从自动化到自治

-Oracle 优化案例实践和 19c 新特性揭秘

盖国强

云和恩墨 创始人，首席架构师

自我介绍

□ 盖国强　云和恩墨信息技术有限公司　创始人

- □ 国内第一个Oracle ACE及ACE总监；
- □ 致力于技术分享与传播
  - ○ 自2001年，技术论坛ITPUB的主要倡导者之一；
  - ○ 自2003年，已经编辑、编译、著作出版了14本技术书籍；
  - ○ 自2010年，和张乐奕共创 Oracle用户组 - ACOUG,开展持续的公益活动；
- □ 走在技术创业的道路上
  - ○ 自2008年，尝试单飞与创业;
  - ○ 自2011年，和创业伙伴发起成立云和恩墨；
  - ○ 自2019年，公司伙伴超过500人；

ORACLE ACE Director

ACOUG
All China Oracle User Group
中国 Oracle 用户组

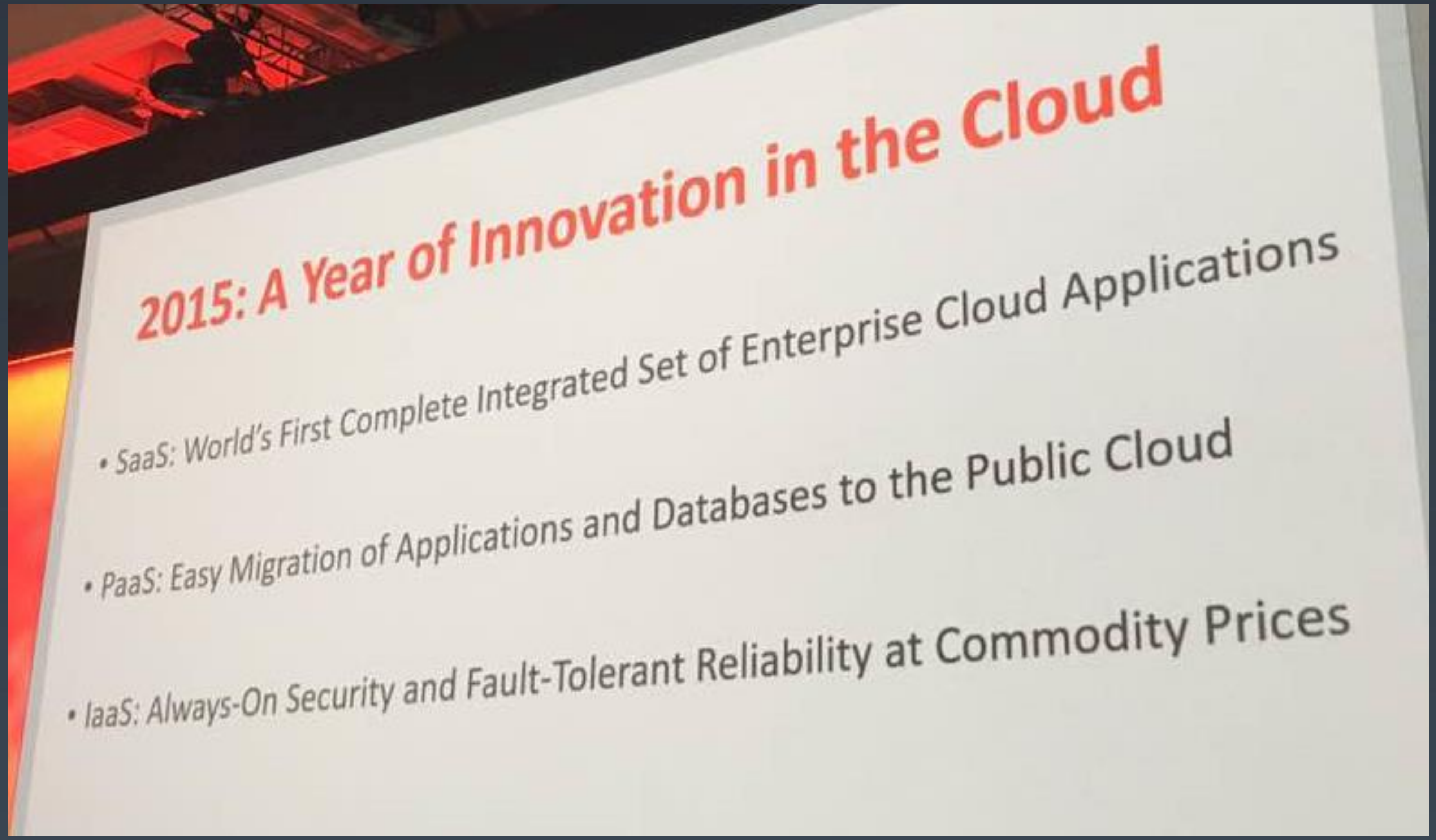
云和恩墨
ENMOTECH

Oracle Performance Tuning
Oracle 数据库性能优化
深入浅出Oracle
DBA 入门、进阶与诊断案例
IT名人堂
循序渐进 Oracle 数据库管理、优化与备份恢复
深入解析 Oracle DBA入门、进阶与诊断案例
Oracle DBA手记
海量数据库解决方案
循序渐进 Oracle 数据库管理、优化与备份恢复
Oracle DBA手记 3
数据安全警示录

目　录

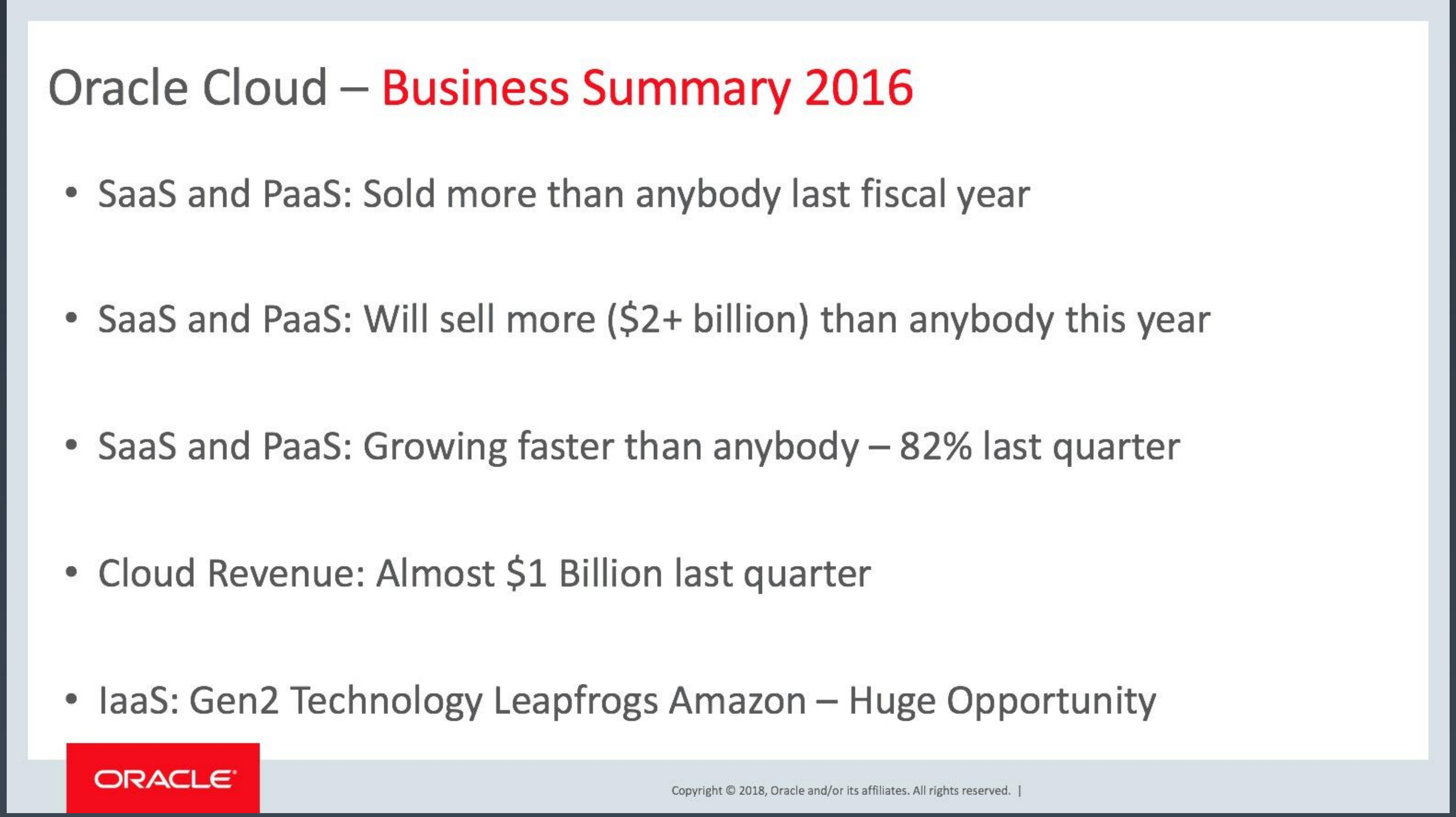
- ➢ **Oracle 的变革之路**
- ➢ 性能优化与数据库进化
- ➢ Oracle 19c新特性

Oracle 的云上变革之路 – 全面转向



2015: A Year of Innovation in the Cloud

- SaaS: World's First Complete Integrated Set of Enterprise Cloud Applications
- PaaS: Easy Migration of Applications and Databases to the Public Cloud
- IaaS: Always-On Security and Fault-Tolerant Reliability at Commodity Prices

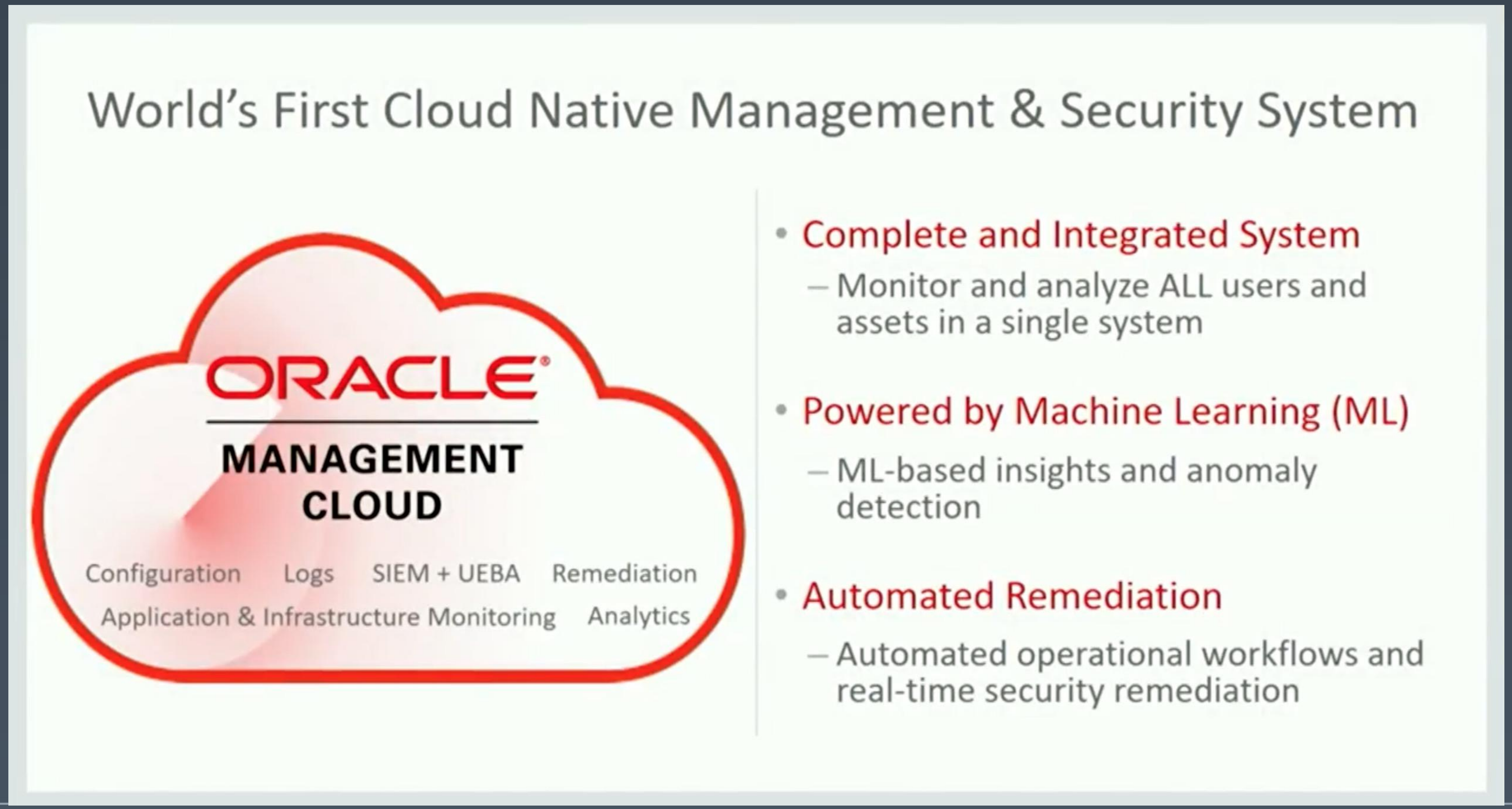


Oracle Cloud – Business Summary 2016

- SaaS and PaaS: Sold more than anybody last fiscal year
- SaaS and PaaS: Will sell more ($2+ billion) than anybody this year
- SaaS and PaaS: Growing faster than anybody – 82% last quarter
- Cloud Revenue: Almost $1 Billion last quarter
- IaaS: Gen2 Technology Leapfrogs Amazon – Huge Opportunity

ORACLE

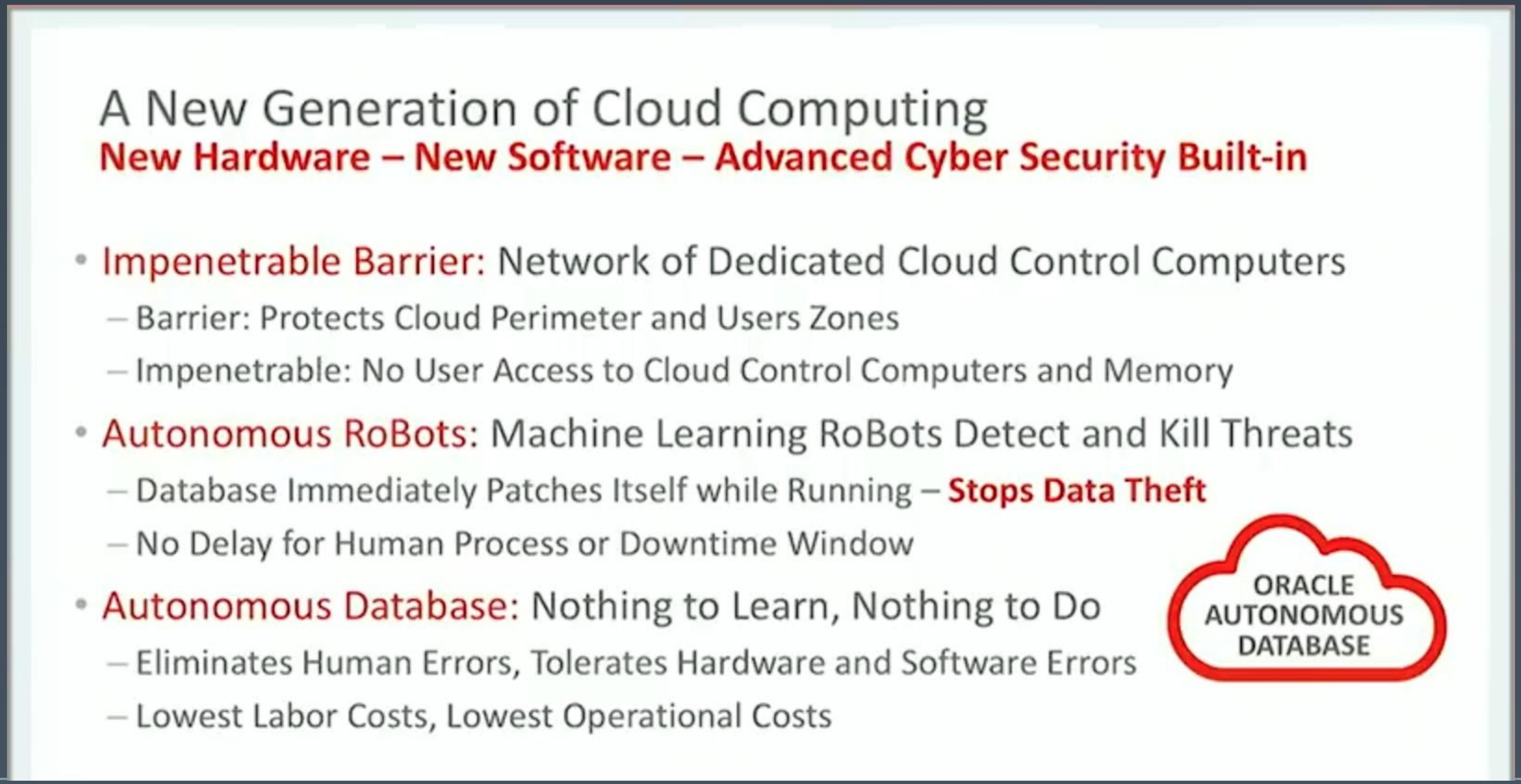
Copyright © 2018, Oracle and/or its affiliates. All rights reserved. |



World's First Cloud Native Management & Security System

ORACLE
MANAGEMENT CLOUD
Configuration
Logs
SIEM + UEBA
Remediation
Application & Infrastructure Monitoring
Analytics

- Complete and Integrated System
  - Monitor and analyze ALL users and assets in a single system
- Powered by Machine Learning (ML)
  - ML-based insights and anomaly detection
- Automated Remediation
  - Automated operational workflows and real-time security remediation

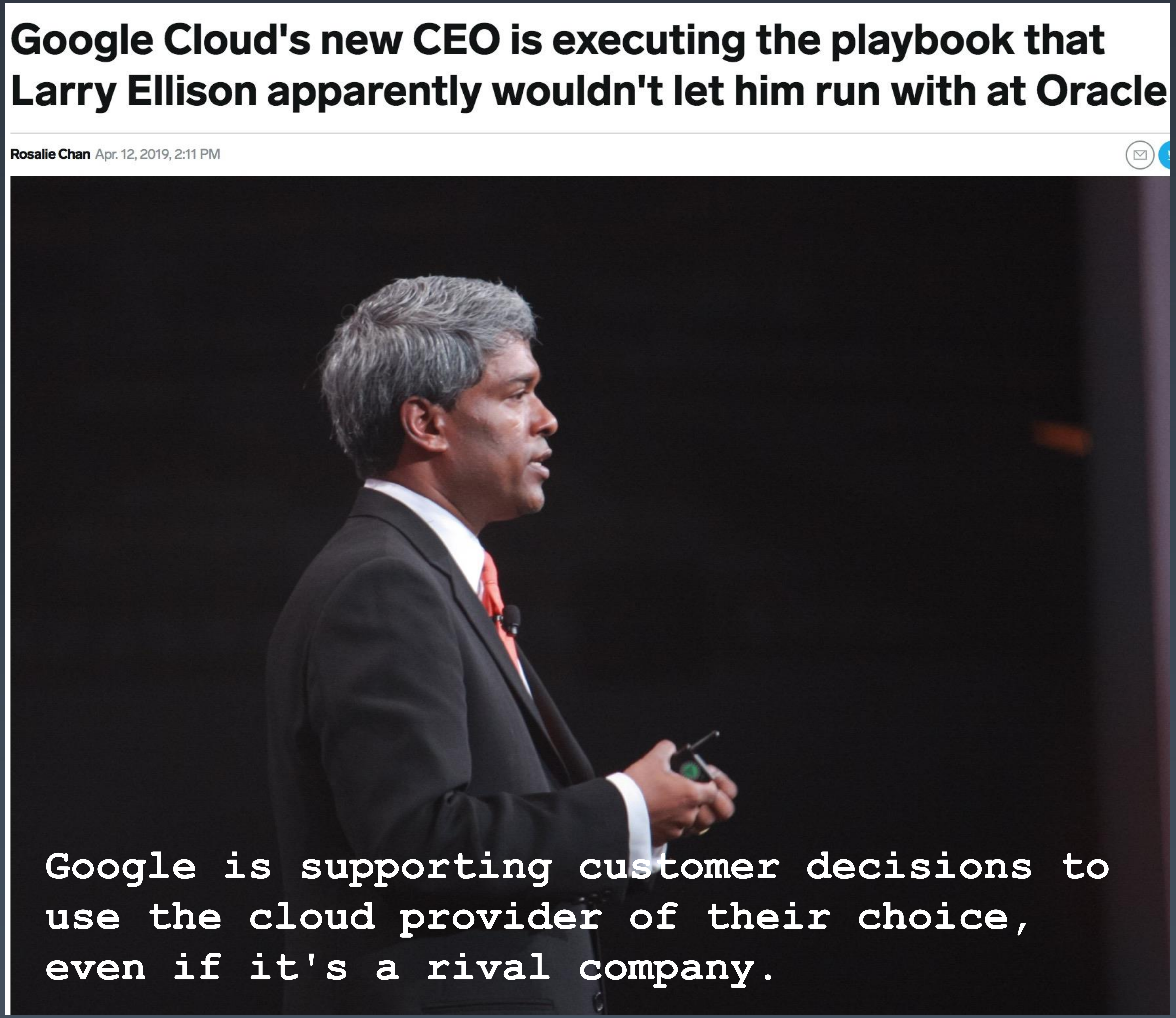


A New Generation of Cloud Computing

New Hardware – New Software – Advanced Cyber Security Built-in

- Impenetrable Barrier: Network of Dedicated Cloud Control Computers
  - Barrier: Protects Cloud Perimeter and Users Zones
  - Impenetrable: No User Access to Cloud Control Computers and Memory
- Autonomous RoBots: Machine Learning RoBots Detect and Kill Threats
  - Database Immediately Patches Itself while Running – **Stops Data Theft**
  - No Delay for Human Process or Downtime Window
- Autonomous Database: Nothing to Learn, Nothing to Do
  - Eliminates Human Errors, Tolerates Hardware and Software Errors
  - Lowest Labor Costs, Lowest Operational Costs

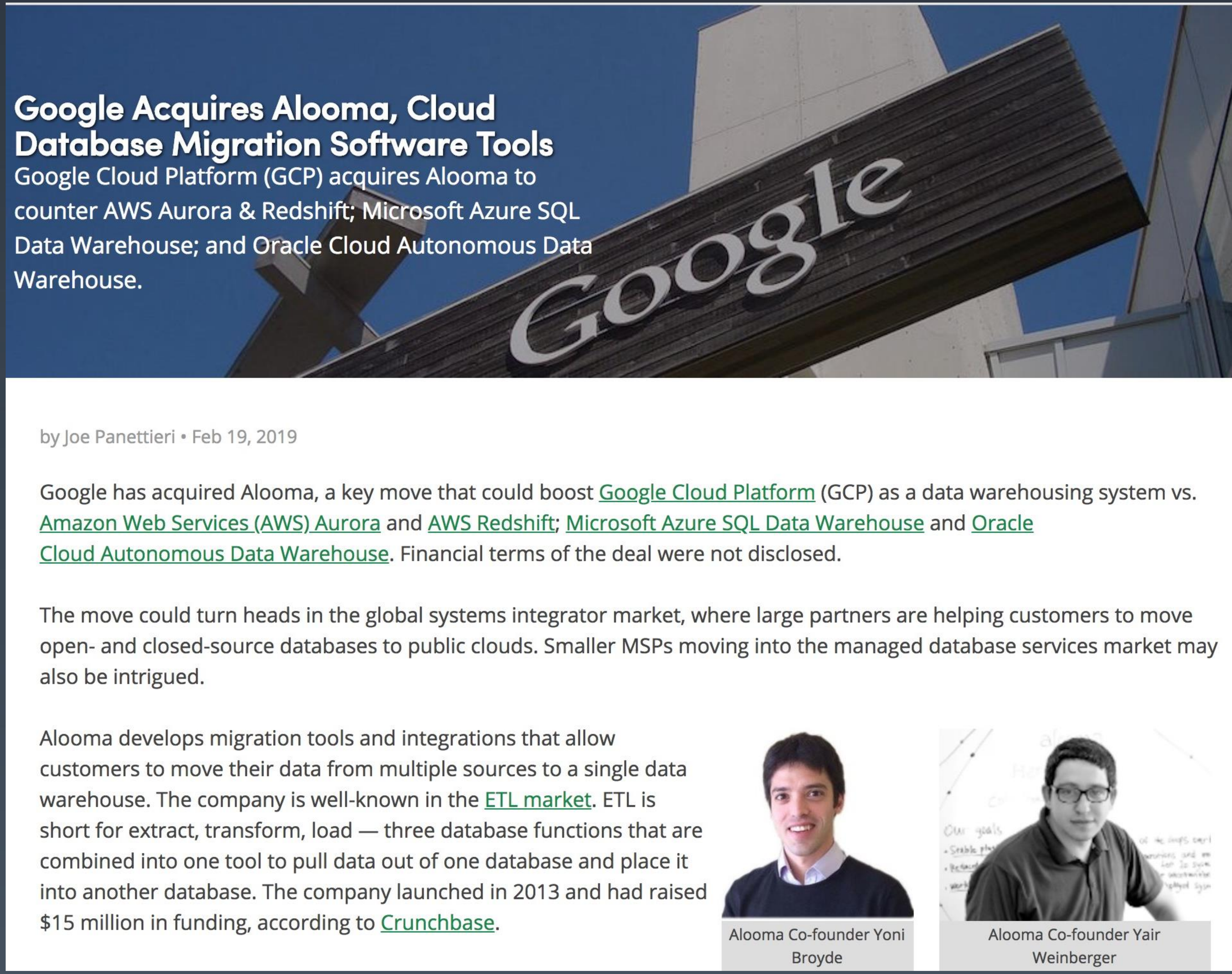
Oracle 的云上变革之路 – 数据为王



Google Cloud's new CEO is executing the playbook that Larry Ellison apparently wouldn't let him run with at Oracle

Rosalie Chan Apr. 12, 2019, 2:11 PM

Google is supporting customer decisions to use the cloud provider of their choice, even if it's a rival company.



Google Acquires Alooma, Cloud Database Migration Software Tools

Google Cloud Platform (GCP) acquires Alooma to counter AWS Aurora & Redshift; Microsoft Azure SQL Data Warehouse; and Oracle Cloud Autonomous Data Warehouse.

by Joe Panettieri • Feb 19, 2019

Google has acquired Alooma, a key move that could boost Google Cloud Platform (GCP) as a data warehousing system vs. Amazon Web Services (AWS) Aurora and AWS Redshift; Microsoft Azure SQL Data Warehouse and Oracle Cloud Autonomous Data Warehouse. Financial terms of the deal were not disclosed.

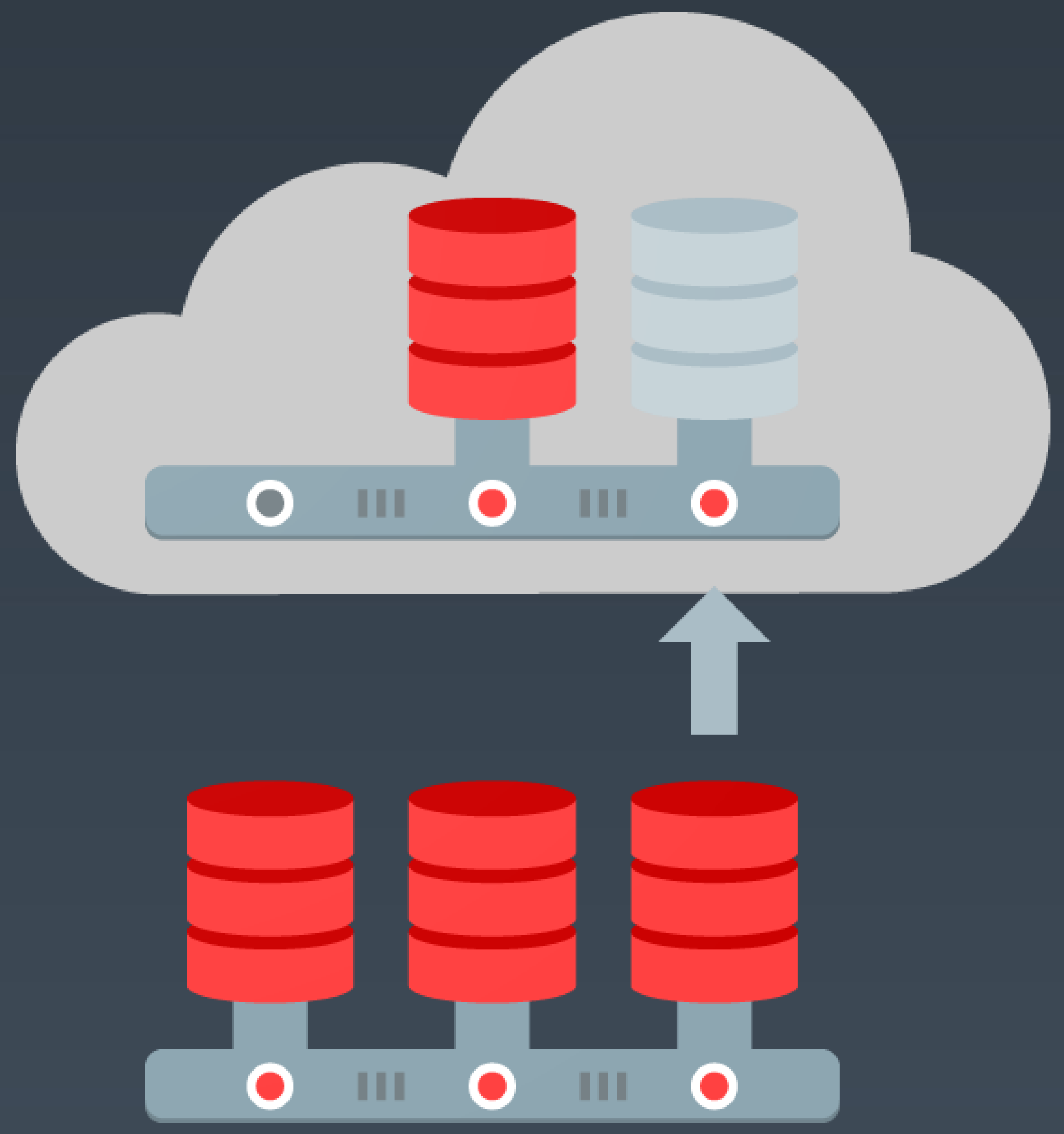
The move could turn heads in the global systems integrator market, where large partners are helping customers to move open- and closed-source databases to public clouds. Smaller MSPs moving into the managed database services market may also be intrigued.

Alooma develops migration tools and integrations that allow customers to move their data from multiple sources to a single data warehouse. The company is well-known in the ETL market. ETL is short for extract, transform, load — three database functions that are combined into one tool to pull data out of one database and place it into another database. The company launched in 2013 and had raised $15 million in funding, according to Crunchbase.

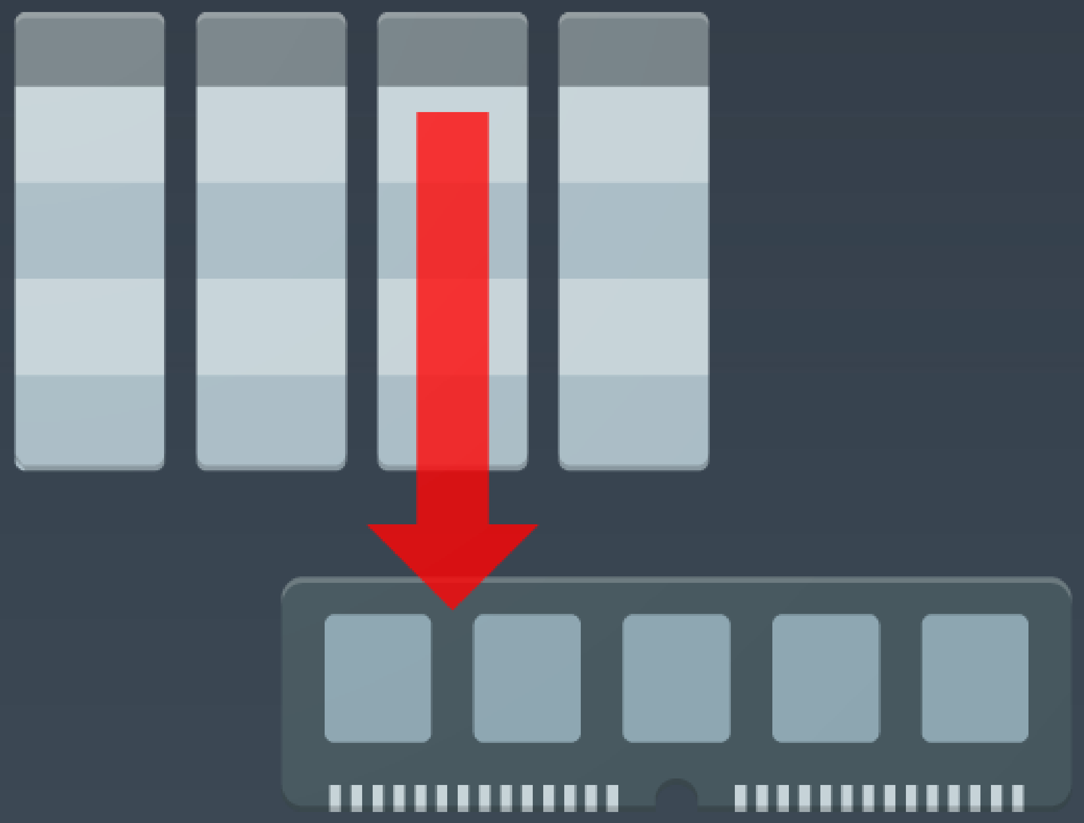
Alooma Co-founder Yoni Broyde

Alooma Co-founder Yair Weinberger

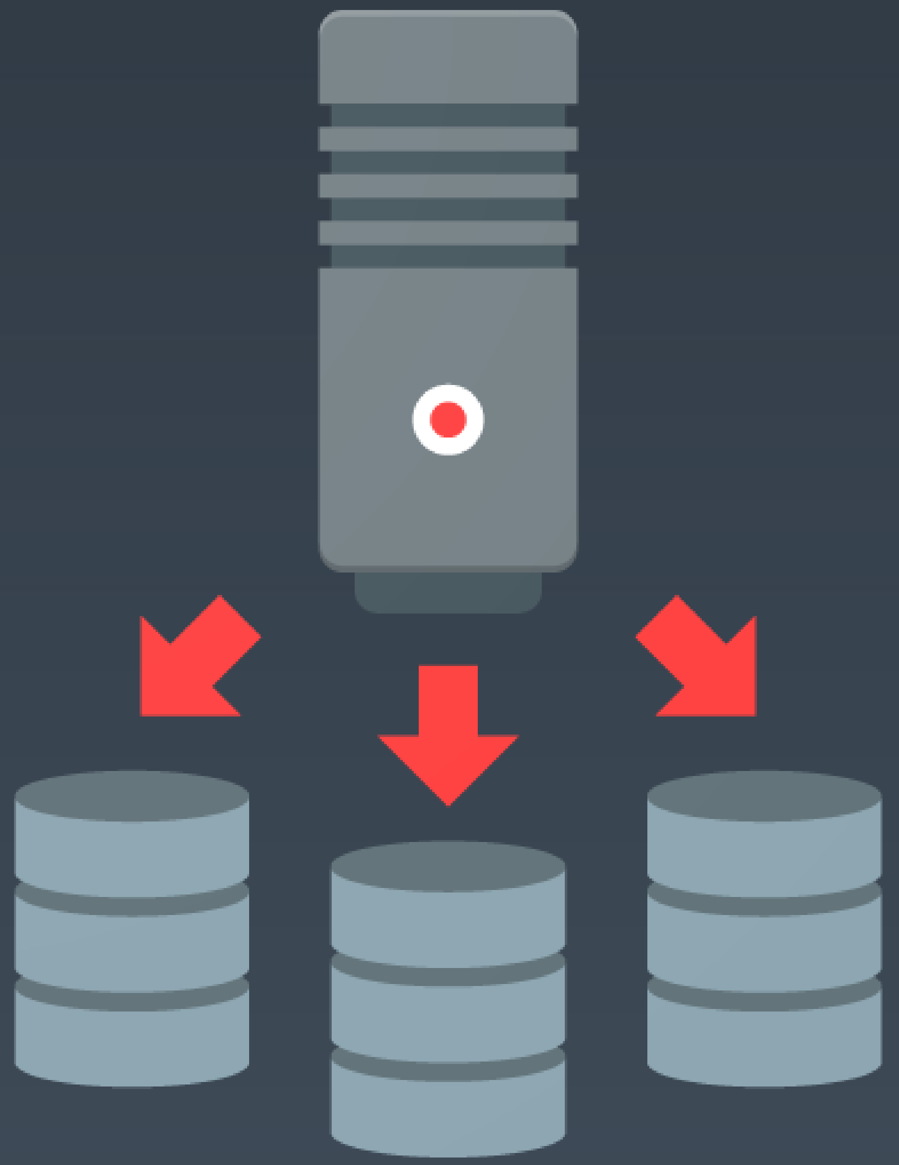
云数据库 – Oracle 12c 的核心特性



多租户：Massive Cost Savings and Cloud Agility with Multitenant



内存存储：Massive Performance with Database In-Memory



数据库分片：Massive Web Applications with Sharding

数据库整合集中、OLTP 和 OLAP 一体、以及大规模并发的负载分散。

自治数据库 – Oracle 18c 的云上部署

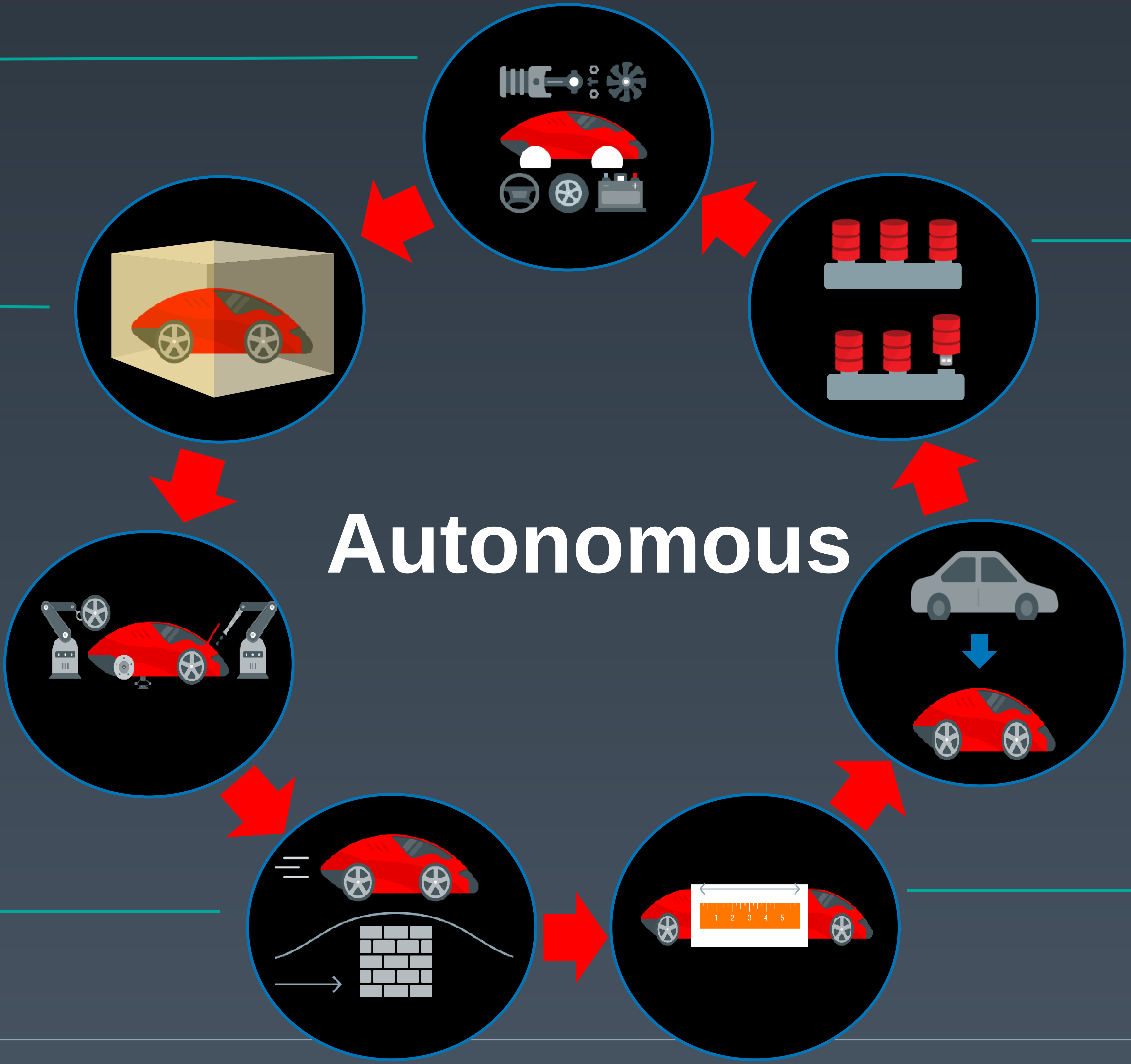
18c

供应 - Provision
Create RAC cluster with Data Guard Standby

安全 - Secure
Encrypt data, Database Vault, apply security patches online

更新 - Update
Online patching and upgrade of database

保护 - Protect
Backup, failover, repair



整合 - Consolidation
User driven PDB and CDB creation

迁移 - Migrate
Easy DB migration, load data from object store

扩展 - Scale
Elastically adjust OCPUs, expand DB

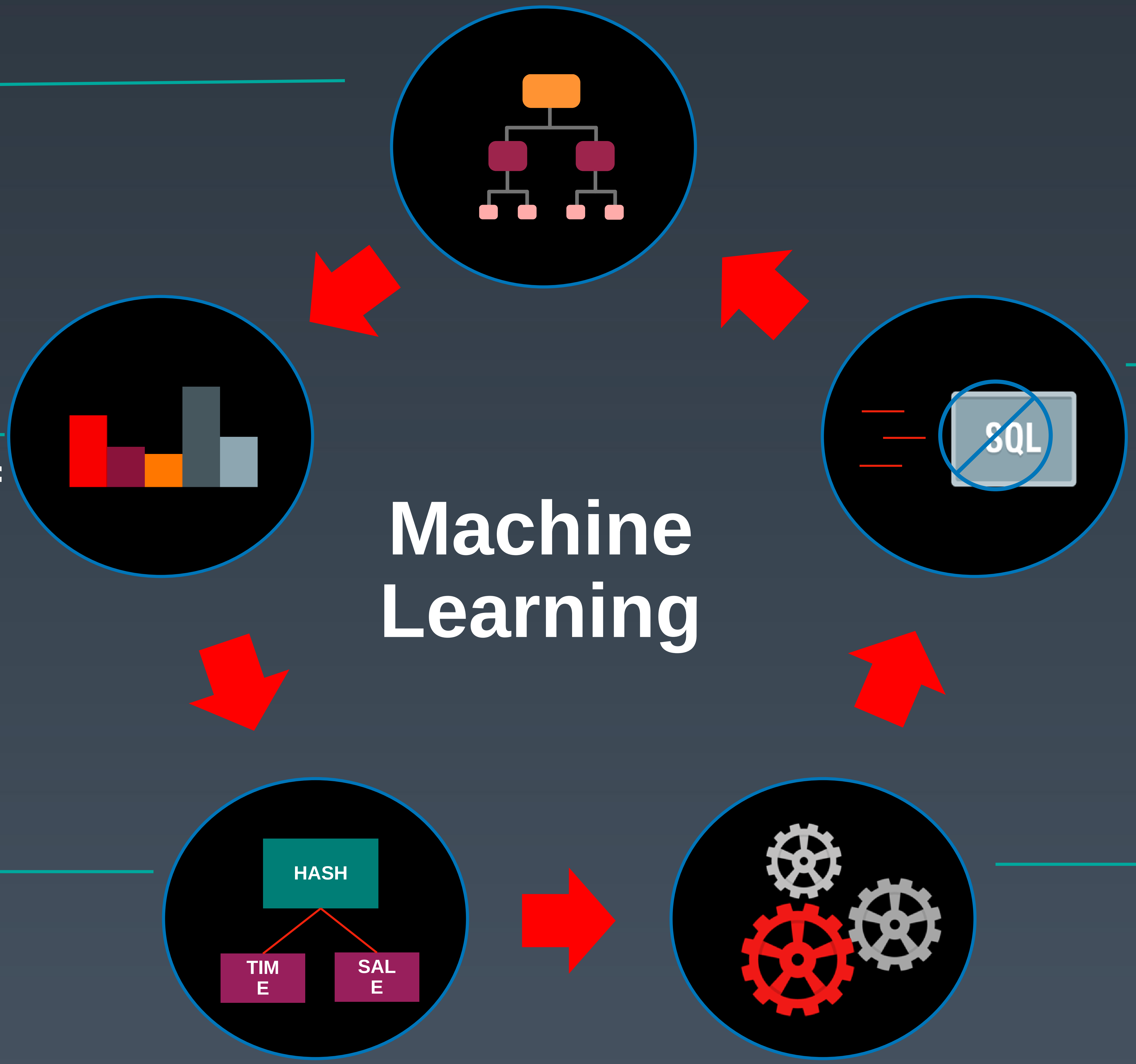
自治数据库 – Oracle 19c 的机器学习

NEW IN 19c

索引 - Indexes
Automatic creation and monitoring of indexes

统计信息 - Statistics
Real-time maintenance of optimizer statistics

执行计划 - Plans
Automatic detection and correction of regressions due to plan changes



控制SQL - Runaway SQL Automatically kill runaway SQL and prevent re-execution until tuned

配置 - Configuration
Automatically tune memory, process, sessions, etc.

目　录

- ➢Oracle 的变革之路

- ➢**性能优化与数据库进化**

- ➢Oracle 19c新特性

我的成长：曾经年少做开发

```
SELECT “SP_TRANS”.“TRANS_NO”, … "SP_ITEM"."CHART_ID","SP_ITEM"."SPECIFICATION"
    FROM  "SP_TRANS", "SP_TRANS_SUB","SP_CHK", "SP_CHK_SUB",
          "SP_RECEIVE", "SP_RECEIVE_SUB", "SP_ITEM"
   WHERE ( "SP_TRANS_SUB"."TRANS_NO" = "SP_TRANS"."TRANS_NO" ) and
           ("SP_TRANS"."BILL_NO" = "SP_CHK"."CHK_NO") and
             ( "SP_CHK_SUB"."CHK_NO" = "SP_CHK"."CHK_NO" ) and
            ( "SP_CHK"."RECEIVE_NO" = "SP_RECEIVE"."RECEIVE_NO" ) and
            ( "SP_RECEIVE_SUB"."RECEIVE_NO" = "SP_RECEIVE"."RECEIVE_NO" ) and
             ……
            ( "SP_CHK_SUB"."COUNTRY" = "SP_RECEIVE_SUB"."COUNTRY" ) and
             ( "SP_CHK_SUB"."PLAN_NO" = "SP_RECEIVE_SUB"."PLAN_NO" ) and
             ( "SP_CHK_SUB"."PLAN_LINE" = "SP_RECEIVE_SUB"."PLAN_LINE" ) and
            (to_char("SP_TRANS"."TRANSDATE" ,'YYYY-MM-DD') >='2003-01-01');

130 rows selected.

Elapsed:  00: 29: 1785.47
```

SQL背后的世界：误入歧途DBA

```
SELECT /*+ ordered */ “SP_TRANS”.“TRANS_NO”, … "SP_ITEM"."CHART_ID","SP_ITEM"."SPECIFICATION"
    FROM  "SP_TRANS", "SP_TRANS_SUB","SP_CHK", "SP_CHK_SUB",
          "SP_RECEIVE", "SP_RECEIVE_SUB", "SP_ITEM"
   WHERE ( "SP_TRANS_SUB"."TRANS_NO" = "SP_TRANS"."TRANS_NO" ) and
           ("SP_TRANS"."BILL_NO" = "SP_CHK"."CHK_NO") and
              ( "SP_CHK_SUB"."CHK_NO" = "SP_CHK"."CHK_NO" ) and
            ( "SP_CHK"."RECEIVE_NO" = "SP_RECEIVE"."RECEIVE_NO" ) and
            ( "SP_RECEIVE_SUB"."RECEIVE_NO" = "SP_RECEIVE"."RECEIVE_NO" ) and
             ……
            ( "SP_CHK_SUB"."COUNTRY" = "SP_RECEIVE_SUB"."COUNTRY" ) and
              ( "SP_CHK_SUB"."PLAN_NO" = "SP_RECEIVE_SUB"."PLAN_NO" ) and
              ( "SP_CHK_SUB"."PLAN_LINE" = "SP_RECEIVE_SUB"."PLAN_LINE" ) and
            (to_char("SP_TRANS"."TRANSDATE" ,'YYYY-MM-DD') >='2003-01-01');

130 rows selected.

Elapsed:  00: 00: 05.67
```

SQL背后的世界：多表如何联合

1 SP_TRANS

2 SP_TRANS_SUB

3 SP_CHK

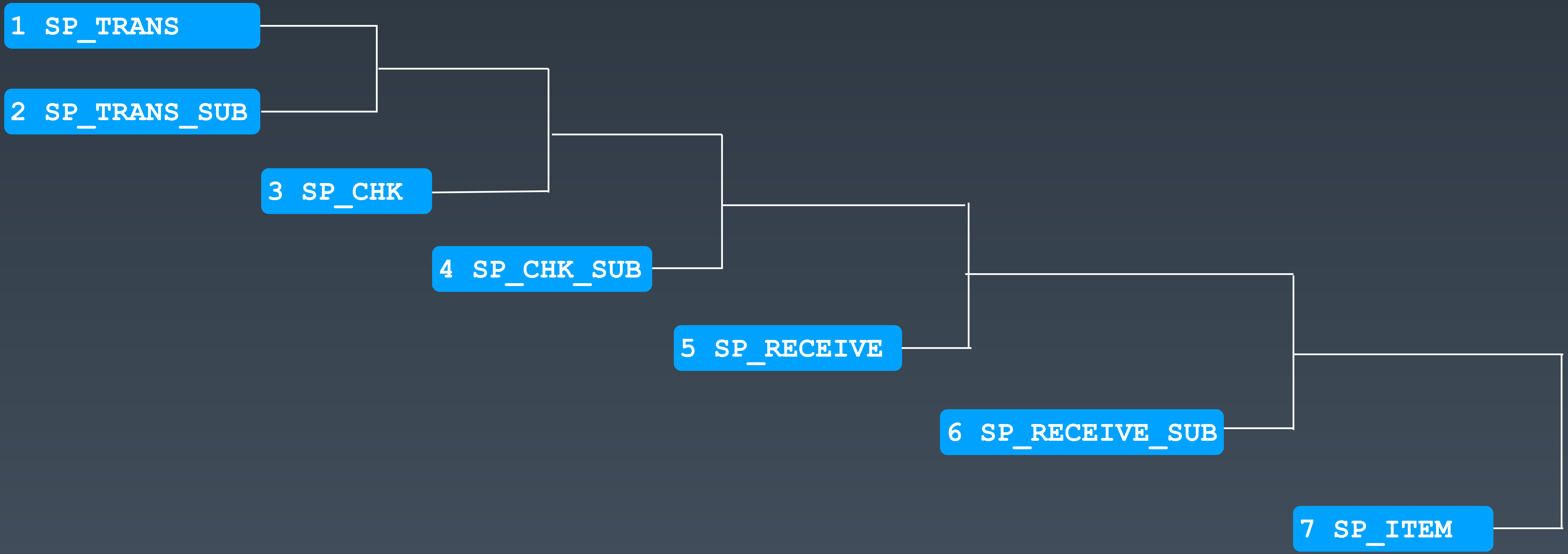
4 SP_CHK_SUB

5 SP_RECEIVE

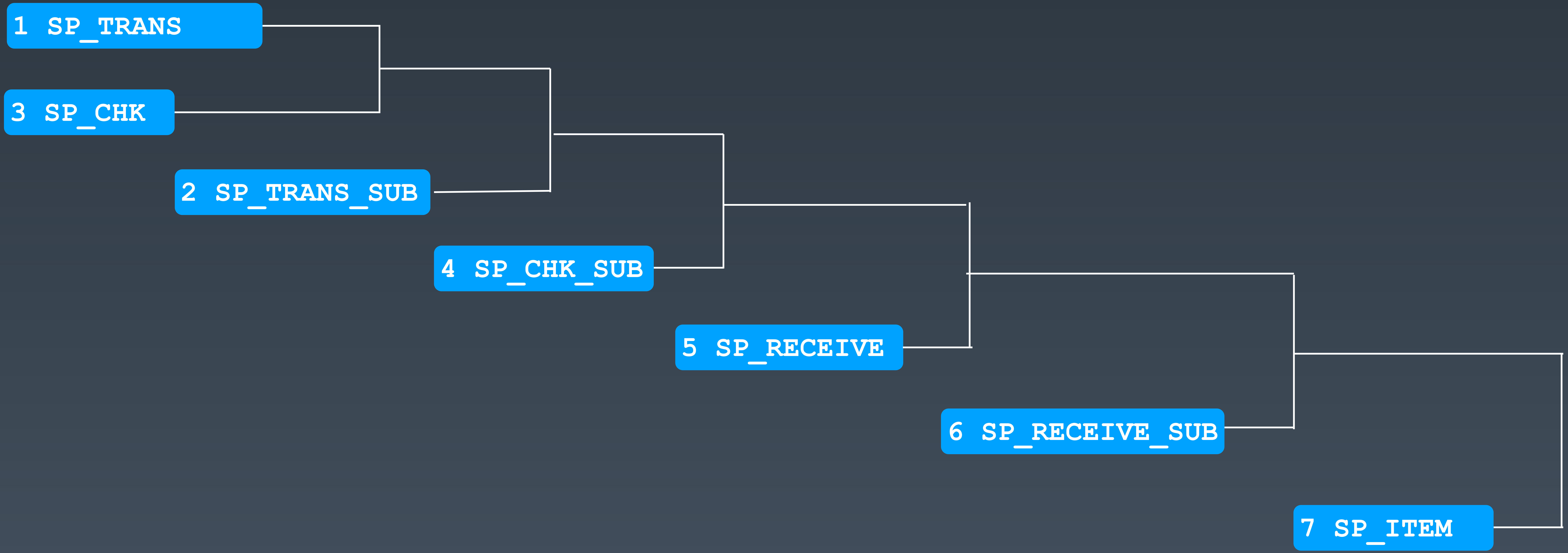
6 SP_RECEIVE_SUB

7 SP_ITEM

SQL背后的世界：多表如何联合



SQL背后的世界：多表如何联合



1 SP_TRANS

3 SP_CHK

2 SP_TRANS_SUB

4 SP_CHK_SUB

5 SP_RECEIVE

6 SP_RECEIVE_SUB

7 SP_ITEM

SQL背后的世界：理解优化器

1 SP_TRANS

2 SP_TRANS_SUB

3 SP_CHK

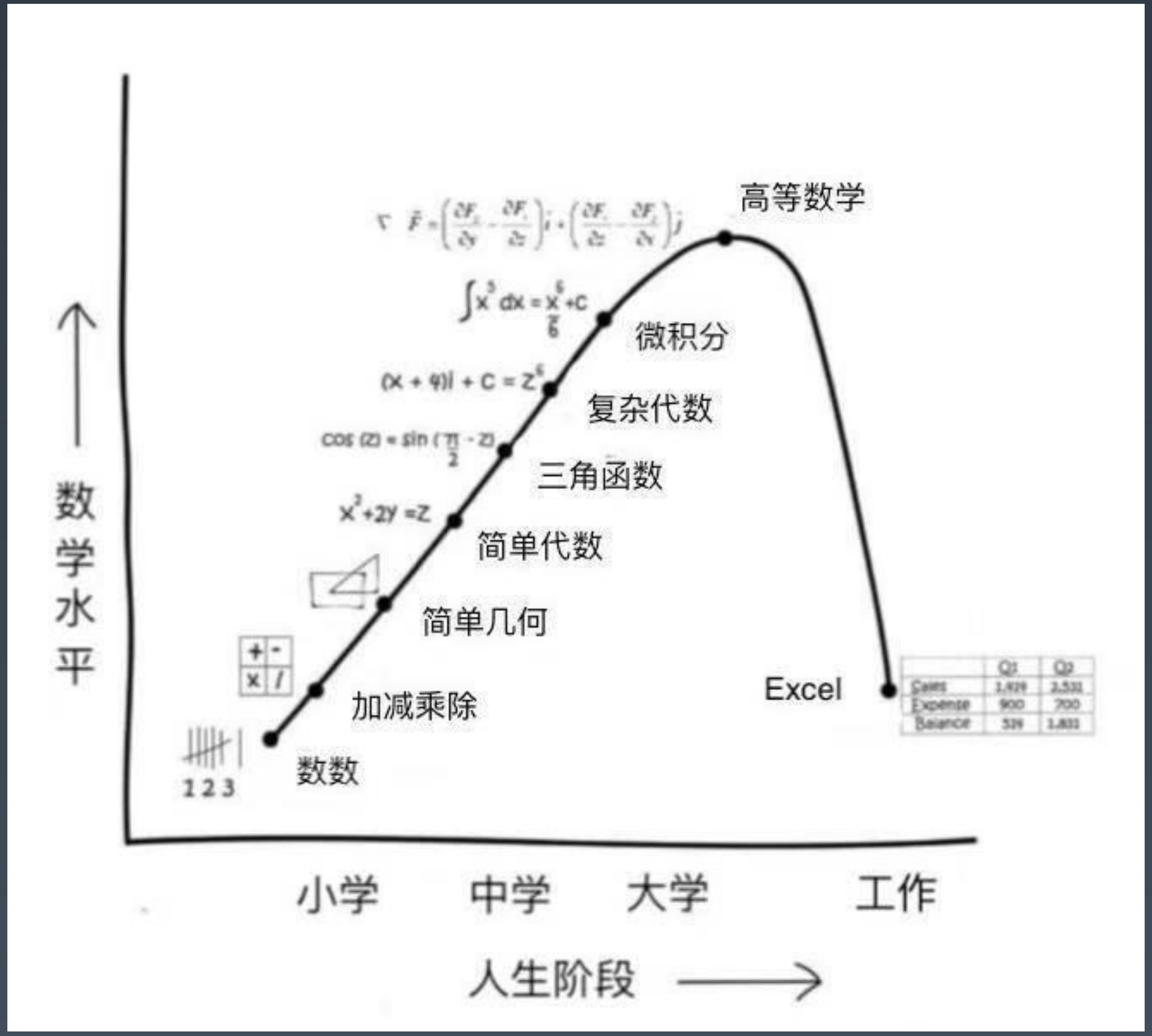
4 SP_CHK_SUB

5 SP_RECEIVE

6 SP_RECEIVE_SUB

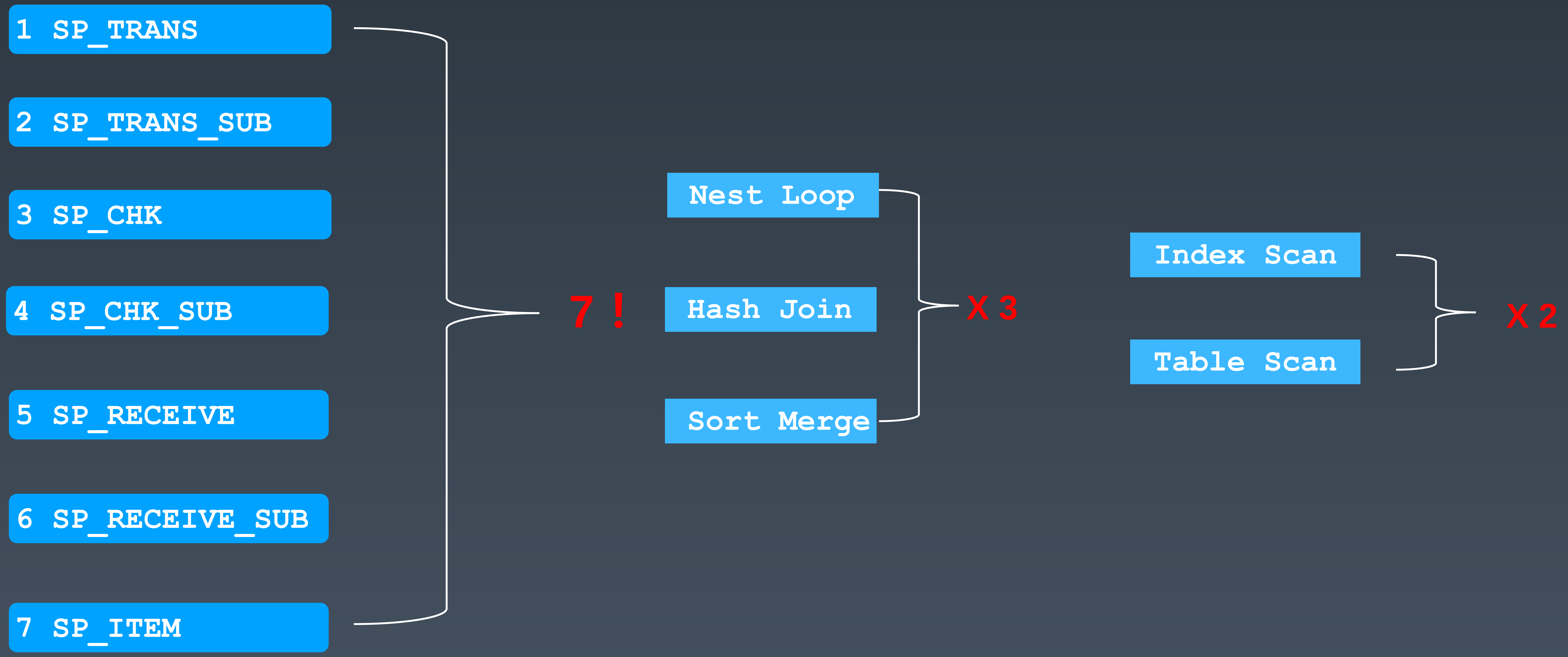
7 SP_ITEM

7 !



_optimizer_max_permutations

SQL背后的世界：理解优化器



1 SP_TRANS

2 SP_TRANS_SUB

3 SP_CHK

4 SP_CHK_SUB

5 SP_RECEIVE

6 SP_RECEIVE_SUB

7 SP_ITEM

7 !

Nest Loop

Hash Join

Sort Merge

X 3

Index Scan

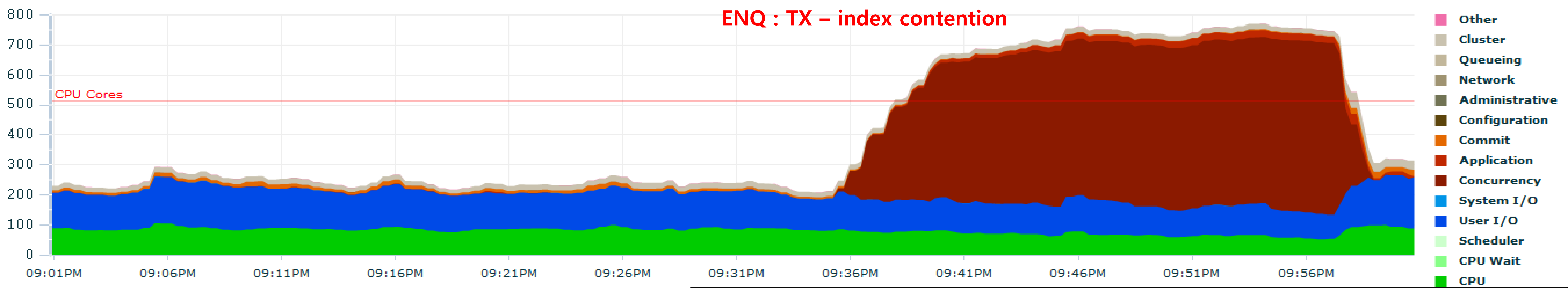
Table Scan

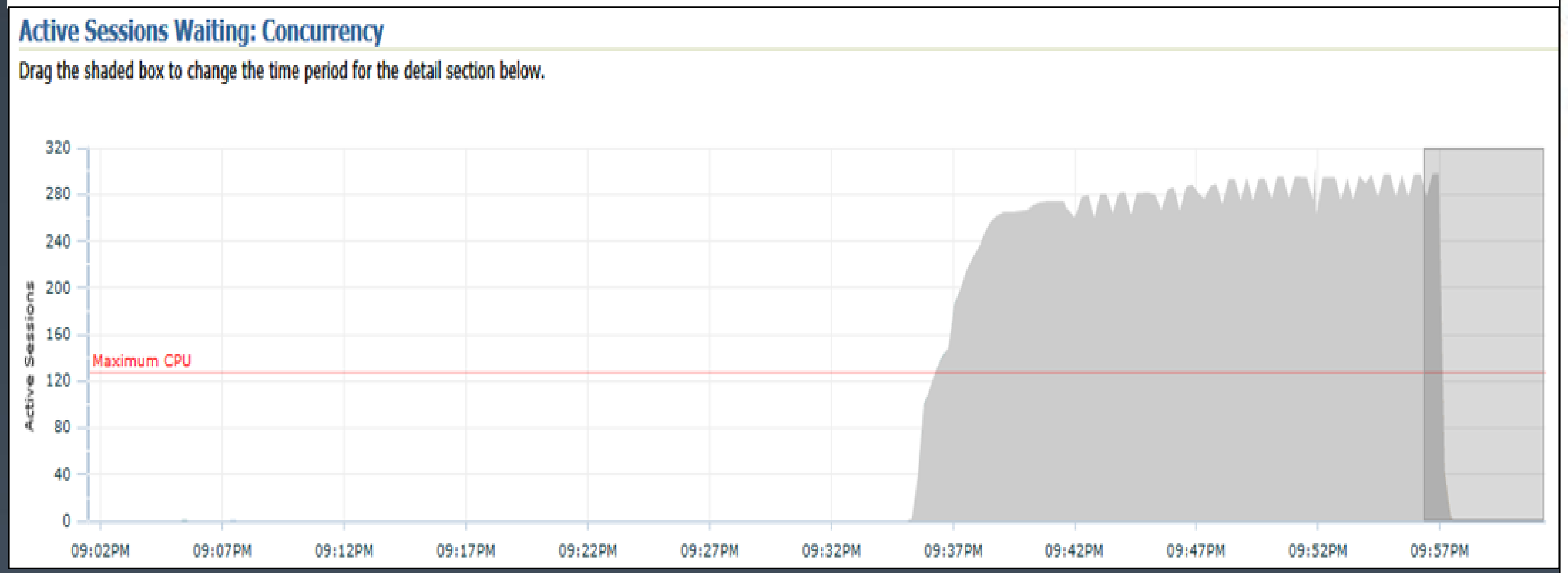
X 2

SQL背后的世界：理解优化器

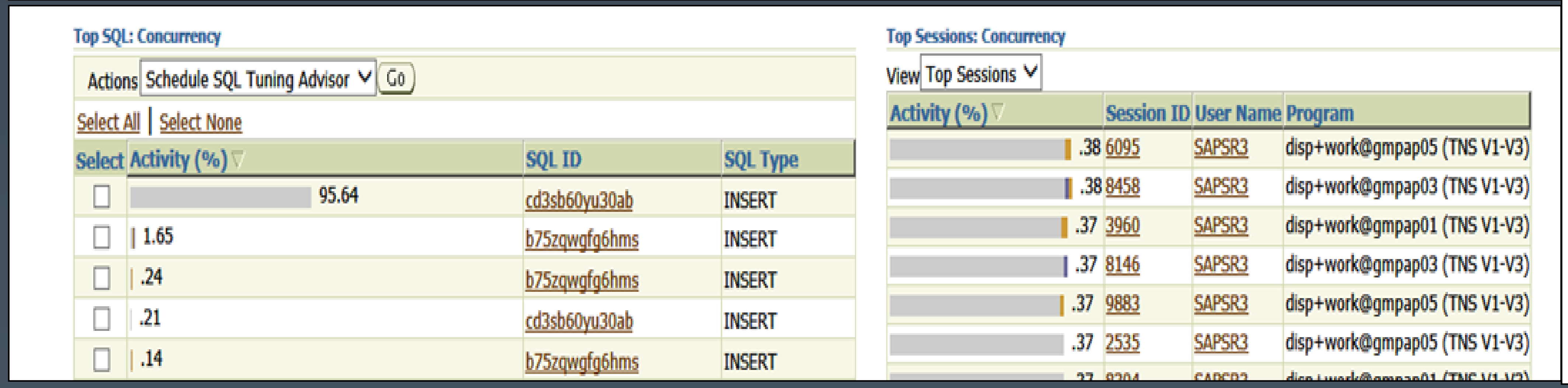
```
Join order[88]: SP_RECEIVE [SP_RECEIVE] SP_RECEIVE_SUB [SP_RECEIVE_SUB] SP_CHK [SP_CHK] SP_CHK_SUB
[SP_CHK_SUB] SP_TRANS [SP_TRANS] SP_TRANS_SUB [SP_TRANS_SUB] SP_ITEM [SP_ITEM]

Now joining: SP_CHK_SUB [SP_CHK_SUB] *******

NL Join
  Outer table: cost: 1863  cdn: 18225  rcz: 124  resp:  1863
  Inner table: SP_CHK_SUB Access path: tsc  Resc: 60 Join resc:  1095363  Resp:  1095363
  Join cardinality:  1 = outer (18225) * inner (36532) * sel (1.3146e-015)  [flag=0]
  Best NL cost: 1095363  resp: 1095363

SM Join
  Outer table: resc: 1863  cdn: 18225  rcz: 124  deg: 1  resp: 1863
  Inner table: SP_CHK_SUB resc: 60  cdn: 36532  rcz: 48  deg:  1  resp: 60
  Merge join  Cost:  3856  Resp:  3856

HA Join
  Outer table: resc: 1863  cdn: 18225  rcz: 124  deg: 1  resp: 1863
  Inner table: SP_CHK_SUB resc: 60  cdn: 36532  rcz: 48  deg:  1  resp: 60
  Hash join one ptn:  3414   Deg:  1   (sides swapped)

  hash_area:  16   buildfrag:  268   probefrag:   303      ppasses:    17
  Hash join   Resc: 5337   Resp: 5337
***********************
```

数据背后：藏身后台的索引和性能




Active Sessions Waiting: Concurrency

Drag the shaded box to change the time period for the detail section below.


Top SQL: Concurrency

Actions Schedule SQL Tuning Advisor Go

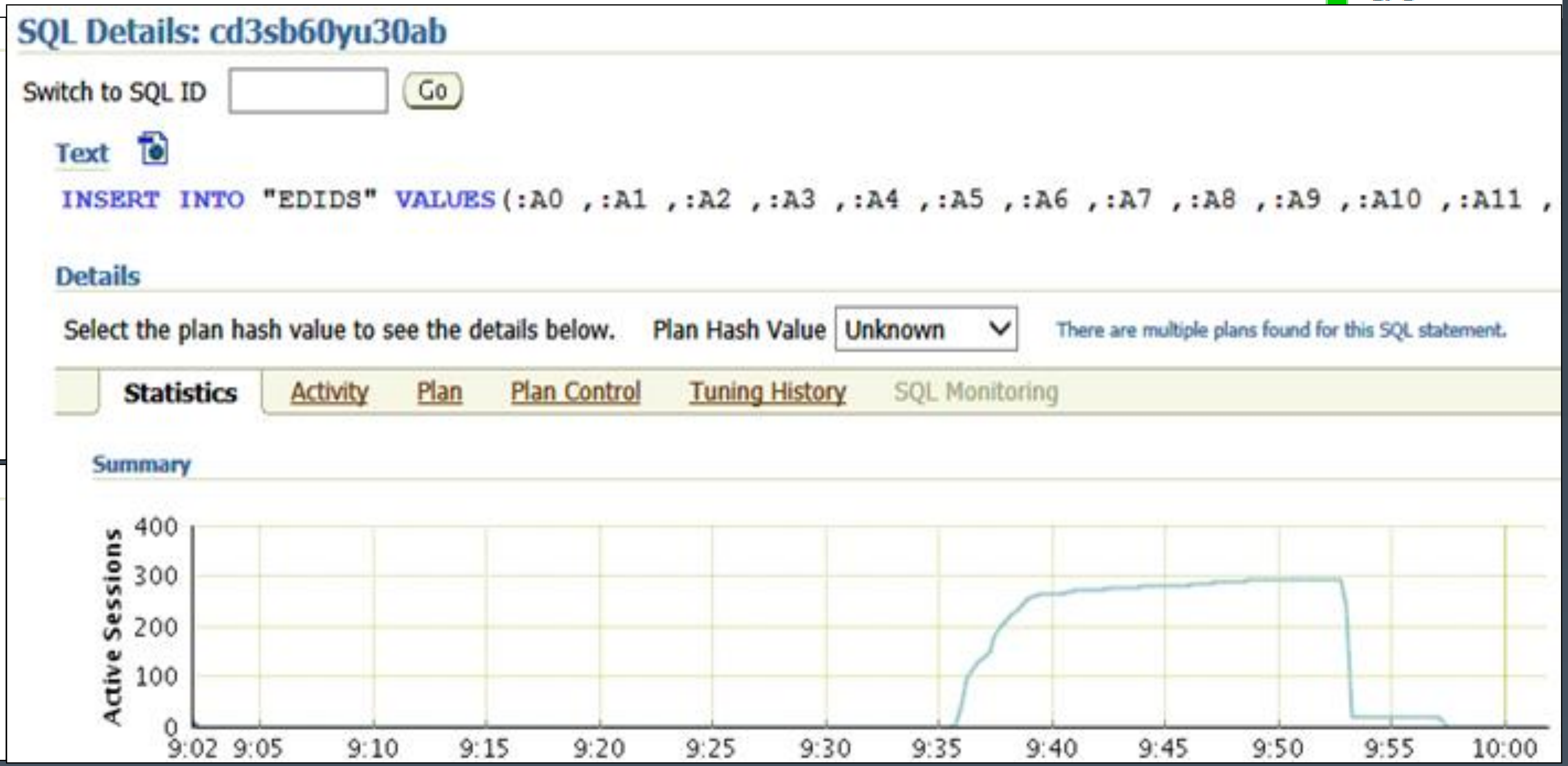
Select All | Select None

Select	Activity (%)	SQL ID	SQL Type
□	95.64	cd3sb60yu30ab	INSERT
□	1.65	b75zqwgfg6hms	INSERT
□	.24	b75zqwgfg6hms	INSERT
□	.21	cd3sb60yu30ab	INSERT
□	.14	b75zqwgfg6hms	INSERT

Top Sessions: Concurrency

View Top Sessions

Activity (%)	Session ID	User Name	Program
.38	6095	SAPSR3	disp+work@gmpap05 (TNS V1-V3)
.38	8458	SAPSR3	disp+work@gmpap03 (TNS V1-V3)
.37	3960	SAPSR3	disp+work@gmpap01 (TNS V1-V3)
.37	8146	SAPSR3	disp+work@gmpap03 (TNS V1-V3)
.37	9883	SAPSR3	disp+work@gmpap05 (TNS V1-V3)
.37	2535	SAPSR3	disp+work@gmpap05 (TNS V1-V3)
.37	8204	SAPSR3	disp+work@gmpap01 (TNS V1-V3)


SQL Details: cd3sb60yu30ab

Switch to SQL ID Go

Text

```
INSERT INTO "EDIDS" VALUES(:A0 ,:A1 ,:A2 ,:A3 ,:A4 ,:A5 ,:A6 ,:A7 ,:A8 ,:A9 ,:A10 ,:A11 ,
```

Details

Select the plan hash value to see the details below. Plan Hash Value: Unknown

There are multiple plans found for this SQL statement.

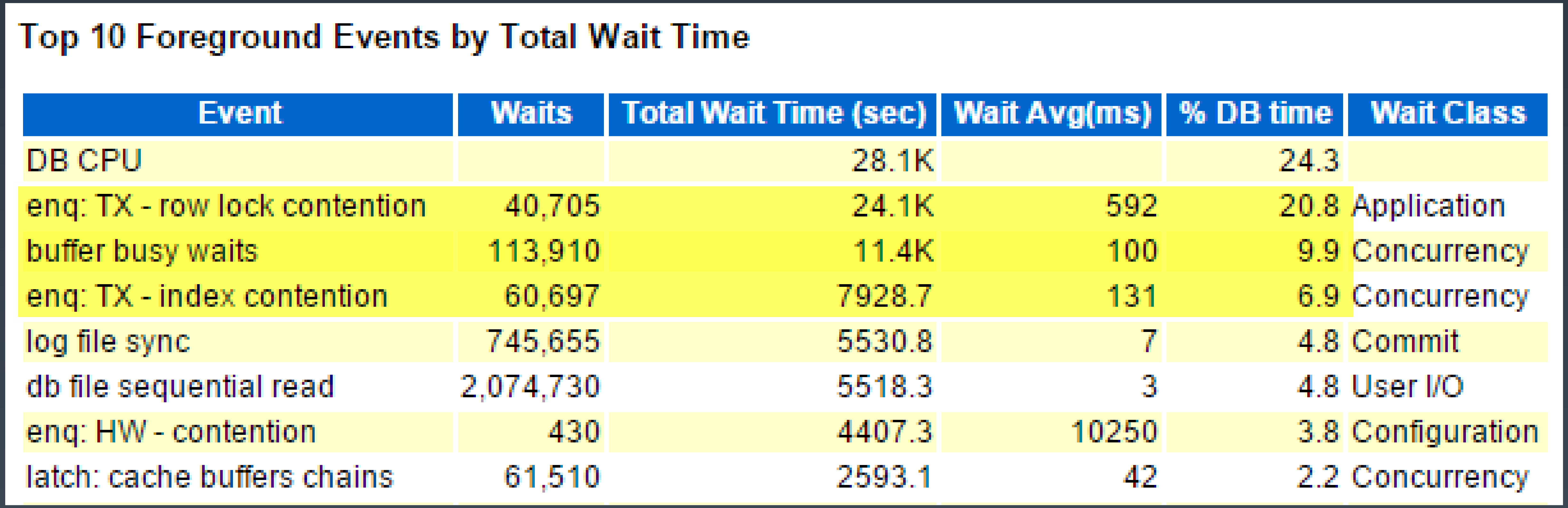
索引分裂：高并发时数据库的表征

高事务并发的典型案例

- 行锁竞争 row lock contention；
- 索引分裂 index contention；
- 前者源自应用，后者源于索引；

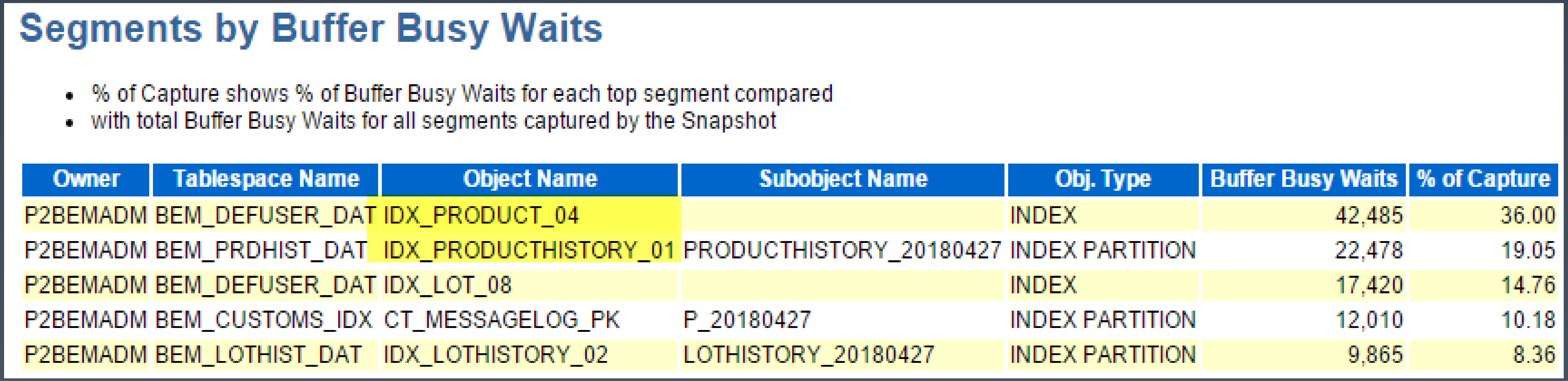
Buffer Busy Waits

- 索引是主要矛盾；



Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
DB CPU		28.1K		24.3	
enq: TX - row lock contention	40,705	24.1K	592	20.8	Application
buffer busy waits	113,910	11.4K	100	9.9	Concurrency
enq: TX - index contention	60,697	7928.7	131	6.9	Concurrency
log file sync	745,655	5530.8	7	4.8	Commit
db file sequential read	2,074,730	5518.3	3	4.8	User I/O
enq: HW - contention	430	4407.3	10250	3.8	Configuration
latch: cache buffers chains	61,510	2593.1	42	2.2	Concurrency

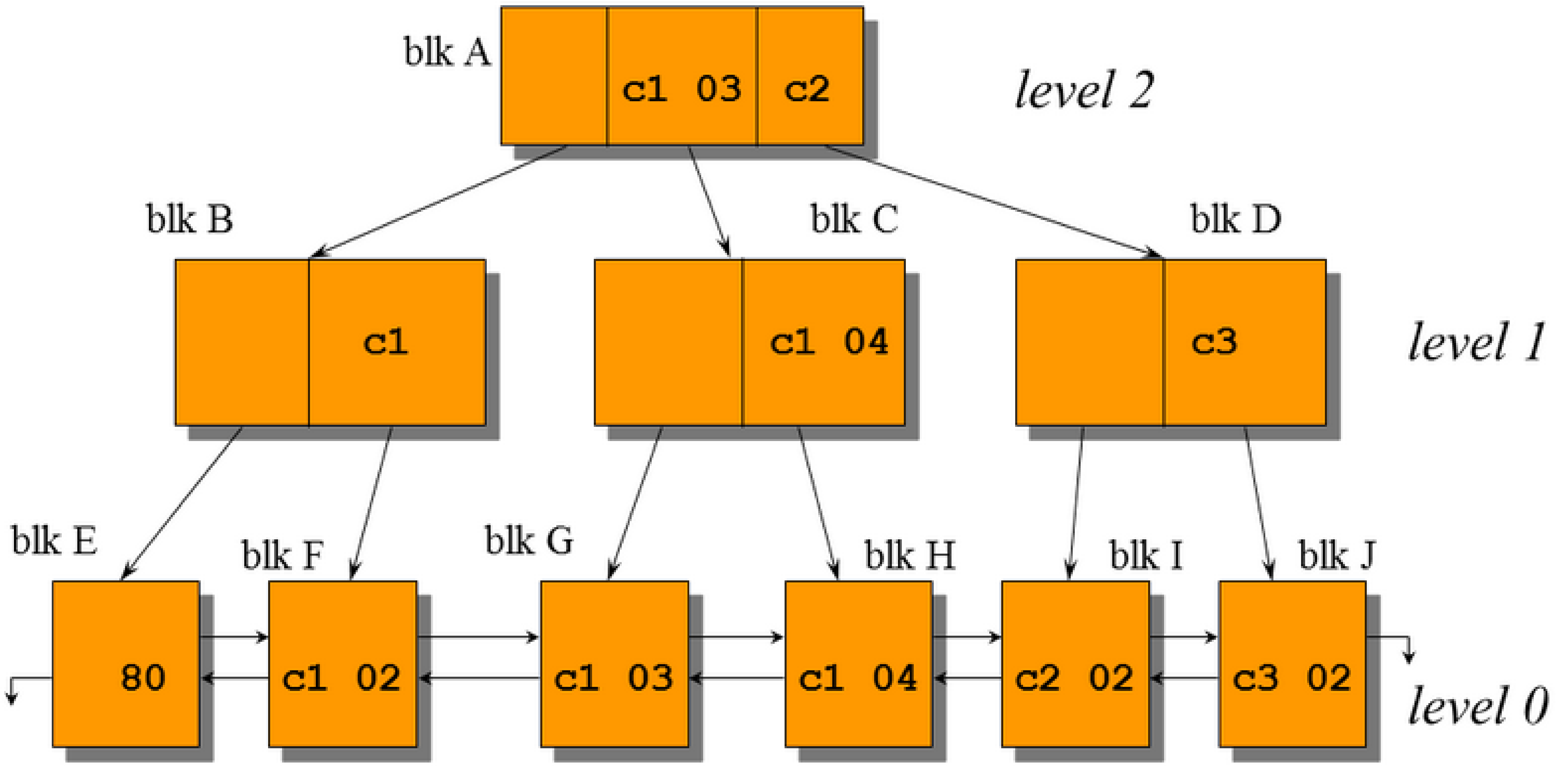


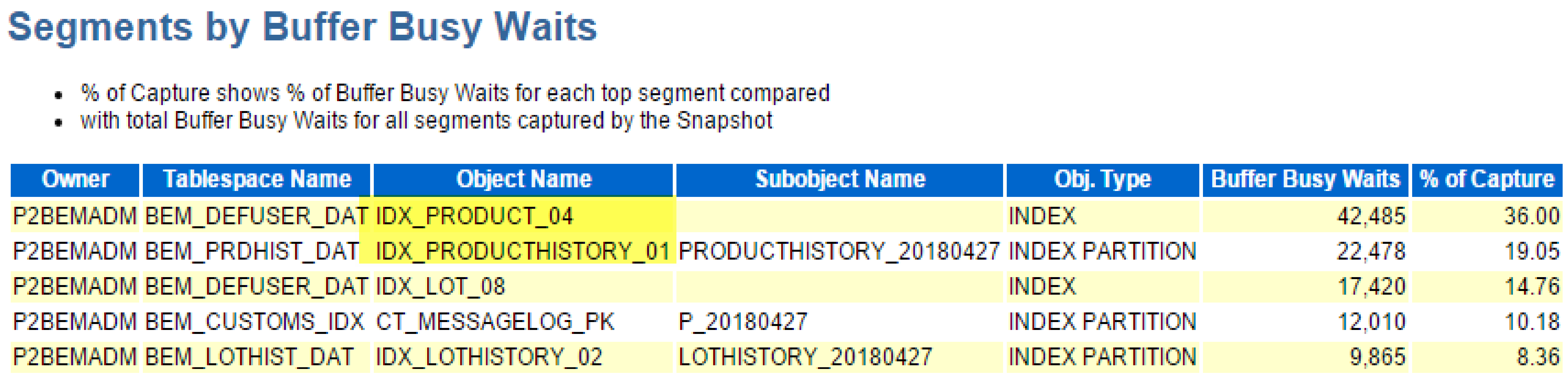
Segments by Buffer Busy Waits

- % of Capture shows % of Buffer Busy Waits for each top segment compared
- with total Buffer Busy Waits for all segments captured by the Snapshot

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Buffer Busy Waits	% of Capture
P2BEMADM	BEM_DEFUSER_DAT	IDX_PRODUCT_04		INDEX	42,485	36.00
P2BEMADM	BEM_PRDHIST_DAT	IDX_PRODUCTHISTORY_01	PRODUCTHISTORY_20180427	INDEX PARTITION	22,478	19.05
P2BEMADM	BEM_DEFUSER_DAT	IDX_LOT_08		INDEX	17,420	14.76
P2BEMADM	BEM_CUSTOMS_IDX	CT_MESSAGELOG_PK	P_20180427	INDEX PARTITION	12,010	10.18
P2BEMADM	BEM_LOTHIST_DAT	IDX_LOTHISTORY_02	LOTHISTORY_20180427	INDEX PARTITION	9,865	8.36

索引分裂：高并发时数据库的表征





Segments by Buffer Busy Waits

- % of Capture shows % of Buffer Busy Waits for each top segment compared
- with total Buffer Busy Waits for all segments captured by the Snapshot

Owner	Tablespace Name	Object Name	Subobject Name	Obj. Type	Buffer Busy Waits	% of Capture
P2BEMADM	BEM_DEFUSER_DAT	IDX_PRODUCT_04		INDEX	42,485	36.00
P2BEMADM	BEM_PRDHIST_DAT	IDX_PRODUCTHISTORY_01	PRODUCTHISTORY_20180427	INDEX PARTITION	22,478	19.05
P2BEMADM	BEM_DEFUSER_DAT	IDX_LOT_08		INDEX	17,420	14.76
P2BEMADM	BEM_CUSTOMS_IDX	CT_MESSAGELOG_PK	P_20180427	INDEX PARTITION	12,010	10.18
P2BEMADM	BEM_LOTHIST_DAT	IDX_LOTHISTORY_02	LOTHISTORY_20180427	INDEX PARTITION	9,865	8.36

- 常规表的单列索引
  - IDX_PRODUCT_04 – product -LASTTIMEKEY;
  - IDX_LOT_08 – lot - LASTTIMEKEY;
- 时间范围分区表的复合索引
  - IDX_PRODUCTHISTORY_01 -producthistory - (timekey,productname)；
  - IDX_LOTHISTORY_02 - lothistory-(timekey,eventname,machinename);
- LASTTIMEKEY/TIMEKEY是时间戳，单调递增;
- Index contention 是索引分裂同时有其他会话尝试更新。最容易产生等待的是单调递增的索引，因为每次插入都在索引的最右边。

解决方案：追根溯源分解竞争

- 数据库级解决方案：降低 buffer busy wait / index contention，也就是打散热点索引，把相近 timekey 对应的索引块分散；
- IDX_PRODUCT_04 / IDX_LOT_08 ，通过建立hash partition index，就可以把索引块分布到不同的分区上，大幅降低争用；
  - Create index IDX_PRODUCT_04 on PRODUCT(lasttimekey) global partition by hash(lasttimekey) partitions 16;
  - Create index IDX_LOT_08 on LOT(lasttimekey) global partition by hash(lasttimekey) partitions 16;
- Producthistory / lothistory 是分区表，无法直接创建hash子分区索引，除非把整个表重建为range-hash的复合分区，但是这么做改动太大；
- 分析SQL，发现producthistory表的访问基本都是两个栏位productname和timekey的，因此把索引重建为 productname 在前，timekey 在后就可解决问题。
  - Create index idx_producthistory_01(productname,timekey);

解决方案：追根溯源分解竞争

- LOTHISTORY 表的索引优化

SQL主要以 timekey 为条件，很多不包括eventname, machinename；

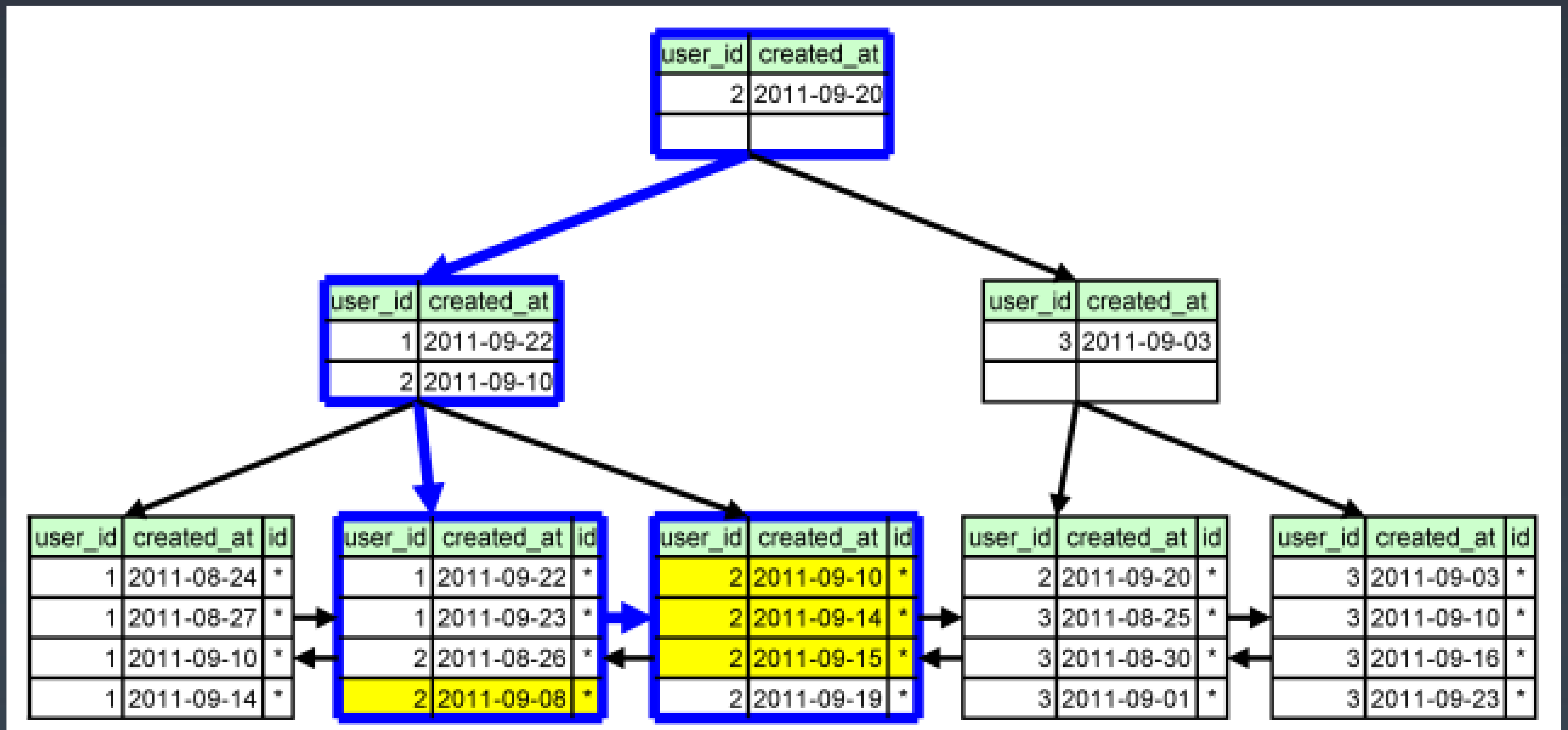
必须有合适的索引可以应用 timekey 条件；

可以增加前缀列，让SQL走index skip scan；

但是前缀列的唯一值不能太大, 否则index skip scan的额外成本高；

前缀列的唯一值也不能太小，否则起不到分散索引块的作用。

考虑使用8-32之间的值。



user_id	created_at
2	2011-09-20

user_id	created_at
1	2011-09-22
2	2011-09-10

user_id	created_at
3	2011-09-03

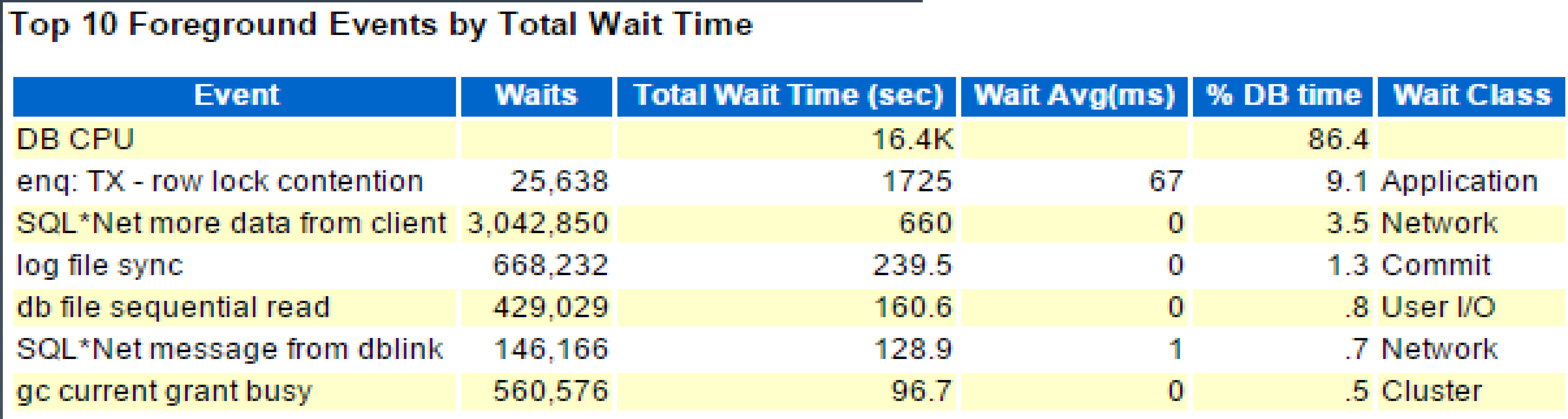
user_id	created_at	id
1	2011-08-24	*
1	2011-08-27	*
1	2011-09-10	*
1	2011-09-14	*

user_id	created_at	id
1	2011-09-22	*
1	2011-09-23	*
2	2011-08-26	*
2	2011-09-08	*

user_id	created_at	id
2	2011-09-10	*
2	2011-09-14	*
2	2011-09-15	*
2	2011-09-19	*

user_id	created_at	id
2	2011-09-20	*
3	2011-08-25	*
3	2011-08-30	*
3	2011-09-01	*

user_id	created_at	id
3	2011-09-03	*
3	2011-09-10	*
3	2011-09-16	*
3	2011-09-23	*



Top 10 Foreground Events by Total Wait Time

Event	Waits	Total Wait Time (sec)	Wait Avg(ms)	% DB time	Wait Class
DB CPU		16.4K		86.4	
enq: TX - row lock contention	25,638	1725	67	9.1	Application
SQL*Net more data from client	3,042,850	660	0	3.5	Network
log file sync	668,232	239.5	0	1.3	Commit
db file sequential read	429,029	160.6	0	.8	User I/O
SQL*Net message from dblink	146,166	128.9	1	.7	Network
gc current grant busy	560,576	96.7	0	.5	Cluster

索引分裂：深入数据库的原理

NEW IN 12.2

- 右向增长索引

  经常在索引分裂时引起严重竞争；

  尤其是以序列单调递增的方式；

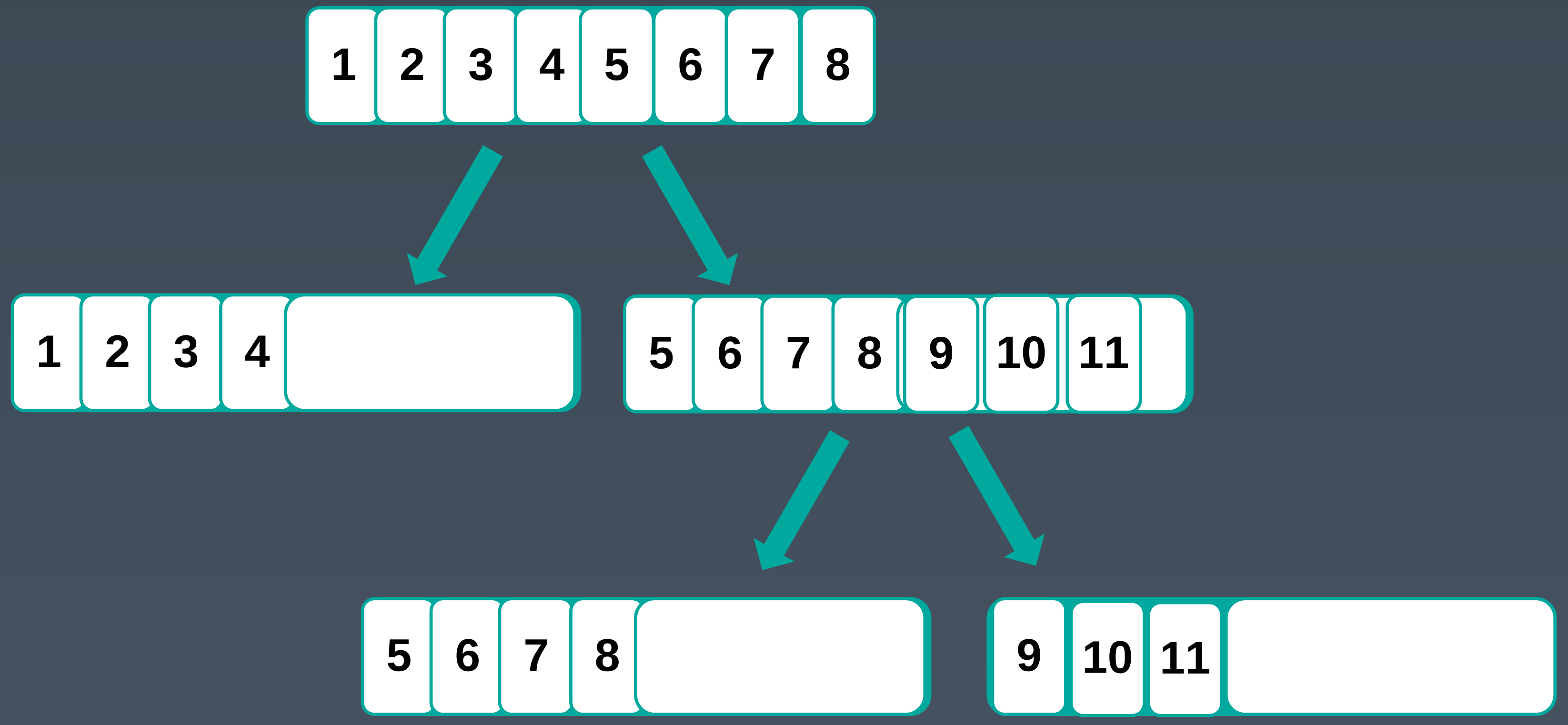
- 优化了叶块分裂算法

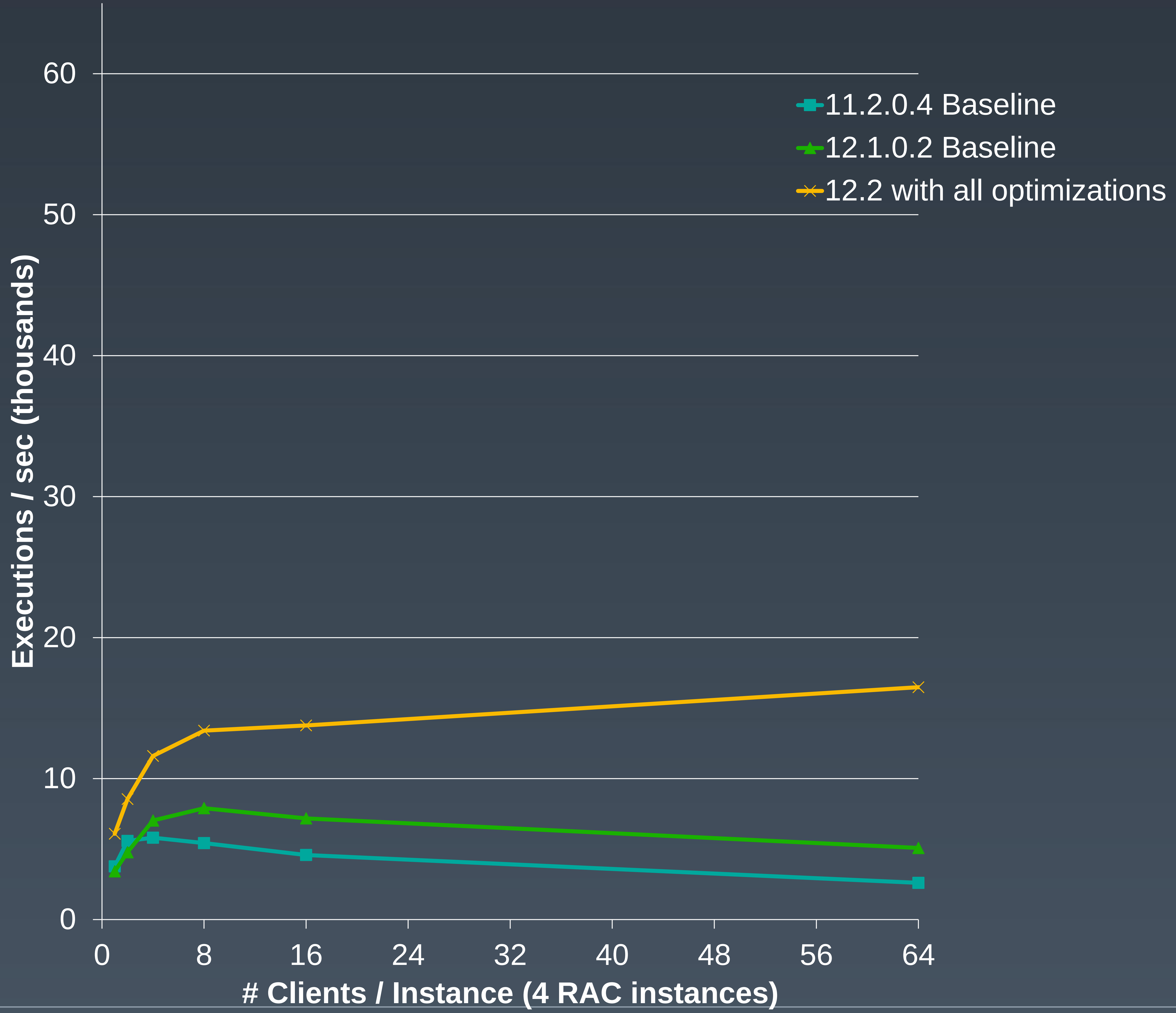
  缩减集群消息传递；

- 使用更佳的索引特性

  18c 的Scalable Sequences；

  模拟 Scalable 方式创建散列索引；



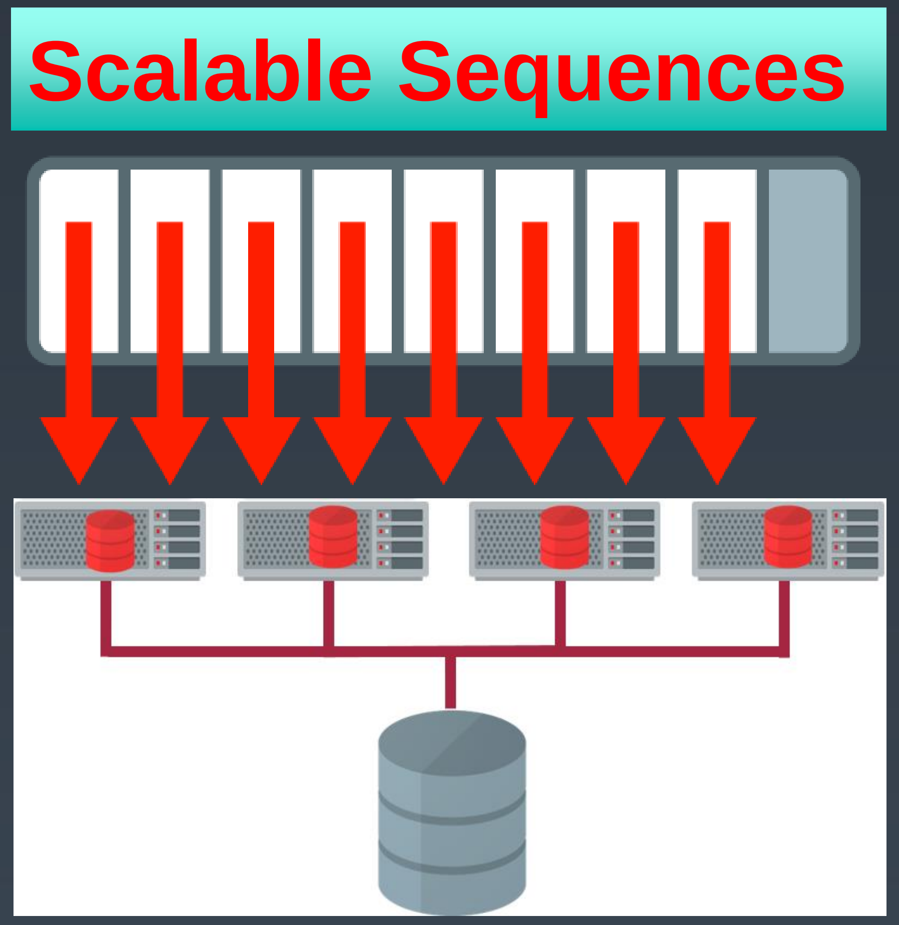


特性增强：可扩展序列分散索引竞争

NEW IN 18c

在Scaleable Sequence 中指定SCALE时，在传统的序列前增加了6位数字

- 前3位是由RAC里的实例号产生
- 随后3位由Session的SID产生



```
SQL> create sequence seq_eygle
     start with 1 increment by 1 scale;
```

```
SQL>select seq_eygle.nextval from dual;
 NEXTVAL
----------
101395000000000000000000001
SQL>select instance_number from v$instance;
1
SQL>select sid from v$mystat where rownum=1;
395
```

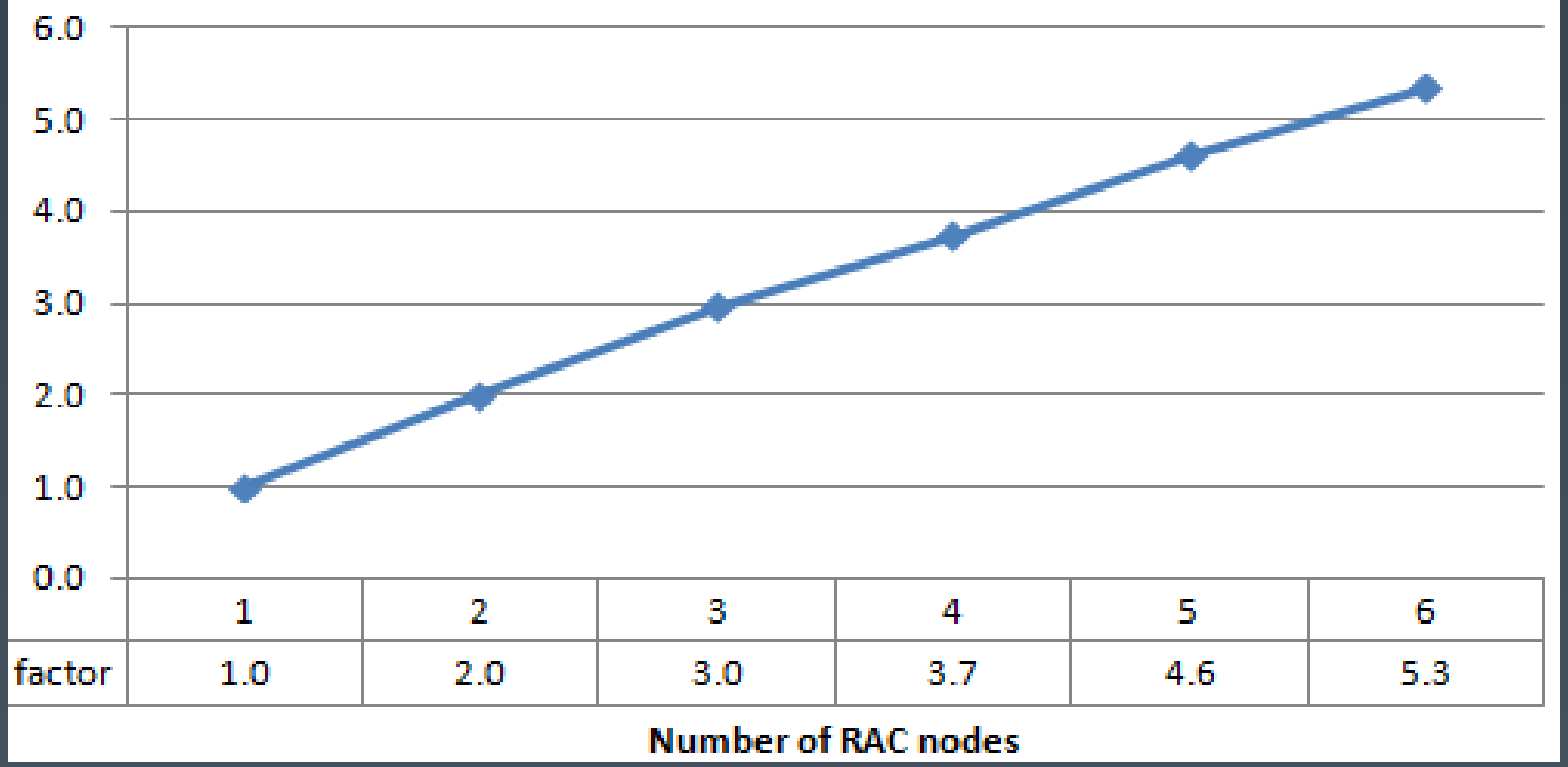
```
SQL>select sid from v$mystat where
rownum=1;
264
SQL>select seq_eygle.nextval from dual;
 NEXTVAL
----------
101264000000000000000000005
```

NEW IN 18c

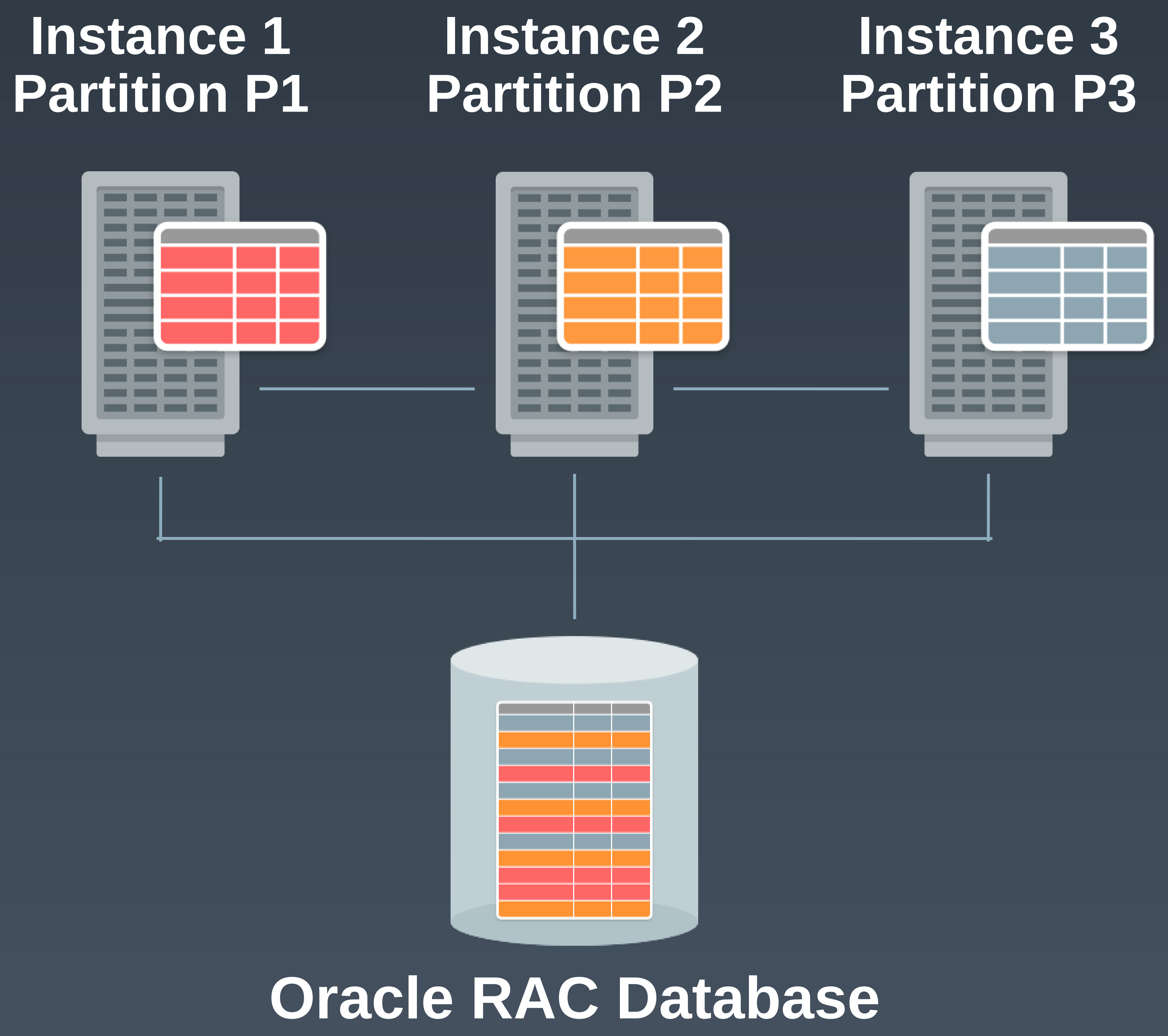
特性增强：Sharded RAC分散集群竞争

- 将分片能力引入到RAC集群实例中
  - 指定了分区键值的SQL查询将被路由到特定的实例;
  - 分区可以避免竞争减少跨实例的访问;
- 不包含分区键值的请求会透明的被处理
- 以最小的应用改变提供分布式性能

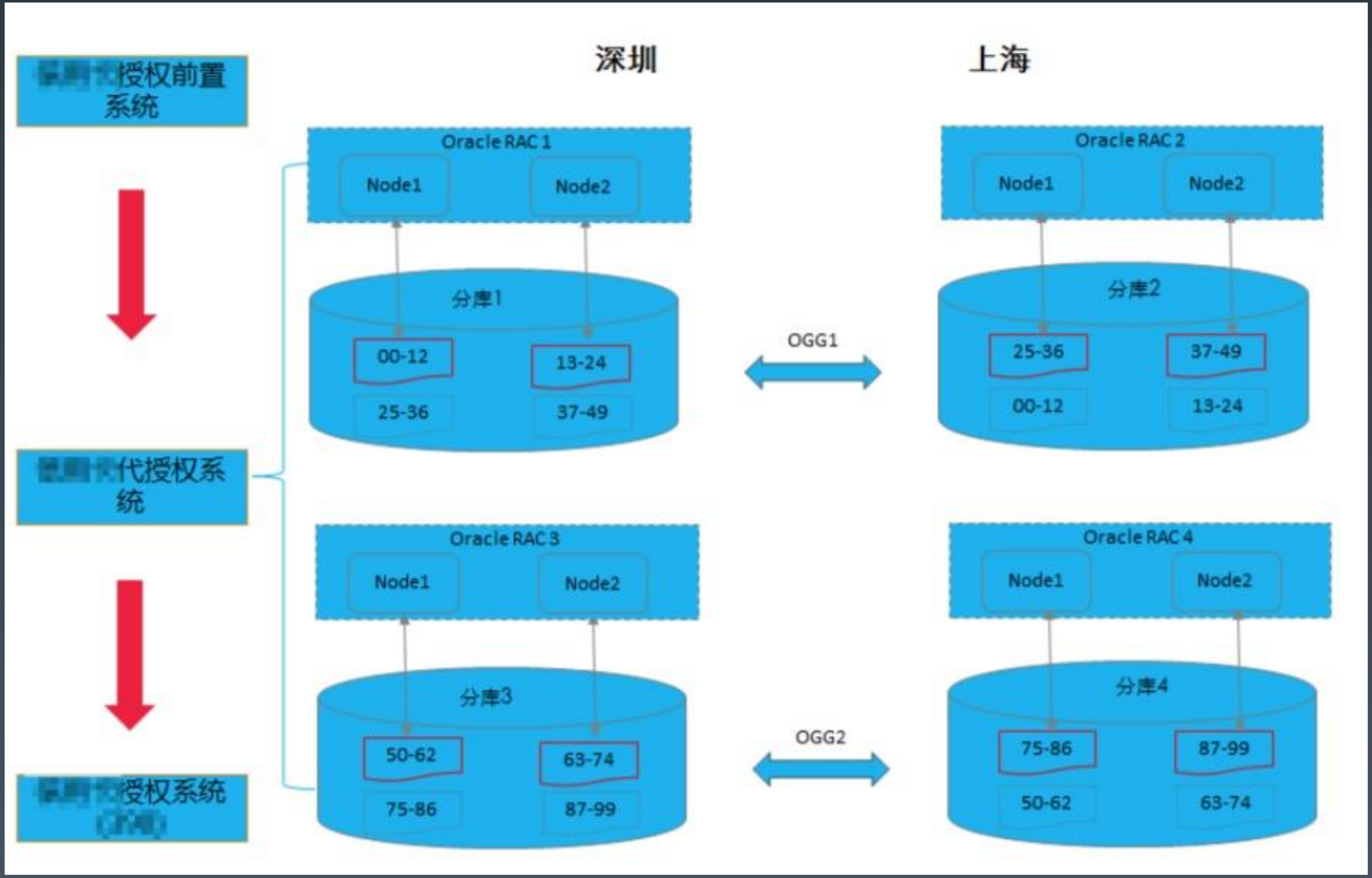
Figure 4 - RAC DB Performance with RAC Affinity



	1	2	3	4	5	6
factor	1.0	2.0	3.0	3.7	4.6	5.3



设计增强：通过分库分表获得线性性能



- 正常情况下
  - 每套RAC双节点；
  - 2套RAC组成双活架构；
  - 通过应用使RAC的各节点访问的数据不交叉；
  - 应用根据客户号后2位分割路由数据；
- 单节点失效
  - 同一RAC中的其他节点接管；
- 站点失效；
  - 切换路由到双活的异地节点；

目　录

- ➢ Oracle 的变革之路
- ➢ 性能优化与数据库进化
- ➢ **Oracle 19c新特性**

数据库上的明珠：SQL 的优化器

SQL

AUTOMATION

备库的自动DML转发

- 使得主备环境的作用进一步增强

失控SQL的执行计划自动隔离

- 实现对于性能的主动防御

自动化的执行计划管理 - Automated Plan Management

- 自动捕获所有可重用SQL的执行计划
- 查找可选计划选择最佳计划

实时的统计信息 - Real Time Statistics

- 近实时的统计信息刷新

SQL ENHANCEMENTS

使用统计信息支持查询

- 使用已经收集的统计信息，避免扫描大量数据；例如，select count(*) from emp
- 可以极大提升某些检查查询的性能，Min, Max, Count, Approximate_Count_distinct

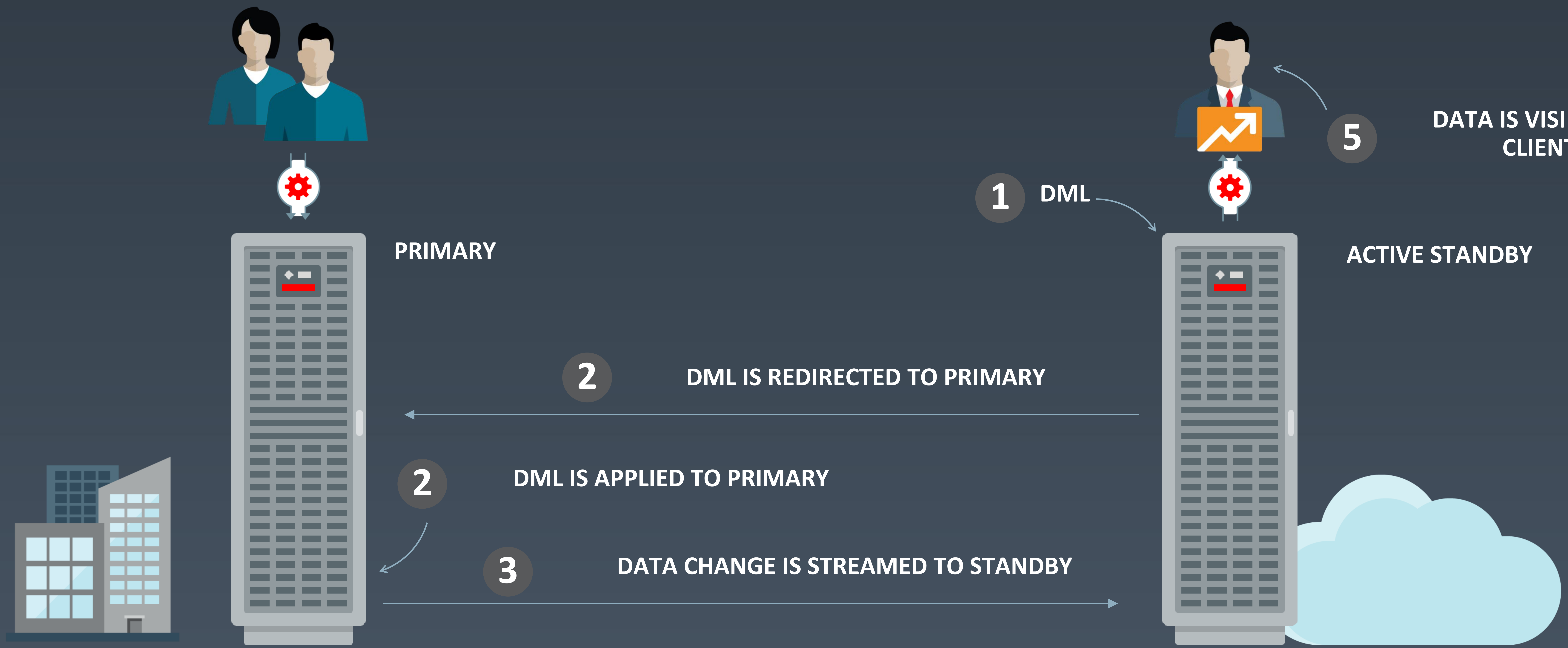
混合表的支持

- 通过分区指向外部存储对象，实现外部数据访问

负载分担：在备库透明的支持DML语句

NEW IN 19c

- 在 ADG 上发出的 DML 重定向到主库执行，备库等待主库日志传递到备库并应用



PRIMARY
ACTIVE STANDBY
1 DML
2 DML IS REDIRECTED TO PRIMARY
2 DML IS APPLIED TO PRIMARY
3 DATA CHANGE IS STREAMED TO STANDBY
5 DATA IS VISIBLE TO CLIENT

负载分担：在备库透明的支持DML语句

NEW IN 19c

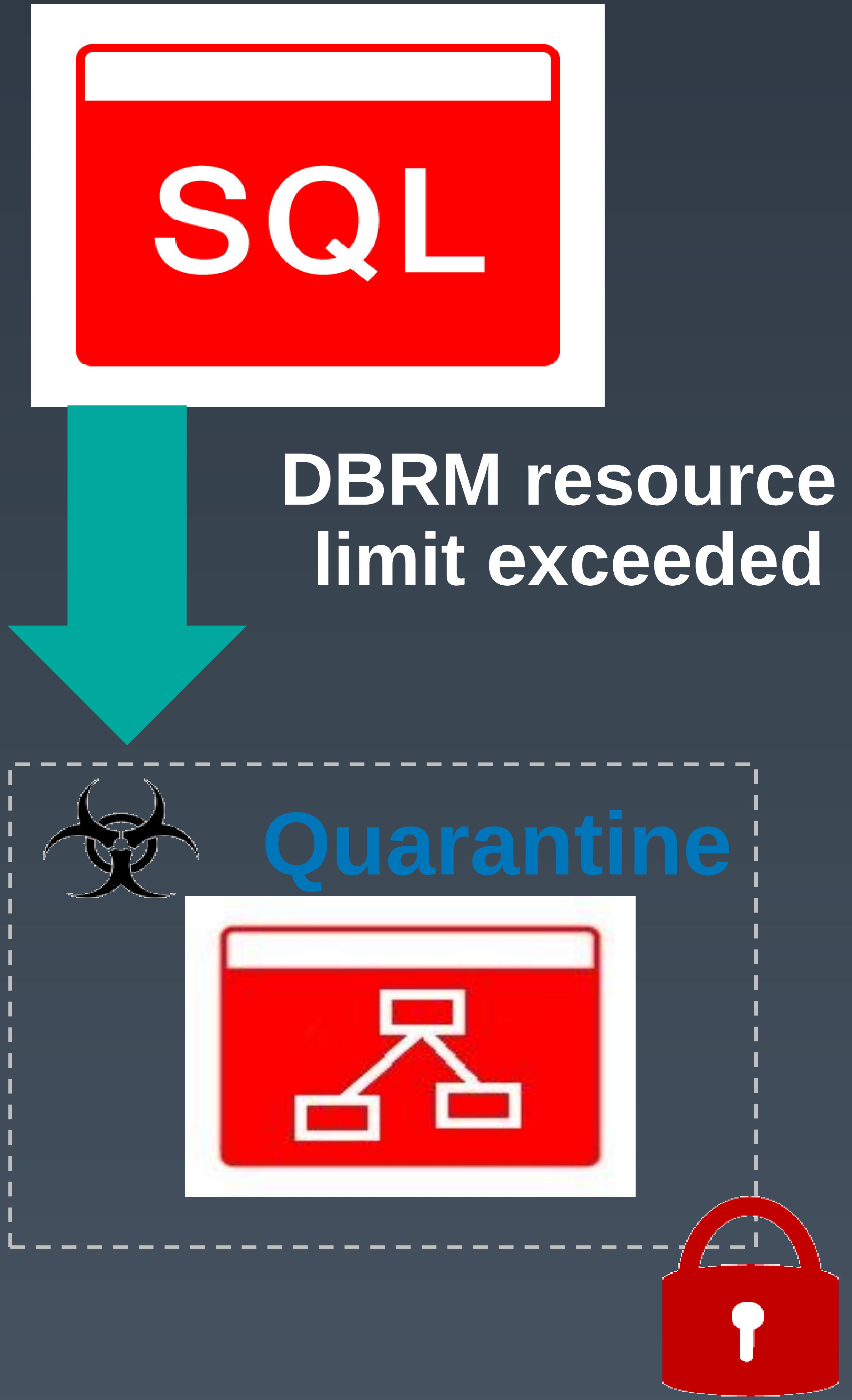
```
SQL> select * from enmotech;
   ID NAME
---------- --------------------
    1 EYGLE
    2 KAMUS
    3 Yangtingkun
SQL> insert into enmotech values(4,'ORA-600');
ORA-16000: database or pluggable database open
for read-only access

SQL> alter session set adg_redirect_dml=true;
Session altered.

SQL> set timing on
SQL> insert into enmotech values(4,'ORA-600');
1 row created.

Elapsed: 00:00:01.21
```

```
SQL> select * from enmotech;
   ID NAME
---------- --------------------
    4 ORA-600
    1 EYGLE
    2 KAMUS
    3 Yangtingkun

Elapsed: 00:00:00.01
SQL> commit;
Commit complete.

Elapsed: 00:00:01.02
```

性能管控：失控SQL的自动隔离

NEW IN 19c

- Oracle Resource Manger 提供SQL监控能力，可以自动终止消耗资源超过一定阈值的SQL。然而，在终止查询之前，大量的资源已经被浪费。
- 在新特性中，执行计划超过DBRM限制将会自动被；
- 隔离的执行计划将被阻止执行；
- SQL 隔离是一个针对失控SQL的自动化解决方案。



性能管控 – 失控计划的隔离

NEW IN 19c

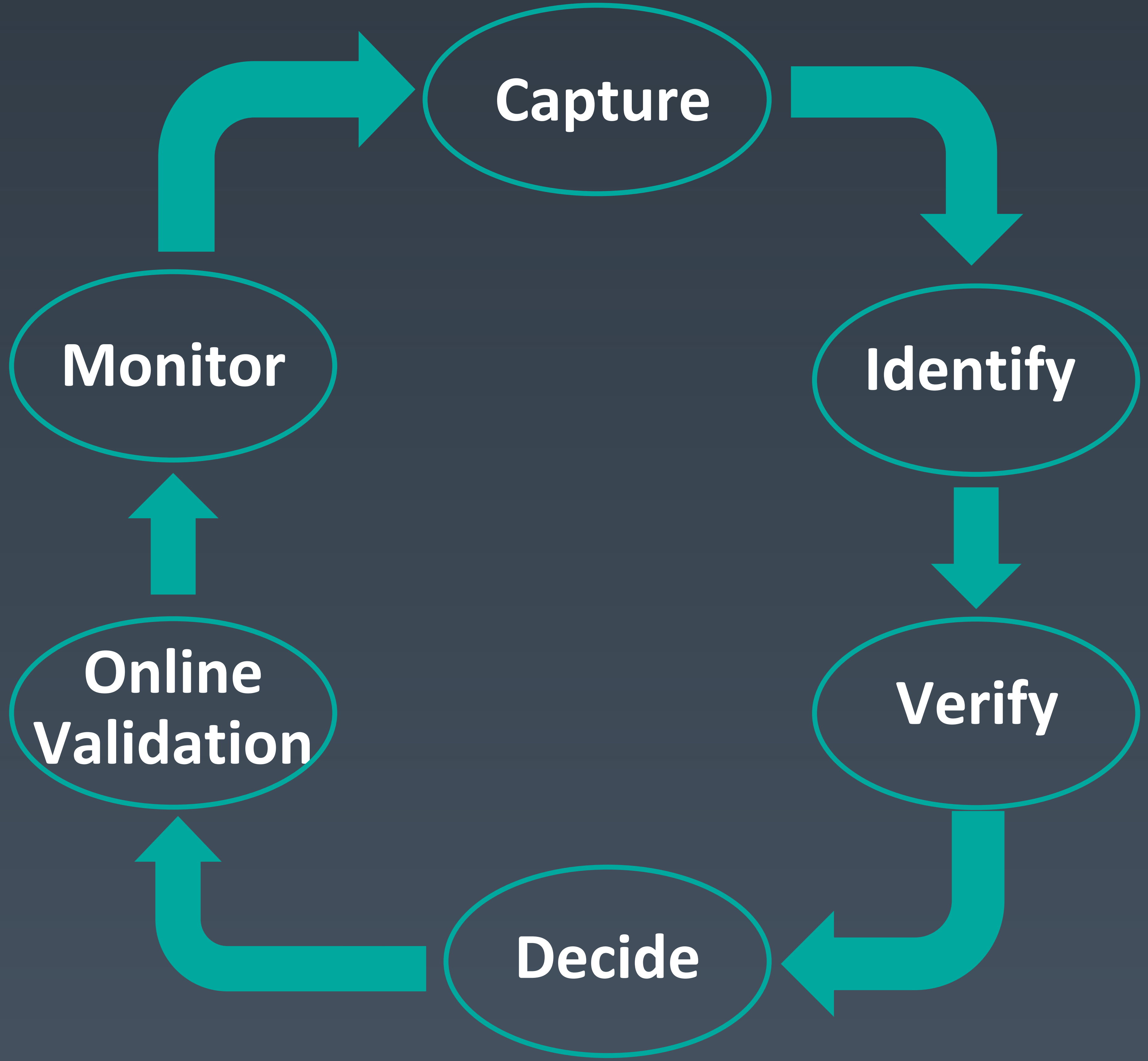
在以下示例中，SQL 因为执行时间超过限制而被隔离，再次执行时提示”使用了被隔离的执行计划”。

```
SQL> select count(*)
  2  from emp emp1, emp emp2, emp emp3, emp emp4, emp emp5, emp emp6, emp emp7, emp emp8
  3  where rownum <= 10000000;

ERROR at line 2:
ORA-00040: active time limit exceeded - call aborted

SQL> select count(*)
  2  from emp emp1, emp emp2, emp emp3, emp emp4, emp emp5, emp emp6, emp emp7, emp emp8
  3  where rownum <= 10000000;

ERROR at line 2:
ORA-56955: quarantined plan used

SQL> select avoided_executions, sql_quarantine
  2  from v$sql vs
  2  where sql_id = 'd0z9zp1h5n799’;

SQL_QUARANTINE                                                    AVOIDED_EXECUTIONS
----------------------------------------------------------------- ------------------
SQL_QUARANTINE_0scf6as37zcu0cfe7a0e4                                               1
```

自动化索引创建和实施 – Automatic Indexing

NEW IN 19c

自动索引是借鉴于人工工作的专家系统 It is an **expert system** that implements what a performance engineer skilled in index tuning would do



- 自动索引技术基于和常规手工SQL优化同样的思路实现；
- 系统自动识别候选索引并在启用前验证索引的效率和性能；
- 整个过程完全是自动化实现的；
- 透明度与复杂的自动化同样重要；
  - 所有的调整活动可以通过报告进行核查；

自动化索引技术的实现过程

- 通过 DBA_AUTO_INDEX_CONFIG 修改和启用自动索引特性

```
CDB$ROOT@SYS>EXEC DBMS_AUTO_INDEX.CONFIGURE('AUTO_INDEX_MODE','IMPLEMENT');
CDB$ROOT@SYS>select * from DBA_AUTO_INDEX_CONFIG;
PARAMETER_NAME                  PARAMETER_VALUE  LAST_MODIFIED                 MODIFIED_BY
------------------------------- ---------------- -------------------------     ------------
AUTO_INDEX_DEFAULT_TABLESPACE
AUTO_INDEX_MODE                 IMPLEMENT        17-FEB-19 10.03.59.000000 PM  SYS
AUTO_INDEX_REPORT_RETENTION     31
AUTO_INDEX_RETENTION_FOR_AUTO   373
```

- 通过测试数据执行测试查询，此时表上不存在索引

```
PDB1>create table test as select * from dba_objects;
PDB1>insert into test select * from test;
PDB1>insert into test select * from test;
PDB1>update test set object_id=rownum;
2316704 rows updated.
PDB1>commit;

PDB1>select object_name from test where object_id=1;
PDB1>select object_type from test where object_id=123;
PDB1>select created from test where object_id=345;
```

自动化索引技术的实现过程

- **检查数据库自动任务执行和过程记录**

```
select * from DBA_AUTO_INDEX_EXECUTIONS;
EXECUTION_NAME                 EXECUTION_START     EXECUTION_END       ERROR_MESSAGE        ST
-----------------------------  ------------------- ------------------- -------------------- --
SYS_AI_2019-02-17/22:51:00     2019-02-17 22:51:00 2019-02-17 22:53:07                      CO

select * from DBA_AUTO_INDEX_STATISTICS where  EXECUTION_NAME='SYS_AI_2019-02-17/22:51:00';
EXECUTION_NAME                           STAT_NAME                          VALUE
---------------------------------------- ----------------------------- ----------
SYS_AI_2019-02-17/22:51:00               Index candidates                       1
SYS_AI_2019-02-17/22:51:00               Indexes created (visible)              1
SYS_AI_2019-02-17/22:51:00               Indexes created (invisible)            0
SYS_AI_2019-02-17/22:51:00               Indexes dropped                        0
SYS_AI_2019-02-17/22:51:00               Space used in bytes             45088768
SYS_AI_2019-02-17/22:51:00               Space reclaimed in bytes               0
SYS_AI_2019-02-17/22:51:00               SQL statements verified                2
SYS_AI_2019-02-17/22:51:00               SQL statements improved                2
SYS_AI_2019-02-17/22:51:00               SQL statements managed by SPM          0
SYS_AI_2019-02-17/22:51:00               SQL plan baselines created             0
SYS_AI_2019-02-17/22:51:00               Improvement percentage               100
```

自动化索引技术的实现过程

- 以 AI 标识的自动索引已经被创建出来

```
PDB1>select command,statement from DBA_AUTO_INDEX_IND_ACTIONS
      where execution_name='SYS_AI_2019-02-17/22:51:00‵ order by action_id;
COMMAND                     STATEMENT
 -------------------- -------------------- ----------------------------------------
CREATE INDEX                CREATE INDEX “EN“.″SYS_AI_18sc6rdkngxkh“ ON ″EN"."TEST"("OBJECT_ID")
REBUILD INDEX               ALTER INDEX ″EN"."SYS_AI_18sc6rdkngxkh"  REBUILD  ONLINE
ALTER INDEX VISIBLE  ALTER INDEX ″EN"."SYS_AI_18sc6rdkngxkh" VISIBLE
```

- 原有查询再次执行将适用智能索引提升性能

```
PDB1>select object_name from test where object_id=1234;
Execution Plan
-------------------------------------------------------------------------------------
| Id  | Operation                           | Name                    | Rows  |Bytes| Cost (%CPU)|
-------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |                         |     1 |   41|     4   (0)|
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| TEST                    |     1 |   41|     4   (0)|
|*  2 |   INDEX RANGE SCAN                  | SYS_AI_18sc6rdkngxkh |     1 |     |     3   (0)|
-------------------------------------------------------------------------------------
```

自动化索引技术的实现效果

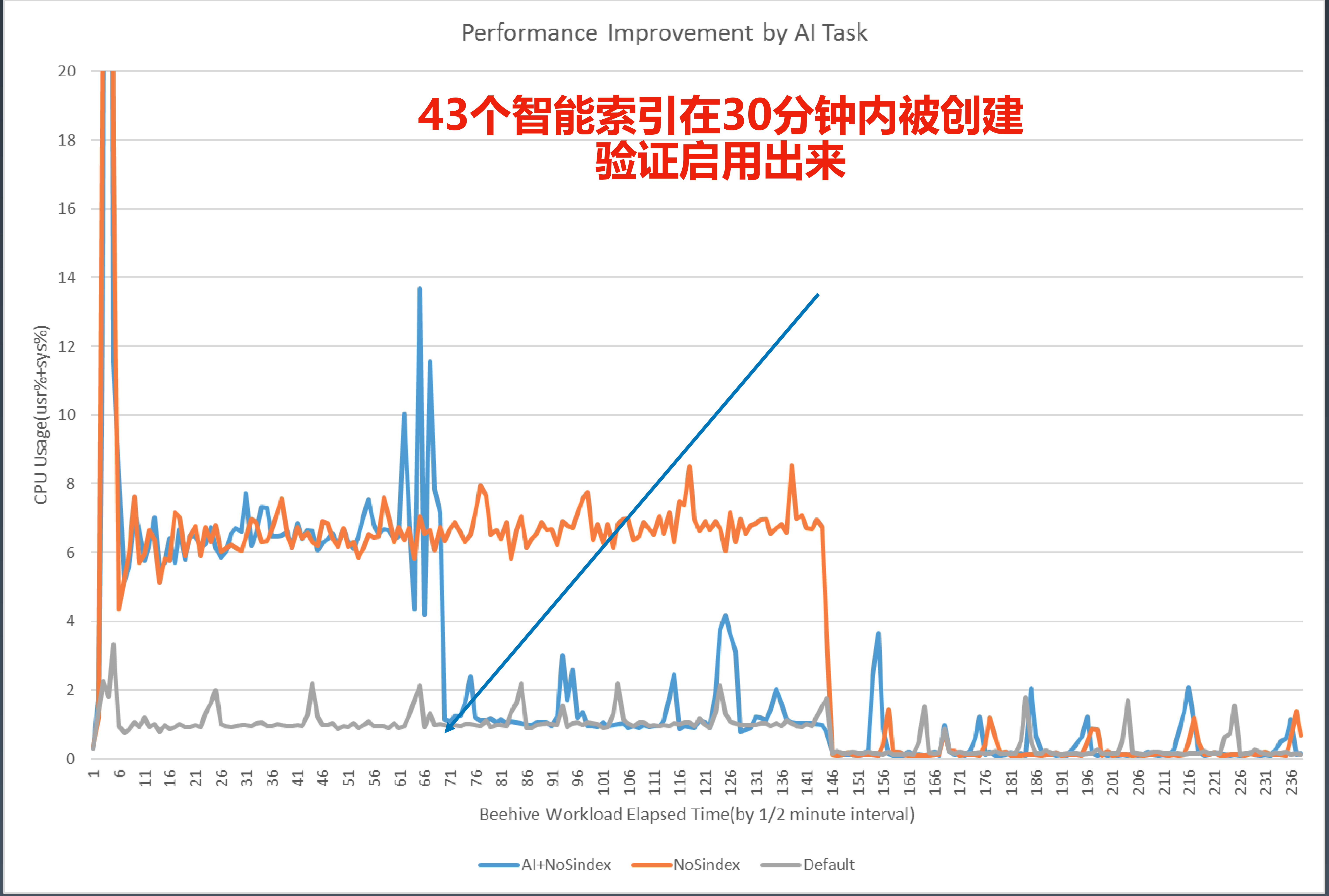
让完全无索引的应用程序持续运行，AI系统在后台自动优化创建索引。

运行效果如下，总运行时间 2 小时

-10 分钟负载高攀；

-50 分钟负载下降；

-60 分钟稳定；



总 结

- ➢Oracle 仍然是最好的数据库产品
- ➢数据库的竞争已经转移到云上
- ➢数据库的未来是自动化

THANKS！

QCon 10th