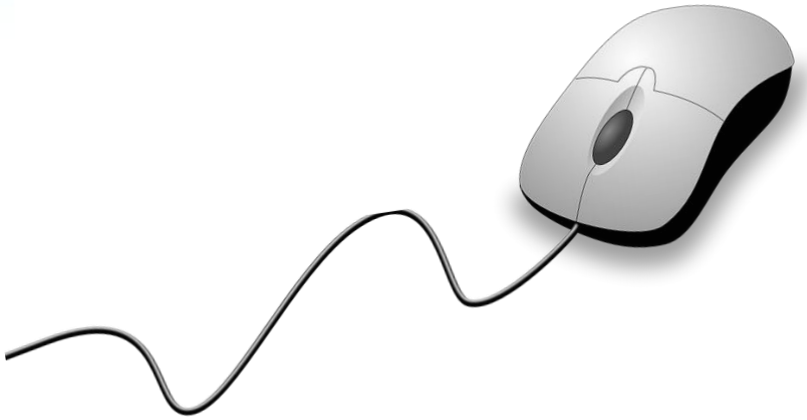
공개SW 솔루션 설치 & 활용 가이드

시스템 SW > 데이터관리



ALTIBASE 제대로 배워보자

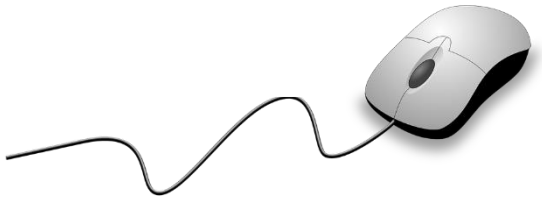
How to Use Open Source Software

Open Source Software Installation & Application Guide

OPENUP

오픈소스 소프트웨어 통합지원센터

Open Source Software Support Center



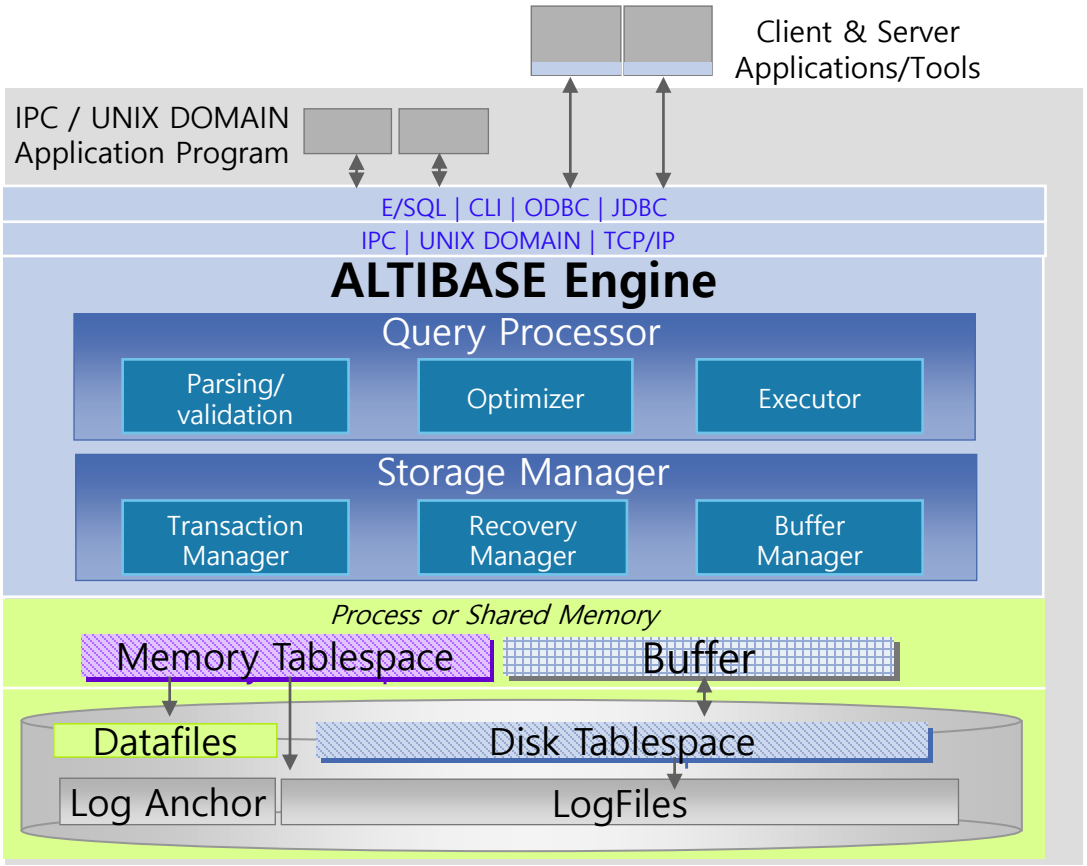
CONTENTS

1. 개요

소개	• ALTIBASE 는 2000년 아시아 최초 제품 출시 후 국내 In-memory DBMS 분야의 선두 주자 • 2005년 세계 최초 Hybrid Database Architecture 를 개발 및 상용화 성공		
주요기능	• 다양한 저장 공간 제공(In-Memory Tablespace, Volatile Memory Tablespace, Disk Tablespace) • SQL Standards 지원 (SQL92, SQL99(partial)) • 네트워크를 통한 복제 기능 • 공간 데이터 처리를 위한 데이터 타입 및 함수 제공 • 이기종 DBMS 연동 (DBLINK, JDBCAdaptor) • 다양한 개발 환경 지원 (Embedded SQL, CLI, ODBC, JDBC, .Net Provider,…) • 보안 (접근제어, 권한 제어, 감사, 데이터 암/복호화, 암호화 통신) • 다양한 백업 기능 제공(aexport, online backup, incremental backup)		
대분류	• 시스템 SW	**소분류**	• 데이터 관리
라이선스형태	• Server : GNU-AGPLv3 • Client : GNU-LGPLv3	**사전설치 솔루션**	• 없음
		버전	• 7.1.0
특징	• Hybrid RDBMS 로 고성능의 메모리 테이블과 대용량의 디스크 테이블을 하나의 엔진으로 운영 • Hybrid partitioned table 지원 • 네트워크를 통한 이중화로 유연한 HA 구축 가능		
개발회사/커뮤니티	• ㈜알티베이스		
공식 홈페이지	• https://altibase.com		

2. 기능요약

- ALTIBASE 전체 구성도



- 하나의 ALTIBASE 엔진에 MMDBMS와 DRDBMS를 제공하기 때문에 별도의 QP (Query Processor), SM (Storage Manager)가 필요한 것이 아니라 하나의 QP, SM을 통해서 효율적인 데이터 관리를 할 수 있습니다.

- 하나의 QP와 SM을 통해서 데이터 처리를 할 수 있기 때문에 쿼리, 트랜잭션, 개발, 복구, 관리 등의 투명성을 제공할 수 있습니다.

- 메모리 기반의 테이블은 디스크 기반의 Buffer와는 달리 Disk I/O가 발생하지 않고 인덱스 관리 방식이 다르기 때문에 고속 처리에 적합합니다.

2. 기능요약

- ALTIBASE 주요 기능 (1/2)

SQL	• SQL92을 지원하며, Sub-query 및 Join에 제한이 없음 • 다양한 built-in 함수를 제공 • 효율적으로 사용하기 위한 In-line View, Hint, SQL 튜닝 정보 등을 지원 • Optimizer 성능향상을 위한 실시간 테이블 통계 정보 자동 수집 기능
프로그래밍 인터페이스	• Embedded SQL, Stored Procedure, ODBC, JDBC, SQLCLI 지원
트랜잭션 기능	• 트랜잭션의 전체/부분 철회 지원(total and partial rollback) • Multi-Version Concurrency Control을 통한 Record Level Lock 지원
DB 이중화	• 무정지 서비스, 부하 분산을 위한 이중화 기능 제공 • Active – Active, Active – Standby의 구성 가능 • 이중화 중 장애발생 시 DB의 일치성 보장 • N-Way 이중화, IP 이중화, 이 기종간의 이중화 지원
실시간 및 대용량 지원	• 실시간 데이터 처리를 위한 메모리 Table과 대용량 데이터 처리를 위한 디스크 Table을 하나의 DBMS 엔진에서 동시에 처리

2. 기능요약

- ALTIBASE 주요 기능(2/2)

Large Record & Internal LOB	• 대용량의 데이터 저장을 위해 data type으로 BLOB, CLOB 지원 • 최대 2GB까지 저장 지원
로그 분석기	• Transaction 기반의 Active Log를 분석하여 API 형태로 Xlog와 Meta를 제공하여 이기종 DBMS 간의 데이터 연동 가능
DB 복구	• 트랜잭션 / 시스템 장애에 대하여 완벽한 복구 지원 • 온라인 백업을 지원하며 이를 이용한 DB 복구 지원 • 정기 및 비정기적으로 체크포인트 기능을 제공하고 체크포인트 수행 중에도 트랜잭션 처리 가능 • 실시간으로 Logging
공간 데이터	• 공간 데이터를 처리하기 위한 기본적인 데이터 타입을 제공 • 공간 데이터들을 저장, 분석에 필요한 함수들을 제공 • ISO SQL/MM 공간 표준 규격과 GIS 국제표준단체인 Open Geospatial Consortium(OGC)의 "Simple Features Specification for SQL" 규격을 준수

3. 실행환경

- Linux 계열
  - ✓ GLIBC 2.12 ~ 2.17 호환성 보장
- 요구사항
  - ✓ 메모리 : 64-bit OS: 최소 1GB 이상 (권장: 2GB 이상, 최대 제한 없음)
  - ✓ 디스크 : 데이터 저장을 위한 테이블스페이스와 트랜잭션 로그 등을 고려하여야 하고 트랜잭션 로그 및 설치를 위해서는 1GB이상이 필요하며, 원활한 운영을 위해서는 12GB 이상의 여유 공간을 확보하는 것을 권장한다. 또한 성능을 위해서는 트랜잭션 로그 저장 영역과 데이터 저장 영역을 물리적으로 분리하는 것을 권장한다.
  - ✓ 네트워크 : 이중화 기능 사용시 전용선 사용을 권장한다.

4. 설치

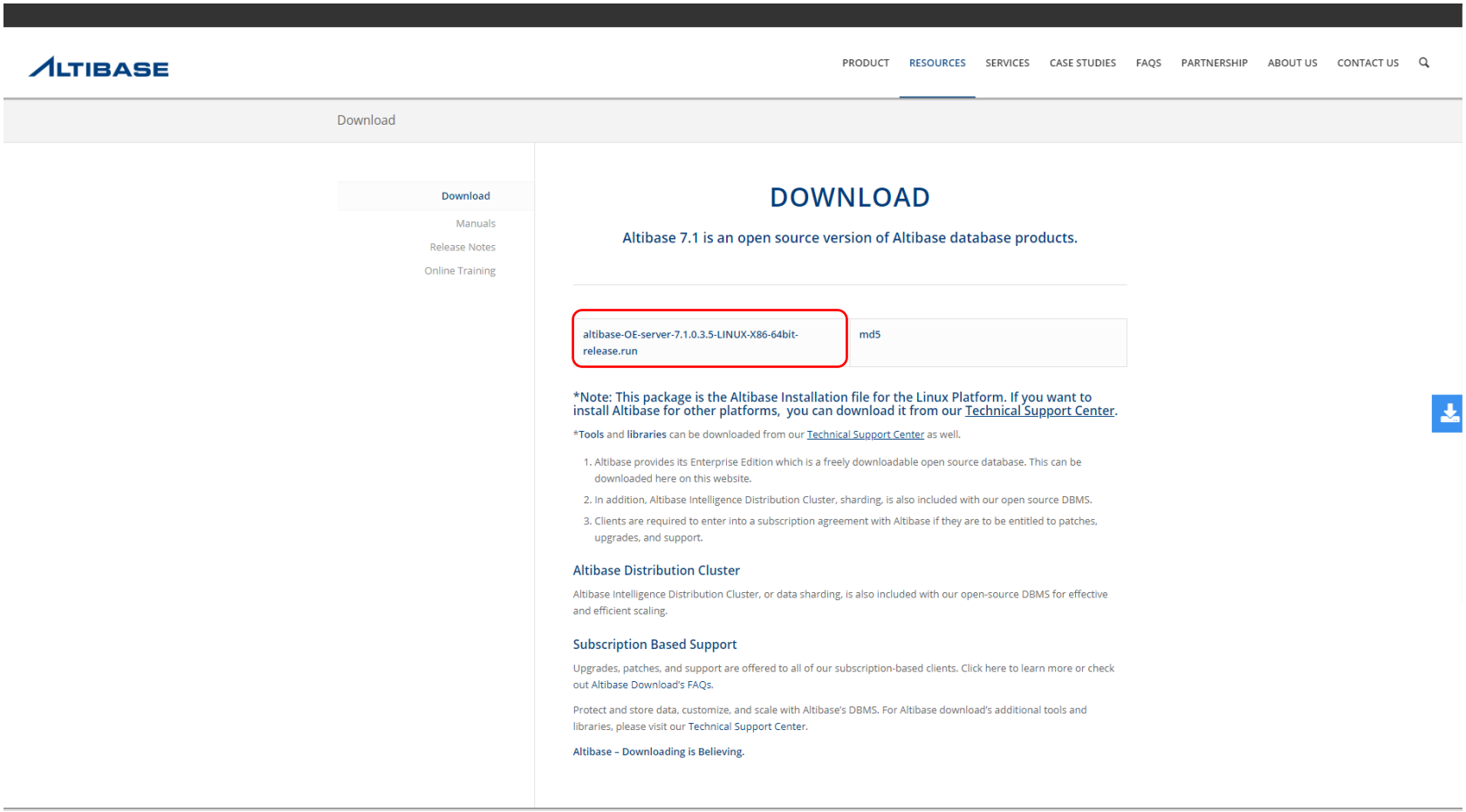
세부 목차

4. 설치

4.1 패키지 다운 받기

- 설치 파일

URL : https://altibase.com/resources/free-download/



1. 사용자 계정의 리소스 한계 값 확인

OS 명령어인 "ulimit"으로 사용자 계정에 설정된 리소스 한계 값을 확인 또는 변경할 수 있다.

- File Size
  프로세스가 생성 가능한 파일의 최대 크기
- Data segment size
  프로세스가 사용 가능한 논리적 메모리의 최대 크기(VSZ측면)
- Max memory size
  프로세스가 사용 가능한 물리적 메모리의 최대 크기(RSS측면)
- Open files (descriptor)
  프로세스가 동시에 접근 가능한 파일 및 소켓의 최대 개수
- Stack size
- 최대 스택 사이즈
- Virtual memory
  프로세스가 사용 가능한 가상 메모리의 최대 크기

사용자 계정의 리소스 한계 값들을 "unlimited"로 설정할 것을 권장한다. 이 때 core file size는 unlimited로 설정하지 않도록 한다. 만일 Altibase 서버가 비정상 종료하여 코어를 덤프 할 경우 메모리 데이터베이스를 모두 core 파일로 저장하기 때문에 unlimited로 설정하면 디스크 부족이 발생할 수 있다. Altibase 클라이언트 제품은 Stack size가 최소 70KB 이상이어야 한다.

2. 커널 설정

1) 공유 메모리 및 세마포어 설정

/proc/sys/kernel 경로에 sem, shmmax, shmmni, swapiness 등의 파일에 설정한다.

권장 값

리눅스 커널 버전이 2.5 이상이 아닐 경우 IPC접속을 사용하는 세션이 갑자기 단절되는 현상이 발생할 수 있다. 서버 부팅 시 자동으로 커널 파라미터가 설정되게 하려면, /etc/rc.d/rc.local 파일 내에 아래의 항목을 추가한다.

```
/etc/rc.d/rc.local 파일 내에 아래의 항목을 추가
echo 2147483648 > /proc/sys/kernel/shmmax
echo 4096 > /proc/sys/kernel/shmmni
echo 200 32000 512 5029 > /proc/sys/kernel/sem
echo 5 > /proc/sys/vm/swappiness
```

2) RemoveIPC 설정

RedHat 7.2 이상의 버전에서는 RemoveIPC 설정 값을 'no'로 설정하는 것을 권장한다(기본 값은 'yes').

RemoveIPC가 'yes'로 설정되면 세마포어가 부족하여 비정상종료가 발생할 수 있기 때문이다.

설정 값을 변경하려면 /etc/systemd/logind.conf에서 RemoveIPC=no로 설정한 후 OS를 다시 시작해야 한다.

3) THP 설정 확인 및 비활성화 방법

THP(Transparent Huge Pages)는 메모리 페이지의 크기를 증가시킴으로써, TLB(Translation Lookaside Buffer)를 조회하는 비용을 줄이기 위한 목적으로 리눅스에서 제공하는 메모리 관리 시스템이다. 원래 의도와 달리 메모리 할당 지연 및 단편화를 유발하여 오히려 시스템 성능이 저하되는 경우가 많다.

THP 설정 확인

THP에서 설정할 수 있는 옵션은 always, madvise, never 3가지이다. [ ] 로 둘러싸인 것이 현재 적용된 옵션이다. 각각의 의미는 아래와 같다.

- madvise: madvise() 함수를 통해 THP 사용을 명시적으로 요청한 프로세스에만 THP가 활성화되는 옵션이다.
- always: 모든 프로세스에 항상 THP가 적용되게 된다.
- never: madvise() 함수 요청과 관계없이 모든 프로세스에서 THP가 비활성화되는 것을 의미한다.

THP 설정 확인 방법은 아래와 같다.

① 아래 명령을 실행한다

```
$ cat /sys/kernel/mm/transparent_hugepage/enabled
```

② 레드햇 리눅스에서는 아래 명령을 실행한다

```
$ cat /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

③ 아래와 같은 결과가 화면에 출력된다.

```
$ [always] madvise never
```

THP 비활성화 방법

Altibase의 운영을 위해서 THP 옵션을 never로 설정할 것을 권고한다.

① root 계정으로 /etc/grub.conf의 kernel boot 끝에 transparent_hugepage=never를 아래처럼 추가한다.

```
.....
kernel /vmlinuz-2.6.32-220.el6.x86_64 ro root=UUID=067b9803-90ca-4875-a018-ff043adde1ed rd_NO_LUKS
LANG=ko_KR.UTF-8 rd_NO_MD quiet rhgb crashkernel=128M KEYBOARDTYPE=pc KEYTABLE=us rd_NO_LVM
rd_NO_DM transparent_hugepage=never
......
```

② 시스템을 재시작 한다.

③ THP 옵션이 never 인지 확인한다.

4) CPU frequency Governor

Altibase에서 권고하는 CPU frequency Governor 설정은 performance이다.

설정 값 확인

```
$ cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor | sort -u
performance
  # CPUfreq 드라이버가 설치되어 있지 않은 경우 아래와 같은 출력 결과를 보일 수 있다.
  # 이 경우 CPU frequency Governor 설정은 고려하지 않아도 된다.
$ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
cat: /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor: 그런 파일이나 디렉터리가 없습니다
```

설정 값 변경 방법

```
$ echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
$ echo performance > /sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
......
or
$ cpupower frequency-set -g performance
```

영구 적용

```
/etc/rc.d/rc.local 파일에 CPU 코어 수만큼 아래 명령어를 추가한다.
echo performance > /sys/devices/system/cpu/cpu0/cpufreq/scaling_governor
echo performance > /sys/devices/system/cpu/cpu1/cpufreq/scaling_governor
......
```

5) swappiness

swapping 발생 시 디스크 I/O로 인한 시스템 성능 저하가 Altibase 서버 성능에 영향을 미치는 것을 최소화 하기 위해 1로 변경하는 것을 권장한다.

설정 값 확인

```
$ cat /proc/sys/vm/swappiness
 or
$ sysctl -a | grep swappiness
```

설정 값 변경 방법

```
$ echo 1 > /proc/sys/vm/swappiness
or
$ sysctl -w vm.swappiness=1
```

영구 적용

```
$ vi /etc/sysctl.conf
vm.swappiness = 1                    # sysctl.conf 파일에 vm.swappiness = 1 를 추가하거나 값을 변경한다.
or
$ vi /etc/rc.d/rc.local
echo 1 > /proc/sys/vm/swappiness     # rc.local 파일에 이 명령어를 추가한다.
```

5) OS Patch

glibc에서 malloc/free 등이 race condition으로 인해 deadlock이 발생할 수 있는 버그가 있어, 해당 버그가 반영된 패치 이상으로 패치 해야 한다. 따라서, glibc-2.12-1.166.el6_7.1 이상으로 glibc 패치를 권고한다.

(참고: https://bugzilla.redhat.com/show_bug.cgi?id=1244002)

6) 참고 자료

Altibase 운영을 위한 Linux 설정 가이드 : https://docs.altibase.com/x/RQbN

Installation Guide :

https://github.com/ALTIBASE/Documents/blob/master/Manuals/Altibase_7.1/kor/Installation.md

1) 설치 모드

제품을 설치하기 위해 Altibase 패키지 인스톨러는 다음 두 가지 모드로 시작할 수 있다.

- 대화형 커맨드 라인 모드: DISPLAY 환경변수를 설정하지 않았을 때 이 모드로 실행된다.
- GUI (Graphical User Interface): DISPLAY 환경변수를 설정했을 때 이 모드로 실행된다.

이 문서에서 Altibase 패키지 인스톨러는 설치 모드를 대화형 커맨드 라인 모드로 설치하는 방법에 대해 설명한다.

2) Altibase 패키지 인스톨러 시작

Altibase 패키지 인스톨러를 다운로드 한 후 chmod로 파일의 실행 권한을 변경해야 한다.

```
$ chmod +x altibase-OE-server-7.1.0.3.5-LINUX-X86-64bit-release.run
```

Altibase 패키지 인스톨러를 실행한다.

```
$ ./altibase-OE-server-7.1.0.3.5-LINUX-X86-64bit-release.run
```

3) 설치 디렉터리 입력 및 설치 타입 선택

Altibase 홈 디렉터리, 즉 Altibase가 설치될 디렉터리와 패키지 설치 타입을 선택한다.

```
$ ./altibase-OE-server-7.1.0.3.5-LINUX-X86-64bit-release.run
--------------------------------------------------------------------------------
Welcome to the Altibase Server 7.1.0.3.5 setup wizard.
--------------------------------------------------------------------------------
Installation Directory

Please specify the installation directory for Altibase Server 7.1.0.3.5

Installation directory [/data/altibase/altibase_OE]: /data/altibase/altibase_OE

Please select the installation type.

Installation type

[1] Full installation: full package install
[2] Patch: patch package install
Please choose an option [1] : 1
```

4) Altibase 프로퍼티 설정

Altibase 프로퍼티를 설정하는 단계는 다음의 세 단계로 구분된다.

Step 1: Basic Database Operation Properties
Step 2: Database Creation Properties
Step 3: Set Database Directories

Step 1: Basic Database Operation Properties

```
Step 1: Basic Database Operation Properties

Database name [mydb]: mydb

Altibase connection port number (1024-65535)      [20300]: 21456

Maximum size of memory database
- MIN value: 16M (K = kB, M = MB, G = GB) [2G]: 2G

Buffer area size for caching disk-based database pages
- MIN value: 1M (K = kB, M = MB, G = GB) [128M]: 512M

Do you want to create a database after the installation process is complete?
[1] YES
[2] NO
Please choose an option [1] : 1
```

Step 2: Database Creation Properties

```
Step 2: Database Creation Properties

Initial database size
- 4M-2G (K = kB, M = MB, G = GB) [10M]: 10M

Database archive logging mode
[1] No archivelog
[2] Archivelog
Please choose an option [1] : 1

Database character set
[1] UTF-8
[2] MS949
[3] US7ASCII
[4] KO16KSC5601
[5] BIG5
[6] GB231280
……
Please choose an option [1] : 2

National character set
[1] UTF-8
[2] UTF-16
Please choose an option [1] : 1
```

Step 3: Database Creation Properties

이 단계에 포함되지 않은 프로퍼티의 값을 설정하거나, 단계1 또는 단계 3에서 설정한 프로퍼티 값을 나중에 변경하려면, $ALTIBASE_HOME/conf/altibase.propertites 파일을 직접 편집하면 된다.

```
Step 3: Set Database Directories

Default disk database directory [/data/altibase/altibase_OE/dbs]: /data/altibase/altibase_OE/dbs
Memory database directory [/data/altibase/altibase_OE/dbs]: /data/altibase/altibase_OE/dbs
Archive log directory [/data/altibase/altibase_OE/arch_logs]: /data/altibase/altibase_OE/arch_logs
Transaction log directory [/data/altibase/altibase_OE/logs]: /data/altibase/altibase_OE/arch_logs

[Log Anchor file directories ]
Altibase maintains three sets of log anchor files. These files contain important
information
about the database. By default, they are located in the "logs" folder.
The location can be changed here or by modifying the contents of the Altibase
properties file,
which is named "altibase.properties".

Directory 1. [/data/altibase/altibase_OE/logs]: /data/altibase/altibase_OE/logs
Directory 2. [/data/altibase/altibase_OE/logs]: /data/altibase/altibase_OE/logs
Directory 3. [/data/altibase/altibase_OE/logs]: /data/altibase/altibase_OE/logs
```

5) Altibase 프로퍼티 확인

```
Property Review

Please check your property settings.

To change these properties after installation is complete,
please modify the following file:
  /data/altibase/altibase_OE/conf/altibase.properties.

1. Altibase Property Settings:
    Step 1: Basic Database Operation Properties
    1) Database name:
        [mydb]
    2) Altibase connection port number (1024-65535):
        [21456]
    3) Maximum size of memory database:
        [2G]
    4) Buffer area size for caching disk-based database pages:
        [512M]

2. Altibase Property Settings:
Press [Enter] to continue :
    Step 2: Database Creation Properties
    1) Initial database size
        [10M]
……
```

6) Altibase 제품 설치

설치가 완료되면, 인스톨러는 아래의 작업들을 수행한다.

- 설정한 프로퍼티가 altibase.properties 파일에 업데이트 된다.
- Altibase 서버 구동을 위한 기본 환경이 수록되어 있는 altibase_user.env 파일이 $ALTIBASE_HOME/conf 디렉터리에 생성된다. 그리고 이 파일을 실행하는 명령어가 사용자 계정의 환경 설정 파일(.bashrc 또는 .bash_profile 또는 .profile 등)에 추가된다.

```
Setup is now ready to install Altibase Server 7.1.0.3.5.

Do you want to continue? [Y/n]: Y

-------------------------------------------------------------------------------
Please wait until the setup wizard finishes installing Altibase Server
7.1.0.3.5.

 Installing
 0% _____________ 50% _____________ 100%
 ##############################################
```

7) Altibase 빠른 설정 가이드 미리 보기

Altibase 빠른 설정 가이드 패널은 설치 성공 후에 Altibase를 어떻게 구동할 것인지를 안내한다.
Altibase 패키지 인스톨러는 사용자들이 좀 더 쉽게 시스템 커널 파라미터와 환경 변수를 설정하도록 다음 두 개의 쉘 스크립트를 제공한다.

- $ALTIBASE_HOME/install/pre_install.sh 이 스크립트는 필수 시스템 커널 파라미터의 최소 집합을 포함하며, 그들의 권장 값과 어떻게 설정하는지에 대해서 설명한다.
- $ALTIBASE_HOME/install/post_install.sh 이 스크립트는 Altibase 프로퍼티 설정 과정에서 단계2를 수행했다면 새로운 데이터베이스를 생성하기 위한 SQL 스크립트를 포함한다.
- $ALTIBASE_HOME/packages/catproc.sql 이 스크립트는 PSM을 사용하기 위한 SQL 스크립트를 포함한다.

```
[ Quick Guide to Making Settings in Altibase ]

1. Set kernel variables using the root user account.
   run the '/data/bluetheme/altibase_OE/install/pre_install.sh' file
   - This script helps you make kernel parameter settings.

================ LINUX ================
[ How to modify kernel parameter values ]

echo 512 32000 512 512 > /proc/sys/kernel/sem
echo 872415232 > /proc/sys/kernel/shmall

# shmall
If it is desired to use Altibase in shared memory mode, the value of 'shmall'
must be set. This value determines the maximum size of an Altibase database.
......
```

8) Altibase 설치 완료

Post_install.sh 와 catproc.sql 을 수행 여부에 따라 완료 과정에서 모두 수행되거나 설치 완료 후 필요한 작업만 직접 수행할 있다.

```
Would you like to launch 'post_install.sh', 'catproc.sql' now ?
 - This will create the Altibase database. Runs all scripts required for or used
with PSM [Y/n]: Y
......
DB Info (Page Size     = 32768)
        (Page Count    = 257)
        (Total DB Size = 8421376)
        (DB File Size  = 1073741824)

        Creating MMDB FILES     [SUCCESS]

        Creating Catalog Tables [SUCCESS]

Press [Enter] to continue :
        Creating DRDB FILES     [SUCCESS]

  [SM] Rebuilding Indices [Total Count:0]  [SUCCESS]

DB Writing Completed. All Done.

Create success.
......
```

5. 기능소개

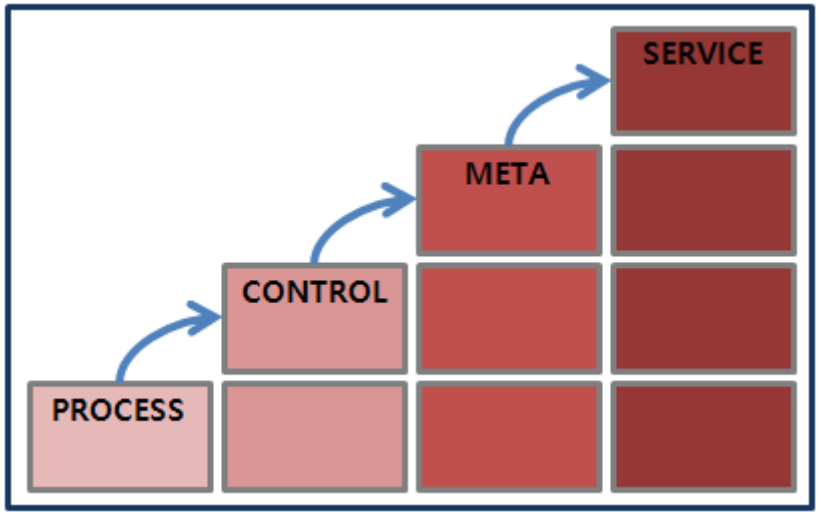
세부 목차

1) Altibase 구동

DBMS 구동 과정이 한 단계가 되면, DBMS가 구동되기 전 할 수 있는 작업과 구동 후 할 수 있는 작업이 엄격히 구분되어 한 번의 명령어 실행으로 DBMS가 구동되기 때문에 DBA가 클라이언트와 구별된 작업을 수행할 수 없다. 이로 인해 메타 마이그레이션, 복구 기능에 대한 제약 사항이 발생하게 된다.
이러한 문제점을 해결하기 위해 Altibase는 DB가 구동되는 단계를 PROCESS, CONTROL, META, SERVICE 의 4 단계로 구분하여 제공한다.
각 단계별로 DBA가 DB의 생성, 메타 업그레이드, 선택적 복구 등의 작업이 가능하다.

Altibase를 구동할 때, 각 단계로 전이하기 위해서는 "STARTUP" 이라는 명령어를 사용하게 되는데, 이 명령어는 iSQL에 SYSDBA 권한으로 접근한 후에 사용할 수 있다.

각 단계로 상태를 변경할 때, 다음 단계로의 전이는 가능하지만 이전 단계로의 복귀는 지원하지 않는다.



- PROCESS 단계

  Altibase를 구동시킬 때의 처음 단계로, “STARTUP PROCESS” 명령어를 통하여 상태를 전이할 수 있다. 이 단계에서는 DBMS에 대한 접근이 불가능하다.PROCESS 단계는 각 모듈(Storage Manager, Query Processor, Communication Manager 등)이 초기화되는 단계이며, 서버와 iSQL이 통신을 하기 위해 Altibase 프로세스를 시작하게 된다.

  - PROCESS 단계에서 수행할 수 있는 작업
    - ✓ CREATE DATABASE 구문을 사용하여 데이터베이스 생성

```
CREATE DATABASE db_name INITSIZE=10M NOARCHIVELOG CHARACTER SET UTF8 NATIONAL CHARACTER SET UTF8;
```

    - ✓ 프로퍼티 값을 변경(iSQL 에서 ALTER 명령어로 변경 가능한 프로퍼티)
    - ✓ 사용자가 조회활 수 있는 성능 뷰 (V$TABLE, V$VERSION, V$PROPERTY, ……)
    - ✓ CONTROL 단계 등의 상위 단계로 전이

- CONTROL 단계

  CONTROL 단계는 PROCESS 단계에서의 작업을 마치고 "STARTUP CONTROL" 명령어를 통해서 상태를 전이할 수 있다.

  해당 단계는 DB의 복구가 가능한 수준까지 각 모듈들을 초기화하고, Storage Manager 모듈을 구성하는 주요 관리자(Buffer, Recovery, Disk 등)를 준비하는 단계이다.

  - CONTROL 단계에서 수행할 수 있는 작업
    - ✓ 백업을 이용한 복구 수행

      ```
      iSQL> ALTER DATABASE RECOVER DATABASE UNTIL TIME '2020-10-30:18:00:00';
      ```

    - ✓ 온라인 로그파일 초기화
    - ✓ 데이터베이스 모드 변경

      ```
      iSQL> ALTER DATABASE ARCHIVELOG;
      ```

    - ✓ 성능 뷰 조회(V$BACKUP_INFO, V$DATAFILES, V$LOG……)
    - ✓ META 단계 등의 상위 단계로 전이

- META 단계

  META 단계는 CONTROL 단계에서의 작업을 마치고 "STARTUP META " 명령어를 통해서 상태를 전이할 수 있다.

  해당 단계는 CONTROL 단계에서 작업한 복구 과정에 대한 완료 단계이며, DBMS 구동을 위해 메모리와 디스크의 데이터 파일들을 체크하고 Restart Recovery 를 수행하는 단계이다.

  - META 단계에서 수행할 수 있는 작업
    - ✓ 모든 성능 뷰 조회
    - ✓ SERVICE 단계로 전이

- SERVICE 단계

  SERVICE 단계는 META 단계에서의 작업을 마치고 "STARTUP SERVICE” 또는 “STARTUP" 명령어를 통해서 상태를 전이할 수 있다. 사용자가 DBMS 를 사용할 수 있는 최종 단계를 의미하며 일반 사용자가 접속할 수 있는 상태가 된다.

  - SERVICE 단계에서 수행할 수 있는 작업
    - ✓ 외부 접속 등 정상적인 서비스가 가능한 상태

1) Altibase 중지

중지는 구동에서 수행한 과정을 역순으로 진행하며 1단계로 종료한다.(SERVICE → META → CONTROL → PROCESS)

현재 구동 중인 Altibase 를 종료하기 위해서는 iSQL에 -sysdba 권한으로 접속하여 "SHUTDOWN" 명령어를 사용한다. "SHUTDOWN" 명령어는 세 가지 옵션(NORMAL, IMMEDIATE, ABORT)을 지정하여 사용할 수 있고 NORMAL, IMMEDIATE 는 Altibase 가 SERVICE 상태일 때만 수행가능하며, ABORT 는 어떤 상태에서도 수행이 가능하다.

Altibase의 중지 명령어는 설치한 OS 계정으로만 수행이 가능하다.

- NORMAL
  Altibase 를 정상적으로 종료하는 방식으로서, Altibase 는 모든 클라이언트들이 접속을 끊을 때까지 종료 작업을 대기한다.

- IMMEDIATE
  현재 연결된 세션들을 강제로 단절시킨 후 Altibase 에서 수행중인 모든 트랜잭션을 롤백 후 종료한다.

- ABORT
  Altibase 를 kill -9 시스템 명령을 사용하여 강제로 종료하는 방법으로 데이터베이스의 일관성에 문제가 발생할 수 있어 이후 구동 때 restart recover 과정을 거치게 된다.

5.2 테이블스페이스 생성 (1/3)

1) 메모리 테이블스페이스 생성

데이터베이스 생성시 기본적인 메모리 테이블스페이스가 생성되지만 필요에 따라 사용자가 생성할 수 있으며 SYS 계정 또는 CREATE TABLESPACE 권한을 가진 계정에서 가능하다.

- 메모리 테이블스페이스 생성 구문

```
CREATE MEMORY [DATA] TABLESPACE tablespace_name
SIZE size (K | M | G)
[AUTOEXTEND [ON [NEXT size] [MAXSIZE size] | OFF) ]
[CHECKPOINT PATH 'path' [SPLIT EACH size]] ;
```

- 메모리 테이블스페이스 생성 예제

초기 크기가 512MB, 최대 1GB까지 128MB 단위로 자동 확장, 이미지 파일 하나의 크기는 256MB이며 3개의 디렉터리에 나누어 저장하는 예제

```
iSQL> CREATE MEMORY TABLESPACE test_mem
      2 SIZE 512M
      3 AUTOEXTEND ON NEXT 128M MAXSIZE 1G
      4 CHECKPOINT PATH '/dbs/path1', '/dbs/path2', '/dbs/path3'
      5 SPLIT EACH 256M ;
Create success.
```

2) 휘발성 메모리 테이블스페이스 생성 (VOLATILE TABLESPACE)

데이터가 메모리에만 상주하고 체크포인트 이미지 파일을 가지지 않는 메모리 테이블 스페이스다. 체크포인트와 로깅의 최소화로 성능을 극대화 할 수 있으나 해당 테이블 스페이스의 데이터들은 DBMS 가 중지되면 사라지게 된다. SYS 계정 또는 CREATE TABLESPACE 권한을 가진 계정에서 가능하다.

- 메모리 테이블스페이스 생성 구문

```
CREATE VOLATILE [DATA] TABLESPACE tablespace_name
SIZE size (K | M | G)
[AUTOEXTEND [ON [NEXT size] [MAXSIZE size] | OFF) ];
```

- 휘발성 메모리 테이블스페이스 생성 예제

  초기 크기가 512MB이고, 128MB 단위로 자동 확장되는 휘발성 데이터 테이블스페이스를 생성

```
iSQL> CREATE VOLATILE DATA TABLESPACE user_data SIZE 512M AUTOEXTEND ON NEXT 128M;
Create success.
```

3) 디스크 테이블스페이스 생성

데이터베이스 생성시 기본적인 디스크 테이블스페이스가 생성되지만 필요에 따라 사용자가 생성할 수 있으며 SYS 계정 또는 CREATE TABLESPACE 권한을 가진 계정에서 가능하다.

- 디스크 테이블스페이스 생성 구문

```
CREATE [DISK] [DATA] TABLESPACE tablespace_name
DATAFILE 'datafile_name '
[SIZE size (K | M | G) ] [REUSE]
[AUTOEXTEND [ON [NEXT size][MAXSIZE size] | OFF)];
```

- 디스크 테이블스페이스 생성 예제

  초기 크기가 100MB, 최대 2GB까지 10MB 단위로 자동 확장, 데이터 파일 test01.dbf, test02.dbf, test03.dbf 로 구성 된 테이블 스페이스 생성 예제

```
iSQL> CREATE TABLESPACE test_disk
    2 DATAFILE 'test01.dbf', 'test02.dbf',  'test03.dbf'
    3 SIZE 100M AUTOEXTEND ON NEXT 10M MAXSIZE 2G ;
Create success.
```

SYS 사용자이거나 사용자 자신의 스키마에 테이블을 생성하려면 CREATE TABLE 또는 CREATE ANY TABLE, 다른 사용자의 스키마에 테이블을 생성하려면 CREATE ANY TABLE 시스템 권한이 있어야 한다. 구문에 테이블 스페이스를 지정할 수 있으며 생략할 경우 테이블은 이 테이블을 생성하려 하는 사용자의 기본 테이블스페이스에 생성된다. 사용자 생성 시 DEFAULT TABLESPACE를 생략했었다면 테이블은 시스템 메모리 기본 테이블스페이스(SYSTEM MEMORY DEFAULT TABLESPACE)에 생성된다.

- 메모리 테이블스페이스 생성 구문

```
iSQL> CREATE TABLE employees(
  eno INTEGER PRIMARY KEY,
  e_lastname CHAR(20) NOT NULL,
  e_firstname CHAR(20) NOT NULL,
  emp_job VARCHAR(15),
  emp_tel CHAR(15),
  dno SMALLINT,
  salary NUMBER(10,2) DEFAULT 0,
  sex CHAR(1) CHECK(sex IN ('M', 'F')),
  birth CHAR(6),
  join_date DATE,
  status CHAR(1) DEFAULT 'H') TABLESPACE SYS_TBS_MEM_DATA;
Create success.
```

- 임시 테이블 생성

```
iSQL> CREATE VOLATILE TABLESPACE my_vol_tbs size 12M AUTOEXTEND ON MAXSIZE 1G;
Create success.
iSQL> CREATE TEMPORARY TABLE t1(
  i1 INTEGER,
  i2 VARCHAR(10)) ON COMMIT DELETE ROWS TABLESPACE my_vol_tbs;
Create success.
iSQL> CREATE TEMPORARY TABLE t2(
  i1 INTEGER,
  i2 VARCHAR(10)) ON COMMIT PRESERVE ROWS TABLESPACE  my_vol_tbs;
Create success.
```

- 디스크 테이블 생성

```
iSQL> CREATE TABLE books(
  isbn CHAR(10) CONSTRAINT const1 PRIMARY KEY,
  title VARCHAR(50),
  author VARCHAR(30),
  edition INTEGER DEFAULT 1,
  publishingyear INTEGER,
  price NUMBER(10,2),
  pubcode CHAR(4)) MAXROWS 2 TABLESPACE SYS_TBS_DISK_DATA;
Create success.
```

- LOB 데이터를 별도의 테이블스페이스에 저장하는 테이블 생성

```
CREATE TABLE lob_products
(
  product_id integer,
  image1 BLOB,
  image2 BLOB
) TABLESPACE SYS_TBS_DISK_DATA
LOB(image1) STORE AS ( TABLESPACE lob_data1 )
LOB(image2) STORE AS ( TABLESPACE lob_data2 );
```

- 파티션 테이블 : 파티션의 테이블스페이스를 지정하여 파티션드 테이블 생성

```
CREATE TABLE T1
(
  I1 INTEGER,
  I2 INTEGER
)
PARTITION BY RANGE (I1)
(
  PARTITION P1 VALUES LESS THAN (100),
  PARTITION P2 VALUES LESS THAN (200) TABLESPACE TBS1,
  PARTITION P3 VALUES DEFAULT TABLESPACE TBS2
) TABLESPACE SYS_TBS_DISK_DATA;
```

- 파티션 테이블 : 범위 파티셔닝(range partitioning)

```
CREATE TABLE range_sales
(
  prod_id NUMBER(6),
  cust_id NUMBER,
  time_id DATE
)
PARTITION BY RANGE (time_id)
(
  PARTITION Q1_2006 VALUES LESS THAN (TO_DATE('01-APR-2006')),
  PARTITION Q2_2006 VALUES LESS THAN (TO_DATE('01-JUL-2006')),
  PARTITION Q3_2006 VALUES LESS THAN (TO_DATE('01-OCT-2006')),
  PARTITION Q4_2006 VALUES LESS THAN (TO_DATE('01-JAN-2007')),
  PARTITION DEF VALUES DEFAULT
) TABLESPACE SYS_TBS_DISK_DATA;
```

- 파티션 테이블 : 리스트 파티셔닝(list partitioning)

```
CREATE TABLE list_customers
(
  customer_id NUMBER(6),
  cust_first_nameVARCHAR(20),
  cust_last_name VARCHAR(20),
  nls_territory VARCHAR(30),
  cust_email    VARCHAR(30)
)
PARTITION BY LIST (nls_territory)
(
  PARTITION asia VALUES ('CHINA', 'THAILAND'),
  PARTITION europe VALUES ('GERMANY', 'ITALY', 'SWITZERLAND'),
  PARTITION west VALUES ('AMERICA'),
  PARTITION east VALUES ('INDIA'),
  PARTITION rest VALUES DEFAULT
) TABLESPACE SYS_TBS_DISK_DATA;
```

- 파티션 테이블 : 해시 파티셔닝(hash partitioning)

```
CREATE TABLE hash_products
(
    product_id          NUMBER(6),
    product_name        VARCHAR(50),
    product_description     VARCHAR(2000)
)
PARTITION BY HASH (product_id)
(
    PARTITION p1,
    PARTITION p2,
    PARTITION p3,
    PARTITION p4
) TABLESPACE SYS_TBS_DISK_DATA;
```

- 파티션 테이블 : HYBRID PARTITIONED TABLE

```
CREATE TABLE part_table
(
  sales_date DATE,
  sales_id NUMBER,
  sales_city VARCHAR(20)
)
PARTITION BY LIST(sales_city)
(
  PARTITION part_1 VALUES ( 'SEOUL' , 'INCHEON' ) TABLESPACE mem_tbs_0,
  PARTITION part_2 VALUES ( 'PUSAN' , 'JUNJU' ) TABLESPACE disk_tbs_1,
  PARTITION part_3 VALUES ( 'CHUNGJU' , 'DAEJUN' ) TABLESPACE mem_tbs_2,
  PARTITION part_def VALUES DEFAULT TABLESPACE disk_tbs_3
) TABLESPACE mem_tbs_4;
```

1) 프로퍼티 설정

이중화를 사용하기 위해서는 $ALTIBASE_HOME/conf/altibase.properties 중 서버에서 사용할 수 있는 포트를 선정하여 REPLICATION_PORT_NO 를 지정해야 한다. 이외 REPLICATION_ 로 시작하는 프로퍼티들은 관련 매뉴얼을 참고하여 필요에 따라 설정하면 된다.

```
#============================================================
# Replication Properties
#============================================================
REPLICATION_PORT_NO              = 30300 # REPLICATION_PORT_NO
REPLICATION_IB_PORT_NO           = 0 # REPLICATION_IB_PORT_NO
......
```

2) 이중화 생성 구문

```
CREATE [LAZY|EAGER] REPLICATION replication_name
[FOR ANALYSIS | FOR PROPAGABLE LOGGING | FOR PROPAGATION | FOR ANALYSIS PROPAGATION]
[AS MASTER|AS SLAVE]
[OPTIONS options_name [option_name ... ] ]
WITH { 'remote_host_ip', remote_host_ port_no [USING conn_type [ib_latency]]}
...
FROM user_name.table_name [PARTITION partition_name] TO user_name.table_name [PARTITION partition_name]
...;
```

3) 이중화 생성 예제

```
CREATE TABLE foo (a DOUBLE PRIMARY KEY, b TIMESTAMP);
CREATE TABLE bar (a DOUBLE PRIMARY KEY, b CHAR(3));
CREATE REPLICATION rep WITH '127.0.0.1', 30300
FROM sys.foo TO sys.foo,
FROM sys.bar TO sys.bar;
```

4) 이중화 구동/중지

```
# rep 이름의 이중화 시작
ALTER REPLICATION rep START;

# rep 이름의 이중화 중지
ALTER REPLICATION rep STOP;
```

6. 활용예제

세부 목차

6.1 ALTIBASE + Spring JPA + Hibernate 연동

1. 사전 준비 사항

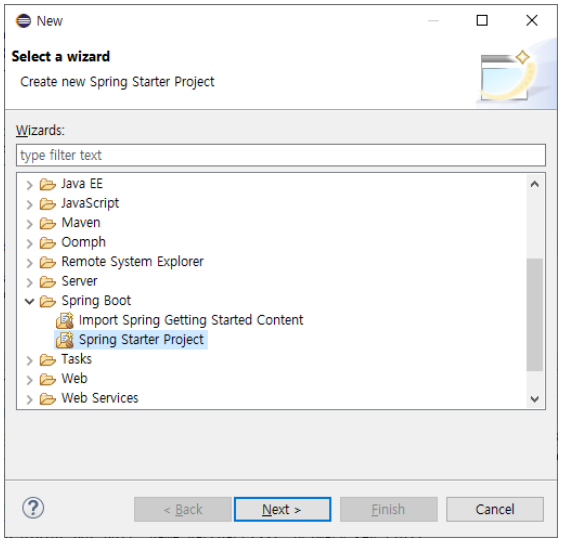
- DEV Tool : eclipse IDE
- STS (Spring Tool Suit) 4 : 4.5.1 (Eclipse Marketplace download)
- ALTIBASE : 7.1.0
- Hibernate : 5.2.17 (5.2 전용 AltibaseDialect 추가)

AltibaseDialect 추가에 대한 참고 URL : https://github.com/ALTIBASE/hibernate-orm/blob/master/ALTIBASE_DIALECT_PORTING.md

2. Spring Starter Porject 생성

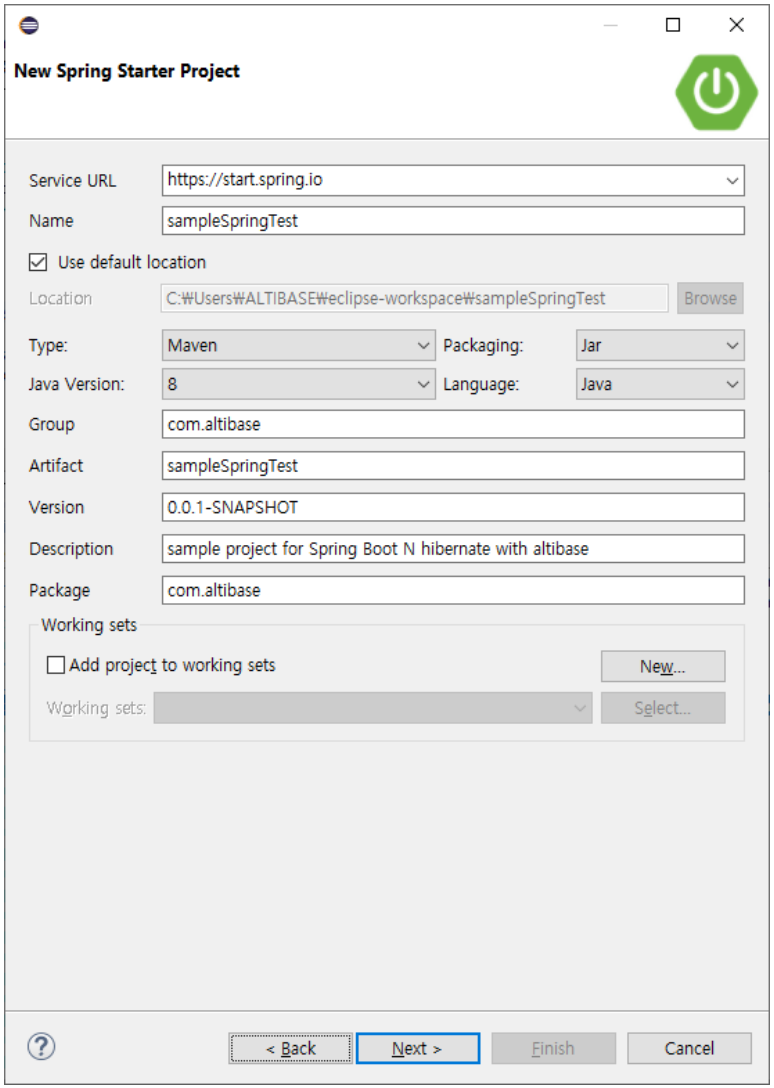
- File > New > Other... 선택 후 Spring Starter Project 선택 > Next
- 다음의 정보를 입력하고 Next

```
Name : sampleSpringTest
packaging : jar
java version : 8
Language : java
package : com.altibase
```



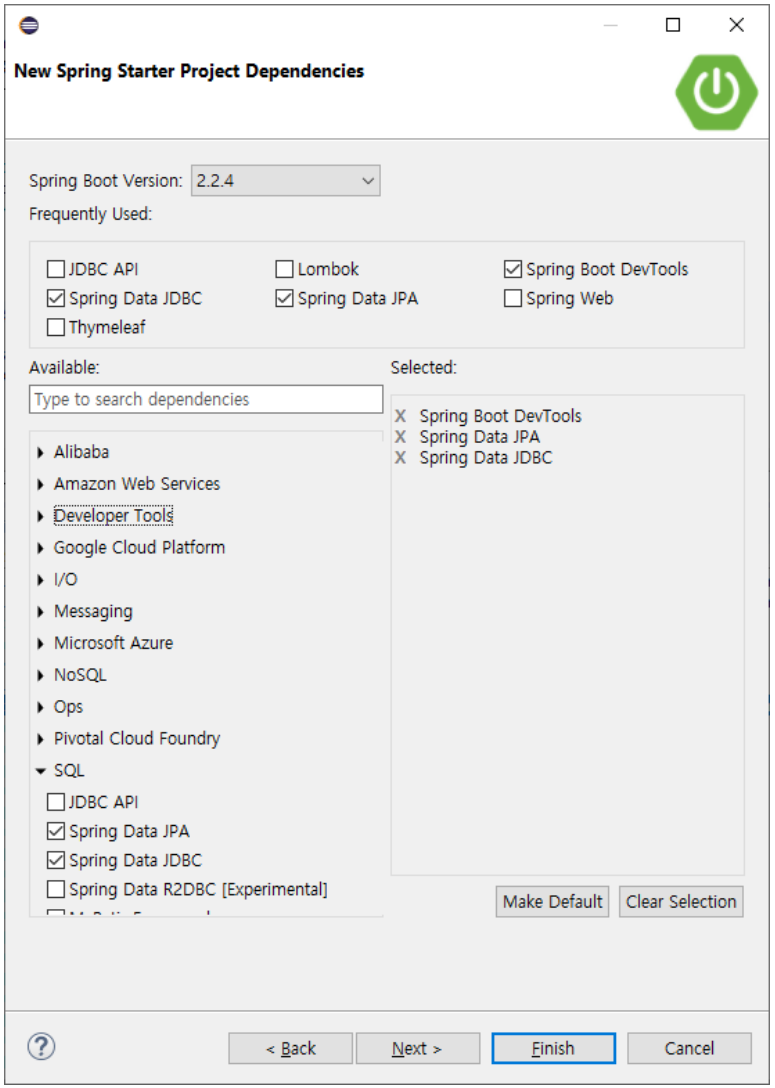
- 다음의 정보를 입력하고 Next

Name : sampleSpringTest
packaging : jar
java version : 8
Language : java
package : com.altibase



- 다음 3개를 선택하고 Next

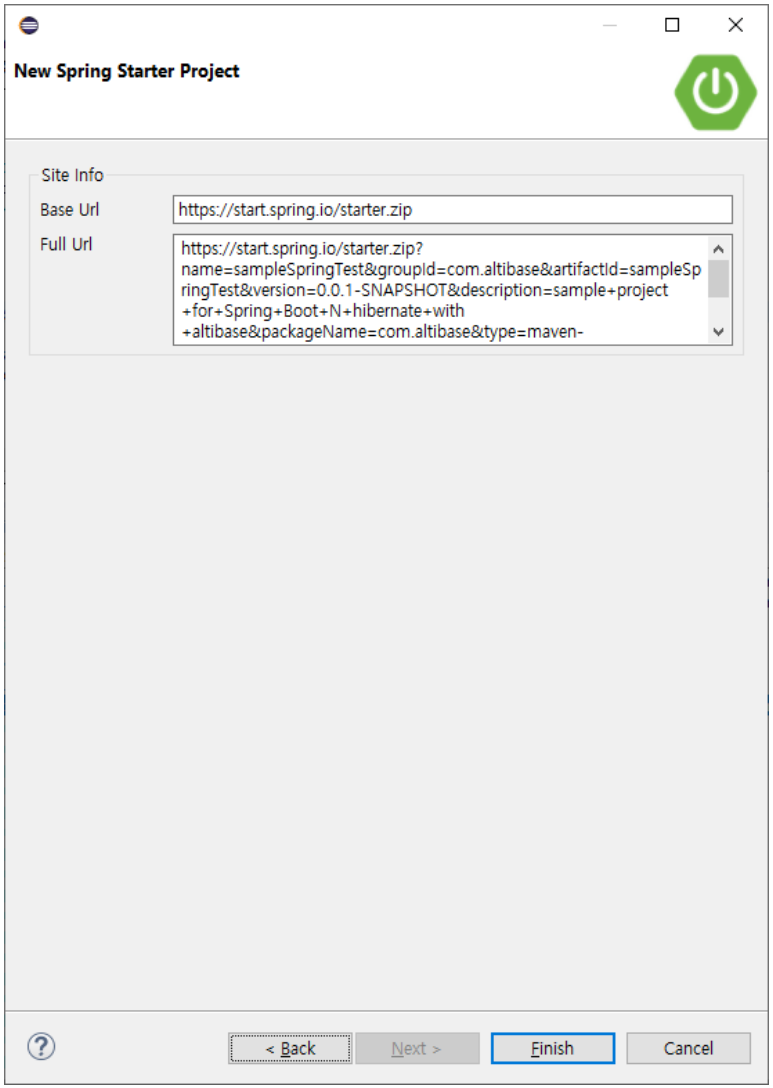
SQL > Spring Data JDBC, Spring Data JPA
Developer Tools > Spring Boot DevTools



6. 활용예제

6.1 ALTIBASE + Spring JPA + Hibernate 연동

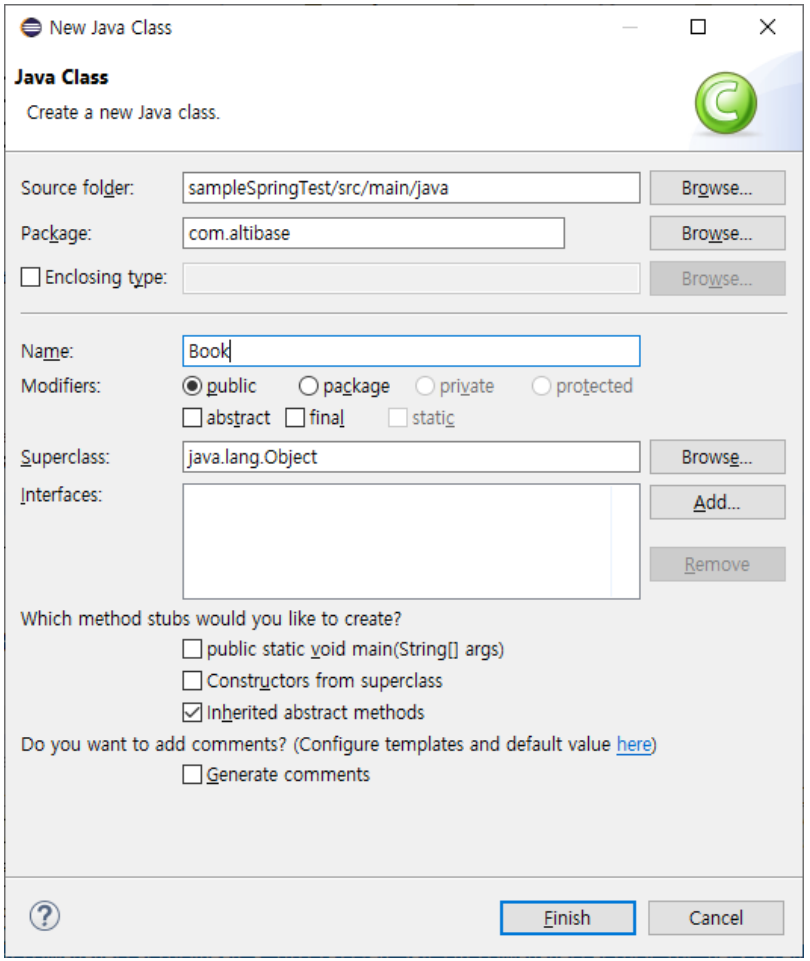
- Finish 로 생성



3. CLASS 생성

- 생성된 프로젝트 sampleSpringTest 를 오른쪽 클릭 > New > Class

Name : Book 입력 후 Finish



- /src/main/java/Book.java 를 다음과 같이 작성한다.

```
package com.altibase;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Book {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String name;

    public Book() {
    }
    public Book(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Book{" + "id=" + id + ", name='" + name + '\'' + '}';
    }
```

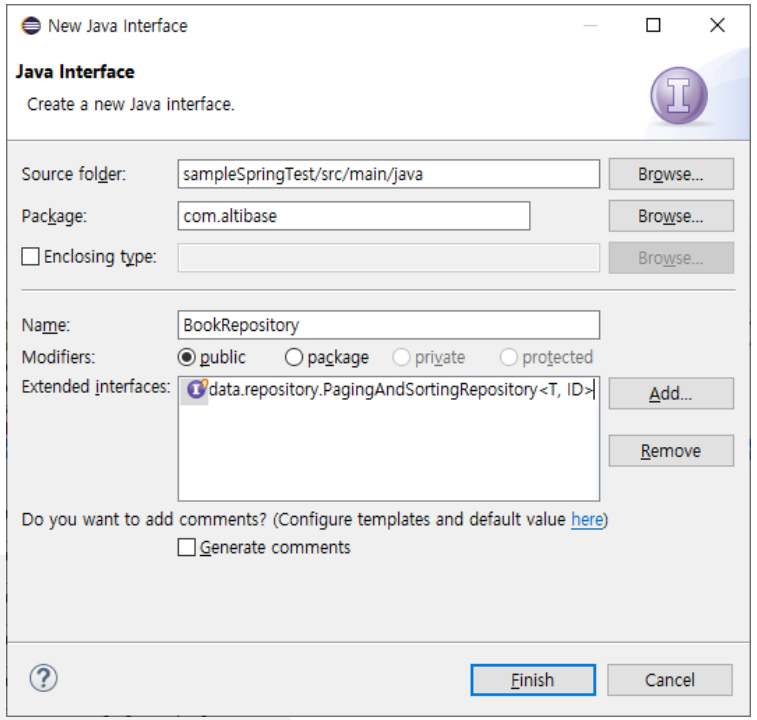
```
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
}
```

4. Repository Interface 생성

   PagingAndSortingRepository 를 상속한 BookRepository interface 를 작성한다.

- 생성된 프로젝트 sampleSpringTest 를 오른쪽 클릭 > New > Interface

Name : BookRepository
Extended interfaces : Add 클릭 > 'PagingAndSortingRepository' 검색 후 Add > OK > Finish



- /src/main/java/BookRepository.java 를 다음과 같이 작성한다.

```
package com.altibase;

import java.util.List;
import org.springframework.data.domain.Pageable;
import org.springframework.data.repository.PagingAndSortingRepository;

public interface BookRepository extends PagingAndSortingRepository<Book, Long> {
   List<Book> findByName(String name, Pageable pageable);
}
```

5. main 을 포함하는 SampleSpringTestApplication.java 수정

   최초 sampleSpringTest 프로젝트 생성시 만들어진 SampleSpringTestApplication.java 를 다음과 같이 작성한다.

```
package com.altibase;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import com.altibase.SampleSpringTestApplication;
import com.altibase.Book;
import com.altibase.BookRepository;
@SpringBootApplication
public class SampleSpringTestApplication implements CommandLineRunner {
    private static final Logger log = LoggerFactory.getLogger(SampleSpringTestApplication.class);
    @Autowired
    private BookRepository repository;

    public static void main(String[] args) {
        SpringApplication.run(SampleSpringTestApplication.class, args);
    }
```

```
@Override
public void run(String... args) {
    log.info("StartApplication...");
    repository.save(new Book("Java"));
    repository.save(new Book("Python"));
    repository.save(new Book("C++"));
    repository.save(new Book("Basic"));
    repository.save(new Book("Perl"));
    repository.save(new Book("Node"));
    repository.save(new Book("Node"));
    repository.save(new Book("Node"));
    repository.save(new Book("Node"));
    repository.save(new Book("Node"));
    repository.save(new Book("PHP"));
    repository.save(new Book("Node"));
    repository.save(new Book("Node"));
    repository.save(new Book("Node"));
    repository.save(new Book("Node"));
    repository.save(new Book("Node"));
    repository.save(new Book("Node"));
    System.out.println("\nfindAll()");
    repository.findAll().forEach(x -> System.out.println(x));
    System.out.println("\nfindById(1L)");
    repository.findById(1l).ifPresent(x -> System.out.println(x));
    System.out.println("\nfindById(4L)");
    repository.findById(4l).ifPresent(x -> System.out.println(x));
```

```
        Pageable firstPageWithTwoElements  = PageRequest.of(0, 2, Sort.by("id"));
        Pageable secondPageWithTwoElements = PageRequest.of(1, 2, Sort.by("id"));
        Pageable thirdPageWithTwoElements  = PageRequest.of(2, 2, Sort.by("id"));
        System.out.println("\nfindByName('Node') : 0,2");
        repository.findByName("Node", firstPageWithTwoElements).forEach(x -> System.out.println(x));
        System.out.println("\nfindByName('Node') : 1,2");
        repository.findByName("Node", secondPageWithTwoElements).forEach(x -> System.out.println(x));
        System.out.println("\nfindByName('Node') : 2,2");
        repository.findByName("Node", thirdPageWithTwoElements).forEach(x -> System.out.println(x));
    }
}
```

6. 수행 환경 설정

1) application.properties 를 다음과 같이 각자의 DB에 맞게 설정한다.

```
logging.level.org.springframework=INFO
logging.level.com.test=INFO
logging.level.com.zaxxer=DEBUG
logging.level.root=ERROR
#spring.datasource.hikari.connectionTimeout=20000
spring.datasource.hikari.maximumPoolSize=5
spring.datasource.hikari.connectionTestQuery=SELECT 1 FROM DUAL
logging.pattern.console=%-5level %logger{36} - %msg%n
```

```
## altibase
spring.datasource.driver-class-name=Altibase.jdbc.driver.AltibaseDriver
spring.datasource.url=jdbc:Altibase://192.168.1.145:21456/mydb
spring.datasource.username=sys
spring.datasource.password=manager
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.AltibaseDialect
spring.jpa.properties.hibernate.temp.use_jdbc_metadata_defaults=false
#drop n create table again, good for testing, comment this in production
spring.jpa.hibernate.ddl-auto=create
spring.jpa.show-sql=true
```

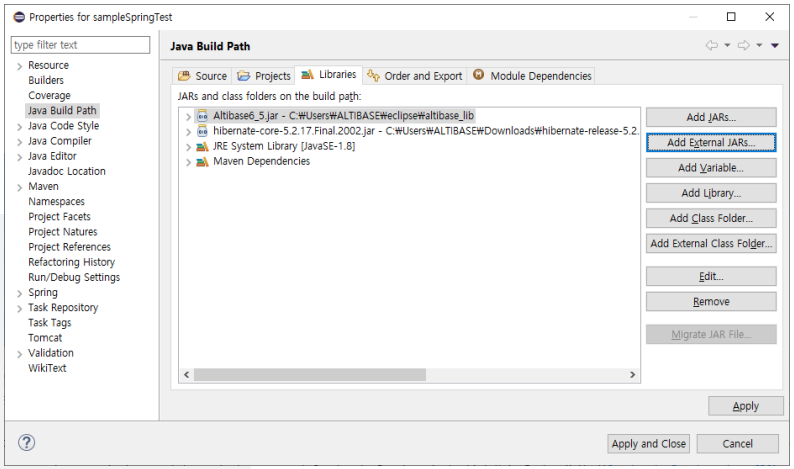
2) 필요한 jar 파일 build path 에 추가

SampleSpringTest 프로젝트의 Properties > Java Build Path > Libraries > Add External JARs...

AltibaseDialect 이 포함 된 hibernate core 파일과

ALTIBASE JDBC 드라이버를 추가한 후 Apply And Close

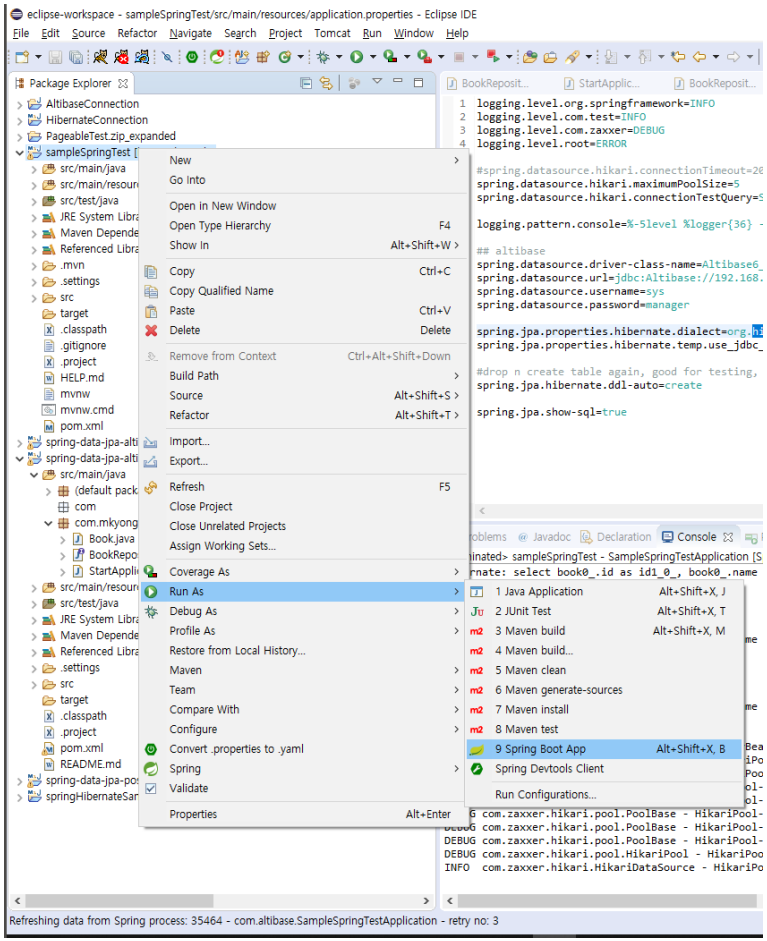
버튼을 클릭하여 적용한다.



❖ JDBC 드라이버의 경우 위 application.properties 에서 정의한 class name 을 사용할 수 있는 드라이버를 선택해야 한다. 알티베이스의 경우 여러 버전을 동시에 접근할 수 있도록 Altibase.jar 외 Altibase6_5.jar 등과 같이 추가 버전을 명시한 드라이버를 제공하고 있다.

7. 수행

SampleSpringTest 프로젝트 오른쪽 클릭 > Run As > Spring Boot App



[수행 결과]

```
penJDK 64-Bit Server VM warning: Options -Xverify:none and -noverify were deprecated in JDK 13 and will likely be removed in a future release.
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.2.4.RELEASE)
INFO  o.s.b.d.restart.ChangeableUrls - The Class-Path manifest attribute in C:\Users\ALTIBASE\.m2\repository\org\glassfish\jaxb\jaxb-runtime\2.3.2\jaxb-runtime-2.3.2.jar referenced one or more files that do not exist: file:/C:/Users/ALTIBASE/.m2/repository/org/glassfish/jaxb/jaxb-runtime/2.3.2/jakarta.xml.bind-api-2.3.2.jar,file:/C:/Users/ALTIBASE/.m2/repository/org/glassfish/jaxb/jaxb-runtime/2.3.2/txw2-2.3.2.jar,file:/C:/Users/ALTIBASE/.m2/repository/org/glassfish/jaxb/jaxb-runtime/2.3.2/istack-commons-runtime-3.0.8.jar,file:/C:/Users/ALTIBASE/.m2/repository/org/glassfish/jaxb/jaxb-runtime/2.3.2/stax-ex-1.8.1.jar,file:/C:/Users/ALTIBASE/.m2/repository/org/glassfish/jaxb/jaxb-runtime/2.3.2/FastInfoset-1.2.16.jar,file:/C:/Users/ALTIBASE/.m2/repository/org/glassfish/jaxb/jaxb-runtime/2.3.2/jakarta.activation-api-1.2.1.jar
.....
DEBUG com.zaxxer.hikari.HikariConfig - dataSourceJNDI.....none
DEBUG com.zaxxer.hikari.HikariConfig - dataSourceProperties.....{password=<masked>}
DEBUG com.zaxxer.hikari.HikariConfig - driverClassName.....Altibase6_5.jdbc.driver.AltibaseDriver"
DEBUG com.zaxxer.hikari.HikariConfig - healthCheckProperties.....{}
DEBUG com.zaxxer.hikari.HikariConfig - healthCheckRegistry.....none
DEBUG com.zaxxer.hikari.HikariConfig - idleTimeout.....600000
DEBUG com.zaxxer.hikari.HikariConfig - initializationFailTimeout.....1
DEBUG com.zaxxer.hikari.HikariConfig - isolateInternalQueries.....false
.....
Hibernate: select hibernate_sequence.nextval from dual
Hibernate: insert into book (name, id) values (?, ?)
Hibernate: select hibernate_sequence.nextval from dual
Hibernate: insert into book (name, id) values (?, ?)
Hibernate: select hibernate_sequence.nextval from dual
Hibernate: insert into book (name, id) values (?, ?)
Hibernate: select hibernate_sequence.nextval from dual
Hibernate: insert into book (name, id) values (?, ?)
Hibernate: select hibernate_sequence.nextval from dual
.....
Book{id=10, name='Node'}
Book{id=11, name='PHP'}
Book{id=12, name='Node'}
Book{id=13, name='Node'}
Book{id=14, name='Node'}
Book{id=15, name='Node'}
findById(1L)
Hibernate: select book0_.id as id1_0_0_, book0_.name as name2_0_0_ from book book0_ where book0_.id=?
Book{id=1, name='Java'}
Hibernate: select book0_.id as id1_0_0_, book0_.name as name2_0_0_ from book book0_ where book0_.id=?
Book{id=4, name='Basic'}
findByName('Node') : 0,2
Hibernate: select book0_.id as id1_0_, book0_.name as name2_0_ from book book0_ where book0_.name=? order by book0_.id asc limit ?
Book{id=6, name='Node'}
Book{id=7, name='Node'}
findByName('Node') : 1,2
Hibernate: select book0_.id as id1_0_, book0_.name as name2_0_ from book book0_ where book0_.name=? order by book0_.id asc limit ?, ?
Book{id=8, name='Node'}
Book{id=9, name='Node'}
findByName('Node') : 2,2
Hibernate: select book0_.id as id1_0_, book0_.name as name2_0_ from book book0_ where book0_.name=? order by book0_.id asc limit ?, ?
Book{id=10, name='Node'}
Book{id=12, name='Node'}
INFO  o.s.o.j.LocalContainerEntityManagerFactoryBean - Closing JPA EntityManagerFactory for persistence unit 'default'
INFO  com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown initiated...
DEBUG com.zaxxer.hikari.pool.HikariPool - HikariPool-1 - Before shutdown stats (total=5, active=0, idle=5, waiting=0)
DEBUG com.zaxxer.hikari.pool.PoolBase - HikariPool-1 - Closing connection Altibase6_5.jdbc.driver.AltibaseConnection@7222a187: (connection evicted)
DEBUG com.zaxxer.hikari.pool.PoolBase - HikariPool-1 - Closing connection Altibase6_5.jdbc.driver.AltibaseConnection@38fae96a: (connection evicted)
DEBUG com.zaxxer.hikari.pool.PoolBase - HikariPool-1 - Closing connection Altibase6_5.jdbc.driver.AltibaseConnection@3cc57aad: (connection evicted)
DEBUG com.zaxxer.hikari.pool.PoolBase - HikariPool-1 - Closing connection Altibase6_5.jdbc.driver.AltibaseConnection@79c7e351: (connection evicted)
DEBUG com.zaxxer.hikari.pool.PoolBase - HikariPool-1 - Closing connection Altibase6_5.jdbc.driver.AltibaseConnection@27aab64d: (connection evicted)
DEBUG com.zaxxer.hikari.pool.HikariPool - HikariPool-1 - After shutdown stats (total=0, active=0, idle=0, waiting=0)
INFO  com.zaxxer.hikari.HikariDataSource - HikariPool-1 - Shutdown completed.
```

Q ALTIBASE 관련 자료는 어디에 있나요?

A ALTIBASE 기술 지원 포탈 : http://support.altibase.com/kr/

ALTIBASE 기술 자료 : https://aid.altibase.com/dashboard.action

Q ALTIBASE 사용 중 성능이 느려지다가 프로세스가 사라졌습니다.

A 메모리 부족으로 인해 스왑을 사용하다가 Linux 시스템이 원활한 시스템 운영을 위해 많은 메모리를 사용하는 프로세스를 정리하는데 이 때 IN-Memory DBMS 인 ALTIBASE 는 OOM Killer에 의해 종료 될 확률이 굉장히 높다. 이를 해결하기 위해서는 시스템 메모리 증설을 권장합니다.

Open Source Software Installation & Application Guide