ORACLE

DATABASE IN-MEMORY

Oracle 白皮书
2014 年 8 月

Oracle Database In-Memory

免责声明

以下内容旨在概述产品的总体发展方向。该内容仅供参考，不可纳入任何合同。其内容不构成提供任何材料、代码或功能的承诺，并且不应该作为制定购买决策的依据。此处所述有关 Oracle 产品的任何特性或功能的开发、发布以及相应的日程安排均由 Oracle 自行决定。

概要

Oracle Database In-Memory 透明地将分析查询的速度提高了若干数量级，从而有助于实时业务决策。使用 Database In-Memory，企业可以即刻运行分析和报告，而以往则需要花数小时或数天的时间。企业受益于实时制定的更好决策，从而降低成本、提高生产效率以及增强竞争优势。

Oracle Database In-Memory 加速了数据仓库和混合负载 OLTP 数据库，可在与 Oracle Database 12*c* 兼容的任何现有应用程序下轻松部署。并且无需更改应用程序。Database In-Memory 使用 Oracle 成熟的纵向扩展、横向扩展和存储分层技术，经济高效地运行任何大小的负载。Oracle 行业领先的可用性和安全性特性都能透明地与 Database In-Memory 协同工作，从而使后者成为市场上最健壮的产品。

能够轻松地对所有现有应用程序同时执行实时数据分析和实时事务处理，从而使组织可以转型为实时企业，以便快速进行数据驱动式决策、即时响应客户需求，并不断优化所有关键流程。

目标读者

从 DBA 或性能专家角度来讲，要求读者具有 Oracle 数据库技术的亲身体验。

引言

现在的信息架构相比几年前更具动态性。业务用户现在要求能够更快获得更多支持决策的信息。为跟上需求增长的步伐，各个公司除了对其数据仓库运行分析之外，还不得不对其运营系统运行分析。这会打破事务负载之间的平衡，受到频繁插入和更新以及对需要扫描大量数据的报告式查询的影响。

通过引入 Oracle Database In-Memory，单个数据库现在可以高效地支持混合负载，从而在为事务提供最佳性能的同时又能支持实时分析和报告。这主要归功于独特的“双格式”架构，这种架构支持以现有的 Oracle 行格式（适用于 OLTP 操作）和新的纯内存中列格式（已针对分析处理进行了优化）维护数据。内存中技术还支持数据集市和数据仓库，以便提供更具即席性的分析，从而最终用户能够在目前仅运行一个查询的时间内运行多个业务驱动式查询。

将内存中列格式嵌入到现有 Oracle 数据库软件可确保它与所有现有特性完全兼容，无需在应用程序层进行任何更改。努力成为实时企业的公司可以更轻松地实现自己的目标，并且不受所运行的应用程序限制。本文将介绍 Oracle Database In-Memory 的主要组件，并提供简单、可重现的示例以便很容易地熟悉这些组件。本文还将概述如何将 Database In-Memory 集成到现有运营系统和数据仓库环境中，以提高性能和可管理性。

Oracle Database In-Memory 选件

行格式与列格式

Oracle 数据库传统上以行格式存储数据。在行格式数据库中，数据库中存储的每个新事务或新记录都表示为表中的一个新行。一行由多个列构成，每一列表示相应记录的一个不同的属性。行格式适用于联机事务系统，因为它允许快速访问记录中的所有列（这是因为给定记录的所有数据一起保存在内存中和存储上）。

列格式数据库在单独的列结构中存储事务或记录的每个属性。列格式适用于分析，因为仅选择几列但查询却要访问数据集的大部分时，列格式可以快速检索数据。

但是，每种格式上发生 DML 操作（插入、更新或删除）时会发生什么情况呢？行格式对于处理 DML 极其高效，因为它在一个操作中就对整个记录进行了处理，即，插入行、更新行或删除行。列格式在处理行式 DML 方面不是那么高效：为插入或删除列格式的单条记录，必须更改表中的所有列结构。

到目前为止，仅允许您选择一种格式，因此实现的 OLTP 或分析性能都不是最佳的。

Oracle Database In-Memory (Database In-Memory) 允许以内存中行格式（缓冲区缓存）和新的内存中列格式同时填充数据，因此兼具两者的优势。

请注意，双格式架构并**不会**带来双倍内存需求。内存中列格式应能够调整大小以容纳必须存储在内存中的对象，但数十年来缓冲区缓存已经过优化，能够高效运行，且其大小比数据库大小小得多。实际上，人们希望双格式架构在总内存需求方面带来的增加低于 20%。对于为所有负载始终提供最佳性能而言，这只是一个很小的代价。

图 1. Oracle 独特的双格式架构

使用 Oracle 提供的独特方法，存储上仍保留表的单个副本，因此不存在额外的存储成本或同步问题。数据库维护行格式和列格式间的完全的事务一致性，就像它维护表和索引间的一致性

一样。Oracle 优化器能够完全感知列格式：它自动将分析查询路由到列格式，将 OLTP 操作路由到行格式，从而确保为所有负载提供出色的性能和完整的数据一致性，且所有应用程序无需进行任何更改。

内存中列存储

Database In-Memory 使用内存中列存储（IM 列存储），它是 Oracle 数据库系统全局区域 (SGA) 的一个新组件，称为*内存中区域*。IM 列存储中的数据不以 Oracle 数据库使用的传统行格式存放，而是使用新的列格式。IM 列存储不会取代缓冲区缓存，而是作为一种补充，以便数据现在可同时以行格式和列格式存储在内存中。

内存中区域是 SGA 内的一个静态池，其大小通过初始化参数 `INMEMORY_SIZE`（默认值为 0）进行控制。可在 `V$SGA` 中查看内存中区域的当前大小。作为静态池，对 `INMEMORY_SIZE` 参数所做的任何更改在重启数据库实例后才会生效。它也不受自动内存管理 (AMM) 的影响或控制。内存中区域的大小必须至少为 100MB。

内存中区域可分为两个池：一个 1MB 池，用于存储填充到内存中的实际列格式数据；一个 64K 池，用于存储填充到 IM 列存储的对象的元数据。可以在 `V$INMEMORY_AREA` 视图中查看每个池的可用内存的数量。两个池的相对大小通过内部试探确定，内存中区域的大部分内存分配给 1MB 池。

```
SQL> Select pool, alloc_bytes, used_bytes, populate_status
  2  From V$INMEMORY_AREA;

POOL        ALLOC_BYTES USED_BYTES POPULATE_STATUS
----------- ----------- ---------- --------------------
1MB POOL     1710227456   16777216 DONE
64KB POOL     419430400    1900544 DONE
```

图 2. `V$INMEMORY_AREA` 中显示的 INMEMORY_AREA 的空间分配详细信息

填充内存中列存储

与纯内存中数据库不同，Oracle 数据库中的所有对象并非都需要填充到 IM 列存储。IM 列存储中应填充数据库中对性能影响最大的数据。对性能影响不太大的数据则可以存放在成本较低的闪存或磁盘上。当然，如果数据库足够小，可以将所有表填充到 IM 列存储。Database In-Memory 为表和物化视图添加了一个新的 INMEMORY 属性。只将具有 INMEMORY 属性的对象填充到 IM 列存储。可以在表空间、表、（子）分区或物化视图上指定 INMEMORY 属性。如

果在表空间级别启用该属性，则默认情况下，表空间中的所有新表和物化视图都将启用 IM 列存储。

```
ALTER TABLESPACE ts_data INMEMORY;
```

图 3. 通过指定 INMEMORY 属性在 ts_data 表空间上启用 In-Memory 属性

默认情况下，具有 INMEMORY 属性的对象的所有列都将填充到 IM 列存储。但是，如果需要，可以只填充部分列。例如，以下语句对 SH 示例模式中的 SALES 表设置 In-Memory 属性，但不包括列 PROD_ID。

```
ALTER TABLE sales INMEMORY NO INMEMORY(prod_id);
```

图 4. 在 sales 表上启用 In-Memory 属性，但不包括 prod_id 列

同样，对于分区表，表的所有分区都继承 In-memory 属性，但可以只填充部分分区或子分区。

要指示某对象不再是候选对象以及要将其立即从 IM 列存储中删除，只需指定 NO INMEMORY 子句即可。

```
ALTER TABLE sales MODIFY PARTITION SALES_Q1_1998 NO INMEMORY;
```

图 5. 通过指定 NO INMEMORY 子句对 sales 表的一个分区禁用 In-Memory 属性

IM 列存储通过一组称为*工作进程* (ora_w001_orcl) 的后台进程进行填充。填充时，数据库处于完全活动状态/可完全访问。使用纯内存中数据库，直到所有数据都填充到内存中后，才可访问数据库，但这会引起严重的可用性问题。

会给每个工作进程提供对象的部分数据库块，以填充到 IM 列存储。填充是一种流式机制，填充的同时对数据采用列格式并进行压缩。

与磁盘上的表空间由多个区段构成一样，IM 列存储由多个内存中压缩单元 (IMCU) 构成。每个工作进程分配自己的 IMCU，并在其中填充部分数据库块。在填充过程中，数据不以任何特定的方式进行排序或排列。以在行格式中的显示顺序读取数据。

在数据库打开后立即以按优先级顺序排序的列表将对象填充到 IM 列存储，或首次扫描（查询）对象后将其填充到 IM 列存储。对象的填充顺序通过关键字 PRIORITY 进行控制，关键字 PRIORITY 有五个级别（参见图 7）。默认 PRIORITY 为 NONE，这表示只在首次扫描对象后才将其填充。
必须先完全填充指定优先级的所有对象，之后才可开始填充较低优先级的任何对象。但是，如果扫描到没有 PRIORITY 的对象，可以废弃填充顺序，从而触发该对象到 IM 列存储的填充。

```
ALTER TABLE customers INMEMORY PRIORITY CRITICAL;
```

图 6. 对具有 critical 优先级的 customers 表启用 In-Memory 属性

优先级	说明
CRITICAL	打开数据库后，立即填充对象
HIGH	填充完所有 CRITICAL 对象后，如果 IM 列存储中仍有可用空间，则填充该优先级的对象
MEDIUM	填充完所有 CRITICAL 和 HIGH 对象后，如果 IM 列存储中仍有可用空间，则填充该优先级的对象
LOW	填充完所有 CRITICAL、HIGH 和 MEDIUM 对象后，如果 IM 列存储中仍有可用空间，则填充该优先级的对象
NONE	如果 IM 列存储中仍有可用空间，只有在首次扫描对象后才将其填充（默认设置）

图 7. INMEMORY 子句的 PRIORITY 分子句控制不同的优先级

限制

数据库中的几乎所有对象都可以填充到 IM 列存储，但有少数情况除外。以下数据库对象无法填充到 IM 列存储：

- 归 SYS 用户拥有并存储在 SYSTEM 或 SYSAUX 表空间中的任何对象
- 索引组织表 (IOT)
- 集群表

以下数据类型在 IM 列存储中也不受支持：

- LONG（从 Oracle Database 8i 开始已弃用）
- 行外 LOB

包含这些数据类型的对象的所有其他列可以填充到 IM 列存储。仅访问位于 IM 列存储中的列的任何查询都将获益于通过列存储访问表数据。对于需要的数据来自具有不受支持列类型的列的任何查询，将通过缓冲区缓存执行。

小于 64KB 的对象不填充到内存中，因为内存是按 1MB 块分配的，这些对象会浪费 IM 列存储中的大量空间。

IM 列存储无法在当前版本的 Active Data Guard 备用实例上使用。但是，它可以在逻辑备用实例以及使用 Oracle Golden Gate 维护的实例中使用。

内存中压缩

通常只将压缩视为节省空间的一种机制。但是，填充到 IM 列存储的数据是使用一组新压缩算法进行压缩的，这谢新压缩算法不仅有助于节省空间，而且还能提高查询性能。这种新的 Oracle 内存中压缩格式允许直接对压缩的列执行查询。这意味着，将对非常少量的数据执行所有扫描和筛选操作。仅当结果集需要数据时才解压缩数据。

使用关键字 MEMCOMPRESS（INMEMORY 属性的分子句）指定内存中压缩。有六个级别，每个级别提供不同的压缩级别和性能。

压缩级别	说明
NO MEMCOMPRESS	不进行任何压缩的情况下填充数据
MEMCOMPRESS FOR DML	针对 DML 性能优化的最小压缩
MEMCOMPRESS FOR QUERY LOW	针对查询性能进行了优化（默认设置）
MEMCOMPRESS FOR QUERY HIGH	针对查询性能以及节省空间进行了优化
MEMCOMPRESS FOR CAPACITY LOW	通过更强调节省空间获得平衡
MEMCOMPRESS FOR CAPACITY HIGH	针对节省空间进行了优化

图 8. INMEMORY 子句的 MEMCOMPRESS 分子句控制不同的压缩级别

默认情况下，使用 FOR QUERY LOW 选项压缩数据，该选项提供最佳的查询性能。该选项利用常用压缩技术，如字典编码、行程编码和位打包。FOR CAPACITY 选项在 FOR QUERY 压缩基础上应用其他压缩技术，这种方式可对性能产生重大影响，因为必须先解压缩每个条目才能应用 WHERE 子句谓词。FOR CAPACITY LOW 选项应用一种称为 OZIP 的专用压缩技术，该技术的解压缩速度极快且已专门针对 Oracle 数据库进行了调优。为了提供更高的压缩率，FOR CAPACITY HIGH 选项应用重量级压缩算法，但会极大影响解压缩速度。

压缩率从 2 倍到 20 倍不等，具体取决于所选的压缩选项、数据类型和表的内容。可以对一个表内的各个列或各个分区使用不同的压缩技术。例如，可以优化表中的一些列以提高扫描速度，而优化其他一些列以节省空间。

```
CREATE TABLE employees
  (  c1 NUMBER,
     c2 NUMBER,
```

```
    c3 VARCHAR2(10),
    c4 CLOB
  )
INMEMORY MEMCOMPRESS FOR QUERY

NO INMEMORY(c4)
INMEMORY MEMCOMPRESS FOR CAPCITY HIGH(c2);
```

图 9. 指示对不同列应用不同压缩技术的 create table 命令

Oracle Compression Advisor

Oracle Compression Advisor (DBMS_COMPRESSION) 已得到增强，可支持内存中压缩。该顾问程序估算通过使用 MEMCOMPRESS 可实现的压缩率。这种估算基于表数据样本分析，并能很好地估算表填充到 IM 列存储后获得的实际结果。由于顾问程序实际上对数据应用了新的 MEMCOMPRESS 算法，因此仅可在 Oracle Database 12.1.0.2（或更高版本）环境中运行。

```
DECLARE
    l_blkcnt_cmp        PLS_INTEGER;
    l_blkcnt_uncmp      PLS_INTEGER;
    l_row_cmp           PLS_INTEGER;
    l_row_uncmp         PLS_INTEGER;
    l_cmp_ratio         PLS_INTEGER;
    l_comptype_str      VARCHAR2(100);
    comp_ratio_allrows  NUMBER := -1;
BEGIN
    dbms_compression.Get_compression_ratio  (
    -- Input parameters
    scratchtbsname  => 'TS_DATA',
    ownname         => 'SSB',
    objname         => 'LINEORDER',
    subobjname      => NULL,
    comptype        => dbms_compression.comp_inmemory_query,
    -- Output parameter
    blkcnt_cmp      => l_blkcnt_cmp,
    blkcnt_uncmp    => l_blkcnt_uncmp,
    row_cmp         => l_row_cmp,
    row_uncmp       => l_row_uncmp,
    cmp_ratio       => l_cmp_ratio,
    comptype_str    => l_comptype_str,
    subset_numrows  => dbms_compression.comp_ratio_allrows);
    dbms_output.Put_line('The  IM compression  ratio is '|| cmp_ratio);
END;
/
```

图 10. 使用 Oracle Compression Advisor (DBMS_COMPRESSION) 确定 MEMCOMPRESS 的影响

请注意，使用 ALTER TABLE 语句更改列的压缩子句会导致重新填充 IM 列存储的所有现有数据。

内存中扫描

分析查询通常只引用表的少数几列。Oracle Database In-Memory 仅访问查询所需的列，并直接对这些列应用任何 WHERE 子句筛选谓词，而无需先对其解压缩。这可极大减少需要访问和处理的数据量。

内存中存储索引

使用内存中存储索引可进一步减少访问的数据量，内存中存储索引在 IM 列存储中的每一列上自动创建和维护。存储索引允许基于 SQL 语句中提供的筛选谓词进行数据修剪。内存中存储索引跟踪 IMCU 中各列的最小值和最大值。如果查询指定了 WHERE 子句谓词，将查看引用列上的内存中存储索引，以便通过比较指定值与存储索引中维护的最小值和最大值来确定具有指定列值的任何条目是否存在于每个 IMCU 中。如果指定列值位于某 IMCU 的最小值和最大值范围之外，则避免扫描该 IMCU。

对于等式谓词、列表谓词和一些范围谓词，还可以通过在使用基于字典的压缩时为每个 IMCU 创建的元数据字典进一步执行数据修剪。元数据字典包含适用于该 IMCU 中的每个列的不同值的列表。这样，通过基于字典的修剪，Oracle 数据库可确定搜索的值是否存在于 IMCU 中，从而确保仅扫描必要的 IMCU。

SIMD 向量处理

对于确实需要扫描 IM 列存储的数据，Database In-Memory 使用 SIMD 向量处理（单个指令处理多数据值）。SIMD 向量处理允许发出一条 CPU 指令来估算一组列值，而不是一次估算列中的一个条目。

IM 列存储中使用的列格式经过专门设计，可以将最多的列条目加载到 CPU 上的向量寄存器并通过一条 CPU 指令进行估算。SIMD 向量处理使 Oracle Database In-Memory 能够每秒扫描数十亿行。

例如，我们假设使用 SH 示例模式中的 SALES 表（参见图 11），假设要求我们统计使用 PROMO_ID 值为 9999 的销售订单的总数。SALES 表已完全填充到 IM 列存储。查询首先只扫描 SALES 表的 PROMO_ID 列。PROMO_ID 列的前 8 个值加载到 CPU 上的 SIMD 寄存器，然后通过一条 CPU 指令与 9999 进行比较（加载的值的数量将根据使用的数据类型和内存压缩而

有所不同）。记录与 9999 匹配的条目数，然后丢弃这些条目，将另外 8 个条目加载到寄存器中进行估算。如此持续，直到估算完 PROMO_ID 列中的所有条目。

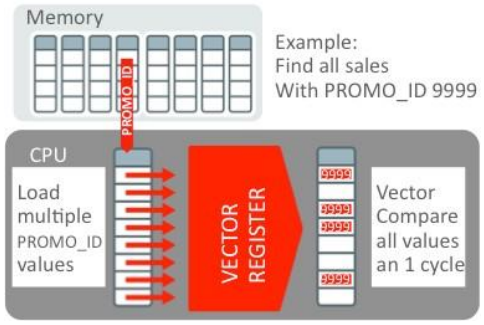


图 11. 使用 SIMD 向量处理能够每秒扫描数十亿行

要确定 SQL 语句是否正在扫描 IM 列存储中的数据，请查看执行计划。

```
-------------------------------------------
| Id  | Operation                   | Name  |
-------------------------------------------
|   0 | SELECT STATEMENT            |       |
|   1 |  SORT AGGREGATE             |       |
|   2 |   PARTITION RANGE ALL       |       |
|*  3 |    TABLE ACCESS INMEMORY FULL| SALES |
-------------------------------------------
```

图 12. 执行计划中的新 IN MEMORY 关键字指明可在内存中执行的操作

您会注意到，执行计划显示一组新关键字“IN MEMORY”。这些关键字指明 **LINEORDER** 表已标记为 IN MEMORY，Oracle 数据库**可以**在该查询中使用列存储。

为确认使用了 IM 列存储，请查看会话级统计信息。可以通过查询性能视图 V$MYSTAT、V$SESSTAT 和 V$STATNAME 监视会话级统计信息。与 IM 列存储相关的所有统计信息都以“IM”开头，如“IM scan rows”— 统计通过内存中扫描处理的行数。

内存中联接

联接多个表的 SQL 语句也可以在 IM 列存储中得到高效处理，因为它们可以利用**布隆筛选器**。布隆筛选器将联接转换为筛选器，然后可在大型表的扫描过程中应用。布隆筛选器最初是在 Oracle Database 10g 中引入的，其目的是增强散列联接的性能，不是特定于 Oracle Database In-Memory 的。但是，可通过 SIMD 向量处理将其高效应用于列格式数据。

当通过散列联接对两个表进行联接时，扫描第一个表（通常是较小的表），然后使用满足 WHERE 子句谓词（针对该表）的行创建内存中散列表（存储在进程全局区域 (PGA) 中）。在创建散列表过程中，还基于联接列创建位向量或布隆筛选器。然后，将位向量作为附加谓词发

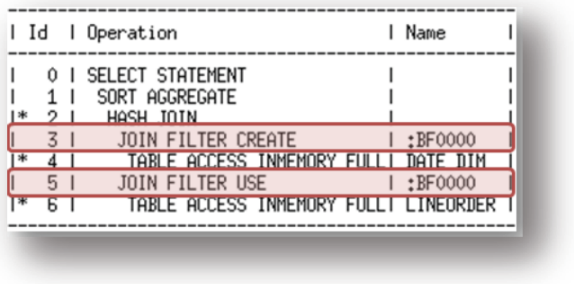
送给第二个表的扫描。将 WHERE 子句谓词应用于第二个表扫描后，将对结果行的联接列执行散列操作，并与位向量中的值进行比较。如果在位向量中找到匹配项，则将该行发送给散列联接。如果未找到匹配项，则将丢弃该行。

识别执行计划中的布隆筛选器是非常容易的。它们将出现在两个位置 — 创建时以及应用时。下面我们以 DATE_DIM 与 LINEORDERS 表之间的简单两表联接为例。

```
SELECT SUM(lo_extendedprice * lo_discount) revenue
FROM   lineorder l,
       date_dim d
WHERE  l.lo_orderdate = d.d_datekey
AND    l.lo_discount BETWEEN 2 AND 3
AND    d.d_date='December 24, 2013';
```

图 13. 将受益于内存中列存储的布隆筛选器的简单两表联接

以下是该查询的执行计划，其中布隆筛选器高亮显示。该执行计划的第一步实际上是第 4 行：内存中全表扫描 DATE_DIM 表。DATE_DIM 表扫描完成后立即创建布隆筛选器 (:BF0000)（第 3 行）。然后在 LINEORDER 表的内存中全表扫描过程中应用该布隆筛选器（第 5 和 6 行）。



Id	Operation	Name
0	SELECT STATEMENT	
1	SORT AGGREGATE	
*2	HASH JOIN	
3	JOIN FILTER CREATE	:BF0000
*4	TABLE ACCESS INMEMORY FULL	DATE_DIM
5	JOIN FILTER USE	:BF0000
*6	TABLE ACCESS INMEMORY FULL	LINEORDER

图 14. 在 DATE_DIM 与 LINEORDER 表的两表联接中创建并使用布隆筛选器

通过查看计划中的谓词信息可以了解用于构建布隆筛选器的联接条件。在筛选谓词中查找“SYS_OP_BLOOM_FILTER”。如果联接转换为布隆筛选器，您可能会奇怪为什么计划中出现 HASH JOIN（第 2 行）。存在 HASH JOIN 是因为布隆筛选器有可能会返回虚假结果。HASH JOIN 确认从 LINEORDER 表扫描返回的所有行对于联接条件都是真匹配项。通常这会消耗极少的工作量。

对于联接多个表的更复杂查询，会发生什么情况呢？这正是 Oracle 30 多年的数据库创新的用武之地。通过将 IM 列存储无缝构建到 Oracle 数据库，我们可以利用自第一版起添加到数据库中的所有优化。使用一系列优化器转换，可以重新编写多个表联接，以便能够创建多个布隆筛选器并将其使用在大型表或事实表的扫描过程中。

注意：从 Oracle Database 12.1.0.2 起，当对填充到 IM 列存储的表执行串行查询时，可对串行查询使用布隆筛选器。为创建并使用布隆筛选器，查询中的所有表并非都需要填充到 IM 列存储。

内存中聚合

分析式查询通常需要的不仅仅是简单筛选和联接。它们需要复杂的聚合和汇总。Oracle Database 12.1.0.2 中引入了一个新优化器转换（称为 *Vector Group By*），能够确保可使用有效利用 CPU 的新算法处理更复杂的分析查询。

Vector Group By 转换流程分为两部分，这一点与星型转换没有什么不同。我们以下面的业务查询为例：统计直销店鞋类产品的总销售额。

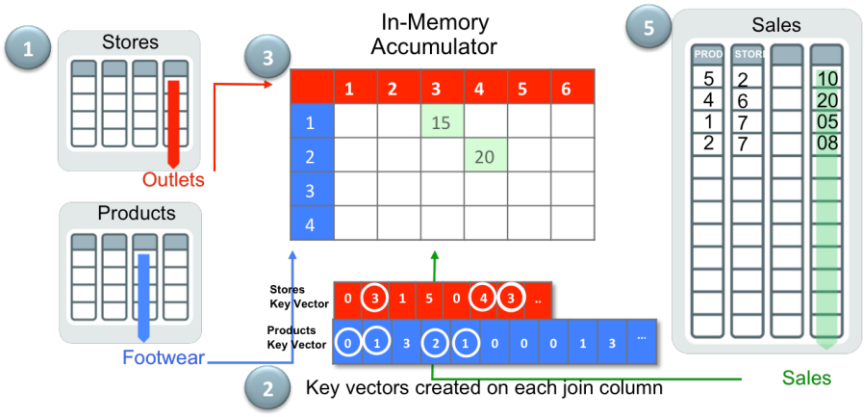


图 15. 内存中聚合示例 — 统计直销店鞋类产品的总销售额

第 1 阶段

1. 查询首先扫描两个维度表（较小的表）STORES 和 PRODUCTS（以下计划中的第 5 和 10 行）。
2. 根据各个扫描结果创建称为*键向量* 的新数据结构（以下计划中的第 4、9 和 13 行）。键向量类似于布隆筛选器，因为它允许在 SALES 表（最大的表）的扫描过程中将联接谓词作为附加筛选谓词进行应用。与布隆筛选器不同，键向量不会返回虚假结果。
3. 还可使用键向量创建另一个结构，称为内存中累加器。累加器是一个在 PGA 中构建的多维数组，支持 Oracle 数据库在 SALES 表扫描过程中执行聚合或 GROUP BY 操作，而不必在扫描之后执行。

4. 第一阶段结束时，会创建临时表，用以存放较小维度表中的有效负载列（SELECT 列表中引用的列）（以下计划中的第 2、6 和 11 行）。请注意，这一步未在上面的图 15 中给出。

第 2 阶段

5. 执行计划的第二部分首先扫描 SALES 表以及应用键向量（以下计划中的第 24-29 行）。对于 SALES 表中与联接条件（直销店与鞋类产品）匹配的每个条目，相应的销售额将添加到内存中累加器的适当单元中。如果相应单元中已存在值，则将两个值相加，然后将结果值放入该单元中。

6. 最后，将大表扫描的结果联接回维度表扫描过程中创建的临时表（第 16、18 和 19 行）。切记，这些临时表仅包含有效负载列。注意，这一步未在上面的图 15 中给出。

这两个阶段结合起来，通过复杂聚合显著提高了多个表联接的效率。

```
-------------------------------------------------------------------------------
| Id  | Operation                          | Name                      |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                           |
|   1 |  TEMP TABLE TRANSFORMATION         |                           |
|   2 |   LOAD AS SELECT                   | SYS_TEMP_0FD9D6635_4F9FC7 |
|   3 |       VECTOR GROUP BY              |                           |
|   4 |     KEY VECTOR CREATE BUFFERED     | :KV0000                   |
|   5 |      TABLE ACCESS INMEMORY FULL    | STORES                    | PHASE 1
|   6 |   LOAD AS SELECT                   | SYS_TEMP_0FD9D6636_4F9FC7 |
|   7 |       VECTOR GROUP BY              |                           |
|   8 |        HASH GROUP BY               |                           |
|   9 |      KEY VECTOR CREATE BUFFERED    | :KV0001                   |
|  10 |       TABLE ACCESS INMEMORY FULL   | PRODUCTS                  |
|  11 |   SORT GROUP BY                    |                           |
|  12 |     HASH JOIN                      |                           |
|  13 |      MERGE JOIN CARTESIAN          |                           |
|  14 |       TABLE ACCESS FULL            | SYS_TEMP_0FD9D6636_4F9FC7 |
|  15 |       BUFFER SORT                  |                           |
|  16 |        TABLE ACCESS FULL           | SYS_TEMP_0FD9D6635_4F9FC7 |
|  17 |      VIEW                          | VW_VT_80F21617            |
|  18 |       VECTOR GROUP BY              |                           |
|  19 |        HASH GROUP BY               |                           |
|  20 |         KEY VECTOR USE             | :KV0000                   |PHASE 2
|  21 |           KEY VECTOR USE           | :KV0001                   |
|  22 |            TABLE ACCESS INMEMORY FULL| SALES                   |
-------------------------------------------------------------------------------
```

图 16. 受益于内存中聚合的查询执行计划

VECTOR GROUP BY 转换是基于成本的转换，这意味着优化器将估算包含和不包含转换的执行计划的成本，然后选择具有最低成本的执行计划。例如，以下情形下可以选择 VECTOR GROUP BY 转换：

- 表之间的联接列包含“主要的”唯一键或数字键
- 事实表（查询中的最大表）至少比其他表大 10 倍

- 表填充到 IM 列存储

以下情形下不太可能选择 VECTOR GROUP BY 转换：

- 在两个或多个非常大的表之间进行联接
- 维度表包含的行数超过 20 亿行
- 系统没有足够的内存资源

DML 和内存中列存储

很明显，IM 列存储可显著提高所有查询类型的性能，但极少数据库环境是只读的。要使 IM 列存储在现代数据库环境中真正有效，它必须能够处理批量数据加载**和**联机事务处理。

批量数据加载

批量数据加载在数据仓库环境中最常用，通常作为直接路径加载执行。直接路径加载解析输入数据，将每个输入字段的数据转换为其对应的 Oracle 数据类型，然后为数据构建列数组结构。可以使用这些列数组结构设置 Oracle 数据块的格式并构建索引键。然后将新格式化数据库块直接写入数据库，绕过标准的 SQL 处理引擎和数据库缓冲区缓存。

直接路径加载操作是一个全有或全无操作。这意味着，所有数据都加载后才可提交该操作。如果操作过程中出现错误，将中止整个操作。为满足这一严格标准，直接路径加载将数据插入高于段高水位（目前对象或段使用最多的数据库块）创建的数据库块中。提交直接路径加载后，将移动高水位以将新创建的块包含到段中，且使这些块对于同一表中的其他 SQL 操作可见。到目前为止，IM 列存储还不知道段上发生的任何数据更改。

提交操作后，IM 列存储立刻知道还没有填充对象的所有数据。所缺数据的多少将在 v$IM_SEGMENTS 视图的 BYTES_NOT_POPULATED 列中看到（请参见监视一节）。如果对象上指定了 PRIORITY，新添加的数据将自动填充到 IM 列存储。否则，下次查询对象时，将触发后台工作进程，以开始填充缺失数据，前提是 IM 列存储中有可用空间。

分区交换加载

强烈建议对数据仓库中较大的表或事实表进行分区。分区的一个优势是能够通过使用交换分区命令快速轻松地加载数据，且对用户的影响最小。通过交换分区命令，可将未分区表中的数据

交换到分区表中的特定分区。该命令并不真正移动数据，而是更新数据字典，将指针从分区交换到表，反之亦然。由于数据没有物理移动，因此交换不生成重做和撤销，瞬间即可完成，且影响性能的可能性远低于 INSERT 等任何传统数据移动方法。

与直接目录操作一样，直到操作完成后，IM 列存储才知道发生了分区交换加载。此时，临时表中的数据是已经成为分区表的一部分。如果对临时表设置了 INMEMORY 属性，且其所有数据已填充到 IM 列存储，则不会发生其他事情。仅通过 IM 列存储即可访问临时表中的数据，下次扫描分区表时可访问分区表中的其他数据。

但是，如果未对临时表设置 INMEMORY 属性，对新交换分区中数据的所有后续访问将通过缓冲区缓存完成。切记，INMEMORY 属性是对象的一个物理属性。如果希望分区在交换后具有该属性，必须在交换之前在临时表上指定该属性。在空分区上指定该属性无法实现此目的。

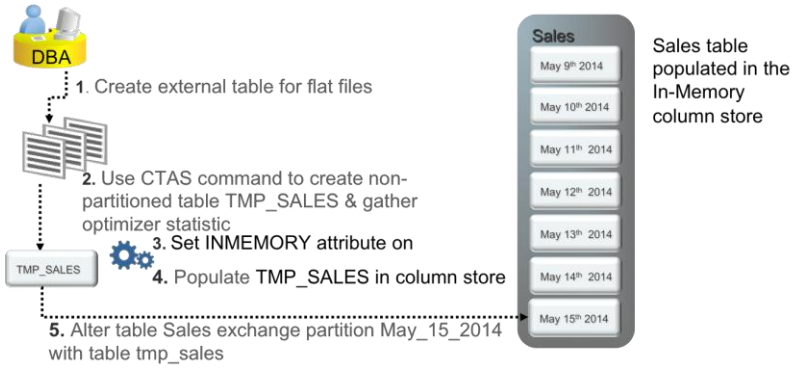


图 17. 在 INMEMORY 表上完成分区交换加载所需的五个步骤

事务处理

单行数据更改操作 (DML) 通过缓冲区缓存执行（OLTP 式更改），这一点与未启用 Database In-Memory 时一样。如果发生 DML 操作的对象填充至 IM 列存储，那么更改在发生时就会在 IM 列存储中反映出来。缓冲区缓存与列存储通过内存中事务管理器在事务上保持一致。与以前一样，所有日志记录都在基表上完成，IM 列存储不需要日志记录。

对 IM 列存储中的每个 IMCU，都会自动创建和维护一个事务日记账（参见图 18）。当 DML 语句更改填充到 IM 列存储中的对象的某行时，该行对应的条目将在 IMCU 中标记为陈旧，且该行的新版本副本将添加到内存中事务日记账中。为提供读取一致性以及维持数据压缩，不立刻替换 IMCU 中的原始条目。如果对 IM 列存储中的对象执行的任何事务是在执行 DML 之前开

始的，则需要查看条目的原始版本。IM 列存储中的读取一致性通过系统更改编号 (SCN) 进行管理，这一点与未启用 Database In-Memory 时一样。

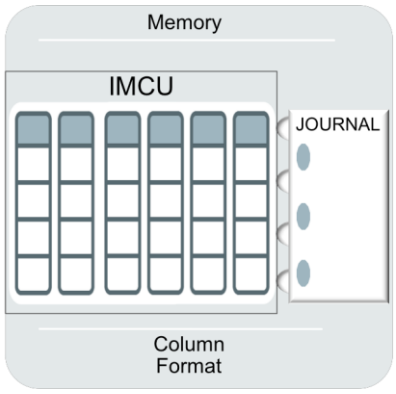


图 18. IM 列存储中的每个 IMCU 都包含对象的部分行和一个事务日记账

当针对对象执行具有新 SCN 的查询时，除陈旧条目之外，该查询将读取 IMCU 中列的所有条目。而从事务日记账或基表（缓冲区缓存）检索那些陈旧条目。

重新填充

IMCU 中的陈旧条目越多，IMCU 的扫描速度就越慢。因此，当 IMCU 中的陈旧条目数达到陈旧程度阈值时，Oracle 数据库将重新填充 IMCU。陈旧程度阈值通过试探方式确定，这种试探方式将考虑 IMCU 的访问频率以及 IMCU 中陈旧行的数量。对于访问频率较高或陈旧行比例较高的 IMCU，重新填充会更频繁。IMCU 的重新填充是一个由后台工作进程执行的联机操作。数据始终可用，且自动记录重新填充过程中对 IMCU 中的行所做的任何更改。

除标准的重新填充算法之外，还有另一种算法，该算法试图使用低优先级后台进程清除所有陈旧条目。IMCO（内存中协调器）后台进程还可能会对 IM 列存储中包含部分陈旧条目但目前尚未达到陈旧程度阈值的任何 IMCU 执行*涓流重新填充*。涓流重新填充是一个持续不断的后台活动。

IMCO 每隔两分钟就唤醒一次，检查是否有任何填充任务需要完成。例如，刚刚对某新对象使用 `PRIORITY` 分子句指定了 `INMEMORY` 属性。IMCO 还将检查 IM 列存储中是否有包含陈旧条目的 IMCU。如果找到一些，它将触发工作进程，对这些 IMCU 进行重新填充。在给定的 2 分钟内通过涓流重新填充所重新填充的 IMCU 的个数受新初始化参数 `INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT` 的限制。该参数控制工作进程可参与涓流重新填充活动的最大时间百分比。参与的工作进程数越多，可涓流重新填充的 IMCU 数

也就越多，但参与的工作进程数越多，CPU 消耗就越多。将 INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT 设为 0，可完全禁用涓流重新填充。

IM 列存储保持事务一致性带来的开销

IM 列存储保持事务一致性带来的开销随应用程序而异，具体取决于多种因素，其中包括：更改速度、为表选择的内存中压缩级别、更改行的位置以及执行的操作类型。具有较高压缩级别的表将比具有较低压缩级别的表产生的开销要多。

共同位于同一块中的更改行将比在表内随机分布的更改行产生的开销要少。共同位于同一块中的更改行的示例有新插入的行，因为数据库通常将这些行组织在一起。另一个示例是使用直接路径加载操作加载的数据。

对于具有高 DML 速率的表，建议使用 MEMCOMPRESS FOR DML，此外，如果可能，还建议使用分区在表内定位更改。例如，可使用范围分区按日期确定数据在表内的位置，以便将大多数更改限于最新分区中存储的数据。日期范围分区还提供许多其他可管理性和性能优势。

RAC 上的内存中列存储

RAC 环境中的每个节点均有其自己的 IM 列存储。强烈建议将每个 RAC 节点的 IM 列存储设置为相同的大小。对于不需要 IM 列存储的任何 RAC 节点，应将 INMEMORY_SIZE 参数设为 0。可以在每个节点上填充完全不同的对象，也可以在集群的所有 IM 列存储间分布较大的对象。还可以在每个节点的 IM 列存储中存储相同的对象（仅限工程化系统）。对象在集群中各个 IM 列存储间的分布通过 INMEMORY 属性的两个附加分子句进行控制：DISTRIBUTE 和 DUPLICATE。

在 RAC 环境中，仅指定了 INMEMORY 属性的对象才可在集群的所有 IM 列存储间分布。对象在集群中的分布方式通过 DISTRIBUTE 分子句进行控制。默认情况下，Oracle 基于使用的分区类型（如果有）决定在集群中分布对象的最佳方式。或者，可以指定 DISTRIBUTE BY ROWID RANGE 按 rowid 范围分布，指定 DISTRIBUTE BY PARTITION 将分区分布给不同的节点，或指定 DISTRIBUTE BY SUBPARTITION 将子分区分布给不同的节点。

```
ALTER TABLE lineorder INMEMORY DISTRIBUTE BY PARTITION;
```

图 19. 该命令按分区在集群的各个 IM 列存储间分布 lineorder 表。

如果按 HASH 对表进行分区或子分区且期望使用智能化分区联接计划，建议使用 DISTRIBUTE BY PARTITION 或 SUBPARTITION。这样，每个分区联接可共同位于一个

节点内。DISTRIBUTE BY ROWID RANGE 可用于未分区表，也可用于使用 DISTRIBUTE BY PARTITION 会导致数据偏差的分区表。

如果对象非常小（仅包含一个 IMCU），该对象将仅填充到集群中一个节点的 IM 列存储中。

当针对分布式对象执行查询时，将利用并行服务器进程在每个 RAC 节点上对位于相应节点的 IM 列存储中的部分对象执行查询。查询协调器将每个并行服务器进程的结果聚合在一起，然后再将其返回给最终用户的会话。

如果针对位于某节点的 IM 列存储中的对象或部分对象发出 DML 语句，则将 IM 列存储中的相应行标记为陈旧，且将新行副本添加到该 IMCU 中的事务日记账中。但是，如果在不同节点上发出 DML 语句，将使用缓存融合保持 IM 列存储在事务上的一致性，即在远程节点的相应 IM 列存储中将具有更改行的数据库块中的列条目标记为陈旧。

内存中容错

在工程化系统上，可以通过指定 INMEMORY 属性的 DUPLICATE 分子句镜像填充到 IM 列存储的数据。这意味着，填充到 IM 列存储的每个 IMCU 都将在 RAC 集群的另外一个节点上存放一个镜像副本。镜像 IMCU 提供了内存中容错功能，因为，即使某节点发生故障也可确保仍能通过 IM 列存储访问数据。它还提高性能，因为查询可随时访问主 IMCU 及其备份副本。

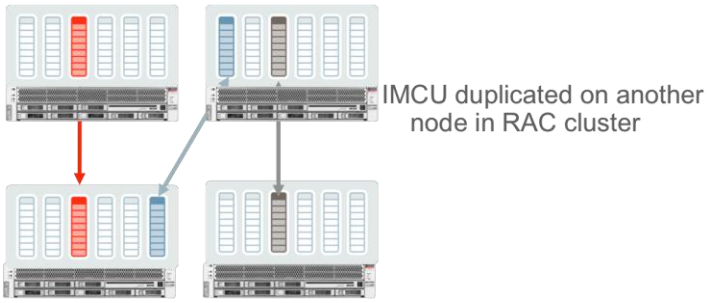


图 20. 可镜像工程化系统上 IM 列存储中的对象以提高容错能力

如果某 RAC 节点发生故障并停用一段时间，唯一的影响将是重新镜像位于该节点上的主 IMCU。只有第二个节点发生故障并停用一段时间才需要重新分布数据。

如果需要额外的容错能力，可以通过指定 INMEMORY 属性的 DUPLICATE ALL 分子句将对象填充到集群中每个节点的 IM 列存储。这将提供最高级别的冗余性，并提供线性可伸缩性，因为查询将能够在单个节点内全部执行。

```
ALTER TABLE lineorder INMEMORY DUPLICATE ALL;
```

图 21. 该命令确保 lineorder 表的每个 IMCU 都将出现在集群的所有 IM 列存储中

DUPLICATE ALL 选项还可帮助将大的分布式事实表与较小的维度表间的联接位于同一位置。通过在较小的维度表上指定 DUPLICATE ALL 选项，这些表的一个完整副本将填充到每个节点的 IM 列存储。

DUPLICATE 分子句仅适用于 Oracle 工程化系统，如果在其他系统上指定则会被忽略。

如果非工程化系统上的某 RAC 节点出现故障，填充到该节点 IM 列存储的数据将在集群的内存中不再可用。针对缺失的对象片段发出的查询将不会失败。它们将改为访问缓冲区缓存或存储中的数据，但这将影响这些查询的性能。如果节点停用一段时间，位于该节点 IM 列存储中的对象或对象片段将填充到集群的其余节点上（假设有可用空间）。为使故障 RAC 节点对性能产生的影响降至最低，建议在集群的每个节点的 IM 列存储中留出一些可用空间。

请注意，某节点或实例发生故障后并不会立即将数据重新分布到集群的其他节点，因为该节点或实例很可能会快速恢复正常使用。如果立即重新分布数据，重新分布进程会向系统增添额外负载，但节点或实例恢复正常使用后将无法撤消这些负载。因此，系统将等待几十分钟，之后才开始重新分布数据。这一等待留出了节点或实例重新联接集群的时间。

节点重新联接集群后，数据将重新分布到新联接的节点上。逐个 IMCU 分布数据，且在该过程中完全可访问对象。

多租户环境中的内存中列存储

Oracle Multitenant[1] 是一个新的数据库整合模型，在该模型中，多个可插拔数据库 (PDB) 整合在一个容器数据库 (CDB) 中。在保持单一数据库的多个隔离特性的同时，它还支持多个 PDB 共享一个通用 CDB 的系统全局区域 (SGA) 和后台进程。因此，多个 PDB 也共享一个 IM 列存储。

[1] 有关 Oracle Multitenant 的更多信息，请参阅白皮书 Oracle Multitenant

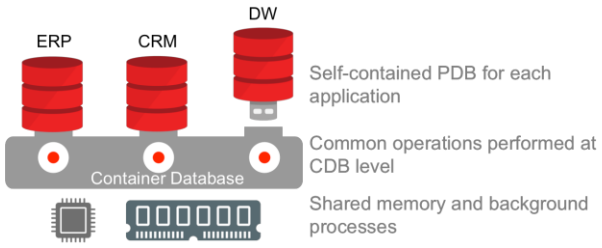


图 22. 一个 Oracle Database 12c 容器数据库中包含三个 PDB

IM 列存储的总大小通过在 CDB 中设置 INMEMORY_SIZE 参数进行控制。通过设置每个 PDB 的 INMEMORY_SIZE 参数，来指定其能够使用的共享 IM 列存储的大小。并非给定 CDB 中的所有 PDB 都需要使用内存中列存储。一些 PDB 可以将 INMEMORY_SIZE 参数设为 0，这意味着它们根本不使用内存中列存储。

PDB 的 INMEMORY_SIZE 参数的总和无需小于或等于 CDB 的 INMEMORY_SIZE 参数的大小。PDB 可以过度使用 IM 列存储。允许过度使用可确保某个可插拔数据库关闭或拔出的情况下不浪费 IM 列存储的宝贵空间。由于 INMEMORY_SIZE 参数是静态的（需要数据库实例重启才能反映更改），最好允许 PDB 过度使用，以便可以使用 IM 列存储的所有空间。

但是，由于这种过度使用，一个 PDB 可能会抢占另一个 PDB 在 IM 列存储中的空间。如果不希望任何 PDB 长时间关闭或拔出，建议不采用过度使用。

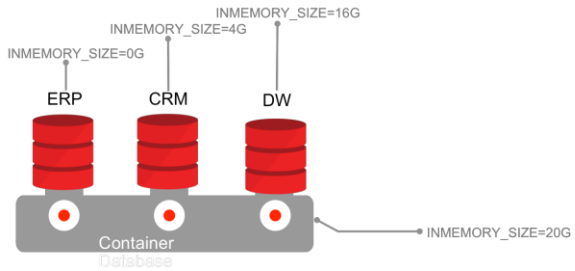


图 23 通过设置 INMEMORY_SIZE 参数，PDB 能够指定可使用的共享 IM 列存储的大小

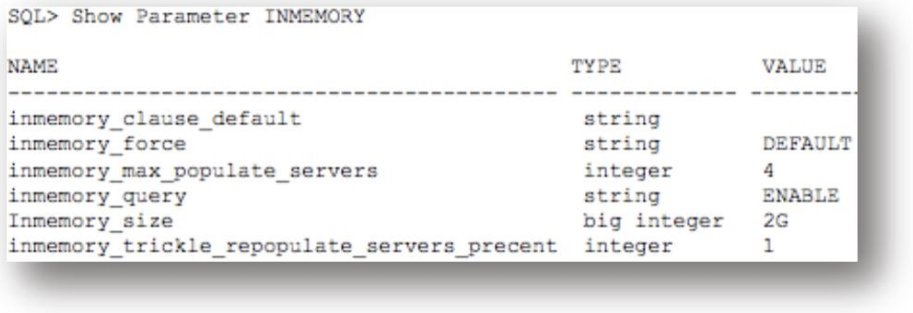
每个可插拔数据库 (PDB) 本身就是一个完整的 Oracle 数据库，因此每个 PDB 都将有其自己的优先级列表。当 PDB 启动时，其优先级列表中的对象将按顺序填充到内存中列存储（假设有可用空间）。

控制 Oracle Database In-Memory 的使用

有多个新初始化参数和优化器提示可用于控制使用 IM 列存储的时间和方式。本节将介绍所有这些初始化参数和优化器提示，并对哪些是核心初始化参数以及哪些是可选初始化参数提供了指导。

核心初始化参数

引入了六个带有 INMEMORY 前缀的新初始化参数，来直接控制新内存中功能的各个方面。还有一个新优化器参数，该参数对查询是否使用 IM 列存储有一定的影响。



```
SQL> Show Parameter INMEMORY

NAME                                 TYPE        VALUE
------------------------------------ ----------- ---------
inmemory_clause_default              string
inmemory_force                       string      DEFAULT
inmemory_max_populate_servers        integer     4
inmemory_query                       string      ENABLE
Inmemory_size                        big integer 2G
inmemory_trickle_repopulate_servers_precent  integer     1
```

图 24. 新的内存中初始化参数

INMEMORY_SIZE

如本文前面所述，INMEMORY_SIZE 参数控制分配给 IM 列存储的内存量。默认大小为 0 字节。该参数只能在系统级进行修改，且需要数据库重启才能生效。INMEMORY_SIZE 参数所需的最小大小为 100 MB。

INMEMORY_QUERY

Oracle 优化器能够感知填充到 IM 列存储的对象，会自动将它认为可受益于内存中列格式的所有查询定向到 IM 列存储。可以在会话级或系统级将 INMEMORY_QUERY 设置为 DISABLE，来禁用 IM 列存储。默认值为 ENABLE。

INMEMORY_MAX_POPULATE_SERVERS

可以启动的最大工作进程数通过 INMEMORY_MAX_POPULATE_SERVERS 进行控制，该参数默认设置为 0.5 X CPU_COUNT。减少工作进程数将减少填充过程中占用的 CPU 资源，但可能会延长填充 IM 列存储所花费的时间。

其他初始化参数

INMEMORY_CLAUSE_DEFAULT

借助 INMEMORY_CLAUSE_DEFAULT 参数，可以通过为在语法中未显式指定的所有 INMEMORY 分子句指定一组有效值来为内存中表指定默认模式。默认值是一个空字符串，这意味着只有显式指定的表才能填充到 IM 列存储。

```
ALTER SYSTEM SET inmemory_clause_default= 'INMEMORY PRIORITY LOW';
```

图 25. 使用 INMEMORY_CLAUSE_DEFAULT 参数将所有新表都标记为 IM 列存储的候选表

参数值的解析方式与 INMEMORY 子句相同，如果未指定某个分子句，则使用相同的默认值。针对内存中列存储显式指定的任何表将继承该参数的任何未指定值。

INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT

该参数控制工作进程可执行涓流重新填充的最大时间百分比。该参数的值是 INMEMORY_MAX_POPULATE_SERVERS 参数的一个百分比。将该参数设为 0 将禁用涓流重新填充；默认值为 1，这意味着工作进程将工作时间的 1% 用于执行涓流重新填充。

INMEMORY_FORCE

默认情况下，指定了 INMEMORY 属性的任何对象都可填充到 IM 列存储。但是，如果 INMEMORY_FORCE 设置为 OFF，那么即使配置了内存中区域，也不会将任何表放入内存中。默认值为 DEFAULT。

OPTIMIZER_INMEMORY_AWARE

如上所述，优化器能够感知 IM 列存储，并在估算 SQL 语句的备选内存中计划的成本后使用内存中特定的成本。通过将 OPTIMIZER_INMEMORY_AWARE 参数设为 FALSE，可以禁用优化器成本模型的所有内存中增强。

优化器提示

内存中的各个方面（内存中扫描、内存中联接和内存中聚合）可以通过使用优化器提示，在语句或语句块级进行控制。与大多数优化器提示一样，以下描述的每个提示的对应否定提示前面都跟有单词“NO_”。切记，优化器提示是将遵循的指令（如适用）。指定内存中提示不会强制将未设置 INMEMORY 属性的对象填充到 IM 列存储。

内存中扫描

有两个新优化器提示可帮助控制内存中扫描的使用。INMEMORY 提示允许在语句块级启用或禁用对内存中对象的扫描。

此外，还可以使用 INMEMORY_PRUNING 提示控制内存中存储索引的使用。

内存中联接

使用布隆筛选器将联接转换为筛选器是一个基于成本的决策。如果优化器不选择布隆筛选器，则可以通过使用 PX_JOIN_FILTER 提示强制选择。

内存中聚合

新的内存中聚合特性 (VECTOR GROUP BY) 是一个基于成本的查询转换，这意味着，即使优化器不认为转换是成本最低的执行计划，也可以强制执行转换。通过指定 VECTOR_TRANSFORM 提示，可以强制执行 VECTOR GROUP BY 计划。

监视和管理 Oracle Database In-Memory

监视哪些对象位于内存中列存储中

存在两个新的 v$ 视图 — v$IM_SEGMENTS 和 v$IM_USER_SEGMENTS，这两个视图可指示目前填充到 IM 列存储的对象。

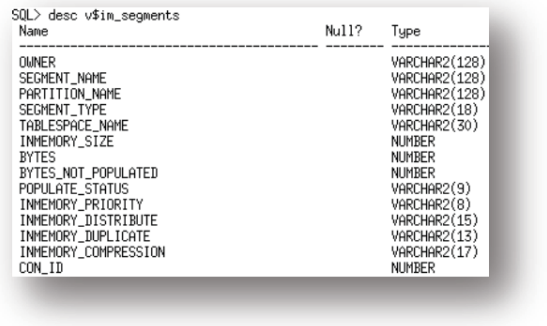


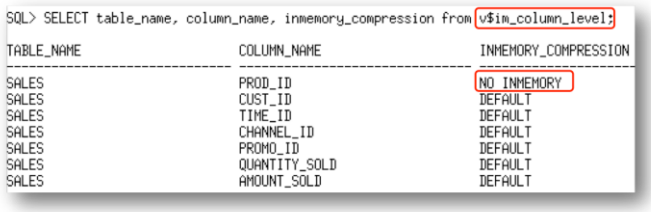
图 26. 新的 v$IM_SEGMENTS 视图

这些视图不仅显示填充到 IM 列存储的对象，而且还指示对象在 RAC 集群中的分布方式以及是否填充了整个对象 (BYTES_NOT_POPULATED)。还可以使用这些视图确定填充到 IM 列存储的每个对象实现的压缩率（假设对象未在磁盘上进行压缩）。

```
SELECT  v.owner,  v.segment_name,
        v.bytes                       orig_size,
        v.inmemory_size               in_mem_size,
        v.bytes  /  v.inmemory_size   comp_ratio
FROM    v$im_segments  v;
```

图 27. 确定填充到 IM 列存储的对象实现的压缩率

另一个新视图 v$IM_COLUMN_LEVEL 包含有关填充到列存储的列的详细信息，因为并非表中的所有列都需要填充到列存储。



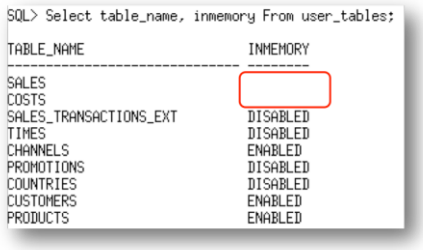
```
SQL> SELECT table_name, column_name, inmemory_compression from v$im_column_level;

TABLE_NAME                     COLUMN_NAME                    INMEMORY_COMPRESSION
------------------------------ ------------------------------ --------------------
SALES                          PROD_ID                        NO INMEMORY
SALES                          CUST_ID                        DEFAULT
SALES                          TIME_ID                        DEFAULT
SALES                          CHANNEL_ID                     DEFAULT
SALES                          PROMO_ID                       DEFAULT
SALES                          QUANTITY_SOLD                  DEFAULT
SALES                          AMOUNT_SOLD                    DEFAULT
```

图 28. PROD_ID 列未填充到 IM 列存储

USER_TABLES

向 *_TABLES 字典表中添加了一个新的布尔型列（称为 INMEMORY），以指示对哪些表指定了 INMEMORY 属性。



```
SQL> Select table_name, inmemory From user_tables;

TABLE_NAME                     INMEMORY
------------------------------ --------
SALES
COSTS
SALES_TRANSACTIONS_EXT         DISABLED
TIMES                          DISABLED
CHANNELS                       ENABLED
PROMOTIONS                     DISABLED
COUNTRIES                      DISABLED
CUSTOMERS                      ENABLED
PRODUCTS                       ENABLED
```

图 29. 添加到 *_TABLES 中的新 INMEMORY 列，指示哪些表具有 INMEMORY 属性

在上例中，您会注意到，COSTS 和 SALES 这两个表的 INMEMORY 列没有值。INMEMORY 属性是一个段级属性。COSTS 和 SALES 都是分区表，因此都是逻辑对象。这些表的 INMEMORY 属性将在分区或子分区级别，在 *_TAB_(SUB)PARTITIONS 中进行记录。

还向 *_TABLES 视图中添加了另外三列 — INMEMORY_PRIORITY、INMEMORY_DISTRIBUTE 和 INMEMORY_COMPRESSION，以指示每个表的当前内存中属性。

管理 IM 列存储填充的 CPU 占用

IM 列存储的初始填充是 CPU 密集型操作，可能会影响同时运行的其他负载的性能。可以使用 Resource Manager[2] 控制 IM 列存储填充操作的 CPU 使用，并根据需要更改操作的优先级。

为此，可启用一个现成的资源计划（如 default_plan）或创建自己的资源计划来启用 CPU Resource Manager。默认情况下，内存中填充在 ora$autotask 用户组中运行，但按需填充除外，它在触发填充的用户组中运行。如果资源计划中不存在 ora$autotask 用户组，填充将在 OTHER_GROUPS 中运行。ora$autotask 中的其他操作包括收集统计信息和段分析等自动维护操作。

可使用 SET_CONSUMER_GROUP_MAPPING 过程更改内存中填充的用户组。

```
BEGIN
    dbms_resource_manager.Set_consumer_group_mapping(
      attribute      => 'ORACLE_FUNCTION',
      value          => 'INMEMORY',
      consumer_group => 'BATCH_GROUP');
END;
```

图 30. 更改 INMEMORY 操作的 Resource Manager 用户组

总结

Oracle Database In-Memory 透明地将分析查询的速度提高了若干数量级，从而有助于实时业务决策。它极大加速了数据仓库和混合负载 OLTP 环境。独特的“双格式”方法自动以现有的 Oracle 行格式（适用于 OLTP 操作）和新的纯内存中列格式（已针对分析处理进行了优化）维护数据。这两种格式同时处于活动状态且保持事务一致性。将列存储嵌入到 Oracle 数据库可确保它与所有现有特性完全兼容，无需在应用程序层进行任何更改。这意味着，当天即可开始充分利用列存储，且与应用程序无关。

[2] 有关使用 Oracle Database Resource Manager 的更多信息，请参阅白皮书使用 Oracle Resource Manager

甲骨文（中国）软件系统有限公司

北京远洋光华中心办公室

地址：北京市朝阳区景华南街5号远洋光华中心C座21层
邮编：100020
电话：(86.10) 6535-6688
传真：(86.10) 6515-1015

北京汉威办公室

地址：北京市朝阳区光华路7号汉威大厦10层1003-1005单元
邮编：100004
电话：(86.10) 6535-6688
传真：(86.10) 6561-3235

北京甲骨文大厦

地址：北京市海淀区中关村软件园24号楼甲骨文大厦
邮编：100193
电话：(86.10) 6106-6000
传真：(86.10) 6106-5000

北京国际软件大厦办公室

地址：北京市海淀区中关村软件园9号楼国际软件大厦二区308单元
邮编：100193
电话：(86.10) 8279-8400
传真：(86.10) 8279-8686

北京孵化器办公室

地址：北京市海淀区中关村软件园孵化器2号楼A座一层
邮编：100193
电话：(86.10) 8278-6000
传真：(86.10) 8282-6401

上海名人商业大厦办公室

地址：上海市黄浦区天津路155号名人商业大厦12层
邮编：200001
电话：(86.21) 2302-3000
传真：(86.21) 6340-6055

上海腾飞浦汇大厦办公室

地址：上海市黄浦区福州路318号腾飞浦汇大厦508-509室
邮编：200001
电话：(86.21) 2302-3000
传真：(86.21) 6391-2366

上海创智天地10号楼办公室

地址：上海市杨浦区淞沪路290号创智天地10号楼512-516单元
邮编：200433
电话：(86.21) 6095-2500
传真：(86.21) 6107-5108

上海创智天地11号楼办公室

地址：上海市杨浦区淞沪路303号创智天地科教广场3期11号楼7楼
邮编：200433
电话：(86.21) 6072-6200
传真：(86.21) 6082-1960

上海新思大厦办公室

地址：上海市漕河泾开发区宜山路926号新思大厦11层
邮编：200233
电话：(86.21) 6057-9100
传真：(86.21) 6083-5350

广州国际金融广场办公室

地址：广州市天河区珠江新城华夏路8号合景国际金融广场18楼
邮编：510623
电话：(86.20) 8513-2000
传真：(86.20) 8513-2380

成都中海国际中心办公室

地址：成都市高新区交子大道177号中海国际中心7楼B座02-06单元
邮编：610041
电话：(86.28) 8530-8600
传真：(86.28) 8530-8699

深圳飞亚达科技大厦办公室

地址：深圳市南山区高新南一道飞亚达科技大厦16层
邮编：518057
电话：(86.755) 8396-5000
传真：(86.591) 8601-3837

深圳德赛科技大厦办公室

地址：深圳市南山区高新南一道德赛科技大厦8层0801-0803单元
邮编：518057
电话：(86.755) 8660-7100
传真：(86.755) 2167-1299

大连办公室

地址：大连软件园东路23号大连软件园15号楼502
邮编：116023
电话：(86.411) 8465-6000
传真：(86.755) 8465-6499

苏州办公室

地址：苏州工业园区星湖街328号苏州国际科技园5期11幢1001室
邮编：215123
电话：(86.512) 8666-5000
传真：(86.512) 8187-7838

沈阳办公室

地址：沈阳市和平区青年大街390号皇朝万鑫国际大厦A座39层3901&3911室
邮编：110003
电话：(86.24) 8393-8700
传真：(86.24) 2353-0585

济南办公室

地址：济南市泺源大街150号中信广场11层1113单元
邮编：250011
电话：(86.531) 6861-1900
传真：(86.531) 8518-1133

南京办公室

地址：南京市玄武区洪武北路55号置地广场19层1911室
邮编：210018
电话：(86.25) 8579-7500
传真：(86.25) 8476-5226

西安办公室

地址：西安市高新区科技二路72号西安软件园零壹广场主楼1401室
邮编：710075
电话：(86.29) 8834-3400
传真：(86.25) 8833-9829

重庆办公室

地址：重庆市渝中区邹容路68号大都会商厦1611室
邮编：400010
电话：(86.23) 6037-5600
传真：(86.23) 6370-8700

杭州办公室

地址：杭州市西湖区杭大路15号嘉华国际商务中心810&811室
邮编：310007
电话：(86.571) 8168-3600
传真：(86.571) 8717-5299

福州办公室

地址：福州市五四路158号环球广场1601室
邮编：350003
电话：(86.591) 8621-5050
传真：(86.591) 8801-0330

南昌办公室

地址：江西省南昌市西湖区沿江中大道258号
皇冠商务广场10楼1009室
邮编：330025
电话：(86.791) 8612-1000
传真：(86.791) 8657-7693

呼和浩特办公室

地址：内蒙古自治区呼和浩特市新城区迎宾北路7号
大唐金座19层北侧1902-1904室
邮编：010051
电话：(86.471) 3941-600
传真：(86.471) 5100-535

郑州办公室

地址：河南省郑州市中原区中原中路220号
裕达国际贸易中心A座2015室
邮编：450007
电话：(86.371) 6755-9500
传真：(86.371) 6797-2085

武汉办公室

地址：武汉市江岸区中山大道1628号
武汉天地企业中心5号大厦23层2301单元
邮编：430010
电话：(86.27) 8221-2168
传真：(86.27) 8221-2168

长沙办公室

地址：长沙市芙蓉区韶山北路159号通程国际大酒店1311-1313室
邮编：410011
电话：(86.731) 8977-4100
传真：(86.731) 8425-9601

石家庄办公室

地址：石家庄市中山东路303号石家庄世贸广场酒店14层1402室
邮编：050011
电话：(86.311) 6670-8080
传真：(86.311) 8667-0618

昆明办公室

地址：昆明市三市街六号柏联广场写字楼11层1103A室
邮编：650021
电话：(86.871) 6402-4600
传真：(86.871) 6361-4946

合肥办公室

地址：安徽省合肥市蜀山区政务新区怀宁路1639号平安大厦18层1801室
邮编：230022
电话：(86.551) 6595-8200
传真：(86.551) 6371-3182

广西办公室

地址：广西省南宁市青秀区民族大道136-2号华润大厦B座2302室
邮编：530028
电话：(86.771) 391-8400
传真：(86.771) 577-5500

Oracle Database In-Memory 选件

2014 年 8 月

作者：Maria Colgan

参与编著：Jesse Kamp, Sue Lee

公司网址：http://www.oracle.com（英文）
中文网址：http://www.oracle.com/cn（简体中文）
销售中心：800-810-0161
售后服务热线：800-810-0366
培训服务热线：800-810-9931

欢迎访问：
http://www.oracle.com（英文）
http://www.oracle.com/cn（简体中文）