# Tibero Active Storage

## 관리자 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 Active Storage  관리자 안내서

발행일: 2024-08-22

소프트웨어 버전: Tibero 7.2.2

안내서 버전: v7.2.2

# 내용 목차

# 그림 목차

# 예 목차

# 안내서에 대하여

## 안내서의 대상

본 안내서는 Tibero® Active Storage(이하 TAS)를 사용해서 Tibero®(이하 Tibero)의 파일을 관리하고자 하는 TAS 관리자를 대상으로 기술한다.

## 안내서의 전제 조건

본 안내서는 TAS를 관리하는 방법을 설명한 안내서이다. 따라서 본 안내서를 원활히 이해하기 위해서는 다음과 같은 사항을 미리 알고 있어야 한다.

- Tibero의 이해
- 운영체제 및 시스템 환경의 이해
- UNIX 계열(Linux 포함)의 기본 지식

## 안내서의 제한 조건

본 안내서는 TAS를 실무에 적용하거나 운용하는 데 필요한 모든 사항을 포함하지 않는다. 따라서 설치, 환경설정 등 운용 및 관리에 대해서는 각 제품 안내서를 참고하기 바란다.

---

**참고**

TAS의 설치 및 환경설정에 관한 내용은 "Tibero 설치 안내서"를 참고하기 바란다.

---

# 안내서 구성

TAS 관리자 안내서는 총 6개의 장과 Appendix로 구성되어 있다.

각 장의 주요 내용은 다음과 같다.

- 제1장: TAS 소개

  TAS의 개념과 기능을 기술한다.

- 제2장: TAS 인스턴스 관리

  TAS 인스턴스 관리 방법을 기술한다.

- 제3장: TAS 디스크 스페이스 관리

  TAS 디스크 스페이스 관리 방법을 기술한다.

- 제4장: TAS 정보 조회

  동적 뷰를 사용한 TAS 정보 조회 방법을 기술한다.

- 제5장: 백업과 복구

  TAS를 사용하는 Tibero의 백업 및 복구 과정을 기술한다.

- 제6장: 커맨드라인 툴

  TAS 파일을 조회하고 관리하는 TASCMD를 사용하는 방법을 기술한다.

- Appendix A: Tibero구성 예제

  TAS를 이용해 Tibero를 구성하는 방법을 예제를 통해 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 |

# 관련 안내서

| 안내서 | 설명 |
|---|---|
| Tibero<br>설치 안내서 | 설치 시 필요한 시스템 요구사항과 설치 및 제거 방법을 기술한 안내서이다. |
| Tibero<br>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<br>유틸리티 안내서 | 데이터베이스와 관련된 작업을 수행하기 위해 필요한 유틸리티의 설치 및 환경설정, 사용 방법을 기술한 안내서이다. |
| Tibero<br>에러 참조 안내서 | Tibero를 사용하는 도중에 발생할 수 있는 각종 에러의 원인과 해결 방법을 기술한 안내서이다. |
| Tibero<br>참조 안내서 | Tibero의 동작과 사용에 필요한 초기화 파라미터와 데이터 사전, 정적 뷰, 동적 뷰를 기술한 참조 안내서이다. |
| Tibero<br>SQL 참조 안내서 | 데이터베이스 작업을 수행하거나 애플리케이션 프로그램을 작성할 때 필요한 SQL 문장을 기술한 참조 안내서이다. |

# 제1장 TAS 소개

본 장에서는 TAS의 개념과 기능을 개략적으로 설명한다.

## 1.1. 개요

TAS는 별도의 외부 솔루션 없이 직접 디스크 장치를 관리하여 Tibero 운용에 필요한 데이터 파일, 로그 파일 등을 저장하기 위한 논리 볼륨 관리자(Logical Volume Manager)와 파일 시스템이다. 또한 공유 디스크를 사용할 경우 Tibero Active Cluster(이하 TAC)기능을 사용할 수 있도록 클러스터링 기능을 제공한다.

---

**참고**

TAC에 대한 자세한 내용은 "Tibero 관리자 안내서"를 참고한다.

---

TAS는 여러 개의 디스크들을 **디스크 스페이스**로 관리한다. 디스크 스페이스는 논리 볼륨 위에 파일 시스템을 생성한 것과 유사하며, Tibero는 이러한 디스크 스페이스에 파일들을 저장한다.

디스크 스페이스를 사용해서 Tibero를 운용하는 도중에 디스크를 추가하거나 제거할 수 있다. 디스크를 추가/제거할 경우 TAS는 자동으로 디스크 스페이스에 있는 모든 디스크들을 공평하게 사용할 수 있도록 저장된 내용을 재배치한다. 디스크 스페이스 관리에 대한 자세한 내용은 “제3장 TAS 디스크 스페이스 관리”를 참고한다.

TAS는 Tibero의 데이터에 대한 미러링 기능을 제공한다. 크게 2개의 복제본을 가지는 2-way 방식(NORMAL option)과 3개의 복제본을 가지는 3-way(HIGH option)방식을 지원한다. 별도의 미러링 솔루션을 사용하고 있다면 TAS의 미러링 기능을 사용하지 않을 수도(EXTERNAL option) 있다.

## 1.2. TAS 개념

본 절에서는 TAS를 이해하는데 필요한 주요 개념들에 대해 설명한다.

## 1.2.1. TAS 인스턴스

TAS는 Tibero를 기반으로 만들어졌다. 그러므로 TAS 인스턴스를 기동하면 Tibero와 거의 동일한 프로세스들이 실행된다. 하지만 TAS는 데이터베이스보다 적은 양의 일을 수행하므로 시스템 자원을 거의 사용하지 않는다. TAS 인스턴스 관리에 대한 자세한 내용은 “제2장 TAS 인스턴스 관리”를 참고한다.

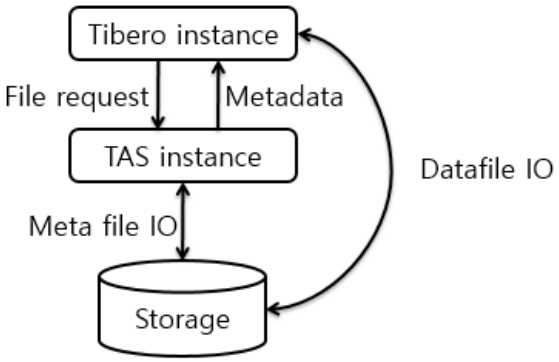
TAS는 디스크 스페이스와 파일 정보등을 기록하기 위한 메타 데이터를 첫 번째 디스크 스페이스에 저장하고 관리한다.

TAS 메타 데이터는 다음과 같은 정보들을 저장한다.

- 디스크 스페이스에 속한 디스크
- 디스크 공간 할당 정보
- 파일과 파일의 별칭
- 파일의 익스텐트 정보
- TAS 메타 데이터를 위한 로그

다음 그림은 TAS 인스턴스와 Tibero, 디스크 사이의 관계를 보여준다. Tibero는 TAS를 통해서 파일을 읽고 쓰는 것이 아니라, 파일의 메타 정보를 사용해서 직접 디스크를 읽고 쓴다. 그러기 위해 Tibero는 TAS로부터 파일의 메타 정보를 받아온다.

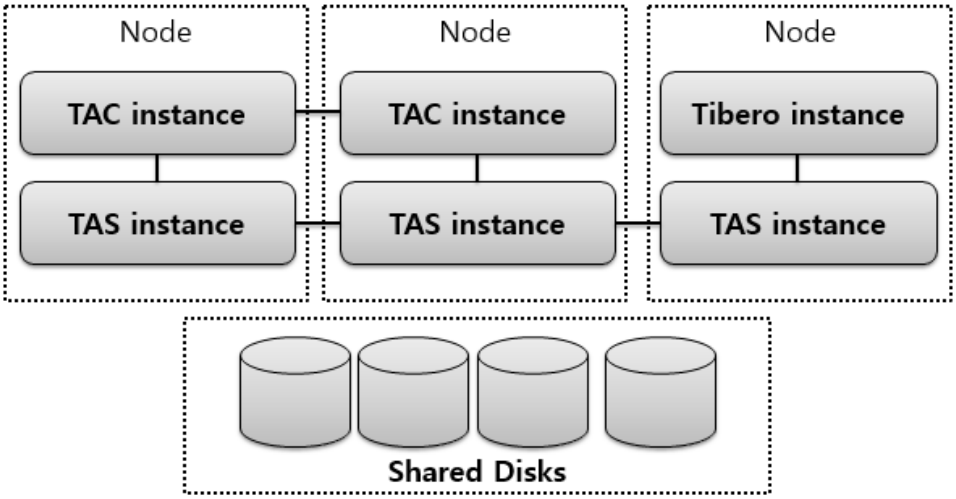
**[그림 1.1] TAS를 사용한 Tibero 구성**




TAS는 공유 디스크를 사용하기 위한 클러스터링 기능을 제공한다. 하나의 디스크 스페이스는 하나의 Tibero 인스턴스만 사용할 수 있다. 여러 Tibero 인스턴스가 하나의 디스크 스페이스를 사용하려면 해당 인스턴스들을 TAC로 구성해야 한다.

다음은 클러스터 기능을 사용한 Tibero와 TAS의 구성을 설명한다.

**[그림 1.2] TAS 클러스터링 기능을 사용한 Tibero와 TAC 구성**




## 1.2.2. TAS 디스크 스페이스

디스크 스페이스는 여러 개의 디스크들로 이루어져있다. 각각의 디스크 스페이스는 해당 디스크 스페이스를 관리하는데 필요한 메타 데이터와 데이터베이스에서 사용하는 파일들을 저장하고있다. 디스크 스페이스 관리에 대한 자세한 내용은 “제3장 TAS 디스크 스페이스 관리”를 참고한다.

## 1.2.3. 미러링과 실패 그룹

TAS는 디스크 스페이스에 저장된 데이터에 대한 미러링 기능을 제공한다. 미러링은 데이터의 복사본을 여러 디스크에 저장하여 데이터를 보호하는 방법이다.

미러링 기능을 사용하기 위해 디스크 스페이스를 생성할 때 다음과 같은 중복 레벨을 설정할 수 있다.

| **레벨** | **미러링 구분** |
|---|---|
| NORMAL | 2-way 미러링 |
| HIGH | 3-way 미러링 |
| EXTERNAL | 미러링하지 않음 |

**참고**

디스크 스페이스를 **EXTERNAL** 중복 레벨로 생성한 경우 해당 디스크 스페이스는 미러링과 관련된 모든 기능을 사용할 수 없다.

미러링된 파일에서 사용할 익스텐트를 할당할 때 파일의 중복 레벨에 따라 익스텐트를 여러 개(2개 또는 3개)를 할당한다. 이때 각 익스텐트를 할당할 디스크를 서로 다른 **실패 그룹**에서 선택한다. 이렇게 함으로

써 특정 실패 그룹에 속한 디스크 혹은 실패 그룹 전체에 장애가 발생하더라도 데이터 손실이 없으며, 해당 디스크 스페이스에 대한 서비스를 중단 없이 제공할 수 있다.

실패 그룹은 디스크 스페이스를 생성할 때 정의하며, 생성할 때 설정한 중복 레벨은 나중에 변경할 수 없다. 실패 그룹 정의를 생략하면 자동으로 각 디스크를 실패 그룹으로 정의한다. 중복 레벨이 **NORMAL**, **HIGH**인 디스크 스페이스는 각각 최소 2개, 3개의 실패 그룹을 정의해야 한다. 중복 레벨이 **EXTERNAL**인 디스크 스페이스는 실패 그룹을 사용하지 않는다.

## 1.2.4. TAS 디스크

TAS는 다음 디스크 장치들을 TAS 디스크로 사용할 수 있다.

- 디스크 전체
- 디스크 파티션
- 논리 볼륨

---

**참고**

위는 모두 OS 입장에서 I/O가 가능한 LUN의 형태로 보여지는 존재이다. 위의 3가지 타입에 대하여 모두 지원가능하다. 현재 Active Storage는 Linux, Solaris, AIX 운영체제를 지원하고 있다.

---

디스크나 Physical Volume에 대한 권한 문제는 **udev**를 통하여 디바이스 노드를 생성하여 해결할 수 있다.

SAN 스토리지 상에서 처음부터 LUN으로 나눠진 디스크에 대해서는 클러스터링 S/W 및 GFS2 없이 TAS 클러스터 구성이 가능하다.

TAS 클러스터 구성할 때 동일한 디스크가 각 노드에서 서로 다른 이름으로 설정되어도 TAS 디스크로 사용할 수 있다. 예를 들어 같은 디스크라도 CLVM이 없다면 1번 node에서 /dev/hdisk1로 보이고 2번 node에서 /dev/hdisk2로 보일 수 있는데 TAS로 구성할 때 이러한 환경에서도 정상적으로 DB를 구성할 수 있다. disk string을 통하여 이러한 TAS 디스크를 검색하게 되므로 클러스터에 참여하는 노드들이 필요한 disk/raw device를 볼 수 있도록만 설정해 주면 된다.

TAS는 파일의 내용을 디스크 스페이스에 속한 모든 디스크들에 분산 저장한다. 이러한 저장 방식은 디스크 스페이스에 속한 모든 디스크들의 공간을 균일하게 사용하도록 하며, 모든 디스크들에 동일한 입출력 부하가 가해지도록 한다. 그러므로 디스크 스페이스를 구성하는 TAS 디스크들은 물리적으로 서로 다른 디스크 장치여야 한다. RAID로 Logical Volume 구성할 때 striping size가 TAS의 AU size의 배수가 되도록 하는 것을 권장한다. 이렇게 해야 디스크의 strping 단위와 TAS의 striping 단위의 align이 맞게 되어 성능이 향상된다.

### 할당 단위

디스크 스페이스에 속한 모든 디스크들의 공간은 할당 단위로 나누어진다. 할당 단위는 디스크 스페이스에서 공간 할당에 사용하는 기본 단위이다. 하나의 파일 익스텐트는 한 개 혹은 여러 개의 할당 단위를 사용하며, 하나의 파일은 한 개 혹은 여러 개의 익스텐트를 사용한다.

디스크 스페이스에서 사용할 할당 단위의 크기는 디스크 스페이스를 생성할 때 **AU_SIZE**속성으로 설정할 수 있으며, 설정 가능한 값은 1, 2 또는 4MB이다.

## 1.2.5. TAS 파일

TAS의 디스크 스페이스에 저장된 파일들을 TAS 파일이라고 한다. Tibero는 TAS에게 파일 정보를 요청하며, 이는 Tibero가 일반 파일 시스템의 파일을 사용하는 것과 비슷하다.

TAS 파일의 이름은 '+{*디스크 스페이스 이름*}/...' 형태이며, 파일 정보에 포함되지 않고 별칭으로 관리한다.

### 익스텐트(Extent)

TAS 파일들은 익스텐트들의 집합으로 디스크 스페이스에 저장된다. 각 익스텐트는 디스크 스페이스에 속한 각 디스크에 저장되어있으며, 한 개 혹은 여러 개의 할당 단위를 사용한다. TAS는 큰 크기의 파일을 지원하기 위해 동일 파일 내에서 파일 크기에 따라 익스텐트 크기를 변경한다.

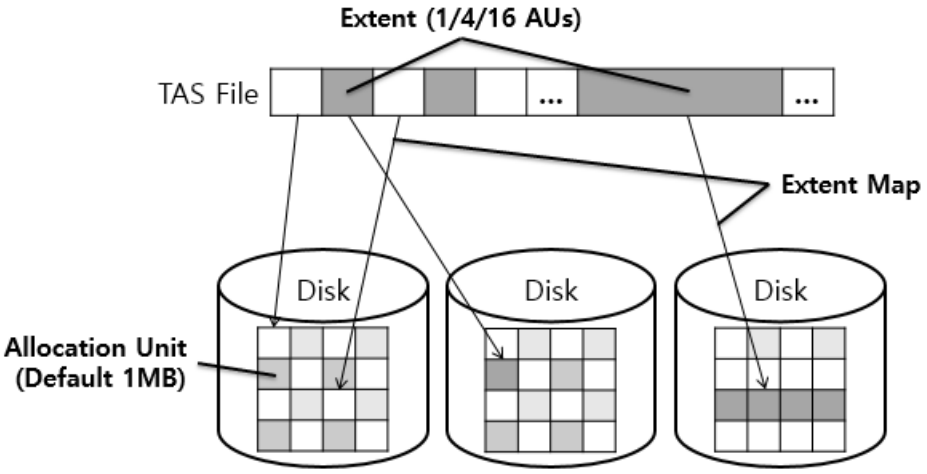
익스텐트 크기를 변경하는 것은 파일 익스텐트 정보를 유지하기 위한 메모리 사용을 줄이는 효과도 있다. 익스텐트의 크기는 파일의 크기가 커짐에 따라 자동으로 커지게 된다. 파일 크기가 작을 때는 디스크 스페이스의 할당 단위(Allocation Unit, 이하 AU)와 동일하며, 파일 크기가 커지면 4AU나 16AU의 크기가 된다.

익스텐트 크기는 아래와 같이 변경된다.

| **파일의 익스텐트 개수** | **익스텐트 크기** |
|---|---|
| 20000개 이하 | 1AU |
| 20000 ~ 40000 | 4AU |
| 40000 개 이상 | 16AU |

다음은 익스텐트와 할당 단위의 관계를 보여준다.

**[그림 1.3] TAS 파일 할당**




# 1.3. TAS 프로세스

TAS를 사용할 때는 추가적으로 ASSD(Active Storage Service Daemon) 프로세스를 실행한다. ASSD 프로세스는 TAC-TAS간의 통신 기능을 수행한다. 이 프로세스 내에서 실행되는 스레드는 다음과 같다.

| 스레드 | 역할 |
|---|---|
| COMM | ASSD 컨트롤 스레드가 받은 메시지를 실질적으로 처리하는 역할을 한다. DB에서 정보를 요청하는 메시지를 보내거나 TAS에서 요청을 처리하고 결과를 반환한다. |
| DMON | 지속적으로 디스크에 접근 가능한 상태인지 체크한다. 일정 횟수 이상 접근이 불가능하면 FAIL 처리하였다가 접근이 가능해지면 리싱크 작업을 수행한다. 이 스레드는 모든 TAS 노드들 중 하나를 선정해 해당 노드에서만 실행되고, 이 노드를 TAS 마스터 노드라고 한다. 마스터 노드가 어떠한 이유로 FAIL 되면 다른 TAS 중 하나가 마스터 노드가 된다. |
| RBAL | 리밸런스 작업을 수행하기 위해 필요한 작업들을 생성해 후술할 RBSL 스레드에 전달한다. |
| RBSL | RBAL 스레드로부터 리밸런스에 필요한 작업을 넘겨받아 실행한다. |

# 제2장 TAS 인스턴스 관리

본 장에서는 TAS의 초기화 파라미터 설정과 TAS 인스턴스를 기동하는 방법을 설명한다. 그리고 TAS를 사용하는 Tibero의 초기화 파라미터 설정과 데이터베이스 생성 방법을 설명한다.

## 2.1. TAS 초기화 파라미터 설정

LISTENER_PORT, MAX_SESSION_COUNT, MEMORY_TARGET, TOTAL_SHM_SIZE를 시스템 구성에 맞게 설정한다.

---

**참고**

기본 초기화 파라미터 설정은 "Tibero 참조 안내서"를 참고한다.

---

추가로 다음 초기화 파라미터들을 설정한다.

| 파라미터 | 설명 |
|---|---|
| INSTANCE_TYPE | 문자열 타입의 값으로 다음 중에 설정한다.<br>– 타입 : 문자열<br>– 설정 가능한 값 : "TIBERO" \| "AS" \| "ACTIVE_STORAGE"<br>– 기본값 : "TIBERO" |
| AS_DISKSTRING | TAS 디스크로 사용할 디스크 장치들의 경로를 설정한다.<br>– 타입 : 문자열<br>– 설정 가능한 값 : 디스크 장치 경로<br>– 기본값 : "" |

다음은 파라미터로 설정 가능한 형태에 대한 예제이다.

- 디스크 한 개를 지정하는 경우

```
"/devs/disk1"
```

- 디스크 한 개 이상을 사용하는 경우(콤마로 구분)

```
"/devs/disk1,/devs/disk2"
```

- 디스크 한 개 이상을 패턴으로 지정하는 경우

```
"/devs/disk*"
```

- 경로 제일 앞에 '?' 문자를 사용하면, TAS의 홈 디렉터리로 치환한다.

```
"?/devs/disk*"
```

다음 초기화 파라미터들은 필요에 따라 설정한다.

| 파라미터 | 설명 |
|---|---|
| AS_ALLOW_ONLY_RAW_DISKS | 'N'으로 설정할 경우 일반 파일 시스템에 있는 파일을 TAS 디스크로 사용할 디스크 장치로 인식하도록 한다.<br>– 타입 : 참, 거짓 값<br>– 설정 가능한 값 : Y \| N<br>– 기본값 : Y |
| AS_WTHR_CNT | TAS 인스턴스를 기동할 때 생성할 리밸런스 스레드 수를 설정한다. 리밸런스 작업 또는 리싱크로나이제이션 작업을 담당한다.<br>– 타입 : 정수<br>– 설정 가능한 값 : 양수<br>– 기본값 : 10 |

# 2.2. TAS 인스턴스 관리

본 절에서는 TAS 인스턴스를 관리하는 방법에 대해서 설명한다.

## 2.2.1. 기동 전 준비 사항

- TB_SID, TB_HOME 환경변수를 설정한다.
- tbsql을 사용한 접속을 위해 $TB_HOME/client/config/tbdsn.tbr 파일에 TAS 접속 정보를 설정한다.
- TAS 인스턴스 설정을 위한 tip 파일을 작성한다.
- TAS 사용을 위해서 최소 한 개 이상의 디스크 장치가 필요하다.
- TAS 클러스터링 기능을 사용하려면 최소 한 개 이상의 공유 디스크가 필요하다.

## 2.2.2. TAS 기동

본 절에서는 TAS 인스턴스를 기동하는데 필요한 환경설정을 예제 파일로 설명한다.

예제의 tip 파일을 사용하는 TAS 인스턴스는 "/devs/disk*"에 해당하는 모든 디스크 장치들을 TAS 디스크로 사용한다.

**[예 2.1] TAS 인스턴스 tip 파일 예제**

```
LISTENER_PORT=9620
MAX_SESSION_COUNT=100

MEMORY_TARGET=2G
TOTAL_SHM_SIZE=1G

INSTANCE_TYPE=AS
AS_DISKSTRING="/devs/disk*"
```

TAS 인스턴스를 최초로 기동할 때 **NOMOUNT** 모드로 기동해야 한다. 이는 TAS 설치를 위하여 최초의 디스크 스페이스를 생성하기 위함이다.

```
[TB_SID=as]$ tbboot nomount
```

NOMOUNT 모드로 기동한 TAS 인스턴스에 tbsql로 접속한 후 CREATE DISKSPACE SQL 문으로 기본 디스크 스페이스를 생성한다.

아래의 DDL 구문은 CREATE DISKSPACE 예제이다.

```
$ CREATE DISKSPACE ds0 NORMAL REDUNDANCY
FAILGROUP fg1 DISK
'/devs/disk101' NAME disk101,
'/devs/disk102' NAME disk102,
'/devs/disk103' NAME disk103,
'/devs/disk104' NAME disk104
FAILGROUP fg2 DISK
'/devs/disk201' NAME disk201,
'/devs/disk202' NAME disk202,
'/devs/disk203' NAME disk203,
'/devs/disk204' NAME disk204
ATTRIBUTE 'AU_SIZE'='4M';
```

기본 디스크 스페이스 생성을 완료하면 TAS 인스턴스가 자동으로 종료된다. 기본 디스크 스페이스가 있으면 TAS 인스턴스를 NORMAL 모드로 기동할 수 있으며, 이 상태에서 디스크 스페이스를 추가로 생성해도 인스턴스가 종료되지 않는다. 디스크 스페이스 생성에 대한 자세한 내용은 “3.2. 디스크 스페이스 생성”을 참고한다.

다음 예제는 TAS 인스턴스 2개를 클러스터로 기동하는데 필요한 인스턴스별 tip 파일을 보여준다. 이 tip 파일을 사용하는 TAS 인스턴스들은 "/devs/disk*"에 해당하는 모든 공유 디스크 장치들을 TAS 디스크로 사용한다. CM_PORT에는 CM_TIP 파일의 CM_UI_PORT에 등록한 포트를 입력해야 한다.

**[예 2.2] TAS 인스턴스 클러스터링 tip 파일 예제 : <<as0.tip>>**

```
LISTENER_PORT=9620
MAX_SESSION_COUNT=100

MEMORY_TARGET=2G
TOTAL_SHM_SIZE=1G

INSTANCE_TYPE=AS
AS_DISKSTRING="/devs/disk*"

CLUSTER_DATABASE=Y
LOCAL_CLUSTER_ADDR=192.168.1.1
LOCAL_CLUSTER_PORT=20000
CM_CLUSTER_MODE=ACTIVE_SHARED
CM_PORT=20005
THREAD=0
```

**[예 2.3] TAS 인스턴스 클러스터링 tip 파일 예제 : <<cm0.tip>>**

```
CM_NAME=cm0
CM_UI_PORT=20005
CM_RESOUCE_FILE=/home/tibero/config/cm0_res.crf
```

**[예 2.4] TAS 인스턴스 클러스터링 tip 파일 예제 : <<as1.tip>>**

```
LISTENER_PORT=9620
MAX_SESSION_COUNT=100

MEMORY_TARGET=2G
TOTAL_SHM_SIZE=1G

INSTANCE_TYPE=AS
AS_DISKSTRING="/devs/disk*"

CLUSTER_DATABASE=Y
LOCAL_CLUSTER_ADDR=192.168.1.2
LOCAL_CLUSTER_PORT=20000
CM_CLUSTER_MODE=ACTIVE_SHARED
CM_PORT=20005
THREAD=1
```

**[예 2.5] TAS 인스턴스 클러스터링 tip 파일 예제 : <<cm1.tip>>**

```
CM_NAME=cm1
CM_UI_PORT=20005
CM_RESOUCE_FILE=/home/tibero/config/cm1_res.crf
```

TAS 인스턴스들을 클러스터로 기동하기 전에 먼저 하나의 TAS 인스턴스를 NOMOUNT 모드로 기동한 후 기본 디스크 스페이스를 생성해야 한다. 기본 디스크 스페이스를 생성한 후 TAS 인스턴스들의 클러스터링을 위한 TBCM을 기동하고 클러스터링을 위한 리소스를 아래 예제와 같이 추가한다.

클러스터 리소스를 추가할 때 cfile attribute의 경우 tip 파일 AS_DISKSTRING에 설정한 디스크 경로를 입력하되 앞에 '+'를 붙여주어야 TAS용 경로로 인식한다. as 리소스를 추가할 때에는 envfile attribute에 [예 2.6]과 같이 AS 바이너리를 실행하기 위해 필요한 환경설정용 파일을 미리 생성하고 이 파일의 경로를 입력해주어야 한다. name attribute는 가동할 인스턴스의 TB_SID와 같아야 한다.

```
[CM_SID=cm0]$ tbcm -b # TBCM 기동
[CM_SID=cm0]$ cmrctl add network --name net0 --ipaddr 192.168.1.1 --portno 20010
[CM_SID=cm0]$ cmrctl add cluster --name cls0 --incnet net0 --cfile "+/devs/disk*"
[CM_SID=cm0]$ cmrctl start cluster --name cls0
[CM_SID=cm0]$ cmrctl add service --type as --name tas --cname cls0
[CM_SID=cm0]$ cmrctl add as --name as0 --svcname tas --envfile
"$TB_HOME/as0.profile" --dbhome "$TB_HOME"
```

**[예 2.6] as 리소스 envfile attribute 환경설정용 파일 예제 : <<as0.profile>>**

```
[CM_SID=cm0]$ cat $TB_HOME/as0.profile
export TB_SID=as0
```

---

**참고**

1. TBCM(Tibero Cluster Manager)에 대한 자세한 설명은 "Tibero 관리자 안내서"를 참고한다.

2. Cluster Manager가 필요한 이유는 가용성을 높이고 관리의 편의를 지원하기 위함이다. 이를 이용하면 여러 Active Storage 인스턴스 중 일부가 다운되더라도 지속적인 운영이 가능하다.

---

이제 TB_SID가 as0인 TAS 인스턴스를 기동할 수 있다.

```
[CM_SID=cm0]$ cmrctl start as --name as0 # 또는 [TB_SID=as0]$ tbboot
[CM_SID=cm0]$ cmrctl show # 리소스 정보 확인
```

TB_SID가 as1인 TAS 인스턴스를 기동하기 전에, 해당 인스턴스의 REDO 스레드를 추가해야 한다.

```
$ tbsql sys/tibero@as0

tbSQL 7

TmaxData Corporation Copyright (c) 2008-. All rights reserved.
```

```
Connected to Tibero.

SQL> ALTER DISKSPACE ds0 ADD THREAD 1;

Diskspace altered.
```

여기서 ds0은 위에서 생성한 기본 디스크 스페이스의 이름이며, REDO 스레드 번호 1은 TB_SID가 as1인 TAS 인스턴스의 tip 파일 [예 2.4]에 있는 THREAD 초기화 파라미터 값과 동일해야 한다.

또한 as0을 기동할 때와 동일한 방식으로 TBCM을 기동하고 클러스터링을 위한 리소스를 아래와 같이 추가해주어야 한다. cm0 설정 때 추가해 주었던 서비스 리소스는 추가할 필요가 없다.

```
[CM_SID=cm1]$ tbcm -b # TBCM 기동
[CM_SID=cm1]$ cmrctl add network --name net1 --ipaddr 192.168.1.2 --portno 20010
[CM_SID=cm1]$ cmrctl add cluster --name cls0 --incnet net1 --cfile "+/devs/disk*"
[CM_SID=cm1]$ cmrctl start cluster --name cls0
[CM_SID=cm1]$ cmrctl add as --name as1 --svcname tas --envfile
"$TB_HOME/as1.profile" --dbhome "$TB_HOME"
```

**[예 2.7] as 리소스 envfile attribute 환경설정용 파일 예제 : <<as1.profile>>**

```
[CM_SID=cm1]$ cat $TB_HOME/as1.profile
export TB_SID=as1
```

이제 TB_SID가 as1인 TAS 인스턴스를 기동할 수 있다.

```
[CM_SID=cm1]$ cmrctl start as --name as1 # 또는 [TB_SID=as1]$ tbboot
[CM_SID=cm1]$ cmrctl show # 리소스 정보 확인
```

## 2.2.3. TAS 종료

종료할 TAS 인스턴스에 대해 **tbdown** 명령을 실행한다. 이때, 해당 TAS 인스턴스를 사용중인 모든 Tibero 인스턴스들이 같이 종료된다.

---

**참고**

Tibero 인스턴스가 사용 중인 TAS 인스턴스를 먼저 강제종료할 경우 Tibero 인스턴스가 비정상 종료하므로 가급적 해당 Tibero 인스턴스를 먼저 종료한 후 TAS 인스턴스를 종료한다.

---

## 2.2.4. TAS 바이너리 교체

이미 TAS 인스턴스를 구성해둔 상태에서 바이너리를 교체할 경우 기존 버전과 신규 버전 사이의 호환성 문제를 해소하기 위해 아래와 같은 동작을 수행해야 한다.

```
$ tbsql sys/tibero@as0

tbSQL 7

TmaxData Corporation Copyright (c) 2008-. All rights reserved.

Connected to Tibero.

SQL> ALTER DISKSPACE ds0 CHECK VERSION;

Diskspace altered.
```

여기서 ds0은 기본 디스크 스페이스의 이름이다.

## 2.3. Tibero 인스턴스 구성

Tibero 인스턴스가 TAS의 디스크 스페이스를 사용하려면 다음 초기화 파라미터를 Tibero의 tip 파일에 설정한다.

| 파라미터 | 설명 |
|---|---|
| USE_ACTIVE_STORAGE | Active Storage 파일 사용 여부를 결정한다. 'Y'로 설정한다.<br>– 타입 : 참, 거짓 값<br>– 설정 가능한 값 : Y \| N<br>– 기본값 : N |
| AS_ADDR | Tibero 인스턴스가 사용할 TAS 인스턴스의 IP 주소를 설정한다.<br>– 타입 : 문자열<br>– 설정 가능한 값 : IP 주소 문자열<br>– 기본값 : "127.0.0.1" |
| AS_PORT | Tibero 인스턴스가 사용할 TAS 인스턴스의 'LISTENER_PORT' 값으로 설정한다.<br>– 타입 : 정수<br>– 설정 가능한 값 : 1024 ~ 65535<br>– 기본값 : 0 |

다음은 TAS 인스턴스를 사용하는 Tibero 인스턴스를 기동하는데 필요한 tip 파일을 보여준다. 이 tip 파일을 사용하는 Tibero 인스턴스는 디스크 스페이스 ds0에 컨트롤 파일을 저장한다.

**[예 2.8] TAS 인스턴스를 사용하는 Tibero 인스턴스 tip 파일 예제**

```
DB_NAME=tibero
CONTROL_FILES="+DS0/c1.ctl"

LISTENER_PORT=8629
MAX_SESSION_COUNT=20

TOTAL_SHM_SIZE=1G
MEMORY_TARGET=2G

USE_ACTIVE_STORAGE=Y
AS_PORT=9620
```

DB_CREATE_FILE_DEST 초기화 파라미터를 '+{*디스크 스페이스 이름*}'으로 설정할 경우 LOG_ARCHIVE_DEST, PSM_SHLIB_DIR, JAVA_CLASS_PATH 초기화 파라미터들을 직접 설정하지 않으면 기본값이 DB_CREATE_FILE_DEST의 하위 디렉터리로 설정된다. 경로를 설정해 준 경우 파일들이 해당 디렉터리 아래에 생성된다.

```
DB_CREATE_FILE_DEST="+DS0"
LOG_ARCHIVE_DEST="/home/tibero/database/tibero/archive"
PSM_SHLIB_DIR="/home/tibero/database/tibero/psm"
JAVA_CLASS_PATH="/home/tibero/database/tibero/java"
```

다음 예제는 TAS 파일을 사용하는 Tibero 인스턴스의 데이터베이스 생성을 보여준다. 데이터베이스를 구성하는 파일들 중 디스크 스페이스 ds0에 저장할 파일들의 경로를 '+DS0/...' 형태로 지정한다.

**[예 2.9] TAS 파일을 사용하는 Tibero 인스턴스의 CREATE DATABASE 예제**

```
CREATE DATABASE "tibero"
    USER SYS IDENTIFIED BY tibero
    MAXINSTANCES 8
    MAXDATAFILES 100
    CHARACTER SET MSWIN949
    LOGFILE GROUP 1 '+DS0/log0001.log' SIZE 100M,
            GROUP 2 '+DS0/log0002.log' SIZE 100M,
            GROUP 3 '+DS0/log0003.log' SIZE 100M
    MAXLOGGROUPS 255
    MAXLOGMEMBERS 8
    NOARCHIVELOG
    DATAFILE '+DS0/system001.dtf' SIZE 100M
            AUTOEXTEND ON NEXT 100M MAXSIZE UNLIMITED
    SYSSUB DATAFILE '+DS0/syssub001.dtf' SIZE 100M
```

```
        AUTOEXTEND ON NEXT 100M
DEFAULT TEMPORARY TABLESPACE tmp
        TEMPFILE '+DS0/temp001.dtf' SIZE 100M
        AUTOEXTEND ON NEXT 100M MAXSIZE UNLIMITED
        EXTENT MANAGEMENT LOCAL AUTOALLOCATE
UNDO TABLESPACE undo00
        DATAFILE '+DS0/undo001.dtf' SIZE 100M
        AUTOEXTEND ON NEXT 100M
        EXTENT MANAGEMENT LOCAL AUTOALLOCATE
DEFAULT TABLESPACE usr
        DATAFILE '+DS0/usr001.dtf' SIZE 100M
        AUTOEXTEND ON NEXT 100M MAXSIZE UNLIMITED
        EXTENT MANAGEMENT LOCAL AUTOALLOCATE;
```

**참고**

Tibero 인스턴스의 초기화 파라미터 설정은 "Tibero 참조 안내서", 데이터베이스 생성 구문은 "Tibero SQL 참조 안내서"를 참고한다.

# 제3장 TAS 디스크 스페이스 관리

본 장에서는 TAS 디스크 스페이스를 관리하는 방법에 대해 설명한다.

## 3.1. 디스크 스페이스 속성

디스크 스페이스 속성은 디스크 스페이스 생성 또는 수정할 때 설정할 수 있다.

디스크 스페이스를 생성하는 경우 **AU_SIZE** 속성을 설정한다. 할당 단위와 익스텐트에 대한 자세한 정보는 “제1장 TAS 소개”의 "할당 단위"와 "익스텐트(Extent)"를 참고한다. 디스크 스페이스에 AU_SIZE를 설정하는 방법은 [예 3.1]를 참고한다.

## 3.2. 디스크 스페이스 생성

본 절에서는 디스크 스페이스 생성 방법을 설명한다.

디스크 스페이스에서 사용할 수 있는 디바이스는 다음과 같다.

- Character Device

  물리 디스크 디바이스를 Charcater Device로 만들어 사용할 수 있다.

- Block Device

  Linux OS에서는 물리 디스크 디바이스를 Block Device로 만들어 사용할 수 있다. Linux 외의 OS에서는 Block device를 사용하는 경우 Direct I/O가 보장되지 않으므로 지원하지 않는다.

### 3.2.1. CREATE DISKSPACE SQL 문 사용

CREATE DISKSPACE SQL 문을 사용하여 디스크 스페이스를 생성할 수 있다.

디스크 스페이스를 생성할 때 다음의 사항들을 고려해야 한다. 자세한 내용은 “1.2.3. 미러링과 실패 그룹”을 참고한다.

- 중복 레벨

  TAS의 미러링 기능을 사용하기 위해 디스크 중복 레벨을 설정한다.

- 실패 그룹 정의

실패 그룹을 설정할 수 있다. 실패 그룹을 나누게 되면, 하나의 실패 그룹에 문제가 발생해도 서비스를 유지할 수 있다. 실패 그룹을 설정하게 되면 미러링을 사용할 때 복제본을 서로 다른 실패 그룹에 만든다. 실패 그룹을 설정하지 않으면 각 디스크가 실패 그룹이 된다.

- 디스크 스페이스 속성

  할당 단위 크기 등의 속성을 설정할 수 있다.

TAS는 각 디스크 장치의 크기를 자동으로 인식한다. 만약 디스크 장치의 크기를 자동으로 인식하지 못했거나 디스크 장치의 사용 공간을 제한하려면, 디스크 스페이스를 생성할 때 디스크에 SIZE 절을 사용할 수 있다. 또한 NAME 절을 사용하여 디스크에 이름을 부여할 수 있다.

```
...
'/devs/disk101' NAME disk101 SIZE 1TB,
...
```

디스크 크기와 이름 정보는 V$AS_DISK 뷰로 확인할 수 있다.

## 3.2.2. 예제

다음은 중복 레벨이 NORMAL이고 실패 그룹 fg1, fg2가 있는 디스크 스페이스 ds0을 만든다. 이때 AS_DISKSTRING 초기화 파라미터를 "/devs/disk*"로 설정하여 발견한 디스크 장치들을 TAS 디스크로 사용한다.

**[예 3.1] 디스크 스페이스 ds0 생성**

```
CREATE DISKSPACE ds0 NORMAL REDUNDANCY
  FAILGROUP fg1 DISK
    '/devs/disk101' NAME disk101,
    '/devs/disk102' NAME disk102,
    '/devs/disk103' NAME disk103,
    '/devs/disk104' NAME disk104
  FAILGROUP fg2 DISK
    '/devs/disk201' NAME disk201,
    '/devs/disk202' NAME disk202,
    '/devs/disk203' NAME disk203,
    '/devs/disk204' NAME disk204
  ATTRIBUTE 'AU_SIZE'='4M';
```

위 예제에서 NAME 절을 사용해서 각 디스크에 이름을 부여했다. 직접 디스크 이름을 부여하지 않으면 "{*디스크 스페이스 이름*}_####" 형태의 기본 이름을 부여하며, '####'은 디스크 스페이스 내에서의 디스크 번호이다. 또한 **AU_SIZE** 속성으로 할당 단위의 크기를 4MB로 설정했다. 디스크 스페이스를 생성할 때 설정한 할당 단위 크기는 V$AS_DISKSPACE 뷰로 확인할 수 있다.

# 3.3. 디스크 스페이스 수정

ALTER DISKSPACE 문으로 디스크 스페이스 설정을 수정할 수 있다. 디스크 스페이스 서비스를 제공하면서 디스크를 디스크 스페이스에 추가하거나 디스크 스페이스로부터 제거할 수 있으며, **디스크 스페이스 리밸런싱**을 수행할 수 있다.

디스크 추가/제거를 수행해도 바로 디스크 스페이스에 반영하는 것은 아니다. 이는 기존 디스크들의 데이터를 추가할 디스크에 재배치해야 모든 디스크를 공평하게 사용할 수 있으며, 제거할 디스크의 데이터를 남은 디스크들에 재배치해야 데이터 손실을 방지할 수 있기 때문이다. 이와 같이 디스크 스페이스에 있는 데이터를 디스크에 공평하게 재배치하는 과정을 리밸런싱이라고 한다. 디스크 추가/제거는 디스크 스페이스 리밸런싱을 통해 디스크 스페이스에 반영된다.

## 3.3.1. 디스크 추가

ALTER DISKSPACE 문의 ADD 절을 사용해서 디스크를 디스크 스페이스에 추가할 수 있다. ADD 절에는 CREATE DISKSPACE 문에서 사용했던 실패 그룹과 디스크 관련 절을 그대로 사용한다.

다음 예제는 디스크 스페이스 ds0에 디스크를 추가하는 SQL 문을 보여준다. 이 예제에서는 새로운 디스크 장치 '/devs/disk105'와 '/devs/disk106'을 디스크 스페이스 ds0의 실패 그룹 fg1에 추가한다. 또한 REBALANCE 절로 디스크 추가와 동시에 디스크 스페이스 리밸런싱을 수행하도록 한다.

**[예 3.2] 디스크 스페이스 ds0에 디스크 추가**

```
ALTER DISKSPACE ds0
    ADD FAILGROUP fg1 DISK
        '/devs/disk105' NAME disk105,
        '/devs/disk106' NAME disk106
    REBALANCE;
```

## 3.3.2. 디스크 제거

ALTER DISKSPACE 문의 DROP 절을 사용해서 디스크를 디스크 스페이스에서 제거할 수 있다.

다음은 디스크 스페이스 ds0에서 디스크를 제거하는 SQL 문이다. 이 예제에서는 디스크 disk105와 disk106을 디스크 스페이스 ds0에서 제거한다.

**[예 3.3] 디스크 스페이스 ds0에서 디스크 제거**

```
ALTER DISKSPACE ds0 DROP DISK disk105;
ALTER DISKSPACE ds0 DROP DISK disk106;
```

디스크 고장 등으로 인해 디스크 상태가 FAIL일 때는 FORCE 옵션을 이용해 디스크를 제거할 수 있다.

다음 예제는 FAIL 상태인 디스크를 제거하는 SQL 문이다.

**[예 3.4] 디스크 스페이스 ds0에서 FAIL 상태인 디스크를 제거**

```
ALTER DISKSPACE ds0 DROP DISK disk105 FORCE;
```

디스크 스페이스에서 특정 실패 그룹에 속한 모든 디스크들을 제거할 수 있다.

다음 예제는 디스크 스페이스 ds0의 실패 그룹 fg1에 속한 모든 디스크들을 제거하는 SQL 문이다.

**[예 3.5] 디스크 스페이스 ds0에서 실패 그룹 fg1의 디스크들 제거**

```
ALTER DISKSPACE ds0 DROP DISKS IN FAILGROUP fg1;
```

디스크 추가/제거를 한 문장의 SQL로 수행할 수 있다.

다음 예제는 디스크 스페이스 ds0에서 디스크 disk204를 제거하고, 디스크 장치 '/devs/disk205'를 디스크 스페이스 ds0의 실패 그룹 fg2에 추가한다.

**[예 3.6] 디스크 스페이스 ds0에 디스크 추가/제거**

```
ALTER DISKSPACE ds0
    DROP DISK disk204
    ADD FAILGROUP fg2 DISK '/devs/disk205' NAME disk205;
```

## 3.3.3. 디스크 크기 변경

ALTER DISKSPACE 문의 RESIZE 절을 통해서 디스크의 크기를 변경할 수 있다.

변경할 크기는 실제 디스크의 물리적 크기보다 작거나 같아야 하며, 아직 기존의 크기보다 더 작은 크기로의 변경은 지원하고 있지 않으므로 기존의 디스크 크기보다는 큰 값을 입력해야만 한다.

**RESIZE** 명령을 수행하면 디스크의 상태는 PREPARE_RESIZE 상태로 변경되며, 이 상태에서는 아직 디스크의 크기가 변경 이전의 크기로 유지된다. 디스크의 크기를 완전히 변경하기 위해서는 **REBALANCE** 명령을 수행해야하며, 크기 변경을 취소하고 디스크의 상태를 원래대로 되돌리려면 **UNPREPARE** 명령을 수행해 준비상태를 취소해야 한다.

다음 예제는 디스크 스페이스 ds0에서 disk105의 크기를 512G로 변경하는 명령어이다.

**[예 3.7] 디스크 스페이스 ds0에서 디스크의 크기를 512G로 변경**

```
ALTER DISKSPACE ds0 RESIZE DISK disk105 size 512G;
```

하나의 디스크 뿐만 아니라 여러 디스크의 크기를 한 번에 변경하는 것도 가능하다.

다음 예제는 디스크 스페이스 ds0에서 실패 그룹 fg1에 포함돼 있는 모든 디스크의 크기를 512G로 변경하는 명령어이다.

**[예 3.8] 디스크 스페이스 ds0에서 실패 그룹 fg1에 속한 모든 디스크의 크기를 512G로 변경**

```
ALTER DISKSPACE ds0 RESIZE DISKS IN FAILGROUP fg1 size 512G;
```

다음 예제는 디스크 스페이스 ds0에서 모든 디스크의 크기를 512G로 변경하는 명령어이다.

**[예 3.9] 디스크 스페이스 ds0에서 모든 디스크의 크기를 512G로 변경**

```
ALTER DISKSPACE ds0 RESIZE ALL size 512G;
```

## 3.3.4. 디스크 추가/제거 준비상태 취소

ALTER DISKSPACE 문의 UNPREPARE 절을 통해서 디스크 추가/제거 준비상태를 취소할 수 있다.

PREPARE_DROP과 PREPARE_RESIZE 상태의 디스크는 ONLINE으로, PREPARE_FORCE 상태의 디스크는 FAIL로, PREPARE_ADD 상태의 디스크는 NOT_USED로 되돌려 준다. PREPARE 상태는 **REBALANCE** 명령을 내리기 전 대기 상태를 의미하므로 **REBALANCE** 명령이 내려진 이후에는 **UNPREPARE** 명령어로 복구하는 것이 불가능하다.

다음 예제는 디스크 스페이스 ds0에서 PREPARE_DROP 상태의 disk105를 ONLINE으로 복구하는 명령어이다.

**[예 3.10] 디스크 스페이스 ds0에 PREPARE_DROP상태의 디스크를 ONLINE으로 복구**

```
ALTER DISKSPACE ds0 UNPREPARE DISK disk105;
```

## 3.3.5. 디스크 스페이스 리밸런싱

디스크 추가/제거와 동시에 리밸런싱을 수행하지 않은 경우와 같이 리밸런싱이 수동으로 필요한 경우 다음과 같이 리밸런싱을 수동으로 수행할 수 있다.

다음 예제는 디스크 스페이스 ds0의 리밸런싱을 수행하는 SQL 문을 보여준다.

**[예 3.11] 디스크 스페이스 ds0 리밸런싱 (1)**

```
ALTER DISKSPACE ds0 REBALANCE;
```

리밸런스 SQL 문에 WAIT를 붙여주면 리밸런싱이 끝나기를 기다릴 수 있다.

**[예 3.12] 디스크 스페이스 ds0 리밸런싱 (2)**

```
ALTER DISKSPACE ds0 REBALANCE WAIT;
```

리밸런스 SQL 문에 FORCE를 붙여주면 추가/삭제되는 디스크가 없더라도 강제로 리밸런스 작업을 수행한다. 이 옵션을 사용하는 경우 이동시키는 익스텐트의 양이 많아 수행 시간이 길어진다.

**V$AS_DISK** 뷰에서 디스크의 상태를 조회함으로써 리밸런싱이 완료되었는지 확인할 수 있으며, **V$AS_OPERATION** 뷰에서 진행 중인 리밸런스 작업을 확인할 수 있다.

### 3.3.6. 파일 삭제

ALTER DISKSPACE 문의 DROP 절을 사용해서 디스크 스페이스에 존재하는 파일을 삭제할 수 있다.

다음 예제는 디스크 스페이스 ds0에서 파일을 삭제하는 SQL 문을 보여준다. 이 예제에서는 파일의 경로가 +DS0/file000.dtf인 파일을 디스크 스페이스 ds0에서 삭제한다.

**[예 3.13] 디스크 스페이스 ds0에서 파일 삭제**

```
ALTER DISKSPACE ds0 DROP FILE '+DS0/file000.dtf';
```

또한 파일 경로를 콤마(,)로 구분하여 여러 개의 파일을 동시에 디스크 스페이스에서 삭제할 수 있다.

다음 예제는 디스크 스페이스 ds0에서 여러 파일을 한 번에 삭제하는 SQL 문을 보여준다. 이 예제에서는 파일의 경로가 +DS0/file000.dtf인 파일과 +DS0/file001.dtf인 파일을 디스크 스페이스 ds0에서 삭제한다.

**[예 3.14] 디스크 스페이스 ds0에서 여러 파일 삭제**

```
ALTER DISKSPACE ds0 DROP FILE '+DS0/file000.dtf', '+DS0/file001.dtf';
```

## 3.4. 디스크 스페이스 삭제

DROP DISKSPACE 문을 사용해서 디스크 스페이스를 삭제할 수 있다. 다음 예제는 디스크 스페이스 ds0을 삭제하는 SQL 문을 보여준다.

**[예 3.15] 디스크 스페이스 ds0 삭제**

```
DROP DISKSPACE ds0;
```

## 3.5. 디스크 장애 관리

TAS는 미러링과 실패 그룹을 활용해 디스크 장애 시에도 가용성을 보장하며, 디스크가 장애 상태에서 복구되면 리싱크 동작을 수행해 장애가 발생한 동안 수정된 데이터의 정합성을 맞춰준다.

### 3.5.1. 디스크 장애 상황에서의 가용성 보장

TAS는 데이터의 복사본을 만들어 저장하는 미러링 기능을 제공하며, 각각의 복사본은 서로 다른 실패 그룹에 저장된다. 따라서 디스크 스페이스의 중복 레벨을 NORMAL 또는 HIGH로 설정했을 경우 특정 실패 그룹에 속한 디스크 혹은 하나(중복 레벨이 HIGH인 경우 두 개까지)의 실패 그룹 전체에 장애가 발생하더라도 다른 실패 그룹에 저장되어 있는 복사본을 사용해 서비스의 가용성을 보장한다.

### 3.5.2. 디스크 장애 후 복구 상황에서의 동작

TAS는 주기적인 DISK 모니터링을 통해 DISK의 상태를 관리하며, DISK의 현재 상태는 “4.1.2. V$AS_DISK”를 통해 확인할 수 있다.

디스크에 장애가 발생하면 디스크의 상태는 FAIL 상태로 전환된다. FAIL 상태의 디스크에 대한 I/O는 별도의 메타데이터에 기록 후 건너뛰며 이렇게 건너뛴 I/O는 디스크가 다시 복구되었을 때 리싱크 절차를 거쳐 반영된다.

### 3.5.3. 리싱크

리싱크는 디스크가 장애 상태에 있는동안 수정된 데이터의 정합성을 맞춰주기 위해 수행되는 동작이며, 중복 레벨을 NORMAL 또는 HIGH로 설정한 경우에만 수행할 수 있다.

TAS는 미러링 기능을 제공하기 때문에 디스크에 장애가 발생하더라도 다른 실패 그룹에 위치한 복사본에는 I/O를 수행할 수 있다. 따라서 장애 디스크가 복구되면 해당 디스크에 위치한 데이터들은 다른 실패 그룹에 위치한 복사본을 통해 최신의 데이터로 복원이 가능하며, 이 동작을 리싱크라고 한다.

### 3.5.4. 디스크 장애 후 복구되지 않는 상황에서의 동작

디스크에 장애가 발생해 디스크의 상태가 FAIL로 바뀐 채로 계속 운영하면 장애 디스크에 위치한 데이터 복사본은 사용할 수 없기 때문에 안정성이 떨어진다. 따라서 이 상태가 장시간 지속되지 않도록 TAS에서는 디스크의 FAIL 상태가 일정 시간 이상 지속되면 해당 디스크를 자동으로 제거하고 리밸런싱을 통해 복사본을 다른 디스크로 이동시킴으로써 미러링을 유지한다.

디스크가 FAIL 상태로 바뀐 이후 자동으로 제거되기까지의 시간은 아래의 파라미터를 통해 설정할 수 있다.

| 파라미터 | 설명 |
|---|---|
| AS_DISK_REPAIR_WAIT_TIME | TAS가 디스크의 상태를 FAIL로 바꾼 이후 해당 디스크를 자동으로 제거하기까지의 대기 시간으로, 0으로 설정하는 경우 자동으로 제거하지 않는다. (단위: 분, 기본값: 0) |

# 제4장 TAS 정보 조회

본 장에서는 동적 뷰를 사용해서 TAS 정보를 조회하는 방법에 대해 설명한다.

## 4.1. TAS 디스크 스페이스 정보 관련 뷰

TAS 디스크 스페이스 정보를 보기위해 다음의 뷰들을 사용할 수 있다.

해당 뷰들은 TAS 인스턴스에서만 조회할 수 있다.

| 뷰 | 설명 |
|---|---|
| V$AS_ALIAS | TAS 인스턴스가 마운트한 모든 디스크 스페이스에 있는 모든 별칭이다. |
| V$AS_DISK | TAS 인스턴스가 발견한 모든 디스크이다. |
| V$AS_DISKSPACE | TAS 인스턴스가 발견한 모든 디스크 스페이스이다. |
| V$AS_OPERATION | TAS가 진행 중인 모든 작업을 보여준다. |
| V$AS_EXTENT_MAP | TAS 인스턴스가 관리하고 있는 모든 익스텐트들의 물리적인 위치 정보를 담고 있다. |
| V$AS_FILE | TAS 인스턴스가 마운트한 모든 디스크 스페이스에 있는 모든 파일이다. |

V$AS_* 뷰들을 조회할 때 디스크 스페이스 번호가 이전과 동일하지 않을 수 있다. 이는 디스크 스페이스를 마운트할 때 디스크 스페이스 번호를 부여하기 때문이다.

### 4.1.1. V$AS_ALIAS

**V$AS_ALIAS** 뷰는 TAS 인스턴스가 마운트한 모든 디스크 스페이스에 있는 모든 별칭을 보여준다.

| 컬럼 | 데이터 타입 | 설명 |
|---|---|---|
| NAME | VARCHAR(48) | 별칭이다. |
| DISKSPACE_NUMBER | NUMBER | 별칭이 속한 디스크 스페이스 번호이다. |
| FILE_NUMBER | NUMBER | 별칭에 해당하는 파일 번호이다. |
| FILE_INCARNATION | NUMBER | 별칭에 해당하는 파일의 인카네이션 번호이다. |
| ALIAS_INCARNATION | NUMBER | 별칭의 인카네이션 번호이다. |

## 4.1.2. V$AS_DISK

**V$AS_DISK** 뷰는 TAS 인스턴스가 발견한 모든 디스크를 보여준다.

| 컬럼 | 데이터 타입 | 설명 |
|---|---|---|
| DISKSPACE_NUMBER | NUMBER | 디스크가 속한 디스크 스페이스 번호이다. |
| DISK_NUMBER | NUMBER | 디스크 스페이스 내에서의 디스크 번호이다. |
| STATE | VARCHAR(16) | 디스크의 상태를 나타낸다.<br>– ONLINE : 디스크가 사용 중이다.<br>– NOT_USED : 디스크 스페이스에 사용되지 않는 디스크를 나타낸다. 이 경우 디스크 스페이스 번호를 255로 표시한다.<br>– PREPARE_ADD : 디스크가 추가될 예정이다. 리밸런싱이 시작되기 전까지 실제로 디스크가 사용되지 않는다.<br>– PREPARE_DROP : 디스크가 제거될 예정이다. 리밸런싱이 완료되기 전까지는 ONLINE 상태와 마찬가지로 디스크가 사용된다.<br>– PREPARE_FORCE : 디스크가 제거될 예정이다. 리밸런싱 중에 실제로 디스크를 사용하지 않는다.<br>– PREPARE_RESIZE : 디스크의 크기가 변경될 예정이다. 리밸런싱 중에는 ONLINE 상태와 마찬가지로 디스크가 사용된다.<br>– ADDING : 리밸런싱을 통해 디스크가 추가되고 있는 중이다.<br>– DROPPING : 리밸런싱을 통해 디스크가 제거되고 있는 중이다.<br>– FORCING : 리밸런싱을 통해 디스크가 제거되고 있는 중이다.<br>– RESIZING : 리밸런싱을 통해 디스크의 크기가 변경되고 있는 중이다.<br>– SYNC : 일시적인 디스크 장애로 인해 반영되지 못한 변경사항을 최신의 상태로 동기화하는 중이다.<br>– FAIL : 디스크에 장애가 발생했거나 디스크를 강제로 오프라인 시킨 상태이다. |

| 컬럼 | 데이터 타입 | 설명 |
|---|---|---|
| OS_MB | NUMBER | OS가 인식하는 디스크 크기이다. |
| TOTAL_MB | NUMBER | 디스크 크기이다. |
| FREE_MB | NUMBER | 디스크에 남은 공간이다. |
| NAME | VARCHAR(48) | 디스크 이름이다. |
| FAILGROUP | VARCHAR(48) | 디스크가 속한 실패 그룹 이름이다. |
| PATH | VARCHAR(256) | 디스크 경로이다. |
| CREATE_DATE | DATE | 디스크가 디스크 스페이스에 추가된 날짜이다. |

### 4.1.3. V$AS_DISKSPACE

V$AS_DISKSPACE 뷰는 TAS 인스턴스가 발견한 모든 디스크 스페이스를 보여준다.

| 컬럼 | 데이터 타입 | 설명 |
|---|---|---|
| DISKSPACE_NUMBER | NUMBER | 디스크 스페이스에 부여한 번호이다. |
| NAME | VARCHAR(48) | 디스크 스페이스 이름이다. |
| SECTOR_SIZE | NUMBER | 물리적 블록 크기이다. (단위: Byte) |
| BLOCK_SIZE | NUMBER | TAS 메타 데이터 블록 크기이다. (단위: Byte) |
| ALLOCATION_UNIT_SIZE | NUMBER | 할당 단위 크기이다. (단위: Byte) |
| STATE | VARCHAR(16) | TAS 인스턴스에서 디스크 스페이스 상태이다.<br>– MOUNTING : 디스크 스페이스를 마운트하는 중이다.<br>– CREATING : 디스크 스페이스를 생성하는 중이다.<br>– MOUNT : 디스크 스페이스가 마운트된 상태이다. |
| TYPE | VARCHAR(8) | 디스크 스페이스의 중복 레벨이다.<br>– EXTERNAL : 디스크 스페이스의 사용자 파일을 중복 저장하지 않는다.<br>– NORMAL : 디스크 스페이스의 사용자 파일을 최소 두 개 이상의 FAILGROUP에 중복 저장한다.<br>– HIGH : 디스크 스페이스의 사용자 파일을 최소 세 개 이상의 FAILGROUP에 중복 저장한다. |
| TOTAL_MB | NUMBER | 디스크 스페이스 크기이다. |
| FREE_MB | NUMBER | 디스크 스페이스에 남은 공간이다. |

| 컬럼 | 데이터 타입 | 설명 |
|---|---|---|
| REQUIRED_MIRROR_FREE_MB | NUMBER | 디스크 장애 상황을 대비해 필요한 여분의 용량이다. |
| USABLE_FILE_MB | NUMBER | 미러링을 고려했을 때 디스크 스페이스의 사용 가능 용량이다. |

## 4.1.4. V$AS_OPERATION

**V$AS_OPERATION** 뷰는 TAS 인스턴스가 마운트한 모든 디스크 스페이스에 진행 중인 모든 작업을 보여준다.

| 컬럼 | 데이터 타입 | 설명 |
|---|---|---|
| DISKSPACE_NUMBER | NUMBER | 파일이 속한 디스크 스페이스 번호이다. |
| OPERATION | VARCHAR(16) | 현재 진행 중인 작업의 종류이다.<br>– REBALANCE : 리밸런스 중이다.<br>– SYNC : fail이었던 디스크를 sync 중이다.<br>– SCRUB : file scrub 중이다. |
| STATE | VARCHAR(8) | 작업의 현재 상태이다. 진행 중인 작업만 보여주므로 RUN 상태이다. |
| POWER | NUMBER | 현재 별도로 operation의 power를 설정하는 기능은 미지원 상태이다. 기본값으로 10이 나타난다. |
| JOB_DONE | NUMBER | 작업 완료된 extent 개수이다. |
| JOB_REMAIN | NUMBER | 앞으로 작업해야 하는 남아있는 extent 개수이다. |
| JOB_TOTAL | NUMBER | 작업의 전체 extent 개수이다. |
| EST_SPEED | NUMBER | 작업의 예상 속도이다. (단위: extent/s) |
| EST_REMAIN_TIME | VARCHAR(32) | 작업의 예상 남은 시간이다. (단위: s) |

다음은 V$AS_OPERATION으로 디스크 스페이스 ds0에서 진행 중인 rebalance 작업을 조회하는 예제이다.

**[예 4.1] 진행 중인 작업 보기**

```
SQL> SELECT diskspace_number, operation, job_remain, est_remain_time
        FROM V$AS_OPERATION;

DISKSPACE_NUMBER OPERATION        JOB_REMAIN EST_REMAIN_TIME
---------------- ---------------- ---------- --------------------------------
               0 REBALANCE                72                         30.6
```

```
1 row selected.
```

## 4.1.5. V$AS_EXTENT_MAP

**V$AS_EXTENT_MAP** 뷰는 TAS 인스턴스가 관리하고 있는 모든 익스텐트들의 물리적인 위치 정보를 담고 있다.

| 컬럼 | 데이터 타입 | 설명 |
|---|---|---|
| DISKSPACE_NUMBER | NUMBER | 파일이 속한 디스크 스페이스 번호이다. |
| FILE_NUMBER | NUMBER | 디스크 스페이스 내에서의 파일 번호이다. |
| EXTENT_NUMBER | NUMBER | 파일의 익스텐트 번호이다. |
| REDUN_NUMBER | NUMBER | 익스텐트의 중복 저장 번호이다. |
| EXTENT_SIZE | NUMBER | 익스텐트의 크기이다. (단위: AU) |
| DISK_NUMBER | NUMBER | 익스텐트가 저장된 디스크 번호이다. |
| AU_NUMBER | NUMBER | 익스텐트가 저장된 디스크에서 해당 익스텐트가 저장된 AU 위치이다. |

## 4.1.6. V$AS_FILE

**V$AS_FILE** 뷰는 TAS 인스턴스가 마운트한 모든 디스크 스페이스에 있는 모든 파일을 보여준다.

| 컬럼 | 데이터 타입 | 설명 |
|---|---|---|
| DISKSPACE_NUMBER | NUMBER | 파일이 속한 디스크 스페이스 번호이다. |
| FILE_NUMBER | NUMBER | 디스크 스페이스 내에서의 파일 번호이다. |
| INCARNATION | NUMBER | 파일의 인카네이션 번호이다. |
| BLOCK_SIZE | NUMBER | 파일의 블록 크기이다. (단위: Byte) |
| BYTES | NUMBER | 파일의 byte 수이다. |
| TYPE | VARCHAR(8) | 파일 타입이다.<br>– DATA : 일반 데이터 파일로 특정 타입의 파일을 제외한 모든 파일이 DATA 타입으로 취급된다.<br>– CTRL : 컨트롤 파일이다.<br>– CM : CM 파일이다.<br>– REDO : REDO 로그 파일이다.<br>– ARCH : ARCHIVE 로그 파일이다. |

| 컬럼 | 데이터 타입 | 설명 |
|---|---|---|
| | | – TEMP : 임시 파일이다. |
| REDUNDANCY | VARCHAR(8) | 파일의 중복 레벨이다.<br>– EXTERNAL : 파일을 중복 저장하지 않는다.<br>– NORMAL : 파일을 최소 두 개 이상의 FAILGROUP에 중복 저장한다.<br>– HIGH : 파일을 최소 세 개 이상의 FAILGROUP에 중복 저장한다. |
| STRIPED | VARCHAR(8) | 스트라이핑 타입이다.<br>– COARSE : 파일이 AU 크기로 스트라이핑되어 저장된다.<br>– FINE : 파일이 AU 크기의 8분의 1 크기로 스트라이핑되어 저장된다. |

다음은 V$AS_ALIAS와 V$AS_DISK에서 모든 디스크 스페이스에 있는 모든 파일의 별칭, 번호, 그리고 크기를 조회하는 예제이다.

**[예 4.2] 파일과 별칭 같이 보기**

```
SQL> SELECT name, a.file_number, bytes
        FROM V$AS_ALIAS a, V$AS_FILE f
     WHERE a.file_number = f.file_number
       AND a.diskspace_number = f.diskspace_number;

NAME                                              FILE_NUMBER      BYTES
------------------------------------------------- ----------- ----------
c1.ctl                                                    256    3145728
log001.log                                                257   52428800
log002.log                                                258   52428800
log003.log                                                259   52428800
system001.dtf                                             266   52428800
undo001.dtf                                               267   85983232
temp001.dtf                                               268    2097152
usr001.dtf                                                269    2097152

8 rows selected.
```

# 제5장 백업과 복구

본 장에서는 Tibero 복구 관리자(이하 RMGR)를 사용하여 TAS의 디스크 스페이스에 저장한 Tibero 데이터베이스 파일의 백업 및 복구 방법을 설명한다.

---

**참고**

RMGR에 대한 자세한 내용은 "Tibero 관리자 안내서"를 참고한다.

---

## 5.1. 백업

다음은 **tbrmgr** 명령을 사용해서 TAS 디스크 스페이스에 저장한 모든 데이터를 '/backupdir' 디렉터리에 백업하는 예이다.

**[예 5.1] TAS 디스크 스페이스에 저장한 데이터 백업**

```
tbrmgr backup -o /backupdir
```

## 5.2. 복구

다음은 **tbrmgr** 명령을 사용해서 TAS 디스크 스페이스에 모든 데이터를 '/backupdir' 디렉터리에 있는 백업본으로 복구하는 예이다.

**[예 5.2] TAS 디스크 스페이스에 저장한 데이터 복구**

```
tbrmgr recover -o /backupdir
```

# 제6장 커맨드라인 툴

본 장에서는 커맨드라인 툴 TBASCMD를 사용하여 디스크 스페이스에 저장된 파일을 조회하고 관리하는 방법을 설명한다.

## 6.1. 개요

TBASCMD는 사용자가 간단한 명령어를 통해서 디스크 스페이스의 정보를 조회하고 디스크 스페이스에 저장된 파일을 조회 및 관리할 수 있도록 해준다. 또한 로컬 디렉터리에 있는 파일을 디스크 스페이스에 저장하거나 디스크 스페이스에 있는 파일을 로컬 디렉터리에 저장할 수 있도록 해준다. 사용자가 명령어를 입력하면 TBASCMD는 TAS 인스턴스에 접속하여 필요한 정보 및 작업을 요청하게 된다.

## 6.2. TBASCMD 실행

TBASCMD를 실행할 때에 정보 및 작업을 요청할 TAS 인스턴스에 대한 접속 정보를 다음과 같이 인자로 주면 된다.

**[예 6.1] TBASCMD 실행 인자**

```
tbascmd (-p) {port}
tbascmd (-p) {port} -c "{cmd}"

$ tbascmd 52000
ASCMD 7

TmaxTibero Corporation Copyright (c) 2008-. All rights reserved.

ASCMD>
```

TBASCMD는 입력한 포트 번호로 로컬의 TAS 인스턴스에 접속하게 된다. 포트 정보는 필수적으로 입력해주어야 한다. 커맨드라인 툴을 실행하기 전에 접속할 TAS 인스턴스가 부팅되어 있어야 한다.

또한 하나의 명령어만 수행하고 TBASCMD를 종료하고자 할 경우 -c "{cmd}" 옵션을 추가하여 TBASCMD 실행과 동시에 명령어를 수행하고 즉시 종료할 수 있으며, 이 때 명령어는 큰따옴표(" ") 또는 작은따옴표(' ') 로 감싸서 입력해야 한다.

# 6.3. TBASCMD 명령어

TBASCMD를 통해 사용할 수 있는 명령어는 다음과 같다.

| 명령어 | 설명 |
|---|---|
| cd | 현재 경로를 변경한다. |
| du | 현재 경로에 존재하는 파일들이 차지하는 디스크 공간의 크기를 보여주거나 디스크 스페이스들의 용량을 조회한다. |
| exit | TBASCMD를 종료한다. |
| help | 명령어들에 대한 도움말을 조회한다. |
| ls | 현재 경로에 있는 모든파일의 목록을 보여주거나 디스크 스페이스의 목록을 조회한다. |
| lsds | 디스크 스페이스들에 대한 정보를 조회한다. |
| pwd | 현재 경로를 조회한다. |
| rm | TAS 파일을 제거한다. |
| cp | TAS 파일을 복사하여 새로운 TAS 파일을 생성한다. |
| cpfromlocal | 로컬 경로의 파일을 복사하여 TAS 파일을 생성한다. |
| cptolocal | TAS 파일을 복사하여 로컬 파일을 생성한다. |
| mkdir | 디렉터리를 생성한다. |
| mv | TAS 파일을 이동시킨다. |
| check | TAS 파일에 대한 정합성 검사를 실시한다. |

**참고**

7.2.1 릴리즈부터 mv, check 명령어를 지원하며 du 명령어는 현재 경로에 존재하는 디렉터리내 파일들이 차지하는 디스크 공간을 보여준다.

명령어에 파일이름이나 경로를 인수로 입력하는 경우에 그 경로를 절대 경로나 상대 경로로 입력할 수 있다.

- 절대 경로

  절대 경로는 경로명 전체를 의미하며 다음과 같이 '+'로 시작하고 디스크 스페이스 이름과 디스크 스페이스 내에서의 경로가 명시된다.

```
current path = +DS0
ASCMD +> cd +DS0/d1/sample
current path = +DS0/d1/sample
```

- 상대 경로

상대 경로는 입력할 전체 경로명 중에서 현재 경로로 지정되지 않은 나머지 경로를 의미하는 것으로 결과적으로 상대 경로는 다음과 같이 현재 경로와 조합되어 명령어를 실행하는 데에 사용된다.

```
current path = +DS0
ASCMD +> cd d1/sample
current path = +DS0/d1/sample
```

상대 경로를 사용하는 경우에 다음의 심볼을 사용할 수 있다.

| 심볼 | 설명 |
|---|---|
| "." | 현재 경로를 나타낸다. |
| ".." | 상위 경로(경로명에서 마지막 '/' 또는 현재 경로가 디스크 스페이스라면 최상위 경로인 '+')를 나타낸다. |

다음과 같이 "."과 ".."뒤에 추가 경로를 입력해줄 수 있다.

```
current path = +DS0/d1
ASCMD> cd ./sample
current path = +DS0/d1/sample
ASCMD> cd ../example
current path = +DS0/d1/example
```

## 6.3.1. cd

cd 명령어를 통해 현재 경로를 변경할 수 있다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
cd {path}
```

| 인자 | 설명 |
|---|---|
| path | 절대 경로 및 상대 경로로 입력될 수 있으며 존재하는 경로라면 입력한 경로로 현재 경로가 변경된다. |

## 6.3.2. du

du 명령어를 사용하여 지정한 경로에 존재하는 파일 또는 디렉터리가 차지하는 디스크 공간의 크기를 조회할 수 있다. 최상위 경로(+)에 대해 명령어를 실행하면 디스크 스페이스들의 용량과 남은 용량을 조회한다.

**참고**

7.2.1 릴리즈부터 현재 경로에 존재하는 디렉터리내 파일들이 차지하는 디스크 공간을 보여준다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
du [-a] [path]
```

| 인자 | 설명 |
|---|---|
| -a | -a 옵션을 사용하는 경우 지정한 경로 뿐만 아니라 하위 디렉터리의 모든 파일에 대해 크기를 조회 한다. |
| path | path는 필수적으로 입력될 필요는 없다. path를 입력하지 않는다면 현재 경로에 대해서 명령어가 수행된다. |

**참고**

-a 옵션은 7.2.1 릴리즈부터 지원하지 않는다.

- 예제

  du 명령어는 해당 경로에 있는 파일들의 크기를 보여줄 뿐만 아니라 미러링된 파일들을 고려한 크기도 조회한다.

  다음 예제에 나타나는 mirror used mb가 이와 같이 mirroring을 고려한 파일 크기이다.

```
current path = +DS0/ARCH
ASCMD> du
            File_name              Used_mb      Mirror used mb
               f5.txt                   12                  24
               f6.txt                    2                   4
               f8.txt                    9                  18

             Total mb                   23                  46
```

```
current path = +
ASCMD> du
diskspace name                          Total_mb    Free_mb
DS0                                         3069        286
diskspace name                          Total_mb    Free_mb
DS1                                         3069       2782
diskspace name                          Total_mb    Free_mb
DS2                                         3069       2720
```

### 6.3.3. exit

exit 명령어를 입력하면 프로그램이 종료된다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
exit
```

### 6.3.4. help

help 명령어를 입력하면 사용할 수 있는 모든 명령어들에 대한 도움말이 출력된다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
help
```

### 6.3.5. ls

ls 명령어를 사용하여 지정한 경로의 파일들의 목록을 조회할 수 있다. 최상위 경로(+)에 대해 명령어를 실행하면 디스크 스페이스들의 목록을 조회한다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
ls [-a] [-s] [-l] [path]
```

| 인자 | 설명 |
|---|---|
| -a | -a 옵션을 사용하는 경우 지정한 경로 뿐만 아니라 하위 디렉터리의 모든 파일 목록을 조회 한다. |
| -s, --size | -s 옵션을 사용하는 경우 지정한 경로 뿐만 아니라 파일의 크기 정보를 조회한다. |
| -l, --long | -l 옵션을 사용하는 경우 지정한 경로 뿐만 아니라 파일의 상세 정보를 조회한다. |
| path | path를 입력하지 않는다면 현재 경로에 대해서 명령어가 수행된다. |

**참고**

-s, -l 옵션은 7.2.1 릴리즈부터 지원한다.

- 예제

  다음은 ls 파일 목록 예제이다.

```
current path = +DS0/d0
ASCMD> ls
File_name
f5.txt
f6.txt
f8.txt
```

  다음은 ls 디스크 스페이스 목록 예제이다.

```
current path = +
ASCMD> ls
DS0
DS1
DS2
```

## 6.3.6. lsds

디스크 스페이스들의 목록과 그 정보를 조회한다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
lsds
```

- 예제

  다음 예제와 같이 디스크 스페이스들의 번호, 이름, 상태, 중복 레벨, 섹터 사이즈, 메타 블록 크기, 디스크 할당 단위의 크기, 총 용량, 여유 용량을 조회한다.

```
ASCMD> lsds
================================================================
Diskspace no.  Diskspace name
0              DS0
Status         Redun_type
MOUNT          NORMAL
Sector_size    Meta_blksize   au_size
512            8192           1048576
Total_mb       Free_mb
3069           286
================================================================
Diskspace no.  Diskspace name
```

```
1               DS1
Status          Redun_type
MOUNT           NORMAL
Sector_size     Meta_blksize    au_size
512             8192            1048576
Total_mb        Free_mb
3069            2782
================================================================
...
```

## 6.3.7. pwd

현재 경로를 조회한다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
pwd
```

- 예제

```
ASCMD> pwd
current path: +DS0/d0
```

## 6.3.8. rm

rm 명령어를 사용하면 TAS 파일을 제거할 수 있다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
rm [-f] [-r] [-v] {files}
```

| 인자 | 설명 |
|---|---|
| -f, --force | -f 옵션을 사용하는 경우 존재하지 않는 파일을 무시하고 에러 메시지를 표시하지 않는다. |
| -r, --recursive | -r 옵션을 사용하는 경우 디렉터리를 제거할 수 있다. |
| -v, --verbose | -v 옵션을 사용하는 경우 작업 결과를 출력해준다. |

| 인자 | 설명 |
| --- | --- |
| files | files는 상대 경로 및 절대 경로로 지정할 수 있으며 여러 파일의 이름을 입력하면 여러 파일을 동시에 제거할 수 있다. |

**참고**

-f, -r, -v 옵션은 7.2.1 릴리즈부터 지원한다.

- 예제

```
current path = +DS0/d0
ASCMD> ls
File_name
f5.txt
f6.txt
f8.txt
ASCMD> rm f5.txt
rm complete: +DS0/d0/f5.txt
ASCMD> ls
File_name1
f6.txt
f8.txt
```

DB가 기동 중인 상황에서는 .arc 파일을 제외하고 에러가 발생한다.

```
ASCMD> rm c1.ctl
ERROR: no such path '+DS0/c1.ctl'
ERROR: rm terminated with error: Invalid argument
ASCMD> rm test001.dtf
ERROR: no such path '+DS0/test001.dtf'
ERROR: rm terminated with error: Invalid argument
ASCMD> rm +DS0/.passwd
ERROR: no such path '+DS0/.passwd'
ERROR: rm terminated with error: Invalid argument
ASCMD> rm log-t0-r0-r1.arc
rm complete: +DS0/archive/log-t0-r0-r1.arc
```

## 6.3.9. cp, cpfromlocal, cptolocal

cp 명령어를 사용하면 TAS 파일을 복사하여 새로운 TAS 파일을 생성할 수 있으며 cpfromlocal 명령어를 사용하면 로컬 파일을 복사하여 TAS 파일을 생성할 수 있다.

cptolocal 명령어를 사용하면 TAS 파일을 로컬 파일로 복사할 수 있다. 생성할 파일명이 이미 존재한다면 복사를 수행을 하지않고 오류 메시지를 출력한다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
cp {infile} {outfile} [-redun] [redun_no]
cpfromlocal {infile} {outfile} [-t] [file_type]
cptolocal {infile} {outfile} [-redun] [redun_no]
```

| 인자 | 설명 |
|---|---|
| infile, outfile | infile과 outfile은 절대 경로와 상대 경로로 지정해 줄 수 있다. infile로 지정한 file을 복사하여 outfile을 생성한다. 또한 복사한 용량, 소요시간, 속도를 출력한다. |
| -redun redun_no | 대상 파일의 미러링 복사본 중 하나를 특정하여 복사하고자 할 때 사용한다. redun_no 번호에 해당하는 미러링 복사본을 복사하게 되며, redun_no는 대상 파일의 중복 레벨에 따라 0~2 사이의 값을 가질 수 있다. |
| -t file_type | 파일이 TAS 상에서 어떤 파일 타입을 가질 것인지 결정하는 옵션이다. (“4.1.6. V$AS_FILE” 참고) |
| -r, --recursive | -r 옵션을 사용하는 경우 디렉터리를 복사할 수 있다. |
| -v, --verbose | -v 옵션을 사용하는 경우 작업 결과를 출력해준다. |
| -b, --blksize | -b 옵션을 사용하는 경우 파일의 블록 크기를 정해준다. |
| --bs BYTES | 입/출력 블록 사이즈의 크기를 설정하며 한번에 <BYTES> 만큼 읽고 쓰며 ibs, obs 설정을 덮어쓴다. |
| --ibs BYTES | infile 입/출력 블록 크기를 설정하며 한번에 <BYTES> 만큼 infile 에서 읽어온다. |
| --obs BYTES | outfile 입/출력 블록 크기를 설정하며 한번에 <BYTES> 만큼 outfile 에 쓴다. |
| --count N | <N> 개의 블록만 복사한다. |
| --skip N | infile의 <N> 번째 블록부터 읽기를 시작한다. |
| --seek N | outfile의 <N> 번째 블록부터 쓰기를 시작한다. |

**참고**

-r, -v, -b, --bs, --ibs, --obs, --count, --skip, --seek 옵션은 7.2.1 릴리즈부터 지원한다.

- 예제

```
current path = +DS0/d0
ASCMD> du
File_name              Used_mb      Mirror used mb
   f6.txt                    2                   4
   f8.txt                    9                  18

 Total mb                   11                  22
ASCMD> cp f6.txt f5.txt
```

```
2097152 bytes (2.00 MB) copied, 0.079874 s, 26.255753 MB/s
ASCMD> du
File_name              Used_mb      Mirror used mb
   f5.txt                    2                   4
   f6.txt                    2                   4
   f8.txt                    9                  18

 Total mb                   13                  26
```

## 6.3.10. mkdir

mkdir 명령어를 이용하면 디렉터리를 생성할 수 있다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
mkdir [-f] {dirname}
```

| 인자 | 설명 |
|---|---|
| -f, --force | -f 옵션을 사용하는 경우 존재하는 디렉터리를 무시하고 에러 메시지를 표시하지 않는다. |
| dirname | dirname에 해당하는 디렉터리를 생성한다. |

**참고**

-f 옵션은 7.2.1 릴리즈부터 지원한다.

- 예제

```
current path = +DS0
<File_name>
c1.ctl
log0001.log
log0002.log
log0003.log
log0004.log
log0005.log
log0006.log
ASCMD> mkdir fd1
ASCMD> ls
<File_name>
c1.ctl
```

```
log0001.log
log0002.log
log0003.log
log0004.log
log0005.log
log0006.log
fd1/
```

## 6.3.11. mv

mv 명령어를 이용하면 파일을 이동시킬 수 있다.

---

**참고**

mv 명령어는 7.2.1 릴리즈부터 지원한다.

---

명령어의 세부 내용은 다음과 같다.

- 사용법

```
mv [-v] {src_file} {dest_file}
```

| 인자 | 설명 |
|---|---|
| -v | -v 옵션을 사용하는 경우 작업 결과를 출력해준다. |
| src_file, dest_file | src_file과 dest_file은 절대 경로와 상대 경로로 지정해 줄 수 있다. src_file로 지정한 file을 dest_file로 변경한다. |

- 예제

```
ASCMD +DS0/test> ls
test.dtf
 - Number of files found: 1
ASCMD +DS0/test> mv test.dtf mv_test.dtf
Moving file: +DS0/test/test.dtf -> +DS0/test/mv_test.dtf
ASCMD +DS0/test> ls
mv_test.dtf
 - Number of files found: 1
```

## 6.3.12. check

check 명령어를 이용하면 사용자가 알고 있는 TAS 파일 정보에 대해 정합성 검사할 수 있다.

**참고**

check 명령어는 7.2.1 릴리즈부터 지원한다.

명령어의 세부 내용은 다음과 같다.

- 사용법

```
check  [--ds_no] [ds_no] [--file_no] [file_no] [--blksize] [blksize] [-size] [size]
 {filename}
```

| 인자 | 설명 |
|---|---|
| --ds_no | 사용자가 알고 있는 디스크 스페이스 번호를 입력한다. |
| --file_no | 사용자가 알고 있는 파일 번호를 입력한다. |
| --blksize | 사용자가 알고 있는 블록 크기를 입력한다. |
| --size | 사용자가 알고 있는 파일 크기를 입력한다. |
| filename | 절대 경로 및 상대 경로로 입력할 수 있으며 검사하고자 하는 파일 이름을 입력한다. |

- 예제

```
ASCMD +DS0> check log0001.log
FILE INFO (SIZE = BYTES)
DS_NO   FILE_NO   BLOCK_SIZE    SIZE              FILE_NAME
    0       257   512           20971520          +DS0/log0001.log
file '+DS0/log0001.log' is consistent!

ASCMD +DS0> check --ds_no 0 --file_no 257 --blksize 512 --size 20971520 log0001.log
FILE INFO (SIZE = BYTES)
DS_NO   FILE_NO   BLOCK_SIZE    SIZE              FILE_NAME
    0       257   512           20971520          +DS0/log0001.log
file '+DS0/log0001.log' is consistent!
```

# 6.4. TBASCMD 에러코드

다음은 TBASCMD를 사용하는 도중에 발생할 수 있는 에러에 대한 원인과 해결 방법에 대한 설명이다.

**참고**

7.2.1 릴리즈 이전까지 해당 에러코드가 적용된다.

- **ERROR CODE -1: invalid tas port number**

| | |
|---|---|
| **원인** | 포트 번호로 0이나 음수 또는 숫자가 아닌 문자열이 입력된 경우이다. |
| **해결 방법** | 포트 번호를 확인한다. |

- **ERROR CODE -2: more arguments needed**

| | |
|---|---|
| **원인** | tbascmd를 실행하는 경우 argument가 하나도 입력되지 않았을 경우이다. |
| **해결 방법** | 포트 번호를 확인한다. |

- **ERROR CODE -3: invalid argument**

| | |
|---|---|
| **원인** | tbascmd를 실행하는 경우 올바르지 않은 argument가 입력된 경우이다. |
| **해결 방법** | 올바른 argument를 입력한다. |

- **ERROR CODE -4: connection failed**

| | |
|---|---|
| **원인** | 포트 번호를 통해 TAS 인스턴스와 연결하는데 실패했을 경우이다. |
| **해결 방법** | 포트 번호 또는 TAS 인스턴스가 실행되어 있는지 확인한다. |

- **ERROR CODE -5: invalid command "INPUT CMD"**

| | |
|---|---|
| **원인** | 유효하지 않은 tbascmd 명령어가 입력된 경우이다. |
| **해결 방법** | 유효한 tbascmd 명령어를 입력한다. |

- **ERROR CODE -6: length of command cannot exceed "ASCMD_CMD_LEN_MAX"**

| | |
|---|---|
| **원인** | 입력한 tbascmd 명령어가 길이 제한을 초과한 경우이다. |
| **해결 방법** | 2048 글자 이하의 tbascmd 명령어를 입력한다. |

- **ERROR CODE -7: invalid option "INPUT OPTION" for command "INPUT CMD"**

| | |
|---|---|
| **원인** | 입력한 tbascmd 명령어와 옵션이 서로 맞지 않은 경우이다. |
| **해결 방법** | tbascmd 명령어와 옵션을 확인한다. |

- **ERROR CODE -8: invalid option "INPUT OPTION"**

| | |
|---|---|
| **원인** | tbascmd에서 정의되어있지 않은 옵션을 사용한 경우이다. |
| **해결 방법** | tbascmd에 정의된 옵션을 입력한다. |

- **ERROR CODE -9: blksize cannot exceed "ASCMD_BLOCK_SIZE_MAX"**

| | |
|---|---|
| **원인** | cpfromlocal 명령어의 옵션으로 입력한 block size가 최대값을 초과한 경우이다. |
| **해결 방법** | 32KB이하의 ASCMD_BLOCK_SIZE를 입력한다. |

- **ERROR CODE -10: invalid file type "INPUT FILE TYPE" valid types [data, redo, ctrl, temp, arch]**

| | |
|---|---|
| **원인** | cpfromlocal 명령어의 옵션으로 지정한 파일 타입이 tbascmd에 정의되어있는 타입에 해당하지 않는 경우이다. |
| **해결 방법** | 명령어의 옵션으로 지정한 파일 타입이 정의된 파일 타입인지 확인한다. |

- **ERROR CODE -12: too many arguments for command "INPUT CMD"**

| | |
|---|---|
| **원인** | tbascmd 명령어의 argument가 요구사항보다 많이 입력된 경우이다. |
| **해결 방법** | 해당 tbascmd 명령어 사용법을 확인한다. |

- **ERROR CODE -13: need more arguments for command "INPUT CMD"**

| | |
|---|---|
| **원인** | tbascmd 명령어의 argument가 요구사항보다 적게 입력된 경우이다. |
| **해결 방법** | 해당 tbascmd 명령어 사용법을 확인한다. |

- **ERROR CODE -14: invalid argument "INPUT ARG" for command "INPUT CMD"**

| | |
|---|---|
| **원인** | argument가 필요없는 명령어에 argument를 전달했을 경우이다. |
| **해결 방법** | 해당 tbascmd 명령어 사용법을 확인한다. |

- **ERROR CODE -15: cannot parse NULL path**

| | |
|---|---|
| **원인** | 파일 경로로 NULL값이 전달된 경우이다. |
| **해결 방법** | NULL이 아닌 파일 경로를 입력한다. |

- **ERROR CODE -16: length of path must be less than "ASCMD_PATH_LEN_MAX"**

| | |
|---|---|
| **원인** | 파일 경로의 길이가 제한 길이를 초과한 경우이다. |
| **해결 방법** | 256 글자 이하의 파일 경로를 입력한다. |

- **ERROR CODE -18: path "INPUT PATH" has invalid form**

| | |
|---|---|
| **원인** | 파일 경로의 형식이 올바르지 않은 경우이다. |
| **해결 방법** | 올바른 파일 경로 형식을 입력한다. |

- **ERROR CODE -19: no such path**

| | |
|---|---|
| **원인** | 입력한 파일 또는 경로가 존재하지 않는 경우이다. |
| **해결 방법** | 존재하는 파일 경로를 입력한다. |

- **ERROR CODE -20: length of diskspace name must be less than "ASCMD_DS_NAME_LEN_MAX"**

| | |
|---|---|
| **원인** | 디스크 스페이스의 이름이 제한 길이를 초과한 경우이다. |
| **해결 방법** | 48 글자 이하의 디스크 스페이스 이름을 입력한다. |

- **ERROR CODE -22: no such diskspace "INPUT DS NAME"**

| | |
|---|---|
| **원인** | 경로에 포함된 디스크 스페이스를 찾지 못한 경우이다. |
| **해결 방법** | 해당 경로에 디스크 스페이스를 생성하거나, 디스크 스페이스가 존재하는 path를 입력한다. |

- **ERROR CODE -23: no diskspace found**

| | |
|---|---|
| **원인** | 디스크 스페이스가 하나도 존재하지 않는 경우이다. |
| **해결 방법** | 디스크 스페이스를 생성한다. |

- **ERROR CODE -24: length of file name must be less than "ASCMD_FILE_NAME_LEN_MAX"**

| | |
|---|---|
| **원인** | 파일명이 제한 길이를 초과한 경우이다. |
| **해결 방법** | 48 글자 이하의 파일명을 입력한다. |

- **ERROR CODE -25: invalid filename ‘FILENAME’**

| | |
|---|---|
| **원인** | 파일명으로 올바르지 않은 문자열이 전달된 경우이다. |
| **해결 방법** | 올바른 파일명을 확인한다. |

- **ERROR CODE -26: file 'FILENAME' already exists**

| | |
|---|---|
| **원인** | cp 명령어을 수행하는 경우 대상 파일과 같은 이름의 파일이 이미 존재할 경우이다. |
| **해결 방법** | 같은 이름의 파일이 이미 존재하므로, 다른 파일명을 입력한다. |

- **ERROR CODE -27: length of directory path must be less than "ASCMD_PATH_LEN_MAX"**

| | |
|---|---|
| **원인** | 입력한 경로의 길이가 제한 길이를 초과한 경우이다. |
| **해결 방법** | 255 글자 이하의 경로를 입력한다. |

- **ERROR CODE -29: cannot use directory with cp command**

| | |
|---|---|
| **원인** | cp 명령어를 수행할 때 두 파일명 중 하나 이상이 디렉터리 경로가 들어온 경우이다. |
| **해결 방법** | 올바른 파일 경로나 파일명인지 확인한다. |

- **ERROR CODE -30: cannot alloc memory**

| | |
|---|---|
| **원인** | 파일 복사에 사용할 버퍼의 메모리 할당에 실패한 경우이다. |
| **해결 방법** | 기술지원을 통한 연구소 분석을 요청한다. |

- **ERROR CODE -31: asfd_pread failed**

| | |
|---|---|
| **원인** | cp 명령어 수행 중 파일 읽기에 실패한 경우이다. |
| **해결 방법** | 올바른 파일 경로나 파일명인지 확인하거나 경로와 파일명의 권한 설정을 확인한다. |

- **ERROR CODE -32: asfd_pwrite failed**

| | |
|---|---|
| **원인** | cp 명령어 수행 중 파일 쓰기에 실패한 경우이다. |
| **해결 방법** | 올바른 파일 경로나 파일명인지 확인하거나 경로와 파일명의 권한 설정을 확인한다. |

- **ERROR CODE -33: file resize failed**

| | |
|---|---|
| **원인** | TAS가 파일의 alignment를 맞추기 위한 resize 작업을 수행하다가 실패한 경우이다. |
| **해결 방법** | 기술지원을 통한 연구소 분석을 요청한다. |

- **ERROR CODE -35: file name is not entered ascmd is not allowed to remove diskspace**

| | |
|---|---|
| **원인** | rm 명령어의 argument로 입력한 경로에 파일명 없이 디스크 스페이스만 주어진 경우이다. |
| **해결 방법** | 올바른 파일명을 입력한다. 만약, 디스크 스페이스를 지우고 싶다면 tbsql에서 DROP DISKSPACE 구문을 사용한다. |

- **ERROR CODE -36: ascmd command should be given after -c, --cmd option**

| | |
|---|---|
| **원인** | tbascmd -c, --cmd 옵션 뒤에 명령어를 입력하지 않았을 경우이다. |
| **해결 방법** | tbascmd 명령어와 옵션을 확인한다. |

- **ERROR CODE -37: cannot open file**

| | |
|---|---|
| **원인** | TAS가 cp 명령어 수행 과정에서 파일을 여는데 실패한 경우이다. 원인이 다양하여 특정할 수 없다. |
| **해결 방법** | 기술지원을 통한 연구소 분석을 요청한다. |

- **ERROR CODE -38: tas failed to get ds list**

| | |
|---|---|
| **원인** | TAS가 디스크 스페이스의 목록을 읽어오다가 실패한 경우이다. |
| **해결 방법** | 기술지원을 통한 연구소 분석 필요 |

- **ERROR CODE -39: tas failed to get file info**

| | |
|---|---|
| **원인** | TAS가 file size, redun number 등의 파일 정보를 얻어오는데 실패한 경우이다. |
| **해결 방법** | 기술지원을 통한 연구소 분석을 요청한다. |

- **ERROR CODE -40: valid input should be given to option 'OPTION'**

| | |
|---|---|
| **원인** | cpfromlocal 명령어 뒤에 사용되는 -b, --blksize, -t, --type 등의 옵션 뒤에 올바르지 않은 값이 입력된 경우이다. |
| **해결 방법** | cpfromlocal 명령어의 사용법을 확인한다. |

# Appendix A. 구성 예제

본 장에서는 TAS를 이용해 Tibero를 구성하는 방법을 모범 예제(best practice)를 통해 설명한다.

예제는 Linux x86_64를 기준으로 작성되었으며, TAS와 Tibero에서 수행하는 SQL을 제외한 명령은 설치를 원하는 운영체제에 맞게 변경해서 수행해야 한다.

---

**참고**

본 장에서 설명하는 예제는 모범 예제(best practice)일 뿐 최소 권장 사양이 아니다. 최소 권장 사양은 "Tibero 설치 안내서"를 참고한다. 그리고 현재 Linux 운영체제의 설치만 지원하고 있다.

---

## A.1. 설치 준비

본 절에서는 데이터베이스 구성 전 준비할 사항들에 대해 설명한다.

### A.1.1. 설치 환경

다음은 본 구성 예제에서 사용할 설치 환경을 나타낸 표이다.

| 항목 | 값 |
|---|---|
| 노드 갯수 | 2 |
| 노드 내부 IP | 100.100.100.11, 100.100.100.12 |
| TAS용 PORT | 9620 |
| TAS용 CM PORT | 20005 |
| TAS용 LOCAL CLUSTER PORT | 20000 |
| 공유 디스크 갯수 | 4 |
| 공유 디스크 크기 | 각 512GB |
| 공유 디스크 경로 | /dev/sdc, /dev/sdd, /dev/sde, /dev/sdf |
| 설치 계정 | dba |
| Tibero 설치 경로 | /home/dba/tibero |

- 구성에 사용할 바이너리는 이미 해당 경로에 설치된 것으로 가정한다.
- 각 노드에서 보이는 공유 디스크의 경로는 같은 것으로 가정한다.

## A.1.2. 디스크 준비

다음은 설치 대상 공유 디스크를 준비하는 과정으로, 모든 노드에서 똑같이 수행한다.

각 디스크의 권한 또는 소유권을 변경해 직접 사용해도 되지만 재부팅시 디스크 이름이 변경되는 문제가 발생할 수 있으므로 이러한 문제를 예방하기 위해 udev를 사용하여 디바이스 노드를 구성하는 방법을 권고한다. 아래 예제에서는 udev를 사용하여 구성한 디바이스 노드를 통해 TAS를 설치하는 방법으로 작성되어 있다.

아래는 udev를 사용하여 구성된 디바이스 노드의 예제이다.

```
$ ls -al /dev/disk*
lrwxrwxrwx. 1 root root       3 Aug 13 19:50 /dev/disk0 -> sdc
lrwxrwxrwx. 1 root root       3 Aug 13 19:50 /dev/disk1 -> sdd
lrwxrwxrwx. 1 root root       3 Aug 13 19:50 /dev/disk2 -> sde
lrwxrwxrwx. 1 root root       3 Aug 13 19:50 /dev/disk3 -> sdf
$ ls -l /dev/sd*
brw-rw----. 1 root disk  8,   0 Jul 13 11:20 /dev/sda
brw-rw----. 1 root disk  8,  16 Jul 13 11:20 /dev/sdb
brw-------. 1 dba disk  8,  32 Jul 13 11:20 /dev/sdc
brw-------. 1 dba disk  8,  48 Jul 13 11:20 /dev/sdd
brw-------. 1 dba disk  8,  64 Jul 13 11:20 /dev/sde
brw-------. 1 dba disk  8,  80 Jul 13 11:20 /dev/sdf
```

위의 예제에서 /dev/disk*로 나타나는 링크들은 udev rules에 따라 생성된 symbolic link이다.

다음은 디바이스 노드를 구성하기 위한 udev rules 파일의 예제이다.

```
$ cat /etc/udev/rules.d/as-disk.rules
KERNEL=="sd?", SUBSYSTEM=="block", PROGRAM=="/lib/udev/scsi_id -g -u -d %N",
RESULT=="35000c50087de8480", SYMLINK+="disk0", OWNER="dba", MODE="0600"
KERNEL=="sd?", SUBSYSTEM=="block", PROGRAM=="/lib/udev/scsi_id -g -u -d %N",
RESULT=="35000c50087de8481", SYMLINK+="disk1", OWNER="dba", MODE="0600"
KERNEL=="sd?", SUBSYSTEM=="block", PROGRAM=="/lib/udev/scsi_id -g -u -d %N",
RESULT=="35000c50087de8482", SYMLINK+="disk2", OWNER="dba", MODE="0600"
KERNEL=="sd?", SUBSYSTEM=="block", PROGRAM=="/lib/udev/scsi_id -g -u -d %N",
RESULT=="35000c50087de8483", SYMLINK+="disk3", OWNER="dba", MODE="0600"
```

udev rules 파일은 /etc/udev/rules.d 폴더(Ubuntu 14.04 기준, OS마다 경로가 다를 수 있다) 안에 .rules라는 확장자로 저장되어야 한다.

예제로 보여준 rule이 의미하는 바는 명시된 SCSI_ID(RESULT=="SCSI_ID")와 일치하는 디바이스를 sd?로 표현되는 커널 이름(KERNEL=="sd?")을 가진 block 디바이스(SUBSYSTEM=="block") 노드 중에 찾아서 소유자, 사용자 권한을 설정하고 주어진 symbolic link를 만들어 주라는 의미이다.

디바이스의 SCSI_ID는 /lib/udev/scsi_id(Ubuntu 14.04기준, OS마다 경로가 다를 수 있다)를 실행하면 확인할 수 있다. 이 프로그램은 관리자 권한으로 실행되어야 하며 아래는 scsi_id를 확인하는 예제이다.

```
$ /lib/udev/scsi_id -g -u -d /dev/sdc
35000c50087de8480
```

### A.1.3. 커널 파라미터 설정

TAS를 이용해 Tibero를 구성하는 경우 "Tibero 설치 안내서"에 명시된 커널 파라미터 설정에 더하여 추가적인 설정이 필요하다. 커널 파라미터 설정 파일 위치는 다음과 같다.(Linux 기준)

```
/etc/sysctl.conf
```

다음과 같이 커널 파라미터를 설정한다.

```
fs.aio-max-nr = 4194304
```

AIX 환경에서 커널 파라미터를 설정할 경우 다음 프러시저를 실행한다.

```
# ioo -p -o posix_aio_maxreqs=131072
```

## A.2. TAS 인스턴스 설치

본 절에서는 두 개의 노드에 TAS 인스턴스를 구성하는 방법에 대해 설명한다.

### A.2.1. 환경변수 설정

해당 운영체제의 사용자 계정별로 존재하는 환경설정 파일(.bashrc 등)에 환경 변수를 설정한다. 본 예제에서는 다음과 같이 환경변수를 설정한다.

모든 노드에서 TAS 인스턴스의 바이너리가 설정된 경로로 TB_HOME 환경변수를 설정한다. 그리고 각 노드의 TAS 인스턴스를 구분하기 위해 TB_SID 환경변수를 노드별로 다르게 설정한다.

다음은 첫 번째 노드 예시이다. 첫 번째 노드의 TAS 인스턴스를 위해 TB_SID 환경변수를 as0으로 설정한다. 첫 번째 노드의 클러스터 매니저를 위해 CM_SID 환경변수를 cm0으로 설정한다.

```
export TB_HOME=/home/dba/tibero
export TB_SID=as0
export CM_SID=cm0
export LD_LIBRARY_PATH=$TB_HOME/lib:$TB_HOME/client/lib
export PATH=$PATH:$TB_HOME/bin:$TB_HOME/client/bin
```

두 번째 노드의 TAS 인스턴스를 위해 TB_SID 환경변수를 as1로 설정한다. CM_SID 환경변수는 cm1로 설정한다.

```
[100.100.100.12]$ export TB_SID=as1
[TB_SID=as1@100.100.100.12]$ export CM_SID=cm1
```

## A.2.2. 초기화 파라미터 설정

TAS 인스턴스의 초기화 파라미터는 기본적으로 Tibero Active Cluster(TAC)의 초기화 파라미터와 같다.

TAS 인스턴스에서 추가로 설정해야 하는 초기화 파라미터는 다음과 같다.

| 파라미터 | 설명 |
|---|---|
| INSTANCE_TYPE | Active Storage 인스턴스임을 나타내기 위해 AS로 설정한다. |
| AS_DISKSTRING | 사용할 디스크를 찾기 위한 경로 패턴을 설정한다. |

다음은 첫 번째 노드의 초기화 파라미터이다.

```
[100.100.100.11]$ cat /home/dba/tibero/config/as0.tip
INSTANCE_TYPE=AS
AS_DISKSTRING="/dev/disk*"
AS_ALLOW_ONLY_RAW_DISKS=N

LISTENER_PORT=9620
TOTAL_SHM_SIZE=3G
MEMORY_TARGET=4G

CLUSTER_DATABASE=Y
LOCAL_CLUSTER_ADDR=100.100.100.11
LOCAL_CLUSTER_PORT=20000
CM_CLUSTER_MODE=ACTIVE_SHARED
CM_PORT=20005

THREAD=0

[100.100.100.11]$ cat /home/dba/tibero/config/cm0.tip
CM_NAME=cm0
CM_UI_PORT=20005
CM_RESOUCE_FILE=/home/dba/tibero/config/cm0_res.crf
```

다음은 두 번째 노드의 초기화 파라미터이다.

```
[100.100.100.12]$ cat /home/dba/tibero/config/as1.tip
INSTANCE_TYPE=AS
AS_DISKSTRING="/dev/disk*"
AS_ALLOW_ONLY_RAW_DISKS=N

LISTENER_PORT=9620
TOTAL_SHM_SIZE=3G
MEMORY_TARGET=4G

CLUSTER_DATABASE=Y
LOCAL_CLUSTER_ADDR=100.100.100.12
```

```
LOCAL_CLUSTER_PORT=20000
CM_CLUSTER_MODE=ACTIVE_SHARED
CM_PORT=20005

THREAD=1

[100.100.100.11]$ cat /home/dba/tibero/config/cm1.tip
CM_NAME=cm1
CM_UI_PORT=20005
CM_RESOUCE_FILE=/home/dba/tibero/config/cm1_res.crf
```

**주의**

1. 일반 파일을 디바이스처럼 사용하기 위해서는 AS_ALLOW_ONLY_RAW_DISKS를 N으로 설정해야 한다.

2. AS_DISKSTRING에 설정한 경로 패턴 포함되지 않은 디스크는 디스크 스페이스 구성에 사용할 수 없다.

## A.2.3. 접속 정보 설정

다음은 접속 정보 설정 파일을 설정한 예이다.

```
$ cat /home/dba/tibero/client/config/tbdsn.tbr
as0=(
    (INSTANCE=(HOST=100.100.100.11)
              (PORT=9620)
    )
)
as1=(
    (INSTANCE=(HOST=100.100.100.12)
              (PORT=9620)
    )
)
```

## A.2.4. 디스크 스페이스 생성과 기동

다음은 디스크 스페이스 생성과 기동하는 과정에 대한 설명이다.

1. 디스크 스페이스를 생성하기 위해서 먼저 첫 번째 노드에서 TAS 인스턴스를 NOMOUNT 모드로 기동한다.

   ```
   [TB_SID=as0@100.100.100.11]$ tbboot nomount
   ```

2. 기동된 인스턴스에 접속하여 디스크 스페이스를 생성한다.

RAID와 같은 외부의 데이터 복제 기능을 사용해 데이터의 가용성을 높인 경우, 내부의 데이터 복제 기능을 사용하지 않도록 하기 위해 EXTERNAL REDUNDANCY로 설정한다.

```
[TB_SID=as0@100.100.100.11]$ tbsql sys/tibero@as0
SQL> CREATE DISKSPACE ds0
     EXTERNAL REDUNDANCY
     DISK '/dev/disk0' NAME disk0 SIZE 512G,
          '/dev/disk1' NAME disk1 SIZE 512G,
          '/dev/disk2' NAME disk2 SIZE 512G,
          '/dev/disk3' NAME disk3 SIZE 512G;
```

디스크 스페이스 생성이 완료되면 자동으로 인스턴스가 중지되며, NORMAL 모드로 기동할 수 있는 상태가 된다.

3. 첫 번째 노드의 클러스터 매니저가 AS 바이너리를 실행하기 위해 필요한 as 리소스 환경설정용 파일을 미리 생성한다.

```
[CM_SID=cm0@100.100.100.11]$ cat $TB_HOME/as0.profile
export TB_SID=as0
```

4. TAS 인스턴스들의 클러스터링을 위해서 클러스터 매니저를 기동하고 리소스를 추가해준다. as 리소스를 추가할 때에는 미리 생성한 환경설정용 파일의 경로를 envfile attribute에 지정해줘야 한다.

```
[CM_SID=cm0@100.100.100.11]$ tbcm -b
[CM_SID=cm0@100.100.100.11]$ cmrctl add network --name net0 --ipaddr
100.100.100.11 --portno 20010
[CM_SID=cm0@100.100.100.11]$ cmrctl add cluster --name cls0 --incnet net0
--cfile "+/dev/disk*"
[CM_SID=cm0@100.100.100.11]$ cmrctl start cluster --name cls0
[CM_SID=cm0@100.100.100.11]$ cmrctl add service --type as --name tas --cname cls0
[CM_SID=cm0@100.100.100.11]$ cmrctl add as --name as0 --svcname tas --envfile
"$TB_HOME/as0.profile" --dbhome "$TB_HOME"
```

5. 인스턴스를 NORMAL 모드로 기동한다.

```
[TB_SID=as0@100.100.100.11]$ tbboot
# 또는 [CM_SID=cm0@100.100.100.11]$ cmrctl start as --name as0
```

6. 기동이 완료되면 인스턴스에 접속해 두 번째 노드의 TAS 인스턴스를 위한 THREAD를 추가한다.

```
[TB_SID=as0@100.100.100.11]$ tbsql sys/tibero@as0
SQL> ALTER DISKSPACE ds0 ADD THREAD 1;
```

7. 두 번째 노드의 클러스터 매니저가 AS 바이너리를 실행하기 위해 필요한 as 리소스 환경설정용 파일을 미리 생성한다.

```
[CM_SID=cm1@100.100.100.12]$ cat $TB_HOME/as1.profile
export TB_SID=as1
```

8. 두 번째 노드에서 클러스터 매니저를 기동하고 리소스를 추가해준다.

```
[CM_SID=cm1@100.100.100.12]$ tbcm -b
[CM_SID=cm1@100.100.100.12]$ cmrctl add network --name net1 --ipaddr
100.100.100.12 --portno 20010
[CM_SID=cm1@100.100.100.12]$ cmrctl add cluster --name cls0 --incnet net1
--cfile "+/devs/disk*"
[CM_SID=cm1@100.100.100.12]$ cmrctl start cluster --name cls0
[CM_SID=cm1@100.100.100.12]$ cmrctl add as --name as1 --svcname tas --envfile
"$TB_HOME/as1.profile" --dbhome "$TB_HOME"
```

9. 두 번째 노드의 TAS 인스턴스를 기동한다.

```
[TB_SID=as1@100.100.100.12]$ tbboot
# 또는 [CM_SID=cm1@100.100.100.12]$ cmrctl start as --name as1
```

# A.3. Tibero 인스턴스 설치

Tibero 인스턴스를 설치하고 기동하는 과정은 TAS 인스턴스를 사용하지 않는 경우와 동일하며 생성할 파일의 경로를 지정하는 방법에만 차이가 있다.

## A.3.1. 환경변수 설정

해당 운영체제의 사용자 계정별로 존재하는 환경설정 파일(.bashrc 등)에 환경 변수를 설정한다. 본 예제에서는 다음과 같이 환경변수를 설정한다.

먼저 모든 노드에서 Tibero 인스턴스의 바이너리가 설정된 경로로 TB_HOME 환경변수를 설정한다. 그리고 각 노드에서 TB_SID 환경변수를 설정한다. 첫 번째 노드의 DB 인스턴스를 위해 TB_SID 환경변수를 tac0으로 설정한다.

```
export TB_HOME=/home/dba/tibero
export TB_SID=tac0
export CM_SID=cm0
export LD_LIBRARY_PATH=$TB_HOME/lib:$TB_HOME/client/lib
export PATH=$PATH:$TB_HOME/bin:$TB_HOME/client/bin
```

두 번째 노드의 DB 인스턴스를 위해 TB_SID 환경변수를 tac1로 설정한다. CM_SID가 설정되어 있지 않다면 cm1으로 설정해준다.

```
[100.100.100.12]$ export TB_SID=tac1
[TB_SID=tac1@100.100.100.12]$ export CM_SID=cm1
```

하나의 서버에 여러 인스턴스를 띄우는 경우에는 SID가 달라야하지만 다른 서버에 각 인스턴스를 띄울때는 SID가 같아도 무방하다.

## A.3.2. 초기화 파라미터 설정

TAS 인스턴스를 사용하기 위해서는 다음의 두 가지 파라미터를 설정한다.

| 파라미터 | 설명 |
|---|---|
| USE_ACTIVE_STORAGE | TAS 인스턴스를 사용하기 위해 Y로 설정한다. |
| AS_PORT | TAS 인스턴스에 접속하기 위해 TAS 인스턴스의 LISTENER_PORT를 설정한다. |

다음은 첫 번째 노드의 초기화 파라미터이다.

```
[100.100.100.11]$ cat /home/dba/tibero/config/tac0.tip
DB_NAME=tibero
LISTENER_PORT=8629
DB_BLOCK_SIZE=32K
MAX_SESSION_COUNT=300
TOTAL_SHM_SIZE=70G
DB_CACHE_SIZE=55G
MEMORY_TARGET=250G
CONTROL_FILES="+DS0/c1.ctl"

USE_ACTIVE_STORAGE=Y
AS_PORT=9620

CLUSTER_DATABASE=Y
LOCAL_CLUSTER_ADDR=100.100.100.11
LOCAL_CLUSTER_PORT=21000
CM_PORT=20005

THREAD=0
UNDO_TABLESPACE=UNDO00
```

다음은 두 번째 노드의 초기화 파라미터이다.

```
[100.100.100.12]$ cat /home/dba/tibero/config/tac1.tip
DB_NAME=tibero
LISTENER_PORT=8629
DB_BLOCK_SIZE=32K
MAX_SESSION_COUNT=300
TOTAL_SHM_SIZE=70G
DB_CACHE_SIZE=55G
MEMORY_TARGET=250G
CONTROL_FILES="+DS0/c1.ctl"

USE_ACTIVE_STORAGE=Y
AS_PORT=9620
```

```
CLUSTER_DATABASE=Y
LOCAL_CLUSTER_ADDR=100.100.100.12
LOCAL_CLUSTER_PORT=21000
CM_PORT=20005

THREAD=1
UNDO_TABLESPACE=UNDO01
```

## A.3.3. 접속 정보 설정

다음은 접속 정보 설정 파일을 설정한 예이다.

```
$ cat /home/dba/tibero/client/config/tbdsn.tbr
tac0=(
    (INSTANCE=(HOST=100.100.100.11)
              (PORT=8629)
    )
)
tac1=(
    (INSTANCE=(HOST=100.100.100.12)
              (PORT=8629)
    )
)
```

## A.3.4. 데이터베이스 생성과 기동

다음은 데이터베이스 생성과 기동하는 과정에 대한 설명이다.

1. 클러스터 매니저가 Tibero 바이너리를 실행하기 위해 필요한 DB 리소스 환경설정용 파일을 미리 생성한다.

   ```
   [CM_SID=cm0@100.100.100.11]$ cat $TB_HOME/tac0.profile
   export TB_SID=tac0
   ```

   ```
   [CM_SID=cm1@100.100.100.12]$ cat $TB_HOME/tac1.profile
   export TB_SID=tac1
   ```

2. 우선 첫 번째 노드에서 TAS 인스턴스를 구성할 때에 부팅한 클러스터 매니저에 Tibero 클러스터링을 위해 사용할 리소스를 생성한다.

   ```
   [CM_SID=cm0@100.100.100.11]$ cmrctl add service --type db --name tibero
   --cname cls0
   [CM_SID=cm0@100.100.100.11]$ cmrctl add db --name tac0 --svcname tibero
   --envfile "$TB_HOME/tac0.profile" --dbhome "$TB_HOME"
   ```

3. 인스턴스를 NOMOUNT 모드로 기동한다.

```
[TB_SID=tac0@100.100.100.11]$ tbboot -t nomount
```

4. 인스턴스에 접속해 데이터베이스를 생성한다.

```
[TB_SID=tac0@100.100.100.11]$ tbsql sys/tibero@tac0
SQL> CREATE DATABASE "tibero"
    USER sys IDENTIFIED BY tibero
    MAXINSTANCES 32
    MAXDATAFILES 2048
    CHARACTER SET MSWIN949
    LOGFILE GROUP 1 '+DS0/log001' SIZE 2G,
            GROUP 2 '+DS0/log002' SIZE 2G,
            GROUP 3 '+DS0/log003' SIZE 2G
    MAXLOGGROUPS 255
    MAXLOGMEMBERS 8
    NOARCHIVELOG
    DATAFILE '+DS0/system.tdf' SIZE 4G
            AUTOEXTEND ON NEXT 64M MAXSIZE UNLIMITED
    SYSSUB DATAFILE '+DS0/syssub.tdf' SIZE 4G
            AUTOEXTEND ON NEXT 64M MAXSIZE UNLIMITED
    DEFAULT TEMPORARY TABLESPACE temp
            TEMPFILE '+DS0/temp00.tdf' SIZE 32G AUTOEXTEND OFF
    UNDO TABLESPACE undo00
            DATAFILE '+DS0/undo00.tdf' SIZE 32G AUTOEXTEND OFF
    DEFAULT TABLESPACE usr DATAFILE '+DS0/usr.tdf' SIZE 4G
            AUTOEXTEND ON NEXT 64m MAXSIZE UNLIMITED;
```

5. 인스턴스를 NORMAL 모드로 재기동한 후 두 번째 노드에서 사용할 UNDO 테이블 스페이스와 REDO THREAD를 생성한다.

```
[TB_SID=tac0@100.100.100.11]$ tbboot
[TB_SID=tac0@100.100.100.11]$ tbsql sys/tibero@tac0
SQL> CREATE UNDO TABLESPACE undo01
            DATAFILE '+DS0/undo01.tdf' size 32G autoextend off;
SQL> ALTER DATABASE ADD LOGFILE THREAD 1 GROUP 4 '+DS0/log004' size 2G;
SQL> ALTER DATABASE ADD LOGFILE THREAD 1 GROUP 5 '+DS0/log005' size 2G;
SQL> ALTER DATABASE ADD LOGFILE THREAD 1 GROUP 6 '+DS0/log006' size 2G;
SQL> ALTER DATABASE ENABLE PUBLIC THREAD 1;
```

6. 두 번째 노드의 클러스터 매니저에 DB 리소스를 추가한다.

```
[CM_SID=cm1@100.100.100.12]$ cmrctl add db --name tac1 --svcname tibero
--envfile "$TB_HOME/tac1.profile" --dbhome "$TB_HOME"
```

7. 두 번째 노드의 인스턴스를 기동한다.

```
[TB_SID=tac1@100.100.100.12]$ tbboot
```

## A.4. TAS 운영 권장사항

TAS를 설치하고 운영하는데 필요한 권장사항이다.

- **초기화 파라미터 설정**

  Active Storage 인스턴스는 데이터베이스 인스턴스보다 더 적은 양의 메모리를 사용한다. 원활한 운영을 위해서 다음 예와 같이 메모리 파라미터를 설정하는 것이 좋다.

| 파라미터 | 값 |
|---|---|
| TOTAL_SHM_SIZE | 1GB 이상 |
| MEMORY_TARGET | 2GB 이상 |

- **디스크 스페이스 REDUNDANCY 설정**

  RAID와 같이 외부적인 데이터 복제 기능을 사용하고 있는 경우 REDUNDANCY를 EXTERNAL로 설정해도 무방하고, 그렇지 않은 경우라면 REDUNDANCY를 NORMAL로 설정하는 것이 좋다. 만약 속도적인 측면보다 데이터의 가용성 측면이 중요한 경우라면 REDUNDANCY를 HIGH로 설정하는 것이 좋다.

- **디스크 스페이스 실패 그룹 설정**

  물리적으로 같은 서버 또는 같은 스위치에 속해 있는 디스크들은 같은 실패 그룹으로 설정하는 것이 좋다. 스위치 고장, 케이블 고장 등이 발생했을 때 이 디스크들은 동시에 접근 불가 상태가 될 확률이 높기 때문이다. Active Storage는 다른 실패 그룹에 복사본을 저장해놓기 때문에 한 실패 그룹에 문제가 생겨도 데이터 접근이 가능하다. 디스크 스페이스의 총 실패 그룹 개수는 3 개 이상으로 구성하는 것이 좋다.

- **디스크 스페이스 용량 설정**

  디스크 스페이스의 용량은 운영할 데이터베이스의 용량에 따라 적절히 구성한다. 이때 한 디스크 스페이스가 가질 수 있는 디스크의 최대 개수는 1024개이며, 최대 용량은 16TB이다.

  디스크 장애가 발생하여 데이터 복사본을 새로 만들어야 하는 경우가 생길 수 있으므로 여유 디스크를 추가할 수 있도록 구성하는 것이 좋다.

- **DBA용 관리 팁**

  SYS 사용자로 SQL을 이용해 Active Storage 인스턴스에 접속을 할 수 있으며 “제4장 TAS 정보 조회”에 설명한 뷰들을 이용해 현재 디스크 스페이스 상태, 디스크 상태를 확인할 수 있다. 만약 디스크 장애가 발생하면 뷰에서 디스크 상태가 FAIL로 바뀌게 된다.

- **디스크 속성**

  디스크 스페이스를 구성하는 디스크들의 속성(크기와 속도)은 비슷한 것이 좋다. 디스크가 크기와 속도가 같으면 각 디스크가 같은 비율로 사용이 되며 이러한 경우 스트라이핑에 효과적이다.

# 색인

## U



## V



## ㄷ



## ㅁ



## ㅅ



## ㅇ



## ㅈ



## ㅍ



## ㅎ