Tibero

IMCS 안내서

Tibero 7

대한민국 경기도 성남시 분당구 황새울로258번길 29, BS 타워 9층 우)13595

Website

http://www.tmaxtibero.com

기술서비스센터

Tel : +82-1544-8629

E-Mail : info@tmax.co.kr

Open Source Software Notice

Some modules or files of this product are subject to the terms of the following licenses. : OpenSSL, RSA Data Security, Inc., Apache Foundation, Jean-loup Gailly and Mark Adler, Paul Hsieh's hash

Detailed Information related to the license can be found in the following directory : ${INSTALL_PATH}/license/oss_licenses

본 제품의 일부 파일 또는 모듈은 다음의 라이선스를 준수합니다. : OpenSSL, RSA Data Security, Inc., Apache Foundation, Jean-loup Gailly and Mark Adler, Paul Hsieh's hash

관련 상세한 정보는 제품의 다음의 디렉터리에 기재된 사항을 참고해 주십시오. : ${INSTALL_PATH}/license/oss_licenses

안내서 정보

안내서 제목: Tibero IMCS 안내서

발행일: 2024-08-22

소프트웨어 버전: Tibero 7.2.2

안내서 버전: v7.2.2

내용 목차

그림 목차

안내서에 대하여

안내서의 대상

본 안내서는 Tibero®(이하 Tibero)에서 제공하는 In-Memory Column Store(이하 IMCS) 기능을 참고하려는 데이터베이스 관리자(Database Administrator, 이하 DBA), 애플리케이션 프로그램 개발자를 대상으로 기술한다.

안내서의 전제 조건

본 안내서를 원활히 이해하기 위해서는 다음과 같은 사항을 미리 알고 있어야 한다.

- 데이터베이스의 이해
- RDBMS의 이해
- SQL의 이해

안내서의 제한 조건

본 안내서는 Tibero를 실무에 적용하거나 운용하는 데 필요한 모든 사항을 포함하고 있지 않다. 따라서 설치, 환경설정 등 운용 및 관리에 대해서는 각 제품 안내서를 참고하기 바란다.

참고

Tibero의 설치 및 환경설정에 관한 내용은 "Tibero 설치 안내서"를 참고한다.

안내서 구성

Tibero IMCS 안내서는 총 4개의 장으로 구성되어 있다.

각 장의 주요 내용은 다음과 같다.

- 제1장: 소개

  In-Memory Column Store의 개념과 기능을 개략적으로 설명한다.

- 제2장: In-Memory Column Store 설정

  In-Memory Column Store 기능을 활성화 및 비활성하고, 오브젝트별 In-Memory Column Store 기능을 설정하는 방법에 대해 설명한다.

- 제3장: In-Memory Column Store의 고가용성

  In-Memory Column Store의 고가용성에 대하여 설명한다.

- 제4장: In-Memory DataBase Reference

  In-Memory Column Store 관련 Initialization Parameter 와 Dynamic Performance View 들에 대해 설명한다.

안내서 규약

표기	의미
<<AaBbCc123>>	프로그램 소스 코드의 파일명
<Ctrl>+C	Ctrl과 C를 동시에 누름
[Button]	GUI의 버튼 또는 메뉴 이름
진하게	강조
" "(따옴표)	다른 관련 안내서 또는 안내서 내의 다른 장 및 절 언급
'입력항목'	화면 UI에서 입력 항목에 대한 설명
하이퍼링크	메일 계정, 웹 사이트
>	메뉴의 진행 순서
+----	하위 디렉터리 또는 파일 있음
|----	하위 디렉터리 또는 파일 없음
참고	참고 또는 주의사항
주의	주의할 사항
[그림 1.1]	그림 이름
[예 1.1]	예제 이름
`AaBbCc123`	Java 코드, XML 문서
[*command argument*]	옵션 파라미터
< xyz >	'<'와 '>' 사이의 내용이 실제 값으로 변경됨
|	선택 사항. 예) A|B: A나 B 중 하나
…	파라미터 등이 반복되어서 나옴
${ }	환경변수

시스템 사용 환경

	요구 사항
Platform	HP-UX 11i v3(11.31)
	Solaris (Solaris 11)
	AIX (AIX 7.1/AIX 7.2/AIX 7.3)
	GNU (X86, 64, IA64)
	Red Hat Enterprise Linux 7 kernel 3.10.0 이상
	Windows(x86) 64bit
Hardware	최소 2.5GB 하드디스크 공간
	1GB 이상 메모리 공간
Compiler	PSM (C99 지원 필요)
	tbESQL/C (C99 지원 필요)

관련 안내서

안내서	설명
Tibero 설치 안내서	설치 과정에 필요한 시스템 요구사항과 설치 및 제거 방법을 기술한 안내서이다.
Tibero tbCLI 안내서	Call Level Interface인 tbCLI의 개념과 구성요소, 프로그램 구조를 소개하고 tbCLI 프로그램을 작성하는 데 필요한 데이터 타입, 함수, 에러 메시지를 기술한 안내서이다.
Tibero 애플리케이션 개발자 안내서	각종 애플리케이션 라이브러리를 이용하여 애플리케이션 프로그램을 개발하는 방법을 기술한 안내서이다.
Tibero External Procedure 안내서	External Procedure를 소개하고 이를 생성하고 사용하는 방법을 기술한 안내서이다.
Tibero JDBC 개발자 안내서	Tibero에서 제공하는 JDBC 기능을 이용하여 애플리케이션 프로그램을 개발하는 방법을 기술한 안내서이다.
Tibero tbESQL/C 안내서	C 프로그래밍 언어를 사용해 데이터베이스 작업을 수행하는 각종 애플리케이션 프로그램을 작성하는 방법을 기술한 안내서이다.
Tibero tbESQL/COBOL 안내서	COBOL 프로그래밍 언어를 사용해 데이터베이스 작업을 수행하는 각종 애플리케이션 프로그램을 작성하는 방법을 기술한 안내서이다.
Tibero tbPSM 참조 안내서	저장 프러시저 모듈인 tbPSM의 패키지를 소개하고, 이러한 패키지에 포함된 각 프러시저와 함수의 프로토타입, 파라미터, 예제 등을 기술한 참조 안내서이다.
Tibero 관리자 안내서	Tibero의 동작과 주요 기능의 원활한 수행을 보장하기 위해 DBA가 알아야 할 관리 방법을 논리적 또는 물리적 측면에서 설명하고, 관리를 지원하는 각종 도구를 기술한 안내서이다.
Tibero 유틸리티 안내서	데이터베이스와 관련된 작업을 수행하기 위해 필요한 유틸리티의 설치 및 환경설정, 사용 방법을 기술한 안내서이다.
Tibero TAS 안내서	Tibero Active Storage(TAS)를 사용해서 Tibero 의 파일을 관리하고자 하는 관리자를 대상으로 기술한 안내서이다.
Tibero 에러 참조 안내서	Tibero를 사용하는 도중에 발생할 수 있는 각종 에러의 원인과 해결 방법을 기술한 안내서이다.

안내서	설명
Tibero 참조 안내서	Tibero의 동작과 사용에 필요한 초기화 파라미터와 데이터 사전, 정적 뷰, 동적 뷰를 기술한 참조 안내서이다.
Tibero SQL 참조 안내서	데이터베이스 작업을 수행하거나 애플리케이션 프로그램을 작성할 때 필요한 SQL 문장을 기술한 참조 안내서이다.
Tibero Spatial 참조 안내서	Tibero에서 Geometry 타입에 대한 설명과 Spatial 기능 관련 프러시저 함수 목록 및 사용 방법 등을 기술한 안내서이다.
Tibero TEXT 참조 안내서	Tibero의 제공하는 Text Index를 소개하고, Text Index를 생성 하고 사용하는 방법을 기술하는 안내서이다.
Tibero TDP.NET 안내서	Tibero Data Provider for .NET 기능을 기술하는 안내서이다.

제1장 소개

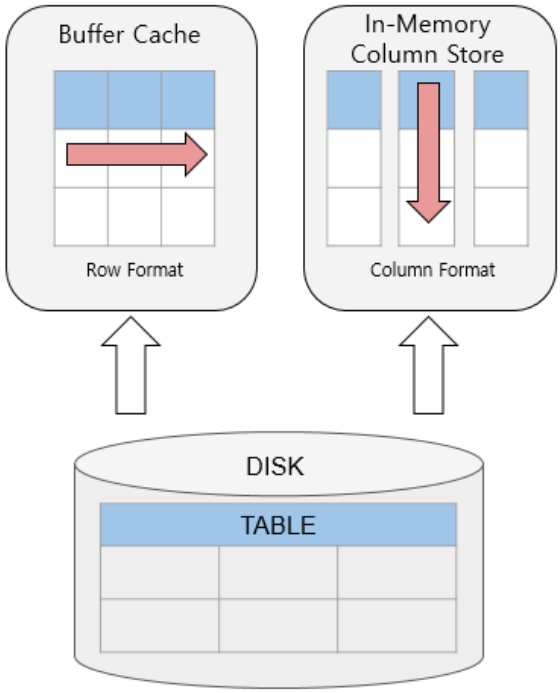
본 장에서는 In-Memory Column Store의 개념과 기능을 개략적으로 설명한다.

1.1. 개요

In-Memory Column Store (IMCS)는 컬럼에 대한 스캔을 최적화하기 위한 메모리 저장구조로 데이터의 복제본을 컬럼 포맷으로 저장하여 관리한다.

기존의 로우 포맷 방식에 컬럼 포맷 방식을 추가로 지원함으로써 OLTP 및 OLAP 업무가 혼재된 환경에서 데이터베이스의 성능을 향상시킬 수 있는 기능이다.

[그림 1.1] 로우 포맷 방식과 컬럼 포맷 방식



In-Memory Column Store의 특징은 다음과 같다.

- In-Memory Area

  In-Memory Column Store는 공유 메모리의 In-Memory Area라는 곳에 존재한다. In-Memory Area는 부팅 시점에 고정된 크기로 할당되며 한번 할당되면 DB 재기동 전까지 크기가 변경되지 않는다.

  Cache Out이 발생하는 버퍼 캐시와는 달리 In-Memory Column Store에는 데이터가 한번 적재되면 유저가 따로 명령을 수행하지 않는 한 데이터가 In-Memory Area에 계속 존재한다. 이 때문에 In-Memory Column Store에 저장된 데이터를 읽을 때는 별도의 I/O 비용이 발생하지 않는다(단, In-Memory 데이터

가 유효하지 않아 버퍼 캐시를 읽어야하는 경우에는 In-Memory 스캔을 수행하더라도 I/O가 발생할 수 있다).

- Columnar Format

  In-Memory Column Store는 데이터의 복제본을 스캔에 최적화된 컬럼 포맷으로 변환하여 저장한다. 컬럼 포맷의 데이터는 내부적으로 고정된 사이즈로 저장되며 이를 통해 스캔 시 필터 연산(e.g. <, >, =)을 효율적으로 처리할 수 있다.

- Compression

  In-Memory Column Store에서 각 컬럼 데이터는 중복 데이터를 제거하는 방법으로 데이터를 압축하여 저장한다. 이를 통해 메모리 효율을 높이고 스캔 시 조회하는 데이터 갯수를 줄임으로써 효율적인 스캔을 지원한다.

- Data Pruning

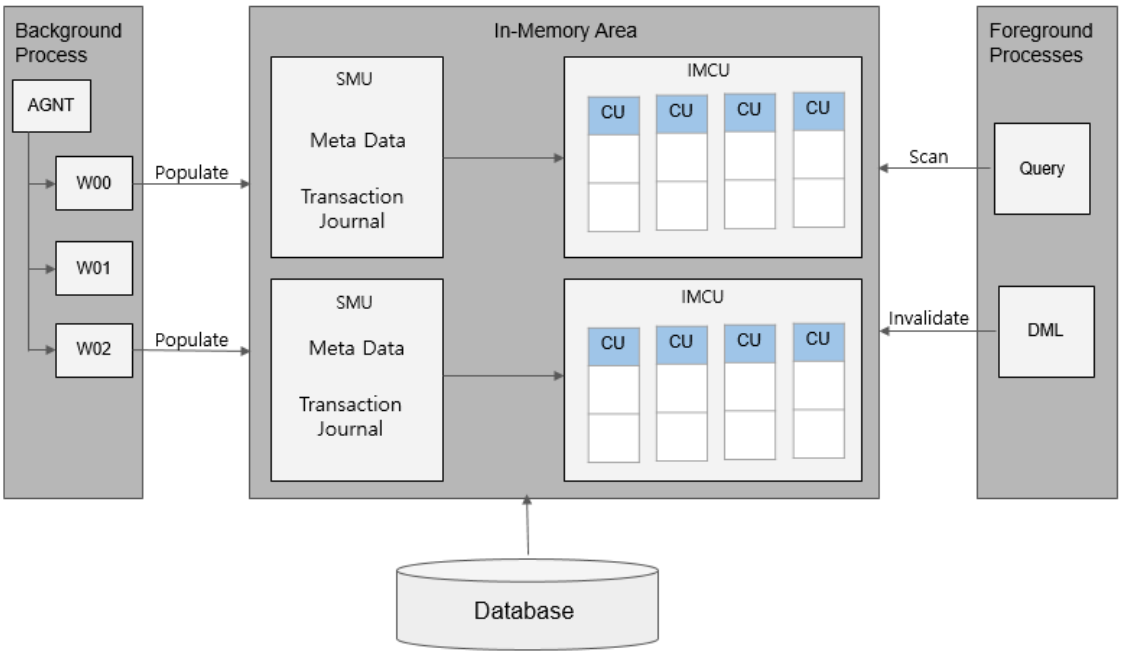
  In-Memory Column Store에서 각 컬럼 데이터는 In-Memory Compression Unit (IMCU)라는 저장 유닛으로 관리되며 IMCU에는 각 컬럼 데이터의 최소 및 최대값 정보가 포함되어 있다. 스캔 시 각 IMCU에 포함되어 있는 데이터의 최소 및 최대값을 확인하여 필요한 IMCU만 조회함으로써 스캔 효율을 높일 수 있다.

1.2. In-Memory Column Store 저장 구조

In-Memory Area는 부팅 시점에 공유 메모리(Shared Memory) 내 고정 (Fixed) 영역에 할당된다. In-Memory Area의 크기는 초기화 파라미터 INMEMORY_SIZE로 설정해준다.

In-Memory Area는 컬럼 데이터를 저장하는 **In-Memory Compression Unit (IMCU)**과 컬럼 데이터의 메타를 저장하는 **Snapshot Metadata Unit (SMU)**로 구성되어 있다.

[그림 1.2] In-Memory Column Store 구조



1.2.1. In-Memory Compression Unit(IMCU)

IMCU (In-Memory Compression Unit)는 In-Memory Column Store (IMCS)의 저장 유닛으로 하나 이상의 컬럼의 데이터를 압축된 형태로 저장하고 있다. IMCU의 크기는 1 MB이며, IMCS는 하나의 오브젝트(테이블, 파티션, 서브파티션)의 데이터를 컬럼 형태로 변환하여 여러 개의 IMCU에 저장한다.

IMCU에는 오직 하나의 오브젝트의 컬럼 데이터만 저장된다. IMCU는 하나 이상의 **Column Compression Unit(CU)**를 포함하고 있다.

- **Column Compression Unit(CU)**

  CU(Column Compression Unit)는 IMCU 내에서 하나의 컬럼 데이터를 저장하는 영역이다. CU 내 저장되는 각 로우의 컬럼 값은 2 byte 값으로 인코딩되어 CU 내 저장된다. 인코딩된 값을 dictionary code라고 하며, CU 안에는 각 dictionary code와 실제 컬럼 데이터 간의 매핑 테이블이 존재한다.

1.2.2. Snapshot Metadata Unit (SMU)

SMU (Snapshot Metadata Unit)은 IMCU에 대한 메타 데이터를 저장하는 영역이다. In-Memory Column Store 공간에서는 각 IMCU 마다 하나의 SMU가 매핑되어있다. SMU에는 오브젝트 정보 및 IMCU에 적재된 블럭의 주소 정보가 들어있으며, 트랜잭션 저널 (transaction journal)이라는 DML이 발생한 로우들에 대한 정보가 기록되어 있다.

1.3. In-Memory Process 구조

In-Memory Column Store 공간에 데이터를 컬럼 포맷으로 적재하는 작업(populate)은 백그라운드 프로세스에서 수행한다. 에이전트 프로세스는 부팅 시점에 In-Memory priority가 지정된 오브젝트에 대해서 populate를 수행하며, priority가 지정되지 않은 오브젝트의 경우 쿼리를 통해 최초 접근 시 에이전트 프로세스에서 해당 오브젝트에 대한 populate 작업을 수행한다.

In-Memory Column Store 공간에 데이터가 적재된 오브젝트에 대해 쿼리를 수행하여 컬럼 포맷의 데이터를 읽거나 혹은 DML을 수행하여 SMU에 트랜잭션 저널을 기록하는 작업은 워커 프로세스에서 수행한다.

1.4. SIMD Vector Processing

SIMD(Single Instruction, Multiple Data) 벡터 연산을 통해서 WHERE 절의 expression 계산 성능을 향상시킨다. 예를 들어 CU를 구성하고 있는 2bytes의 고정된 byte의 데이터를 CPU에 8개씩 로드를 하고 한번의 비교 연산으로 8개의 결과를 출력한다.

SIMD를 수행하기 위한 조건은 In Memory Column과 바인드 파라미터 또는 상수 값의 비교 연산하는 경우 이다. 가능한 비교 연산자는 >, >=, =, <, <=, AND 이다.

제2장 In-Memory Column Store 설정

본 장에서는 In-Memory Column Store 기능을 활성화 및 비활성하고, 오브젝트별 In-Memory Column Store 기능을 설정하는 방법에 대해 설명한다.

2.1. In-Memory Column Store 기능 활성화 및 비활성화

In-Memory Column Store 기능을 활성화 및 비활성화 하기 위해서는 INMEMORY_SIZE라는 파라미터의 값을 변경해주어야 한다.

INMEMORY_SIZE 파라미터

INMEMORY_SIZE 파라미터는 기본값이 0이며, INMEMORY_SIZE 파라미터 값이 0일 때는 In-Memory Column Store 기능이 비활성화된 상태이다. In-Memory Column Store 기능을 활성화시키기 위해서는 INMEMORY_SIZE 파라미터 값을 100M 이상의 값으로 설정 후 데이터베이스를 기동해주어야 한다.

INMEMORY_SIZE 파라미터는 동적으로 변경할 수 없으며, 변경을 위해서는 데이터베이스 재기동이 필요하다.

INMEMORY_SIZE 파라미터의 최댓값은 (TOTAL_SHM_SIZE - _MIN_SHARED_POOL_SIZE)이다.

또한 INMEMORY_SIZE 파라미터 설정을 위한 권장 설정은 다음과 같다.

```
INMEMORY_SIZE 파라미터를 Y로 설정한다면 다음과 같은 다음과 같은 파라미터 설정이 권장된다.
1) TOTAL_SHM_SIZE
 - 기존에 사용하던 값에 Y를 더한 값으로 설정
2) DB_CACHE_SIZE
 - single 환경인 경우: (TOTAL_SHM_SIZE - Y) * (2/3)
 - TAC 환경인 경우   : (TOTAL_SHM_SIZE - Y) / 2

예를 들어, 기존에 TOTAL_SHM_SIZE를 100GB로 설정하여 DB를 구성하고 있었을 때
INMEMORY_SIZE로 30GB를 추가로 설정하려고 한다면,
TOTAL_SHM_SIZE를 기존 값에서 30GB를 더한 130GB으로 설정하는 것을 권장하며
DB_CACHE_SIZE는 single 환경인 경우 (130 - 30) * (2/3) = 66GB,
TAC 환경인 경우 (130 - 30) / 2 = 50GB로 설정하는 것을 권장한다.
```

1. INMEMORY_SIZE 파라미터 설정 변경

   설정파일(tip 파일)에 INMEMORY_SIZE 를 변경한다.

   ```
   INMEMORY_SIZE = 500M
   ```

2. 데이터베이스 기동 종료

3. 데이터베이스 재기동

2.1.1. In-Memory Column Store 기능 활성화

In-Memory Column Store 기능 활성화를 위해서는 데이터베이스 재기동이 필요하다.

1. INMEMORY_SIZE 파라미터 설정

   데이터베이스 기동 전 설정파일(tip 파일)에 INMEMORY_SIZE 를 100M 이상의 값으로 설정해준다.

   ```
   INMEMORY_SIZE = 100M
   ```

2. 데이터베이스 기동 중이라면 데이터베이스 기동 종료

3. 데이터베이스 기동

2.1.2. In-Memory Column Store 기능 비활성화

In-Memory Column Store 기능 비활성화를 위해서는 데이터베이스 재기동이 필요하다.

1. INMEMORY_SIZE 파라미터 설정

   데이터베이스 기동 전 설정파일(tip 파일)에 INMEMORY_SIZE를 0으로 설정하거나 INMEMORY_SIZE 파라미터를 삭제한다.

   ```
   INMEMORY_SIZE = 0
   ```

   ```
   #INMEMORY_SIZE = 500M
   ```

2. 데이터베이스 기동 종료

3. 데이터베이스 재기동

2.2. 오브젝트별 In-Memory Column Store 기능 활성화

본 절에서는 오브젝트별 In-Memory Column Store 기능 활성화 및 비활성화 방법과 priority, compression 옵션에 대해 설명한다.

2.2.1. In-Memory Population

In-Memory Population(이하 population)이란 데이터베이스에서 디스크로부터 row-format 데이터를 읽어 columnar-format으로 변경 후 In-Memory Column Store에 적재하는 것이다. 테이블, 파티션, 서브파티션만 population이 가능하다.

2.2.1.1. In-Memory Population 동작

옵션별 priority 옵션 설정에 따라 데이터베이스 기동 또는 In-Memory 오브젝트에 접근하는 경우 population이 가능하다.

In-Memory Population 우선순위

INMEMORY PRIORITY 절이 포함된 DDL 구문으로 population의 우선순위를 설정할 수 있다.

우선순위의 설정은 table, partition, subpartition에 대해 가능하며 column별로 우선순위 설정은 불가능하다. 오브젝트에 inmemory 옵션을 설정하는 것은 population의 대상이 되는 것이지, 그 즉시 population 되는 것은 아니라는 점에 유념한다.

참고

Segment의 크기가 64KB 이하라면 해당 segment는 population 되지 않는다. 따라서 inmemory 옵션이 설정되어 있더라도 population이 되지 않는 오브젝트가 존재할 수 있다.

우선순위에 따른 population 동작은 다음과 같다.

동작	설명
On-demand population	INMEMORY PRIORITY 옵션의 기본값은 NONE이다. 이 경우 해당 오브젝트에 대해 inmemory scan을 통해 접근하였을 경우만 population이 된다. 오브젝트에 대해 접근이 없었거나, index scan, table full scan을 통해 접근이 되었을 경우 population이 되지 않는다.
Priority-based population	INMEMORY PRIORITY 옵션의 값이 NONE 이외의 값으로 설정되었을 경우, 오브젝트에 대한 접근이 없어도 population이 된다. Priority level에 따라서 level이 높은 순으로 population시키며 priority level이 같을 경우 population의 순서는 보장할 수 없다. In-Memory Column Store의 공간이 부족할 경우 더 이상 population시키지 않는다.

PRIORITY Options

Prioirty 옵션별 동작은 다음과 같다.

옵션	설명
PRIORITY NONE	오브젝트 대해 접근시에만 population을 한다.
PRIORITY LOW	오브젝트에 대해 접근과는 무관하게 population을 한다.

옵션	설명
	다음 priority level의 population이 모두 완료된 후 population이 진행된다. – MEDIUM – HIGH – CRITICAL
PRIORITY MEDIUM	오브젝트에 대해 접근과는 무관하게 population을 한다. 다음 priority level의 population이 모두 완료된 후 population이 진행된다. – HIGH – CRITICAL
PRIORITY HIGH	오브젝트에 대해 접근과는 무관하게 population을 한다. 다음 priority level의 population이 모두 완료된 후 population이 진행된다. – CRITICAL
PRIORITY CRITICAL	오브젝트에 대해 접근과는 무관하게 population을 한다. 최우선으로 population이 진행된다.

PRIORITY Option 설정 예시

- CREATE TABLE 구문

```
CREATE TABLE inmemory_test_priority (
     id           NUMBER(5) PRIMARY KEY,
     test_column  VARCHAR2(15))
  INMEMORY PRIORITY HIGH;
```

- ALTER TABLE 구문

```
ALTER TABLE inmemory_test_priority INMEMORY PRIORITY HIGH;
```

2.2.1.2. In-Memory Population 제어

DDL 구문의 INMEMORY 절을 이용하여 테이블 스페이스, 테이블, 파티션, 서브파티션에 INMEMORY 옵션을 설정할 수 있다.

INMEMORY 절

INMEMORY 절은 기본적으로 segment 단위로만 설정할 수 있으며, column 단위로 지정할 경우 적용할 수 있는 옵션이 제약이 된다. Column 단위 INMEMORY 절은 추후 설명한다.

INMEMORY 옵션을 설정하기 위해서는 아래의 구문에서 INMEMORY 절을 명시해준다.

- CREATE TABLESPACE or ALTER TABLESPACE

  테이블 스페이스에 INMEMORY 옵션을 사용하면, 해당 테이블 스페이스에서 만들어지는 테이블들은 테이블 스페이스의 INMEMORY 옵션을 따른다. 테이블에 INMEMORY 옵션을 명시할 경우 테이블 스페이스의 INMEMORY 옵션을 override 한다. 테이블 스페이스에 대한 INMEMORY 옵션은 만들어질 테이블의 default INMEMORY 옵션에만 영향을 준다. 따라서 ALTER 테이블 스페이스를 통해 INMEMORY 옵션을 사용하여도 이미 만들어진 테이블들은 INMEMORY 옵션이 설정되지 않으며, ALTER TABLESPACE를 통해 NO INMEMORY로 변경하여도 이미 INMEMORY 옵션이 설정된 테이블들은 NO INMEMORY로 변경되지 않는 것에 유의한다.

- CREATE TABLE or ALTER TABLE

  테이블에 INMEMORY 옵션을 사용하면, 기본값으로 nonvirtual column들은 population 대상이 된다. Column 단위 INMEMORY 절로 특정 column 만 population 대상이 되게 설정할 수 있다. Partitioned 테이블의 경우 partition별로 INMEMORY 옵션을 지정할 수 있다. 명시하지 않은 INMEMORY 옵션에 대해서는 테이블의 옵션을 상속받으며, 명시한 INMEMORY 옵션은 테이블의 옵션을 override한다.

2.2.1.3. 테이블의 In-Memory Column Store 설정 예시

- CREATE TABLE 구문

```
CREATE TABLE inmemory_test (
     id           NUMBER(5) PRIMARY KEY,
     test_column  VARCHAR2(15))
  INMEMORY;
```

- ALTER TABLE 구문

```
ALTER TABLE inmemory_test INMEMORY;
```

2.2.1.4. Column의 In-Memory Column Store 설정 예시

Column 단위 INMEMORY 절로 특정 column이 inmemory population 대상이 되지 않도록 설정할 수 있다.

- CREATE TABLE 구문

```
CREATE TABLE inmemory_test (
     id           NUMBER(5) PRIMARY KEY,
     test_column  VARCHAR2(15),
     no_inmemory_column VARCHAR2(20))
```

```
  INMEMORY
  NO INMEMORY (no_inmemory_column);
```

- ALTER TABLE 구문

```
ALTER TABLE inmemory_test NO INMEMORY (no_inmemory_column);
```

2.2.1.5. Tablespace의 In-Memory Column Store 설정 예시

테이블 스페이스의 INMEMORY 절은 DEFAULT 절이 선행되어야 한다.

- CREATE TABLESPACE 구문

```
CREATE TABLESPACE inmemory_ts
   DATAFILE 'imts01.dbf' SIZE 40M
   ONLINE
   DEFAULT INMEMORY;
```

- ALTER TABLESPACE 구문

```
ALTER TABLESPACE inmemory_ts DEFAULT INMEMORY;
```

제3장 In-Memory Column Store의 고가용성

본 장에서는 In-Memory Column Store의 고가용성에 대하여 설명한다.

3.1. TAC 환경에서의 In-Memory Column Store 기능

Tibero Active Cluster(이하 TAC) 환경에서 IMCS 기능을 사용할 경우, 실행 중인 모든 인스턴스는 자신만의 In-Memory(IM) Column Store 공간을 가진다. 기본적으로 TAC 환경에서 오브젝트에 대한 populate를 수행할 경우 해당 오브젝트의 데이터는 각 노드에 분산되어 저장된다.

Tibero에서 모든 TAC 노드의 INMEMORY_SIZE 파라미터의 값은 동일해야한다.

3.1.1. In-Memory Distribution

In-Memory 오브젝트에 지정하는 DISTRIBUTE 옵션은 오브젝트의 데이터가 각 TAC 노드에 어떻게 분산되어 저장될지를 결정한다.

다음은 DISTRIBUTE 옵션에 대한 설명이다.

옵션	설명
AUTO	Tibero 내부 규칙에 따라 데이터를 각 노드별로 균등하게 분산시킨다. (기본값)
By Rowid Range	데이터를 rowid의 range로 나누어 각 노드 별로 분산시킨다. 현재 Tibero에서 미지원 상태이다.
By Partition	데이터를 파티션 별로 나누어 각 노드별로 분산시킨다. 현재 Tibero에서 미지원 상태이다.
By Subpartition	데이터를 서브파티션 별로 나누어 각 노드 별로 분산시킨다. 현재 Tibero에서 미지원 상태이다.

3.1.2. In-Memory Duplication

In-Memory 오브젝트에 지정하는 DUPLICATE 옵션은 각 TAC 노드가 데이터의 복제본을 저장 여부 및 방식을 결정한다.

다음은 DUPLICATE 옵션에 대한 설명이다.

옵션	설명
NO DUPLICATE	In-Memory 데이터의 복제본이 존재하지 않는다. (기본값)
DUPLICATE	각 In-Memory 데이터는 하나의 복제본이 존재한다. duplicate 옵션이 지정된 In-Memory 오브젝트에 대한 populate 수행 시 2개의 TAC 노드에 같은 데이터가 저장된다. 현재 Tibero에서 미지원 상태이다.
DUPLICATE ALL	각 In-Memory 데이터는 노드 갯수만큼의 복제본이 존재한다. 모든 인스턴스는 In-Memory 오브젝트의 복사본을 자신의 In-Memory Column Store 공간에 저장하고 있다.

3.1.3. TAC 환경에서의 Parallelism

기본적으로 TAC 환경에서 In-Memory 데이터는 각 TAC 노드 별로 분산되어 저장되므로 In-Memory 데이터에 대한 쿼리 수행은 병렬적으로 수행된다. 예를 들어, 3-node TAC 환경에서 populate가 완료된 테이블에 대한 In-Memory scan을 수행하면, 각 TAC 노드에서 저장하고 있는 해당 테이블의 데이터를 읽어서 전송해주게 된다. 즉, TAC 환경에서 In-Memory scan을 수행하면 노드 갯수만큼의 DOP(Degree Of Parallelism)를 가지고 병렬적으로 스캔을 수행한다.

제4장 In-Memory DataBase Reference

본 장에서는 In-Memory Column Store 관련 Initialization Parameter 와 Dynamic Performance View 들에 대해 설명한다.

4.1. In-Memory Initialization Parameters

4.1.1. INMEMORY_SIZE

INMEMORY_SIZE는 인스턴스에 할당할 In-Memory Column Store의 크기를 지정하는 파라미터이다. 기본값은 0이며, 이 경우 In-Memory Column Store를 위한 메모리 공간을 따로 할당하지 않는다.

INMEMORY_SIZE로 지정가능한 최소값은 100MB이며 DB 운영 중에 파라미터 값을 동적으로 변경할 수 없다. TAC를 사용하는 경우, 모든 TAC 인스턴스가 동일한 INMEMORY_SIZE 파라미터 값을 가져야하는 제약이 있다.

4.2. In-Memory Dynamic Performance Views

4.2.1. V$IM_COLUMN_LEVEL

Column별 In-Memory Compression 옵션과 No inmemory 여부를 확인 할 수 있는 View이다. TAC 환경에서도 node 별로 다른 정보가 없기에 GV$IM_COLUMN_LEVEL은 없다.

항목	설명
OWNER	테이블의 Owner 이름이다.
OBJ_NUM	테이블의 오브젝트 ID이다.
TABLE_NAME	테이블 이름이다.
SEGMENT_COLUMN_ID	Column의 Segment Column Number이다.
COLUMN_NAME	Column 이름이다.
INMEMORY_COMPRESSION	Column에 적용된 In-Memory Compression 옵션이다. No inmemory 여부를 포함한다.
CON_ID	데이터가 속해 있는 container의 ID이다.

4.2.2. V$IM_SEGMENTS

In-Memory Area에 존재하는 세그먼트의 정보를 조회할 수 있는 View이다. V$IM_SEGMENTS 뷰를 통해 populate된 세그먼트에 대한 정보를 확인할 수 있다.

항목	설명
OWNER	Segment의 Owner 이름이다.
SEGMENT_NAME	Segment의 이름이다.
PARTITION_NAME	해당 Segment가 속하는 Partition 이름이다. Non-Partitioned라면 NULL이다.
SEGMENT_TYPE	Segment의 타입이다. – Table – Table Partition – Table Subpartition
TABLESPACE_NAME	Segment가 속한 테이블 스페이스의 이름이다.
INMEMORY_SIZE	Segment가 In-Memory Area에서 차지하고 있는 공간의 크기이다. (단위: Byte)
BYTES	Segment의 총 크기이다. (단위: Byte)
BYTES_NOT_POPULATED	Segment의 데이터 중 In-Memory Area에 적재되지 않은 부분의 크기이다. (단위: Byte)
POPULATE_STATUS	Segment의 population의 상태이다. – STARTED – COMPLETED – FAILED
INMEMORY_PRIORITY	Segment의 population 우선순위이다. – LOW – MEDIUM – HIGH – CRITICAL – NONE
INMEMORY_DISTRIBUTE	Segment에 지정된 In-Memory Distribute 옵션이다.

항목	설명
	– AUTO – BY ROWID RANGE – BY PARTITION – BY SUBPARTITION
INMEMORY_DUPLICATE	Segment에 지정된 In-Memory Duplicate 옵션이다. – NO DUPLICATE – DUPLICATE – DUPLICATE ALL
INMEMORY_COMPRESSION	Segment에 지정된 In-Memory Compression 옵션이다. – NO MEMCOMPRESS – FOR QUERY LOW – FOR QUERY HIGH