설치

Tibero
물리설계 안내서

Copyright © 2024 TmaxTibero. All Rights Reserved

TINP016

Copyright Notice
Copyright © 2024 TIBERO Co., Ltd. All Rights Reserved.
대한민국 경기도 성남시 분당구 황새울로 258 번길 29, 티맥스수내타워 우)13595

Website
www.tmaxtibero.com

Restricted Rights Legend
All TIBERO Software (Tibero®) and documents are protected by copyright laws and international convention. TIBERO software and documents are made available under the terms of the TIBERO License Agreement and may only be used or copied in accordance with the terms of this agreement. No part of this document may be transmitted, copied, deployed, or reproduced in any form or by any means, electronic, mechanical, or optical, without the prior written consent of TIBERO Co., Ltd.
이 소프트웨어(Tibero®) 사용설명서의 내용과 프로그램은 저작권법과 국제 조약에 의해서 보호받고 있습니다. 사용설명서의 내용과 여기에 설명된 프로그램은 TIBERO Co., Ltd.와의 사용권 계약 하에서만 사용이 가능하며, 사용권 계약을 준수하는 경우에만 사용 또는 복제할 수 있습니다. 이 사용설명서의 전부 또는 일부분을 TIBERO 의 사전 서면 동의 없이 전자, 기계, 녹음 등의 수단을 사용하여 전송, 복제, 배포, 2 차적 저작물 작성 등의 행위를 하여서는 안 됩니다.

Trademarks
Tibero® is a registered trademark of TIBERO Co., Ltd. Other products, titles or services may be registered trademarks of their respective companies.
Tibero®는 TIBERO Co., Ltd.의 등록 상표입니다. 기타 모든 제품들과 회사 이름은 각각 해당 소유주의 상표로서 참조용으로만 사용됩니다.

안내서 정보
안내서 제목: Tibero 물리설계 안내서
발행일: 2024-02-14
소프트웨어 버전: Tibero7 FS02
안내서 버전: 1.0

제, 개정 이력

안내서 버전	개정일자	개정 사유 및 내용	비고
1.0	2024.02.14	최초 제정	작성자:한준석

목록

제 1장 물리 설계소개

1.1. 개요

본 안내서는 Tibero 를 설치 및 운영하기전 시스템을 보다 효율적으로 사용하기 위한 물리설계 방법을 제시한다. Redo Log Files, Undo 및 Temp 테이블 스페이스 그리고 메모리 영역의 크기 산정을 위해서는 Tibero의 특성을 먼저 이해해야 하며 이와 관련된 정보는 다음의 매뉴얼을 참고한다.

- Tibero 관리자 안내서의 "3. 파일과 데이터 관리", "4. 스키마 객체의 관리"

Table 및 Index 크기를 산정하기 위해 필요한 정보와 산정방식은 Tibero의 Data Block Architecture 특성을 고려하여야 한다.

본 문서는 다음과 같은 항목으로 구성되어 있다.

- 테이블 크기 산정
- 인덱스 크기 산정
- 데이터 타입별 크기 산정
- Redo 로그 파일 크기 산정
- Undo 테이블 스페이스 크기 산정
- Temp 테이블 스페이스 크기 산정
- Shared Memory 크기 산정
- SQL Execution Work Area 크기 산정

제 2장 테이블 크기 산정

2.1. 사전정보

Tibero의 테이블 크기를 산정하기 위해 필요한 정보와 방법에 대해 설명한다.

2.1.1. Row의 데이터길이

테이블 크기를 산정하기 위해서는 컬럼의 데이터 타입별 실제 소비 크기와 저장되는 길이를 산정해야 한다. 이 정보는 테이블 크기를 산정하기 위한 항목별 크기 산정방법에 사용된다. 데이터 타입별 크기는 "제 4장 데이터 타입별 크기 산정"을 참고한다.

다음은 컬럼 정의 및 계산방식의 예이다. 크기산정 테이블은 5개의 컬럼으로 구성되었다고 가정한다.

이름	데이터타입	데이터길이	비고
c1	CHAR(10)	10	
c2	DATE	8	
c3	NUMBER(7,2)	4	roundup(7/2)
c4	CHAR(500)	500	
c5	VARCHAR(3000)	2100	3,000 * 0,7
Row의 데이터 길이		2,622	c1 + c2+ c3 + c4 + c5

"제4장 데이터 타입별 크기 산정"의 데이터 타입별 사이즈 표를 참고하여 위 테이블을 구성하는 각 컬럼의 데이터 타입별 실제 소비사이즈를 산정하면 아래와 같다.

구분	설명
CHAR	문자열의 길이는 byte와 문자를 기준으로 지정한다. CHAR(10) = 10, CHAR(500) = 500
DATE	고정 8bytes 이다.
NUMBER	크기를 2로 나누어 반올림한 값이다. NUMBER(7,2) = roundup(7/2) = 4
VARCHAR	지정된 길이의 70%로 가정한다. VARCHAR(3000) = 3,000 * 0,7 = 2,100

2.1.2. 항목별 크기 산정

테이블 크기를 산정하기 위해 필요한 항목은 다음과 같다.
크기 예는 각 항목별로 기본으로 정해진 크기, 파라미터로 설정할수 있는 크기와 운영할 시스템에서 사전 조사된 수치를 나타낸다.

항목	크기 예	단위	설명
Row Directory	2	bytes	row 하나당 bytes
Row Header	3	bytes	row 하나당 bytes
250bytes 이하 컬럼개수	3	개수	위의 Row의 데이터 길이 테이블에 c1, c2, c3가 대상
250bytes 초과 컬럼개수	2	개수	위의 Row의 데이터 길이 테이블에 c4, c5가 대상
Block Size	8192	개수	기본값: 8192 (파라미터)
Block Header	48	bytes	고정 (Fixed Header)
INITRANS	2	개수	기본값: 2 (파라미터)
PCTFREE	10	%	기본값: 10% (파라미터)
ITL 공간	48	bytes	24 * 2 (24 * INITRANS)
초기 예상 건수	100,000	Rows	사용자 환경
월중 예상 추가 건수	20,000	Rows	사용자 환경
데이터 보관 월 수	120	개월	사용자 환경

250bytes 이하 및 초과 컬럼 개수는 Row의 데이터 길이 테이블을 참고한다. 초기 예상 건수, 월중 예상 추가 건수, 데이터 보관 월 수는 운영할 시스템의 사용자 환경의 정보이다.

2.2. 테이블 크기 산정방법

본 절에서는 Tibero에서 테이블 크기를 산정하는 방법에 대해서 설명한다.

2.2.1. 한 Row 저장에 필요한 공간

산정공식은 다음과 같다.

```
한 Row 저장에 필요한 공간 = Row Directory + Row Header + Row의 데이터 길이
+ (1 * <250bytes 이하 컬럼개수>) + (3 * <250bytes 초과 컬럼개수>)
```

항목	사전 정보 크기	실제 저장 크기	비고
Row Directory		2	2

Row Header	3	3	
Row의 데이터 길이	2,622	2622	
250bytes 이하 컬럼개수	3	1 * 3	1 * 250bytes 이하 컬럼개수
250bytes 초과 컬럼개수	2	3 * 2	3 * 250bytes 초과 컬럼개수
Row Space		2,636	2+3+2622+(1*3)+(3*2)

2.2.2. 한 블록의 데이터 공간

block당 사용 가능한 데이터 공간 크기이다.
산정공식은 다음과 같다.

```
한 블록의 데이터 공간 = (Block Size - Block Header - ITL공간) * (100 - PCTFREE) / 100
```

항목	사전 정보 크기	실제 저장 크기	비고
Block Size	8192	8192	
Block Header	48	48	
ITL공간	48	48	24 * 2 (24 * INITRANS)
PCTFREE	10	10	
Data Space per Block		7,286	(8192-48-48)*(100-10)/100

2.2.3. 한 블록에 들어갈 수 있는 Row 개수

산정공식은 다음과 같다

```
/ Row Space ( 한 블록의 데이터 공간 / 한 Row의 저장에 필요한 공간 )
```

항목	사전 정보 크기	실제 저장 크기	비고
Data Space per Block	7,286	7,286	
Row Space	2,636	2,636	
Row Size per Block		2	7,286 / 2,636

2.2.4. 총 데이터 Row 개수

산정공식은 다음과 같다.

`총 데이터 Row 개수 = <초기 예상 건수> + <월중 예상 추가 건수> * (<데이터 보관 월 수> + 1)`

항목	사전 정보 크기	실제 저장 크기	비고
초기 예상 건수	100,000	100,000	
월중 예상 추가 건수	20,000	20,000	
데이터 보관 월 수	120	120+1	<데이터 보관 월 수> + 1
Total Row Size		2,520,000	100,000 +20,000 *(120+1)

2.2.5. 필요한 총 데이터 공간

산정공식은 다음과 같다.

```
필요한 총 데이터 공간 = Block Size * Total Row Size / Row Size per Block (Block Size
* 총 데이터 Row 개수 / 한 블록에 들어갈 수 있는 Row 개수)
```

항목	사전 정보 크기	실제 저장 크기	비고
Block Size	8,192	8,192	
Total Row Size	2,520,000	2,520,000	
Row Size per Block	2	2	
Total Table Size		10,321,920,000 (9.613037GB)	8,192 * 2,520,000 / 2

제 3장 인덱스 크기 산정

본 장에서는 Tibero의 인덱스 크기를 산정하기 위해 필요한 정보와 방법에 대해 설명한다.

3.1. 사전 정보

Tibero에서 인덱스 크기 산정에 필요한 사전 정보로 Key Column 총 길이, Base RowID 산정 기준 그리고 항목별 크기 산정 정보가 필요하다.
다음은 Key Column 총 길이, Base RowID 산정 기준 그리고 항목별 크기 산정의 예이다.

3.1.1. Key Column 총 길이

인덱스 크기를 산정하기 위해서는 컬럼의 데이터 타입별 실제 소비 크기와 저장되는 길이을 산정해야 하며 이 정보는 인덱스 크기를 산정하기 위한 항목별 크기 산정방법에 사용된다. 인덱스 크기를 산정할 때 컬럼에 지정된 데이터 타입별 저장되는 크기 산정이 필요하다.
다음은 컬럼 정의 및 계산방식의 예이다. 크기산정 테이블은 5개의 컬럼으로 구성되었다고 가정한다.

이름	데이터타입	데이터길이	비고
c1	CHAR(10)	10	
c2	DATE	8	
c3	NUMBER(7,2)	4	roundup(7/2)
c4	CHAR(500)	500	
c5	VARCHAR(3000)	2,100	3,000*0,7
Key Column 총 길이		18	c1 + c2

"제4장 데이터 타입별 크기 산정"의 데이터 타입별 사이즈 표를 참고하여 위 테이블을 구성하는 각 컬럼의 데이터 타입별 실제 소비사이즈를 산정한 수치이다. 자세한 항목별 수치 산정방법은 "2.1.1. Row의 데이터 길이"를 참고한다.

3.1.2. Base RowID 산정 기준

크기 선정 대상인 위 테이블의 인덱스는 Local Index와 Not Unique Index로 선별 되었다.

Name	Type	데이터길이	비고
A	Global Index	10	Global, Local 선별
	Unique Index	6	

B	Unique Index	0	Unique, Not Unique 선별
	Not Unique Index	1	
Base RowID 총 길이		8	A + B의 합계

3.1.3. 항목별 크기 산정

인덱스 크기를 산정하기 위해 필요한 항목은 다음과 같다.
크기 예는 각 항목별로 기본으로 정해진 크기, 파라미터로 설정할수 있는 크기와 운영할 시스템에서 사전조사된 수치를 나타낸다.

항목	크기 예	단위	설명
Row Directory	2	bytes	기본값: 2
Index Header	2	bytes	기본값: 2
250bytes 이하 컬럼 개수	2	개수	위의 Key Column 총 길이 테이블에 c1, c2 가 대상이다.
250bytes 초과 컬럼 개수	0	개수	
Block Size	8192	개수	기본값: 8192 (파라미터)
Block Header	224	bytes	
INITRANS	2	개수	기본값: 2 (파라미터)
PCTFREE	10	%	기본값: 10% (파라미터)
ITL공간	48	bytes	24 * 2 (24 * INITRANS)
초기 예상 건수	100,000	Rows	사용자 환경
월중 예상 추가 건수	20,000	Rows	사용자 환경
데이터 보관 월 수	120	개월	사용자 환경
Base RowID	7	bytes	6 + 1

3.2. 인덱스 크기 산정방법

본 절에서는 Tibero에서 인덱스 크기를 산정하는 방법을 설명한다.

3.2.1. 한 Index Entry의 저장에 필요한 공간

산정 공식은 다음과 같다.

```
Row Directory + Index Header + Base RowID
+ Key Column 총 길이 + (1 * <250bytes 이하 컬럼개수>) + (3 * <250bytes 초과 컬럼개수>)
```

항목	사전 정보 크기	실제 저장 크기	비고
Row Directory	2	2	
Index Header	2	2	
Base RowID	7 7	6 +1	
Key Column 총 길이	18	18	
250bytes 이하 컬럼개수	2	1 * 2	1 * 250bytes 이하 컬럼개수
250bytes 초과 컬럼개수	0	3 * 0	3 * 250bytes 초과 컬럼개수
Index Entry Space		31	2+2+7+18+(1*2)+(3*0)

3.2.2. 한 블록의 데이터 공간

산정공식은 다음과 같다.

```
(Block Size - Block Header - ITL 공간) * (100 - PCTFREE) / 100
```

항목	사전 정보 크기	실제 저장 크기	비고
Block Size	8192	8192	
Block Header	224	224	
ITL공간	48	48	24 * 2 (24 * INITRANS)
PCTFREE	10	10	
Data Space per Block		7,128	(8192-224-48)*(100-10)/100

3.2.3. 한 블록에 들어가는 Entry 개수

산정공식은 다음과 같다.

```
Data Space per Block / Index Entry Space (한 블록의 데이터 공간
/ 한 Index Entry의 저장에 필요한 공간)
```

항목	사전 정보 크기	실제 저장 크기	비고
Data Space per Block	7,128	7,128	
Index Entry Space	31	31	
Entry Size per Block		229.9355	7,128 / 31

3.2.4. 총 데이터 Row 개수

산정공식은 다음과 같다.

```
초기 예상 건수 + 월중 예상 추가 건수 * (데이터 보관 월 수 + 1)
```

항목	사전 정보 크기	실제 저장 크기	비고
초기 예상 건수	100,000	100,000	
월중 예상 추가 건수	20,000	20,000	
데이터 보관 월 수	120	120+1	<데이터 보관 월 수> + 1
Total Row Size		2,520,000	100,000 + 20,000 * (120+1)

3.2.5. Index Leaf 블록에 필요한 Index Branch 블록의 비율

Leaf Block 일정수마다 Branch Block이 생성된다.

"3.2.6. 필요한 총 데이터 공간"의 산정방식 (Block Size * (A+B)) 중 A는 Leaf Block을 B는 Branch Block을 의미한다.
산정공식은 다음과 같다.

```
1 / Entry Size per Block (1 / 한 블록에 들어가는 Entry 개수)
```

항목	사전 정보 크기	실제 저장 크기	비고
Entry Size per Block	229.9355	229.9355	
Index Branch Ratio		0.004349046	1 / 229.9355

3.2.6. 필요한 총 데이터 공간

산정공식은 다음과 같다.

```
Block Size *( Total Row Size / Entry Size per Block + Total Row Size
/ Entry Size per Block ^ 2 )
```

항목	사전 정보 크기	실제 저장 크기	비고
Block Size	8,192	8,192	
Total Row Size	2,520,000	2,520,000	
Entry Size per Block	229.9355	229.9355	
Total Index Size		90,161,152	8192*(TRUNC(2520000/229.9355,0) +TRUNC(2520000/229.9355^2,0))

3.2.7. Index Split이 일어난 경우

Index Split이 일어난 경우 실제 블록의 50% 공간만 차지하게 되므로 사용 공간 최대 값은 최종 계산 결과 * 2를 해준다.

1. 인덱스의 경우 Create Index로 데이터를 처음 적재하는 경우 PCTFREE로 지정된 비율을 제외하고 블록을 최대한 사용하여 데이터를 저장한다.
2. 인덱스 생성 이후 신규로 인덱스 키 데이터가 추가되는 경우 해당 인덱스 키가 추가되어야 하는 블록에 여유공간이 없을 때 SPLIT이 발생하게 된다
3. SPLIT이 발생하게 되면 새로운 블록이 할당되고 기존 블록에 있던 인덱스 키 데이터를 통상 5:5 비율로 분할하여 나누어 가지기 때문에 두 개의 블록에 빈 공간이 50% 생긴다.

따라서, 인덱스에 데이터 추가가 계속된다고 하면, 특정 시점 이후에 Index Split이 얼마나 발생했는지는 정확하게 예측하기는 어렵다. 다만, 처음 SPLIT이 발생된 이후, 다음 번 SPLIT이 발생되기까지 빈 공간이 0% ~ 50% 범위에서 유지되기 때문에 평균적으로 절반 정도의 블록이 SPLIT이 되어 있다라고 가정하는 것이 가장 근사하게 추정하는 방법일 것이다.
Index Size시트의 크기 산정 방식에서 최종 계산 결과(Total Table Size for Index Spli)에 2를 곱한 것은 모든 블록이 SPLIT되어 있을 때 최대 사용공간을 가정하여 산출한것이며 만약 평균 개념으로 추정한다면 1.5를 곱해서 계산하는 것이 현실적일수 있다.

산정 공식은 다음과 같다.

```
Index Split이 일어난 경우 데이터 공간 = Total Index Size * 2 (필요한 총 데이터 공간 * 2 )
```

항목	사전 정보 크기	실제 저장 크기	비고
Total Index Size	90,161,152	90,161,152	
Total Index Size for Index Split		180,322,304	90,161,152 *2

제 4장 데이터 타입별 크기 산정

다음 표는 테이블과 인덱스 크기 시트의 계산식에서 적용되는 데이터 타입별 실제 소비 사이즈이며 저장되는 크기는 산정 길이 +1로 한다. 예를 들면 데이터의 산정 길이는 8bytes이며 실제 저장 데이터는 8+1=9bytes이다.
다음은 각 데이터 타입별 크기를 산정방법이다.

- 문자형

타입	고정/가변	길이 산정 예	설명
CHAR	고정/가변	CHAR(10) = 10bytes	– CHAR(10 BYTE), CHAR(10 CHAR) 문자열 의 길이는 byte와 문자를 기준으로 지정한다. – CHAR(10)의 형태로 선언하면, byte로 문자 열의 길이가 지정된다. – CHAR로 길이를 선언하면 문자가 몇 byte로 표현되는지에 따라 그 길이가 달라진다.
VARCHAR	가변	VARCHAR 가변 VARCHAR(10) = (10*0,7)= 7bytes	지정된 데이터 길이의 0.7로 가정
NCHAR	고정	NCHAR(10) = 30bytes(UTF8), 20bytes(UTF16)	– UTF8 : size의 최대 3배 – UTF16 : size의 최대 2배
NVARCHAR	가변	NVARCHAR(10) = (10*0.7)*3 = 21bytes(UTF8), 14bytes(UTF16)	UTF8 : size의 최대 3배 – UTF16 : size의 최대 2배
RAW	가변		실제로 저장되는 데이터의 길이
LONG	가변		
LONG RAW	가변		

- 숫자형

타입	고정/가변	길이 산정 예	설명
NUMBER	가변	NUMBER(7) =roundup(7/2) = 4bytes	roundup(size/2) size가 38 자리수보다 크면 38 로 정의하고 38 자리수 미만의 음수는 1byte를 더한다
INTEGER	가변		NUMBER와 같다.
FLOAT	가변		

- 날짜형

타입	고정/가변	길이 산정 예
DATE	고정	8bytes
TIME	고정	8bytes
TIMESTAMP	가변	12bytes
TIMESTAMP WITH TIME ZONE	고정 1	7bytes
TIMESTAMP WITH LOCAL TIME ZONE	고정	12bytes

- 간격 형

타입	고정/가변	길이 산정 예
INTERVAL YEAR TO MONTH	고정	5bytes
INTERVAL DAY TO SECOND	고정	12bytes

- 대용량 객체형

타입	고정/가변	길이 산정 예	설명
CLOB	가변		실제로 저장되는 데이터의 길이 ("LOB Type 저장에 필요한 overhead" 참조)
BLOB	가변		
XMLTYPE	가변		CLOB과 같다.

- 내재형

타입	고정/가변	길이 산정 예	설명
ROWID	고정	10bytes	

4.1. LOB Type 저장에 필요한 overhead

다음은 LOB Type 저장에 필요한 overhead에 대한 설명이다.

– NULL일 때
`0byte`

– 크기가 0일 때
`16bytes(lob locator)`

– 크기가 4000보다 작을 때

`16bytes(lob locator) + 16bytes(lob inode)`

– 크기가 96KB 보다 작을 때

`16bytes(lob locator) + 16bytes(lob inode) + 48bytes(12 page dbas)`

– 그 이상일 때

`16bytes(lob locator) + 16bytes(lob inode) + 48bytes * (LOB 컬럼 크기 / 64KB)(lob index)`

위 계산에서 LOB 데이터를 별도 segment로 저장하기 위한 overhead는 계산하지 않는다.

long의 경우 데이터의 길이가 길어지면 chained row piece로 나뉘어서 저장되는데, row piece를 연결시키기 위한 추가 overhead는 고려하지 않는다.

제 5장 Redo 로그 파일 크기 산정

본 절에서는 Tibero의 Redo 로그 파일 크기 산정방법에 대해 설명한다.

5.1. 개요

Redo 로그 파일은 최소 2개 그룹을 권장하며, Archivelog 모드인 경우 최소 3개 그룹을 권장한다. 이는
Redo 로그가 순환할 때 rewrite 방지를 위한 hangup 현상을 막기 위함이며 최소 2개의 멤버를 권장한다.
성능을 위해 각각 다른 물리 디스크에 배치하도록 한다.
Redo 로그 파일의 크기를 사전에 정확하게 추측하기가 어렵기 때문에 수행하는 업무의 일부를 수행한후
Redo 로그 양을 측정하고 전체 업무를 수행했을때의 Redo 로그 파일의 크기를 예측하는 것을 권장한다.

5.2. 고려사항

다음은 Redo 로그 파일 크기를 산정할 때 고려할 사항이다.

- 트랜잭션 처리 중 Log Switch가 발생하는 경우 트랜잭션 성능 저하가 발생한다.
- OLTP 중심의 시스템에서 트랜잭션 처리의 피크 타임에 Log Switch가 빈번하게 일어나지 않도록 Redo 로그 크기의 조정이 필요하다.
- Redo 로그 파일의 크기가 크면 Log Switch 발생 가능성이 낮은 반면 Recovery 시간은 길어진다.

위 고려사항을 참고해서 다음의 모드 중에 하나를 선택한다.

- LOGGING / NOLOGGING 모드
    - 테이블/인덱스에 대한 모든 변경사항은 기본적으로 Redo 로그로 기록되지만 NOLOGGING 테이블/인덱스에 대해 Direct-Path(DP) 방식으로 데이터를 추가하는 경우 Redo 로그를 기록하지 않는다.

      DP에 대한 Redo 로그가 남지 않는 경우에 Media Recovery가 수행될 때 DP로 추가한 테이블 데이터 가 복구되지 않기 때문에 DP로 데이터를 추가한 후에는 백업을 받도록 권장한다.

다음은 LOGGING 모드에서 하나의 DML당 로그 발생량 및 확인 방법이다.

- 로그의 발생량을 DML 한건으로 측정하기가 쉽지 않기때문에 여러 건의 DML을 수행한 후 전체 로그 발생량을 측정해서 건수로 나누는 방법을 권장한다.
- 로그 발생량은 v$sysstat 테이블의 'name' 컬럼이 'Redo log size'인 항목으로 확인 가능하며 alter system clear stat 명령으로 v$sysstat 테이블의 내용을 clear할 수 있다.
- ARCHIVELOG / NOARCHIVELOG 모드
  Redo 로그 파일은 Cache Recovery될 때 더 이상 필요없는 경우에 재사용된다. (보통 3개의 Redo 로그 파일들을 생성해서 순차적으로 돌면서 재사용하는 구조이다).

    - ARCHIVELOG 모드
      Media Recovery를 수행해야하는 경우 DB를 백업받은 이후 생성된 전체 Redo 로그파일들이 필요하며 Redo 로그파일들을 재사용하기 전 Media Recovery에서 사용할 목적으로 Redo 로그들을를 다른곳에 복사해두는 방식이다.
        - NOARCHIVELOG 모드 및 LOGGING 모드
          NOARCHIVELOG 모드인 경우 Media Recovery가 지원되지 않기 때문에 LOGGING 테이블인 경우라도 성능개선을 목적으로 DP를 수행할 때 Redo 로그를 남기지 않는다.
          ARCHIVELOG/NOARCHIVELOG 모드는 Redo 로그를 재사용하기 전 백업 수행 여부에 대한 판단이기 때문에 LOGGING/NOLOGGING과는 개념상 큰 연관성은 없다.
- Direct-Path Loading / Direct-Path Insert (DPL/DPI) 모드
    - DP로 데이터를 추가하는 경우 데이터를 row 단위로 건건이 추가하는 방식이 아니라 batch 단위로 추가하기 때문에 성능이 빠르며 Redo 로그의 양도 row 단위로 건건이 추가할 때보다 적게 생성된다.

5.3. 용량 산정

Redo 로그에는 사용자 데이터에 가해지는 변경사항이 저장되며 DML의 종류에 따라 다음과 같은 내용이 저장된다.

- INSERT

```
INSERT 하는 ROW의 내용(TABLE row 전체 + INDEX key 전체) +
INSERT에 대한 UNDO(delete할 rowid + delete할 INDEX key)
```

- UPDATE

```
UPDATE 내용(ROW의 내용 중 새로 변경되는 내용 + 변경되는 컬럼의 INDEX key delete/insert)
+ UPDATE에 대한 UNDO(row 변경을 되돌리는 내용 + INDEX key delete/insert)
```

- DELETE

```
DELETE되는 ROW에 대한 내용(rowid 정보 + INDEX key 전체)
+ DELETE에 대한 UNDO(row 전체 + INDEX key 전체)
```

- 추가로 다음과 같은 내용을 포함한다.
    - 트랜잭션 처리(begin/commit)를 위한 Redo 로그
    - Segment의 공간 관리(extent add, block format, index split)에 대한 Redo 로그
    - Undo 블록에 대한 변경사항
    - 블록 내 각 ROW에서 수행했던 트랜잭션의 commit 여부를 확인하고 변경하는 Redo 로그

5.4. Redo 로그 제한 사항

다음은 Redo 로그의 제한 사항이다.

구분	설명
로그파일 최대 개수	CREATE DATABASE 절의 MAXLOGFILES 파라미터
Group당 로그 파일 최대개수	Unlimited
최소 크기	4MB
최대 크기	– Tibero 5 SP1 : 4GB – Tibero 6 이상 : 2TB - Tibero 7 : 2TB

5.5. Redo 로그 크기

Redo 로그 파일에 쓰여지는 데이터량은 SQL 및 갱신 데이터량에 의해 크게 바뀌기 때문에 운영전 사전 에 정확한 추측이 불가능하다.
시스템의 크기에 따라서 다음과 같이 사전 설정 후 실제 운영 중 REDO 발생량을 보고 조정해야 한다.

구분	설명
소규모 시스템	4MB~100MB
중규모 시스템	100MB~1GB
대규모 시스템	1GB 이상

제 6장 Undo 테이블 스페이스 크기 산정

본 절에서는 Tibero의 Undo 테이블 스페이스 크기 산정방법에 대해 설명한다.

6.1. 개요

데이터의 갱신이 많은 시스템인 경우 Undo 테이블 스페이스의 I/O가 많이 발생하므로 Undo 테이블 스페이스를 여러 개의 데이터 파일로 작성하여 데이터 파일을 분산시킨다.
자동 관리는 다음의 장점을 갖는다.

- 설계 및 관리 용이(rollback,segment)
- UNDO Data의 overwrite 방지
- 영역확장 용이
- 플래시백 쿼리 이용가능

참고
Tibero는 수동 관리는 지원하지 않는다.

6.2. 고려사항

다음은 Undo 테이블 스페이스 크기를 산정할 때 고려할 사항이다.

- 데이터 파일 용량, UNDO_RETENTION, Undo Segment 수
- 자동 관리일 경우 Undo Segment의 개별 공간은 자동으로 관리되므로 전체 Undo Segment의 최소/최대 개수와 데이터 파일 용량만 설정한다.
    - 최소 Undo Segment 개수(_USGMT_ONLINE_MIN)는 보통 10(default)으로 설정한다.
      단, 동시 수행 트랜잭션 수가 많은 경우 부하 중에 추가 Undo Segment를 생성하는 오버헤드가 있을 수 있다. 이 경우 예상되는 동시 수행 트랜잭션 수만큼 최소 Undo Segment 개수를 설정해두면 Undo Segme 추가 생성 오버헤드를 줄일 수 있다.
      최소 Undo Segment 설정 개수가 이미 만들어져 있는 Undo Segment 개수보다 많은 경우 부팅 과정에서 Undo Segment를 추가로 생성한다.
      TAC의 경우 각 노드별로 적용한다. 예를 들어 NODE1은 _USGMT_ONLINE_MIN=30, NODE2는 40으로 설정하고 부팅하면 각각 30개와 40개의 Undo Segment가 생성되어 있는 것을 확인할 수 있다.

6.3. 용량 산정

Undo 테이블 스페이스를 생성하는데 필요한 최소 크기와 예상 동시 수행 트랜잭션수를 기준으로 한 예상 크기, 그리고 APM 정보를 이용한 실제 사용 크기로 나누어 생각해 볼 수 있다.
최초로 DB를 설치하는 경우 최소 크기 및 예상 크기중 큰 값을 기준으로 Undo 테이블 스페이스 크기를 산정하고 이후 APM에서 최대 부하가 들어올때 초당 소모하는 UNDO block 수를 기준으로 실제 크기와

산정한 크기가 맞는지 검증할 수 있다.

다음은 Undo 테이블 스페이스의 산정방법이다.

- 최소 크기

```
(Undo 테이블 스페이스당 Undo Segment 최소 개수 * _USGMT_UNIFORM_EXTSIZE * 2
* Block size(8KB)) + 여유값
```

- 예상크기

```
(Undo 테이블 스페이스당 Undo Segment 최대 개수 * _USGMT_UNIFORM_EXTSIZE * 2
* Block size(8KB)) + 여유값
```

- 실제 크기

```
(초당 UNDO Block 소모량 * UNDO retention * Block size(8KB)) + 여유값
```

다음은 산정 공식에 사용되는 항목에 대한 설명이다.

항목	설명
Undo 테이블 스페이스당 Undo Segment 최소 개수	Single/TAC 모두 경우 _USGMT_ONLINE_MIN 값이다.
Undo 테이블 스페이스당 Undo Segment 최대 개수	예상되는 최대 동시 수행 트랜잭션 수이다. – Single의 경우 : _USGMT_ONLINE_MAX 값 – TAC의 경우 : _USGMT_ONLINE_MAX / 노드수
_USGMT_UNIFORM_EXTSIZE * 2	Undo Segment는 최소 2개의 UNDO extent가 필요하고 Undo extent 하나의크기는 _USGMT_UNIFORM_EXTSIZE이다.
초당 UNDO block 소모량	운영 중 최대 부하 당시의 APM 리포트에서 Undo Statistics의 Undo blocks값을 APM 구간(초)로 나눈 값이다
UNDO_RETENTION	TIP 설정 값이다. (기본값: 900초)
여유값	KB 값이다. Long 트랜잭션이 비정기적으로 들어올 경우 (INDEX rebuild등도 포함)를 대비하여 충분한 크기를 잡아 두는 것이 좋다.

제 7장 Temp 테이블 스페이스 크기 산정

본 절에서는 Tibero의 Temp 테이블 스페이스 크기 산정방법에 대해 설명한다.

7.1. 개요

Join이나 대량 Sort 처리가 많은 경우 Temp 테이블 스페이스에서의 I/O가 다발적으로 발생한다. (CREATE INDEX / ANALYZE / SELECT DISTINCT / ORDER BY / GROUP BY / UNION / SORT-MERGE)
Temp 테이블 스페이스 크기 산정을 위해서는 SQL Execution Work Area와 Temp 테이블 스페이스간의 관계 이해가 필수이다.

7.2. 고려사항

다음은 Temp 테이블 스페이스의 크기를 산정할 때 고려할 사항이다.

- 시스템유형 : OLTP, DSS, DW, Batch Job
- User Load
- Transaction Volume
- Data Size
- Query & Statements 튜닝
- Temp 테이블 스페이스 자체를 여러 개의 데이터 파일로 작성해서 분산 관리 또는 User별 다른 Temp 테이블 스페이스 할당
- Temp 테이블 스페이스 Group 설정

7.3. 용량 산정

다음은 Temp 테이블 스페이스의 용량을 사용하는 방법에 대한 설명이다.

구분	설명
EXTENT SIZE	SORT_AREA_SIZE*2 + Block Size (기본 1MB 권장)
SORT AREA의 최대 크기	SORT_AREA_SIZE → 6KB, 세션 → 1000이라면 Sort 영역의 최대크기는 16MB가 된다. (기본값: (MEMORY_TARGET - TOTAL_SHM_SIZE) * 0.3)

구분	설명
SORT SEGMENT	인스턴스 시작 후 첫 SQL이 Temp 테이블 스페이스를 이용하여 Sort Segment를 생성한다. DB가 shutdown되면 release된다. – V$SORT_SEGMENT : SORT_SEGMENT영역 할당 및 해제 확인한다. – V$SORT_USAGE : SORT_SEGMENT를 사용하고 있는 사용자를 확인한다. – V$TEMPFILE, DBA_TEMP_FILES : Temp 테이블 스페이스에 할당된 tempfiles을 확인한다. – DBA_TEMP_FREE_SPACE

참고
Temp 테이블 스페이스의 용량을 산정하기 위해서 MEMORY_TARGET 파라미터를 이용하고, Autoextending Temp Files을 설정할 것을 권장한다

제 8장 Shared Memory 크기 산정

본 장에서는 Tibero의 SQL Execution Work Area 크기 산정방법에 대해 설명한다.

8.1. 개요

Tibero Shared Memory는 부팅할 때 고정된 용도로 사용되는 Fixed 영역과 운영 중에 자동으로 할당되어 사용되는 Shard Pool 영역으로 나뉜다.

- Fixed Memory
  DB가 부팅할 때 파라미터 값을 바탕으로 크기가 고정되는 영역이다.
  - Buffer Cache(Database Buffer)
  - Log Buffer
  - 그 외 항목 : 전역 변수를 위한 공간, 워킹 스레드 정보와 세션 정보를 관리하는 영역
- Shared Pool Memory
  운영 중에 동적으로 할당되어 사용되는 영역이다.
  - DD Cache
  - PP Cache(Library Cache)
  - 그 외 항목

8.2. 고려사항

다음은 Shared Memory 크기를 산정할 때 고려해야할 사항이다.

- Shared Memory 크기를 운영 중에 동적으로 늘릴 수 없다.
- Buffer Cache와 Shared Pool 사용 패턴을 분석하여 전체 Shared Memory 크기를 알맞게 정해야 한다.
- Buffer Cache
  - 주요 Workload를 돌려본 후 APM 리포트의 Buffer Cache Hit율을 보고 판단한다.
  - Hit율이 90 % 이하 : Buffer Cache 크기 늘려야 한다.
- Shared Pool Memory
  - Workload를 돌려본 후 v$sga의 'SHARED POOL MEMORY' 항목의 사용률 보고 판단한다.
  - Shared Pool 사용률이 너무 높으면 (Free 영역이 적은 경우) Shared Pool 크기를 늘려야 한다.
  - 세션당 최소 1M 이상의 Shared Pool 영역 확보가 필요하다.

8.3. 용량 산정

Shared Memory 각 영역의 크기는 v$sga에서 확인할 수 있다.

```
SQL> select * from v$sga;

NAME                                         TOTAL       USED
---------------------------------------- ---------- ----------
SHARED MEMORY                            1258291200 1258291200
FIXED MEMORY                             1032962080 1032962080
SHARED POOL MEMORY                        225312768  222298112
SHARED POOL ALLOCATORS (LC)                25165824   24048688
SHARED POOL ALLOCATORS (DD)                16777216    3620240
SHARED POOL ALLOCATORS (SLAB)              79691776   76092320
SHARED POOL ALLOCATORS (MISC)             100663296   98956536
SHARED POOL ALLOCATORS (Total)            222298112  202717784
Database Buffers                          838860800  838860800
Redo Buffers                               10485760   10485760
10 rows selected.
```

Shared Memory의 크기는 초기화 파라미터로 설정해서 조회할 수 있다.

- Shared Memory
  다음은 TOTAL_SHM_SIZE 파라미터를 설정해서 조회하는 예이다.

```
SQL> show param total_shm_size

NAME                           TYPE     VALUE
------------------------------ -------- ----------------------------------------
TOTAL_SHM_SIZE                 INT64    1258291200
```

- Buffer Cach
  다음은 DB_CACHE_SIZE 파라미터를 설정해서 조회하는 예이다.

```
SQL> show param db_cache_size

NAME                           TYPE     VALUE
------------------------------ -------- ----------------------------------------
DB_CACHE_SIZE                  UINT64   838860800
```

- 싱글 모드 : TOTAL_SHM_SIZE의 2/3
- TAC 모드 : TOTAL_SHM_SIZE의 1/2

- Log Buffer
  다음은 LOG_BUFFER 파라미터를 설정해서 조회하는 예이다. (LOG_BUFFER 파라미터의 기본값 :10MB)

```
SQL> show param log_buffer

NAME                           TYPE     VALUE
------------------------------ -------- ----------------------------------------
FLASHBACK_LOG_BUFFER           UINT32   0
LOG_BUFFER                     UINT32   10485760
```

- Shared Pool Memory
  전체 Shared Memory 크기에서 Fixed 영역을 뺀 나머지 영역이다.

```
TOTAL_SHM_SIZE - [Fixed Memory]
```

- 싱글 모드
  전체 shared pool 크기가 *MIN_SHARED_POOL_SIZE보다 커야 한다. (*MIN_SHARED_POOL_SIZE 기본값: 1M * MAX_SESSION_COUNT)

- TAC 모드
  TAC를 위한 CCC와 CWS 공간이 충분히 확보되어야 한다. CCC와 CWS를 위해 전체 Buffer Cache 크기의 25% 가량을 Shared Pool에서 사용한다.
  CCC와 CWS를 위한 공간을 제외한 Shared Pool 영역이 _MIN_SHARED_POOL_SIZE 보다 커야 한다.

```
[전체 Shared Pool 크기] = _MIN_SHARED_POOL_SIZE + [CCC 공간] + [CWS 공간]
+ [Free Shared Pool 크기]
```

PP Cache와 DD Cache 등을 위한 Free Shared Pool 크기는 CCC와 CWS 공간보다 커야 한다.

```
[Free Shared Pool 크기] > ([전체 Shared Pool 크기] - _MIN_SHARED_POOL_SIZE) / 2
```

제 9장 SQL Execution Work Area 크기 산정

본 장에서는 Tibero의 SQL Execution Work Area 크기 산정방법에 대해 설명한다.

9.1. 개요

다른 워킹 프로세스 및 워킹 스레드와 공유되지 않은 독립적인 메모리 공간으로 세부 영역은 다음과 같다.

- 세션 정보 영역
  로그온 및 세션 관련 정보
- 커서정보영역(SQL영역)
  SQL 문 처리할 때 필요한 메모리 공간
- SQL Work Area(SQL 작업 영역)
  Sort, hash-Join, Bitmap 생성 및 Merge(병합) 작업할 때 사용되는 메모리 영역

다음은 관련 파라미터에 관한 설명이다.

구분	설명
총 SQL Execution Work Area	– 산정방법 `MEMORY_TARGET – TOTAL_SHM_SIZE`
MEMORY_TARGET	총 메모리 사용량이다.
TOTAL_SHM_SIZE	총 공유 메모리 사용량이다.
_OP_MEM_MAX_SIZE	하나의 operation(정렬, 조인, 분석)에서 사용할 최대메모리 사용량이다. (Tibero 7 기준)
_OP_MEM_PX_MAX_SIZE	하나의 operation(정렬, 조인, 분석)이 parallel로 수행될 때 slave들이 사용할 수 있는 총 최대 메모리 사용량이다. (Tibero 7 기준)
EX_MEMORY_OPERATION_LIMIT	하나의 operation에서 사용할 최대 메모리 사용량을 계산한다. (단위: MB) parallel 쿼리에서 slave들도 위의 최대 메모리 사용량을 쓰려고 하기 때문에 과다 메모리를 사용하려고 하고 memory tunner에 의해 나중에 수행되는 slave 의 경우 메모리 사용에 불이익을 받게 될 수도 있다.

9.2. 고려사항

다음은 SQL Execution Work Area 크기를 산정하기 위해 고려할 사항이다.

- 일반적인 온라인 업무 환경(OLTP)
  온라인 업무에서는 대량의 데이터를 읽어서 정렬하거나 분석하는 쿼리보다는 특정 값을 조회 및 수정 하기 위해 index를 이용하는 쿼리가 대부분이다. 이런 경우에는 SQL에서 독립적으로 사용하는 SQL Execution Work Area의 사용량이 많지 않고 오히려 Index를 이용한 Table Random Access

IO를 줄이 기 위해 Buffer Cache 영역을 대부분 사용하는 것이 바람직하다.

- 데이터 분석 업무 환경(OLAP)

대용량의 데이터를 Full Table Scan으로 읽어들여 정렬, 조인 및 분석을 수행하는 쿼리가 대부분이고 Parallel Execution을 이용하는 환경이다. 읽어들이는 데이터의 크기가 물리적인 메모리 크기에 담을 수 없을 정도의 크기인 경우가 많기 때문에 효율적인 Buffer Cache 이용을 기대할 수가 없는 상황이므로 공유 메모리 영역에서 Buffer Cache의 크기를 줄이고 독립적으로 사용 가능한 SQL Execution Work Area 메모리 크기에 따라 성능이 크게 좌우되는 정렬, 조인 및 분석 작업을 위해 대부분의 물리 메모리 를 SQL Work Area에 할당하는 것이 효율적이다.

9.3. 용량 산정

다음은 SQL Execution Work Area 크기 산정하는 방법이다.

항목	설명

일반적인 온라인 업무환경 (OLTP)	-SQL Execution Work Area (SQL 실행 작업 영역) : 20%
	-TOTAL_SHM_SIZE : MEMORY_TARGET 값의 80% (OS 및 다른 서비스 사용 영역 제외)
	`EX_MEMORY_OPERATION_LIMIT : 5000`
	- _OP_MEM_MAX_SIZE (Tibero 7 기준)
	`(IPARAM(MEMORY_TARGET) - IPARAM(TOTAL_SHM_SIZE)) * 0.05`
	- _OP_MEM_MAX_SIZE (Tibero7 기준)
	`(IPARAM(MEMORY_TARGET) - IPARAM(TOTAL_SHM_SIZE)) * 0.3`

데이터 분석 업무 환경 (OLAP)	-SQL Execution Work Area (SQL 실행 작업 영역) : 80%
	(가용 메모리 한계 내에서 최적의 작업 영역을 지정하여 디스크 I/O를 줄이 고 SQL의 수행속도를 높일 수 있도록 조절하기 위해서)
	-TOTAL_SHM_SIZE : MEMORY_TARGET 값의 20%
	-EX_MEMORY_OPERATION_LIMIT : 20000 (Tibero 7 기준 Deprecated)
	-_OP_MEM_MAX_SIZE(Tibero 7 기준)
	`(IPARAM(MEMORY_TARGET) - IPARAM(TOTAL_SHM_SIZE)) * 0.2`
	- _OP_MEM_MAX_SIZE(Tibero 7 기준)
	`(IPARAM(MEMORY_TARGET) - IPARAM(TOTAL_SHM_SIZE)) * 0.3`

총 SQL Execution Work Area	최대 MEMORY_TARGET - TOTAL_SHM_SIZE만큼 사용이 가능하다.