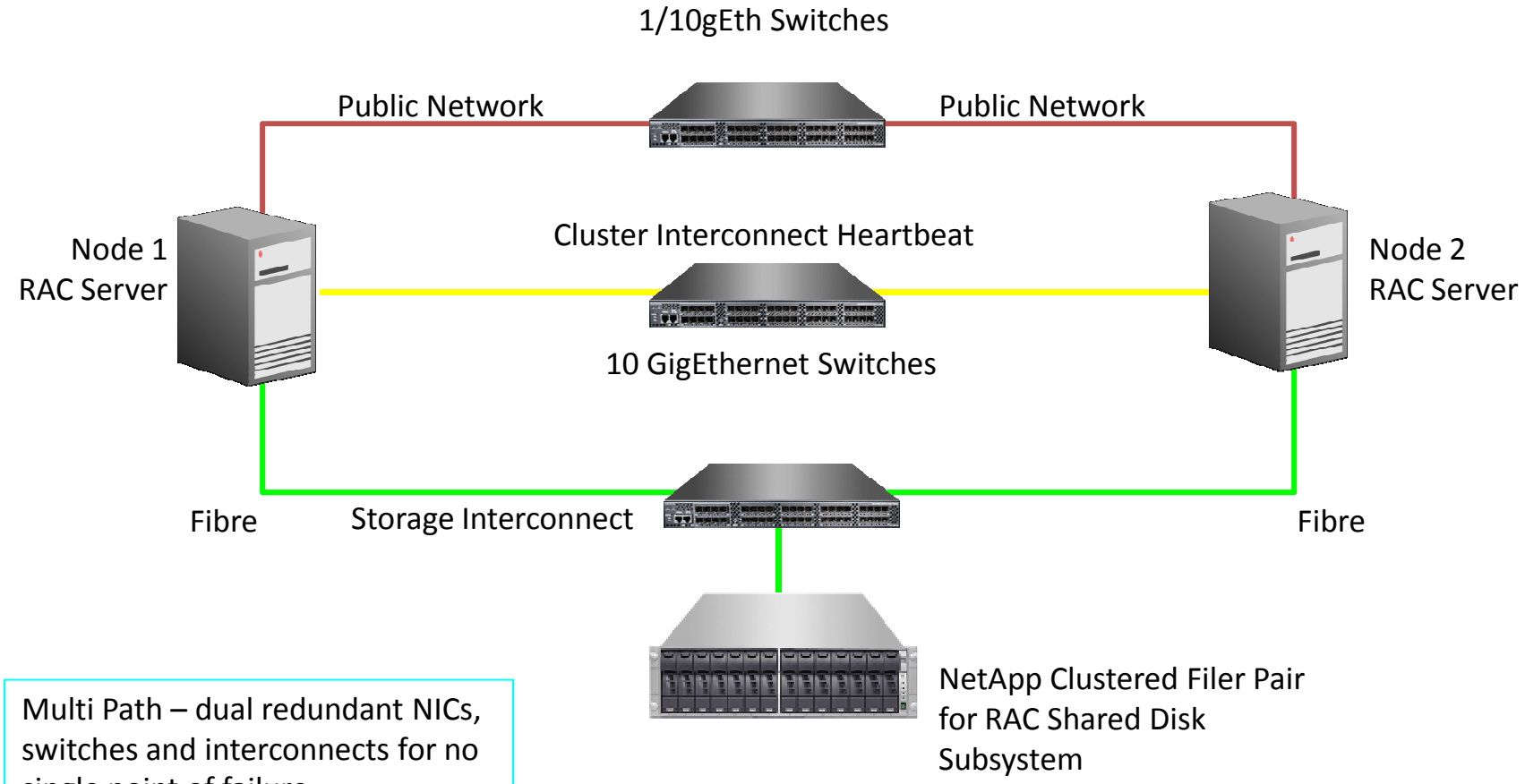
- Provide BOTH availability and scalability
- Maintained like a single database
- More reliable with more servers



Topology

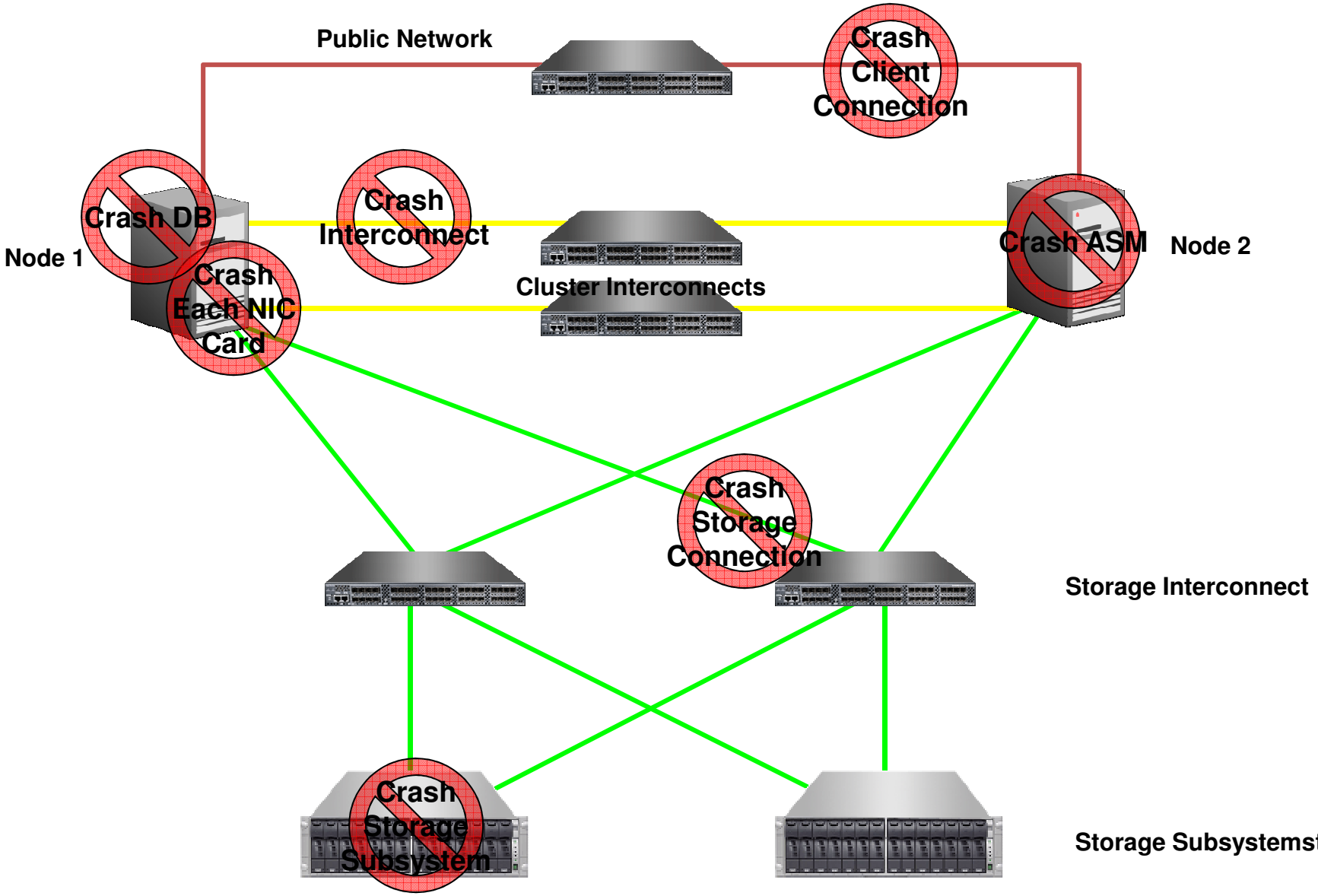

Centralized Management Console
Application Servers Network
Users
High Speed Switch or Interconnect
Low Latency Interconnect
No Single Point Of Failure
Clustered Database Instances
Shared Cache
Hub or Switch Fabric
Storage Network (SAN or NAS)
Mirrored Disk Subsystem

Basic Topology



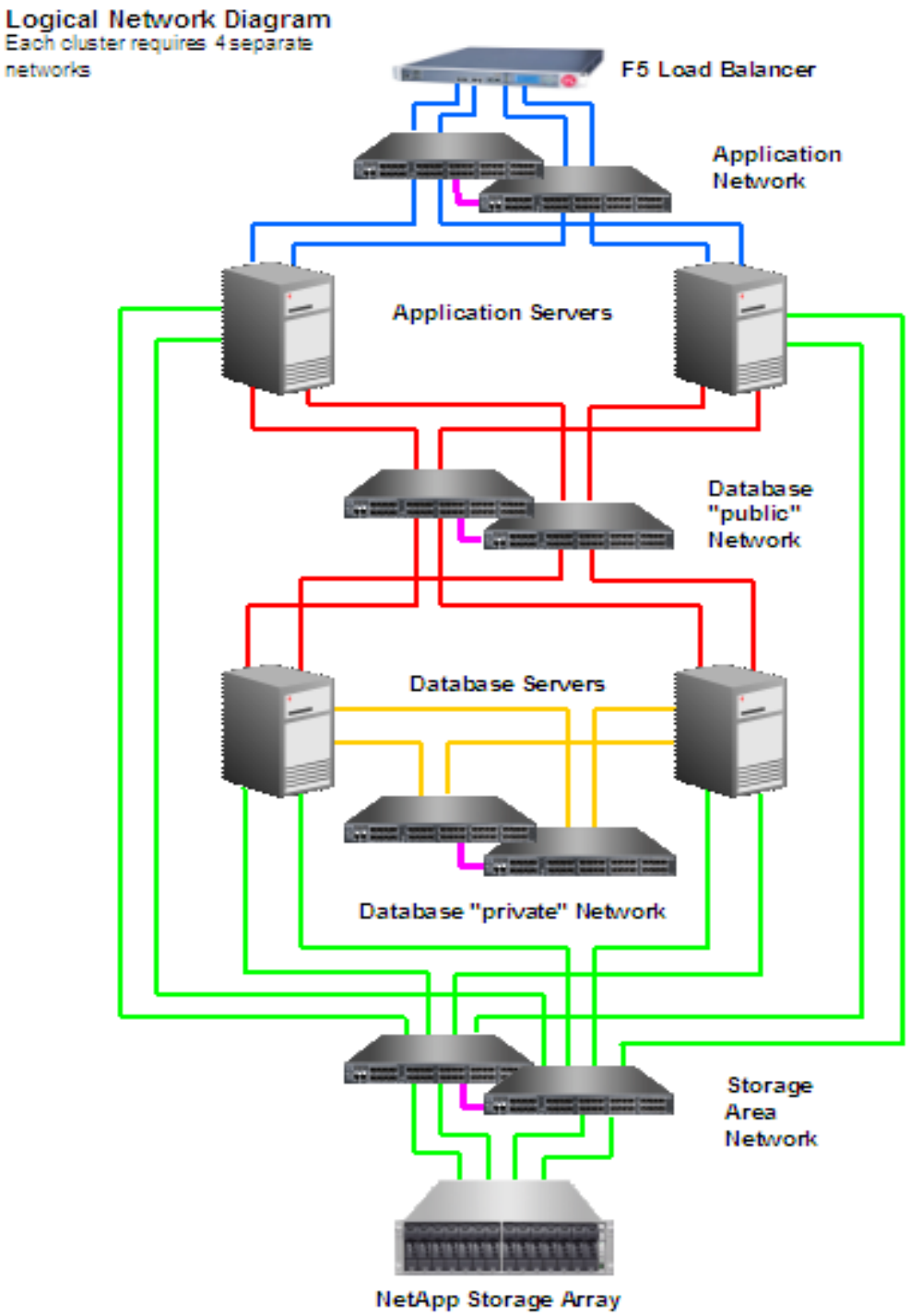
Multi Path – dual redundant NICs, switches and interconnects for no single point of failure
A clustered filer pair is needed for No single point of failure

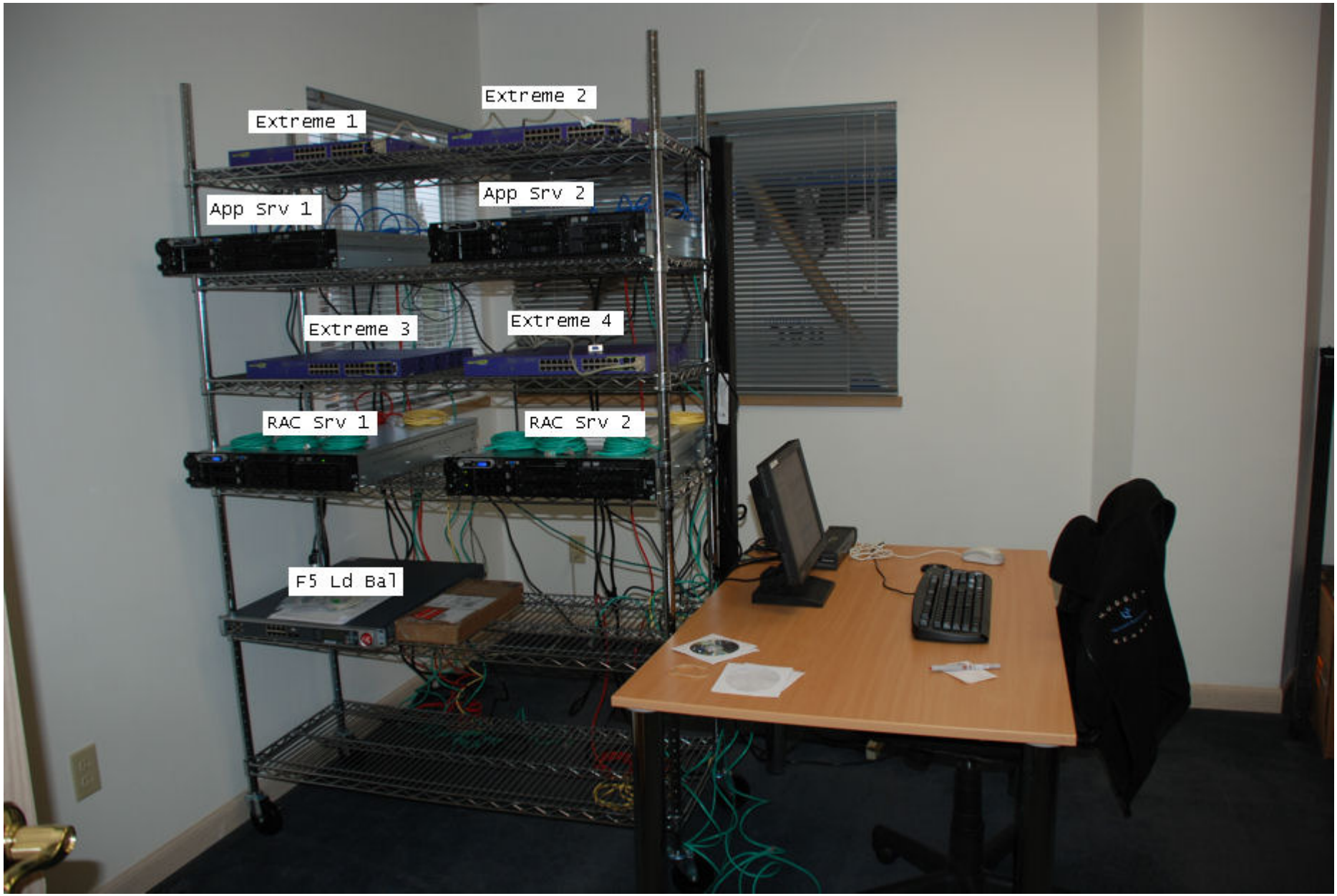
Destructive Testing



Interconnect Features

- Low latency for short messages
- High speed and sustained data rates for large messages
- Low Host-CPU utilization per message
- Flow Control, Error Control, High throughput
- Private to the nodes
- Use Jumbo Frames (MTU 8960)


Logical Network Diagram
Each cluster requires 4 separate networks
F5 Load Balancer
Application Network
Application Servers
Database "public" Network
Database Servers
Database "private" Network
Storage Area Network
NetApp Storage Array


Extreme 2
Extreme 1
App Srv 2
App Srv 1
Extreme 3
Extreme 4
RAC Srv 1
RAC Srv 2
F5 Ld Bal

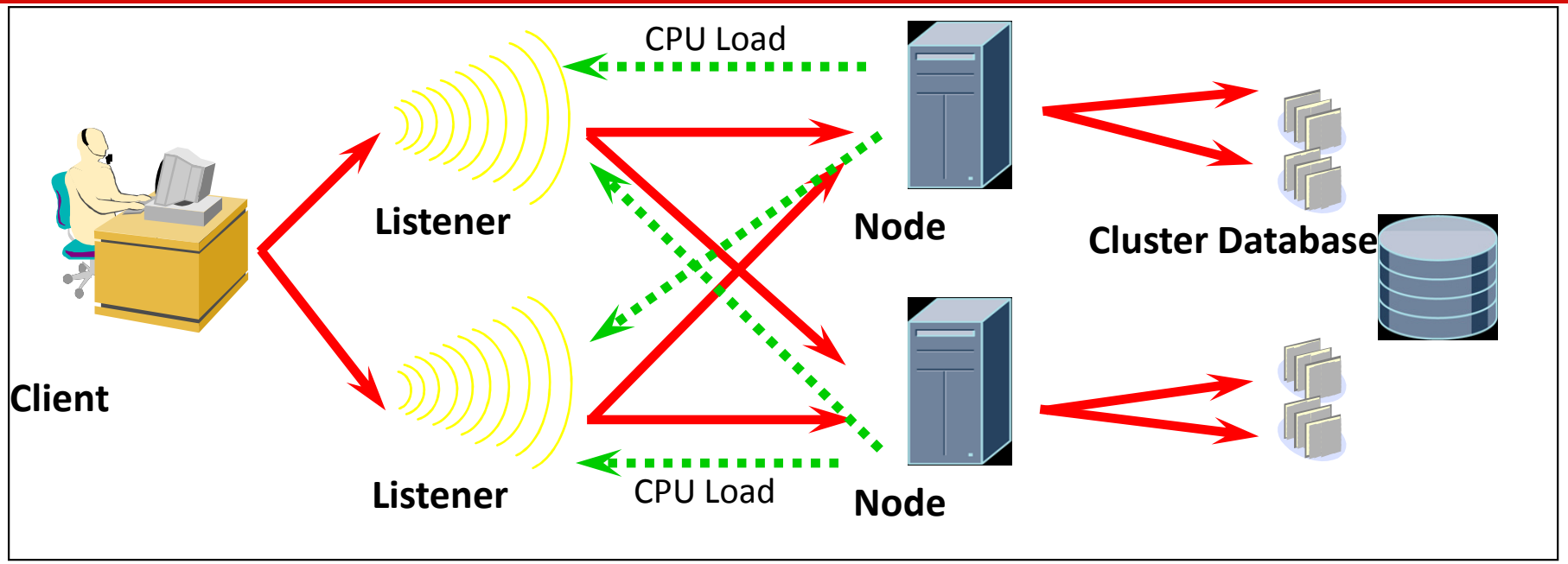
Key Concepts

- Connection Load Balancing
- Transparent Application Failover (TAF)
- Fast Application Notification (FAN)
- Fast Connection Failover (FCF)
- Load Balancing Advisory
- Runtime Connection Load Balancing
- Database Services
- Adaptive Parallel Query

Connection Load Balancing

- Client side
  - Randomly distributes connection requests across listeners on all available nodes
  - Connect time failover automatically chooses another listener if the first one is not available
- Server side
  - Listener subscribes to Load Balance Advisory
  - Connection request is redirected to least loaded, running instance (via its listener)
  - Define goal of LONG or SHORT for each service
- Oracle recommends using both

Connection Load Balancing



- Databases register with listeners when started (PMON)
- Nodes report CPU usage back to registered listeners (SMON)
- Client randomly distributes connections across listeners
- Listener chooses least used node and redirects connection
- Supports both Shared Server and Dedicated Server Configurations
- Uses built-in Load Balancing Advisory

Transparent Application Failover - TAF

- Configured as part of a Service
- Uses PREFERRED and AVAILABLE instances
- Uses BASIC or PRECONNECT policy
- Upon failure of an instance/node:
  - Connections re-established with a surviving instance
  - SELECT statements may resume from the same point using the same SCN
  - Transactions are rolled back
  - Session state (variables, ALTER SESSION, etc) is lost
- Implemented at the OCI/Net layer on a per-connection basis
- Suitable for traditional client-server connections

Fast Application Notification - FAN (1:4)

- Method by which applications are notified immediately of cluster changes
  (eg. node UP/DOWN events, workload)
- Applications can react immediately
- Part of High Availability Framework
- Uses Advanced Queuing (AQ) and Oracle Notification Server (ONS)

Fast Application Notification - FAN (2:4)

- Three categories of FAN events
  1. Service events (eg. service up/down)
  2. Node events (eg. cluster membership)
  3. Load Balancing Events

Also integrated with the Listener and Configuration Manager at the Net Services Layer

Fast Application Notification - FAN (3:4)

- Three ways FAN can be utilized
  1. Use FCF client - JDBC, ODP.NET, OCI
  2. Server side callouts
  3. Application can subscribe to FAN events using the ONS API

Fast Application Notification - FAN (4:4)

- Typical responses to FAN events include:
  - Load balancing connections
  - Load balancing work requests
  - Removing connections from pool when instances fail
  - Adding new connections when instances become available and rebalancing load
- Use server side callouts to:
  - Opening fault tracking tickets
  - Sending messages to e-mail/pagers
  - Start and stop server-side applications
  - Relocate low-priority services when high priority services come online

Fast Connection Failover - FCF

- Subscriber of FAN events
- Built-in to:
  - JDBC (thick & thin)
  - ODP.NET
  - OCI
- Cleans up connections when failures occur
- Distributes work requests across available instances

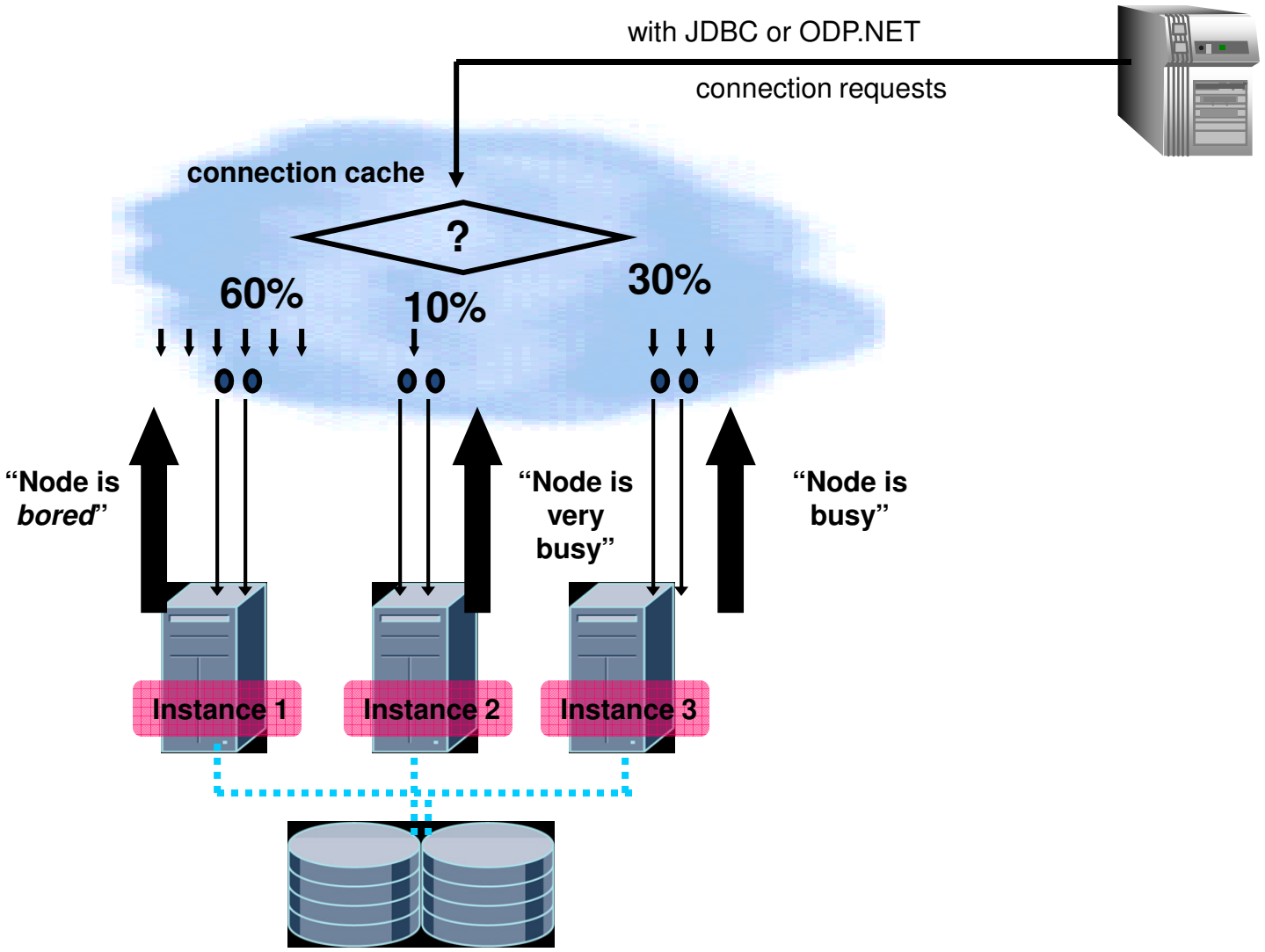
Load Balancing Advisory

- Monitors workload activity for a service across all instances in the cluster
- Analyzes the service level for each instance based on defined goal (service time or throughput)
- Publishes FAN events describing the current service level, amount of work to be sent to each instance and data quality flag
- Integrated with AWR
- If no goal defined, no load balancing events sent.

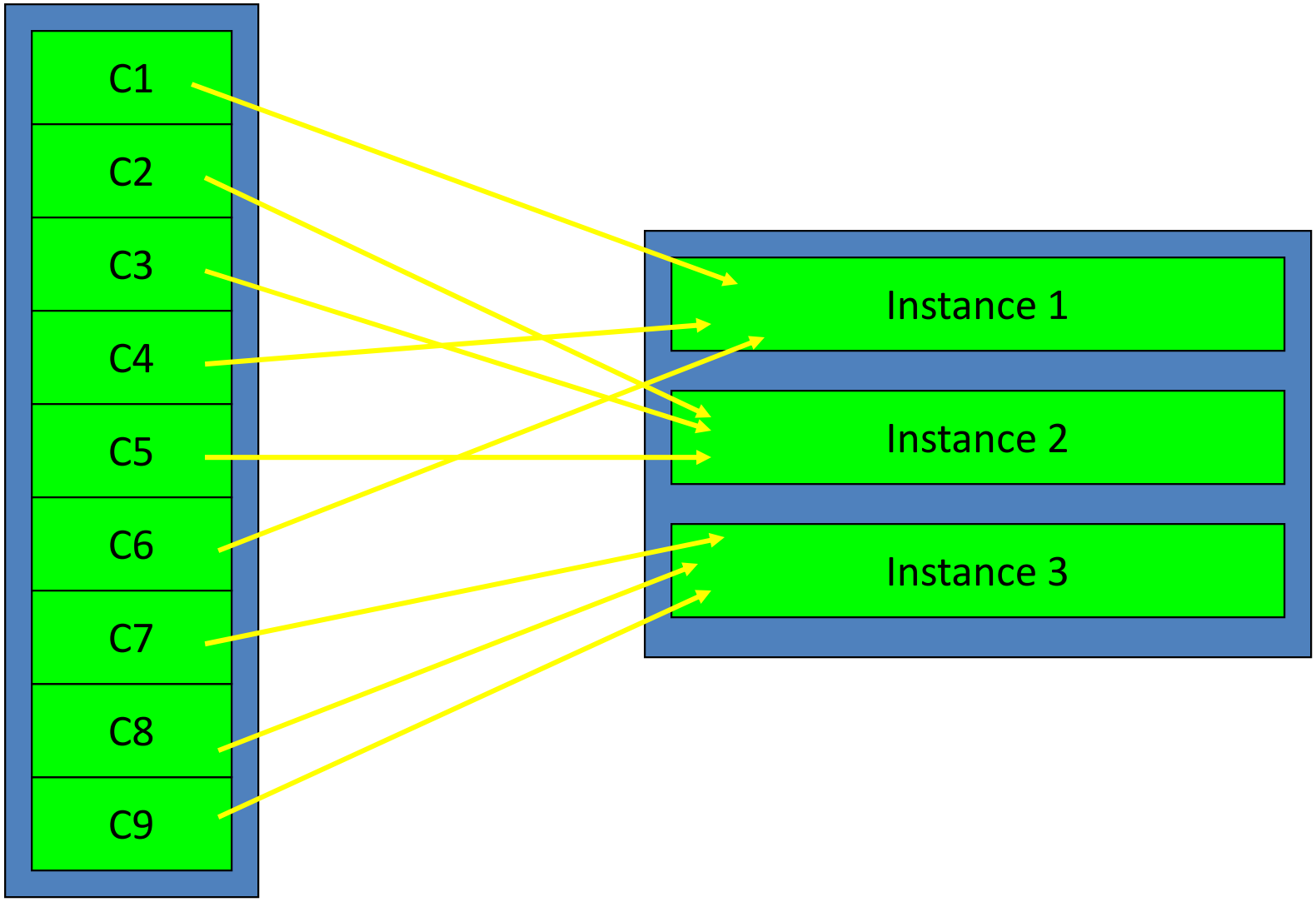
Runtime Connection Load Balancing

- Works with the connection pool (eg. App Server layer)
- Distributes “units of work” across instances in a cluster according to the Load Balancing Advisory
- Most suitable instance is chosen for each work request

Runtime Connection Load Balancing



Node Joins ... Clients Join

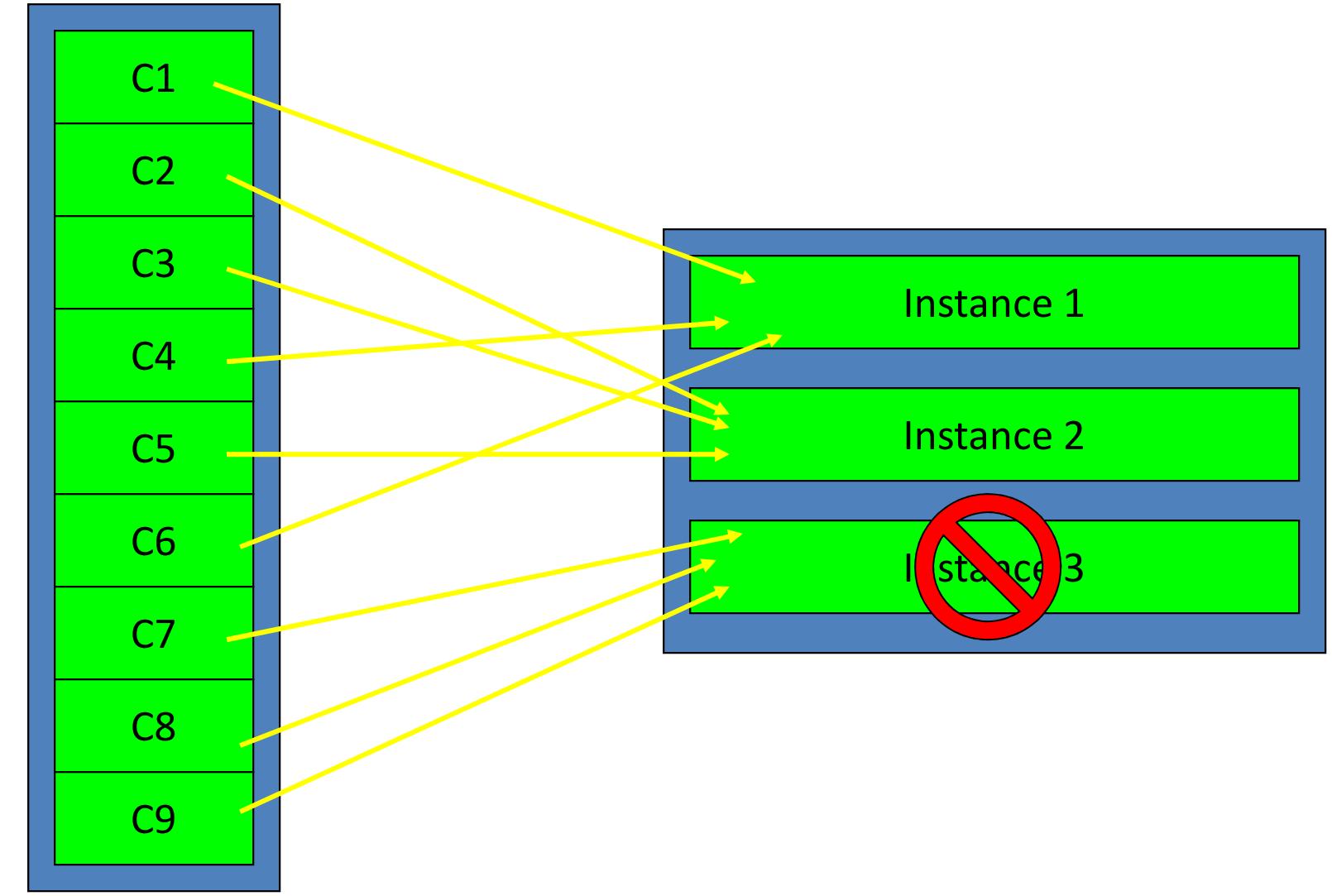


Connection Pool

Rebalancing Upon Instance-Client Join

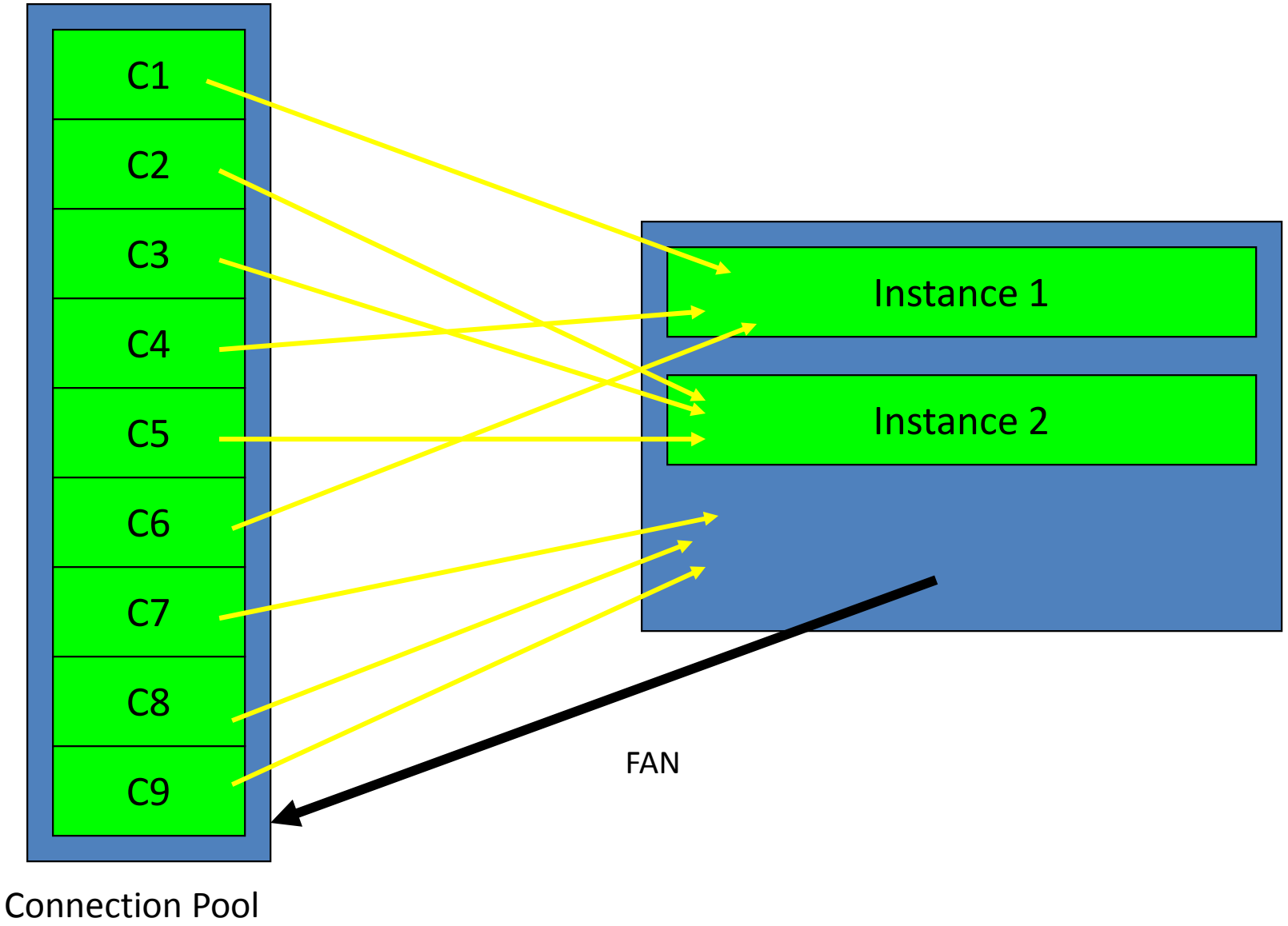
- Desire: Create new connections to new instances and potentially disable some old ones.
- Method:
  - 10g and above ... Fast Application Notification (FAN)
    - Using the HA up and down events for RAC services

FAN & Connection Pools

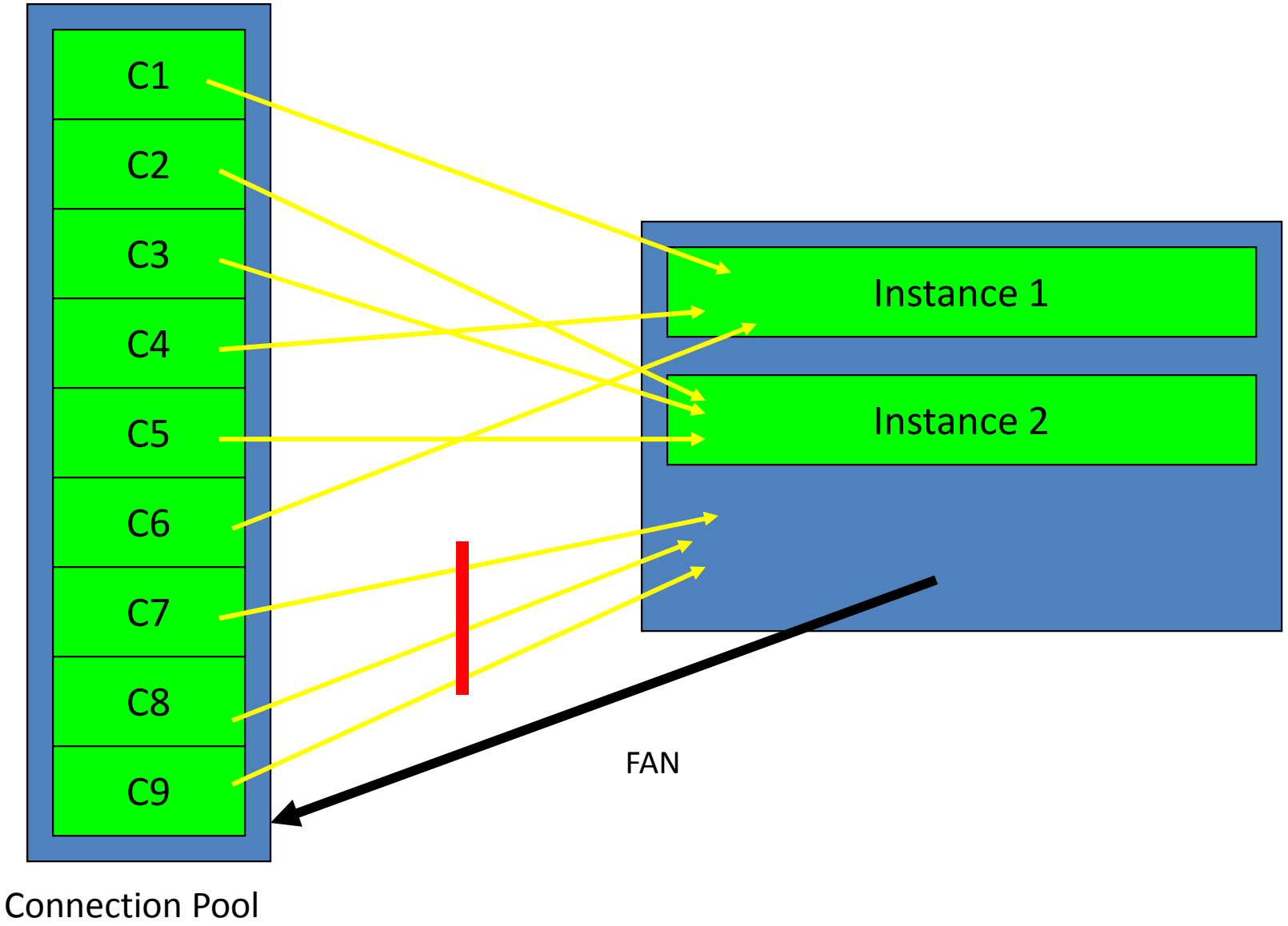


Connection Pool

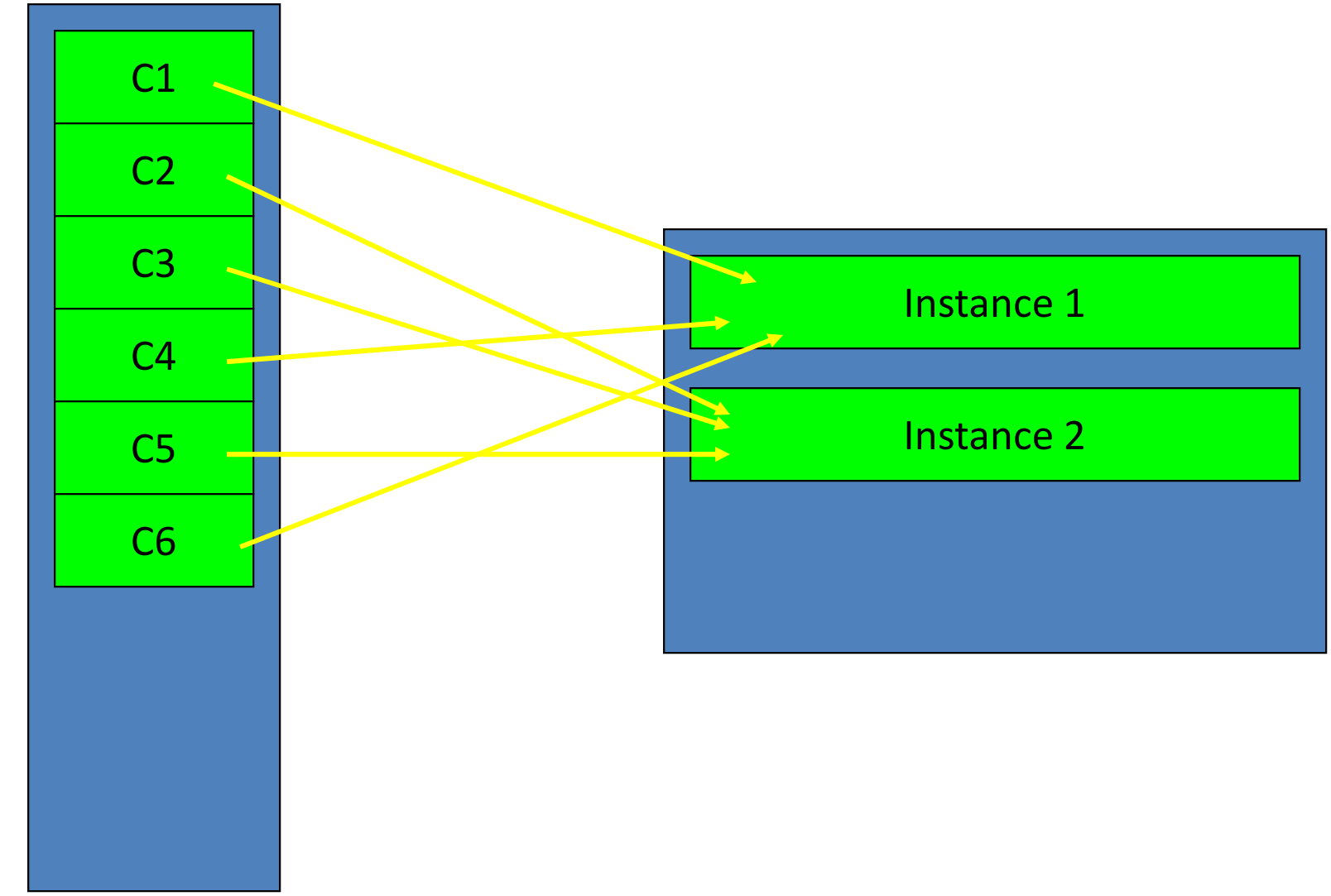
Node Leaves


C1
C2
C3
C4
C5
C6
C7
C8
C9
Instance 1
Instance 2
FAN
Connection Pool

Node Leaves


C1
C2
C3
C4
C5
C6
C7
C8
C9
Instance 1
Instance 2
FAN
Connection Pool

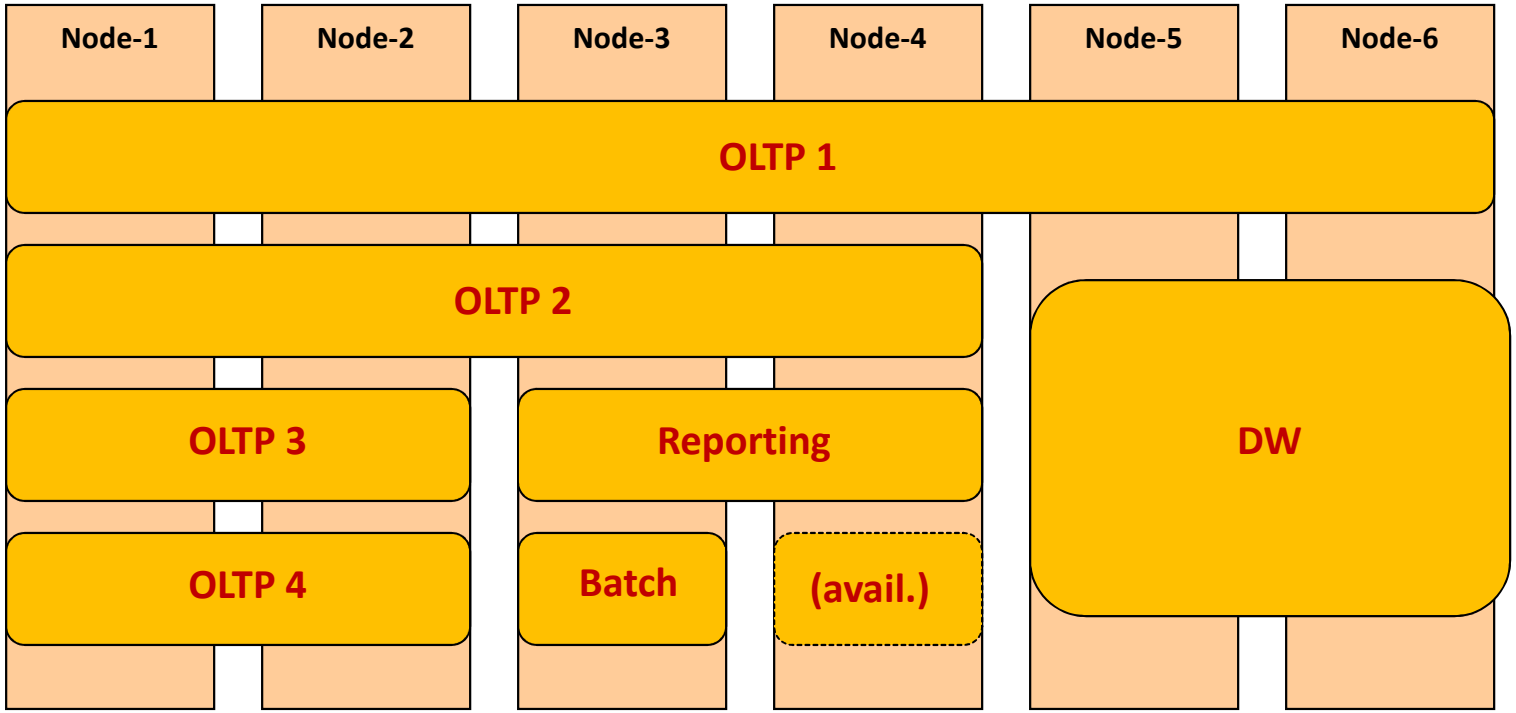
Node Leaves



Services

- Applications with similar needs connect to a Service rather than a database or instance
- One Service can be spread across multiple instances
- Multiple Services (eg. oltp, batch, reports) can be defined and mapped to different instances
- When an instance fails, the Service is not necessarily interrupted
- Services are required to take advantage of High Availability features

Services and Node Affinity



Services may be mapped to instances and resource plans

References

Bryn Llewellyn's White Papers

Caveat

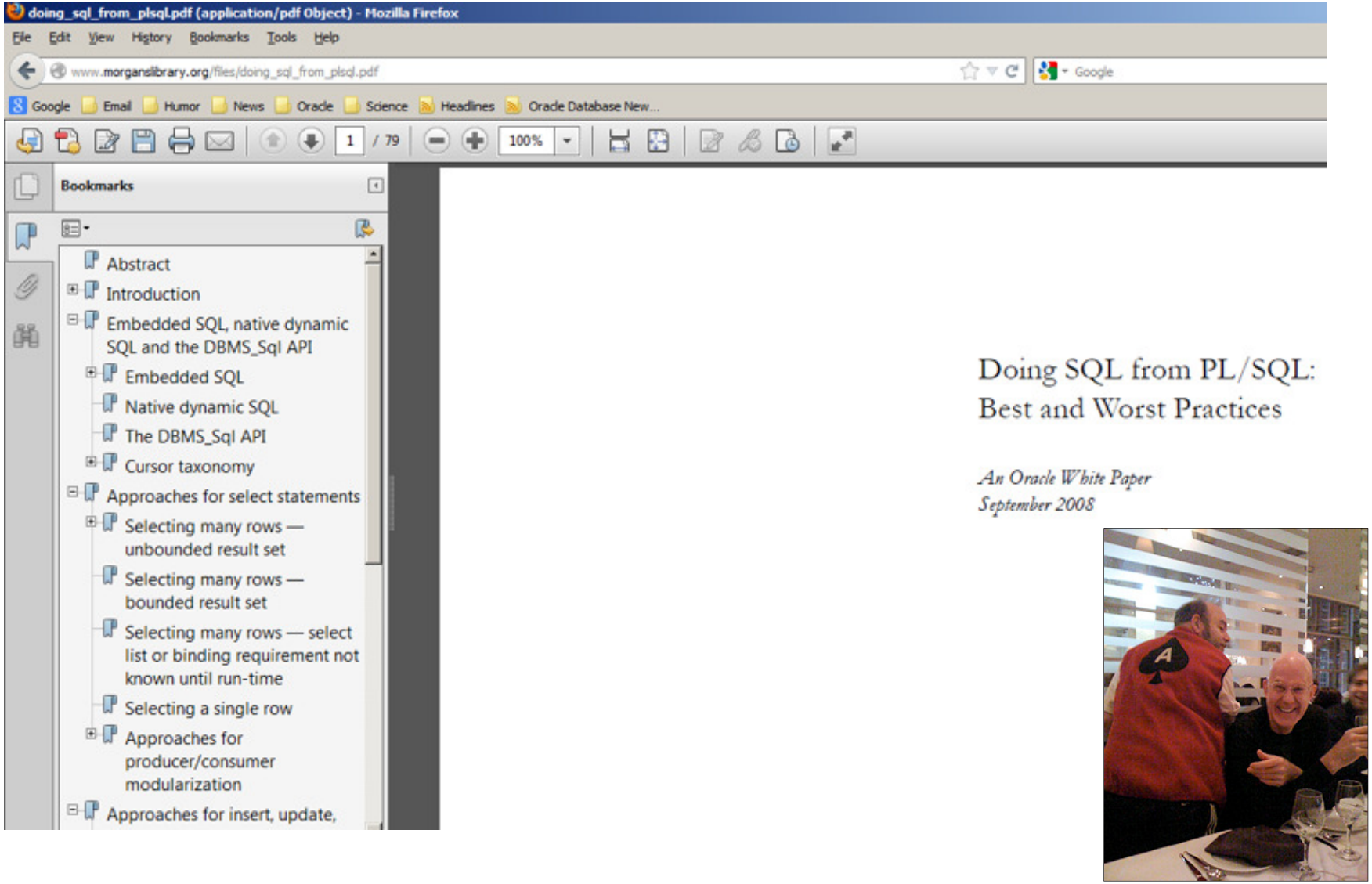
Prescribing best practice principles for programming any 3GL is phenomenally difficult. One of the hardest challenges is the safety of the assumption that the reader starts out with these qualities:

- Has chosen the right parents[3].
- Has natural common sense coupled with well-developed verbal reasoning skills.
- Has an ability to visualize mechanical systems.
- Requires excellence from self and others.
- Has first class negotiating skills. (Good code takes longer to write and test than bad code; managers want code delivered in aggressive timeframes.)
- Has received a first class education.
- Can write excellent technical prose. (How else can you write the requirements for your code, write the test specifications, and discuss problems that arise along the way?)

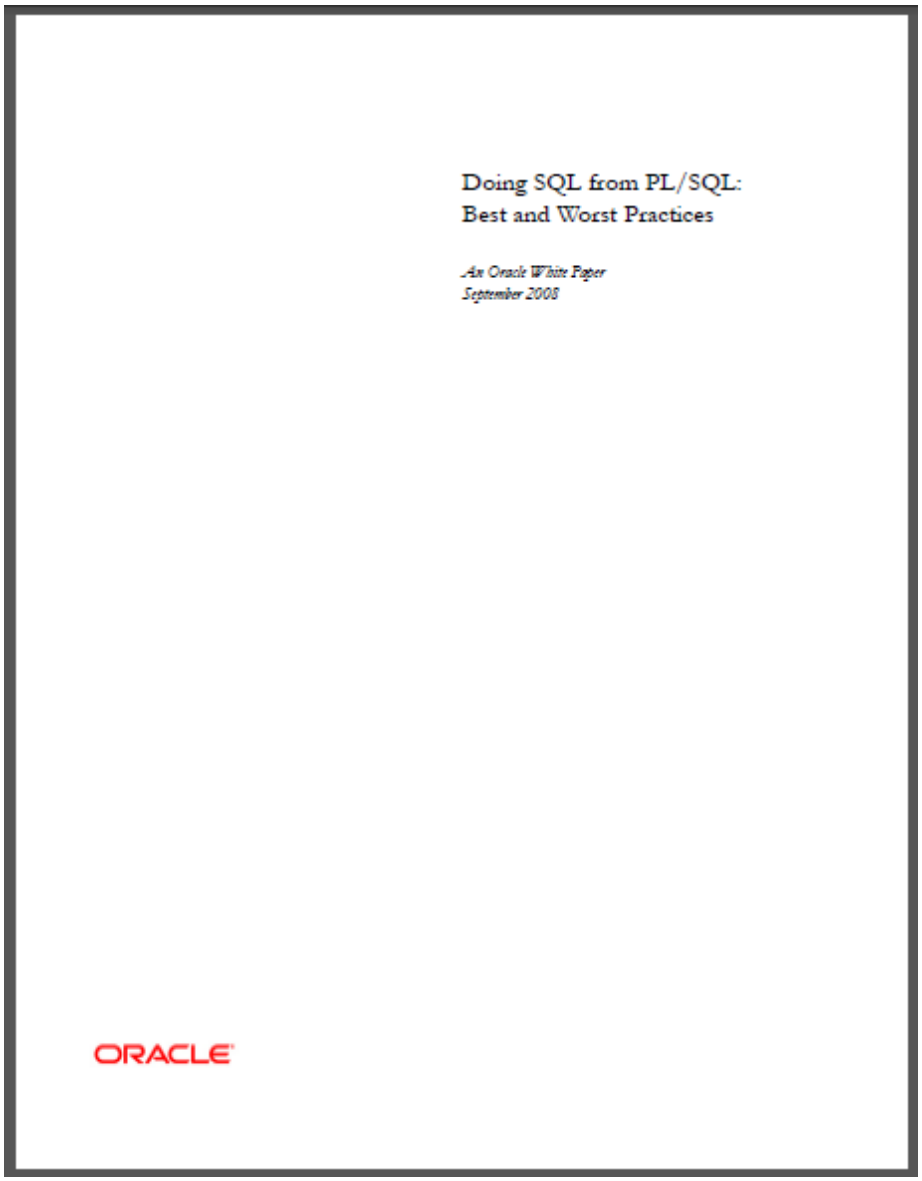
Then, the reader would be fortunate enough to work in an environment which provides intellectual succor:

- Has easy access to one or several excellent mentors.

Bryn Llewellyn's White Paper



Brynn's White Paper



Other References

- http://tahiti.oracle.com
- Any book written by Tom Kyte
- http://asktom.oracle.com
- http://morganslibrary.com/library.html

Questions

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

Thank you