User Manual

오디세우스/COSMOS

Version 3.0

Manual Release 1

2016년 8월

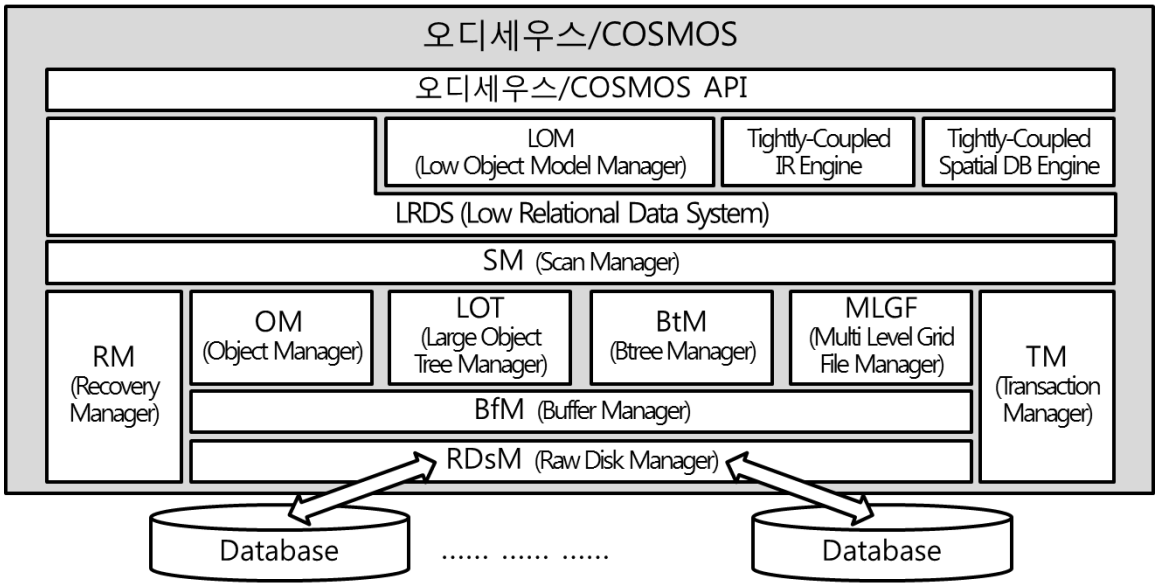
Advanced Information Technology Research Center (AITrc)

KAIST

목 차

1. 오디세우스/COSMOS

오디세우스/COSMOS는 텍스트 정보 검색 엔진 및 공간 DB 엔진과 밀결합(tight coupling)된 정보 검색용 저장 시스템으로서, 각종 데이터베이스 응용 소프트웨어의 하부 구조로 사용되는 핵심 소프트웨어이다. 아래 그림은 오디세우스/COSMOS의 아키텍처를 나타낸다. 오디세우스/COSMOS는 디스크를 관리하는 Raw Disk Manager(RDsM), 버퍼를 관리하는 Buffer Manager(BfM), 데이터를 객체 단위로 관리하는 Object Manager(OM), 멀티미디어 데이터와 같은 대용량 데이터를 관리하는 Large Object Tree Manger(LOT), 효율적인 검색 지원을 위한 Btree Manager(BtM), Multi Level Grid File Manager(MLGF), 파손 회복 지원을 위한 Recovery Manager(RM), 트랜잭션 지원을 위한 Transaction Manager(TM), 편리한 저장 시스템 사용 지원을 위한 Scan Manager(SM), 릴레이셔널 모델 지원을 위한 Low Relational Data System(LRDS), 객체 지향 모델 지원을 위한 Low Object Model Manager(LOM), 지리정보시스템을 위한 Tightly-Coupled Spatial DB Engine, 빠른 텍스트 정보 검색을 위한 Tightly-Coupled IR Engine, 그리고 편리한 사용을 위한 오디세우스/COSMOS API로 구성된다.



오디세우스/COSMOS는 Red Hat 사의 Linux 운영체제를 지원한다. 오디세우스/COSMOS는 C 언어를 사용하여 개발되었다. 오디세우스/COSMOS 약 178,000 라인의 규모를 가진다.

오디세우스/COSMOS는 볼륨 단위 locking을 사용하는 Coarse granularity locking 버전과 레코드 단위 locking을 사용하는 Fine granularity locking 버전 (2016년 8월 현재 release되지 않음) 이 있다. 두 버전은 서로 상이하게 구현되어 있으나 사용 방법은 큰 차이가 없으므로, 본 문서에서는 필요한 경우를 제외하고는 구분하여 설명하지 않는다.

오디세우스/COSMOS는 OS 환경, 사용 가능한 데이터베이스의 최대 용량 등에 따라 오디세우스/COSMOS 32bit, 오디세우스/COSMOS 64bit 대용량화, 오디세우스/COSMOS 64bit 비대용량화로 구분할 수 있다. 이에 대해서는 1.1절, 1.2절, 1.3절에서 자세히 설명한다.

본 문서에서는 오디세우스/COSMOS의 컴파일 및 사용 방법에 관한 내용을 중점적으로 설명한다. 오디세우스/COSMOS의 설계 및 구현, 모듈 별 함수 설명에 대한 내용은 다른 문서들을 참고하도록 한다. 아래의 리스트는 다른 문서들과 각 문서의 내용을 간략히 기술한 것이다.

- User Manual: 컴파일 및 사용 방법
- Functional Specification: 모듈 별 함수 설명
- Reference Manual: API들의 사용 방법

1.1. 오디세우스/COSMOS 32bit

오디세우스/COSMOS 32bit는 32bit OS 환경에서 동작한다. 오디세우스/COSMOS에서 사용가능한 버퍼의 크기는 메모리 크기에 의해 결정되는데, 32bit OS환경에서 사용가능한 최대 메모리의 크기는 4GB이므로, 사용가능한 버퍼의 크기도 최대 4GB이다. 또한, 사용가능한 데이터베이스 볼륨의 크기는 OID의 크기에 의해 결정되며, OID의 크기는 엔진 내부적으로 사용하는 정수형 타입의 크기에 의해 결정된다. 오디세우스/COSMOS 32bit에서 엔진 내부적으로 사용하는 정수형 타입의 크기는 2byte, 4byte로, 사용가능한 데이터베이스 볼륨의 크기는 최대 8TB이다. 64bit 컴퓨터와 크기가 큰 메모리 및 디스크의 등장에 따라 사용 가능한 버퍼의 크기와 데이터베이스 볼륨의 크기를 확장시키기 위해 오디세우스/COSMOS 64bit 대용량화가 개발되었다.

1.2. 오디세우스/COSMOS 64bit 대용량화

오디세우스/COSMOS 64bit 대용량화는 64bit OS 환경에서 동작한다. 64bit OS환경에서 사용가능한 최대 메모리의 크기는 16EB(2^{60})이므로, 사용가능한 버퍼의 크기도 최대 16EB이다. 또한, 엔진 내부적으로 사용하는 정수형 타입의 크기는 4byte, 8byte이므로, 사용가능한 데이터베이스의 크기는 최대 32ZB(2^{70})이다. 오디세우스/COSMOS 64bit 대용량화는 대용량의 버퍼와 데이터베이스를 사용할 수 있지만, 오디세우스/COSMOS 32bit에 비해 엔진 내부적으로 사용하는 정수형 타입의 크기가 2배로 증가하여 OID의 크기가 2배 증가하므로, 동일한 데이터에 대한 데이터베이스 크기가 최대 2배 증가하며, 오디세우스/COSMOS 32bit와 데이터베이스를 호환하여 사용하는 것은 불가능하다. 데이터베이스 크기를 줄이고, 64bit OS 환경에서 오디세우스/COSMOS 32bit와 데이터베이스를 호환하기 위해 오디세우스/COSMOS 64bit 비대용량화가 개발되었다.

1.3. 오디세우스/COSMOS 64bit 비대용량화

오디세우스/COSMOS 64bit 비대용량화는 64bit OS 환경에서 동작하며, 데이터베이스를 오디세우스/COSMOS 32bit와 호환하여 사용할 수 있다. 64bit OS 환경에서 동작하기 때문에 사용가능한 버퍼 크기는 최대 16EB이다. 또한, 엔진 내부적으로 사용하는 정수형 타입의 크기를 오디세우스/COSMOS 32bit와 동일하게 구현하여, 오디세우스/COSMOS 32bit와 OID의 크기가 동일하므로, 사용가능한 데이터베이스 크기도 동일하게 8TB이다.

1.1, 1.2, 1.3에서 설명한 오디세우스/COSMOS 32bit, 오디세우스/COSMOS 64bit, 오디세우스/COSMOS 64bit 비대용량화를 비교하면 아래 표와 같다.

	오디세우스/COSMOS 32bit	오디세우스/COSMOS 64bit	오디세우스/COSMOS 64bit 비대용량화
최대 버퍼 크기	4GB	16EB(2^{60})	16EB(2^{60})
최대 데이터베이스 볼륨 크기	8TB	32ZB(2^{70})	8TB
데이터베이스 호환성	오디세우스/COSMOS 64bit 비대용량화와 호환 가능함	불가능함	오디세우스/COSMOS 32bit와 호환 가능함

2. 오디세우스/COSMOS 컴파일

2.1. 실행환경

오디세우스/COSMOS는 Red Hat 사의 Linux 운영체제 에서 사용이 가능하다. 오디세우스/COSMOS를 실행하기 위한 환경은 다음과 같다.

- 플랫폼: Linux 2.6

2.2. 개발환경

오디세우스/COSMOS는 Red Hat 사의 Linux 운영체제에서 개발되었다. 오디세우스/COSMOS의 개발환경은 다음과 같다.

32bit

- 플랫폼: Linux 2.6
- 하드웨어: Linux server
- 컴파일러: gcc 4.1.2 Compiler

32bit

- 플랫폼: Linux 2.6
- 하드웨어: Linux server
- 컴파일러: gcc 4.4.7 Compiler

2.3. 컴파일 방법

오디세우스/COSMOS는 소스 코드를 컴파일 한 후 데이터베이스 응용 소프트웨어에 링크시켜서 사용한다. 컴파일 방법을 본절에서 설명하고, 응용 소프트웨어와 링크시켜서 사용하는 방법을 3장에서 설명한다.

Linux 운영 체제에서의 컴파일

오디세우스/COSMOS 소스 코드 디렉토리에서 다음 과정을 거쳐서 컴파일 한다. 단, 번호 뒤에 [Coarse]라고 쓴 것은 오디세우스/COSMOS Coarse granularity locking 버전에만 해당하는 항목이고, [Fine]이라고 쓴 것은 오디세우스/COSMOS Fine granularity locking 버전에만 해당하는 항목이며, 나머지는 공통이다. 오디세우스/COSMOS 32bit는 32bit Linux에서 컴파일하며, 오디세우스/COSMOS 64bit는 64bit Linux에서 컴파일한다.

① Header/param.h 파일을 열어서 파라미터 설정을 변경한다. 삭제해야 할 매크로와 추가해야 할 매크로는 다음과 같다.

삭제해야 할 매크로	추가해야 할 매크로
#define SOLARIS64 #define AIX64 #define WIN64	#define LINUX64
#undef READ_WRITE_BUFFER_ALIGN_FOR_LINUX	#define READ_WRITE_BUFFER_ALIGN_FOR_LINUX

② Err/update-cosmos_r.py 파일을 열어서 python 스크립트 언어를 실행할 프로그램의 경로를 수정한다. 즉, 사용할 python 프로그램의 경로 앞에 #!를 붙여서 Err/update-cosmos_r.py 파일의 첫 줄을 대체시킨다. 예를 들어서 사용할 python 프로그램의 경로가 /usr/bin/python이면 Err/update-cosmos_r.py 파일의 첫 줄을 #!/usr/bin/python으로 대체한다.

③ [Coarse] Err/Err_Error.pl 파일을 열어서 perl 스크립트 언어를 실행할 프로그램의 경로를 수정한다. 즉, 사용할 perl 프로그램의 경로 앞에 #!를 붙여서 Err/Err_Error.pl 파일의 첫 줄을 대체시킨다. 예를 들어서 사용할 perl 프로그램의 경로가 /usr/bin/perl이면 Err/Err_Error.pl 파일의 첫 줄을 #!/usr/bin/perl로 대체한다.

④ [Fine] Makefile 파일을 열어서 perl 스크립트 언어를 실행할 프로그램의 경로를 수정한다. 즉, Makefile 내의 PERL 변수의 값을 사용할 perl 프로그램의 경로로 변경한다. 예를 들어서 사용할 perl 프로그램의 경로가 /usr/bin/perl이면 Makefile 파일의 PERL = /usr/local/bin/perl라고 적힌 줄을 PERL = /usr/bin/perl로 수정한다.

⑤ [Coarse] BfM/Makefile 파일을 열어서 어셈블리어 컴파일러의 경로를 수정한다. 즉, BfM/Makefile 내의 AS 변수의 값을 사용할 어셈블리어 컴파일러의 경로로 변경한다. 예를 들어서 사용할 어셈블리어 컴파일러의 경로가 /usr/bin/as이면 BfM/Makefile 파일의 AS = /usr/ccs/bin/as라고 적힌 줄을 AS = /usr/bin/as로 수정한다.

⑥ [Fine] SHM/Makefile 파일을 열어서 어셈블리어 컴파일러의 경로를 수정한다. 즉, SHM/Makefile 내의 AS 변수의 값을 사용할 어셈블리어 컴파일러의 경로로 변경한다. 예를 들어서 사용할 어셈블리어 컴파일러의 경로가 /usr/bin/as이면 SHM/Makefile 파일의 AS = /usr/ccs/bin/as라고 적힌 줄을 AS = /usr/bin/as로 수정한다.

⑦ [Fine] Misc/Makefile 파일을 열어서 Solaris 용 컴파일 옵션을 Linux 용 컴파일 옵션으로 수정한다. 즉, Misc/Makefile 내의 CC = cc –mt 를 CC = gcc –pthread로 수정하고, LIBS = -lm -lsocket -lnsl -lintl -lpthread –lrt를 LIBS = -lm -lnsl -lpthread –lrt로 수정한다.

⑧ Header/param.h 파일을 열어서 오디세우스/COSMOS 32bit와 오디세우스/COSMOS 64bit 비대용량화는 SUPPORT_LARGE_DATABASE2 매크로를 undefine하고, 오디세우스/COSMOS 64bit는 SUPPORT_LARGE_DATABASE2 매크로를 define한다.

⑨ make를 수행하여 컴파일 한다. 컴파일 한 후에 오디세우스/COSMOS 오브젝트 파일(cosmos.o)이 생성되었는지 확인한다. cosmos.o 파일이 생성되어 있지 않다면 컴파일이 정상적으로 이루어지지 않은 것이므로 컴파일 오류를 확인하여 문제를 해결한 후에 다시 컴파일 해야 한다.

3. 사용방법

오디세우스/COSMOS는 응용 소프트웨어에 링크시켜서 사용한다. 오디세우스/COSMOS를 응용 소프트웨어에 링크시키기 위해서는 오디세우스/COSMOS의 Header 파일을 링크시킬 응용 소프트웨어의 소스 코드에 포함(include)시켜야 하고, 응용 소프트웨어에서는 오디세우스/COSMOS의 API 함수를 호출해야 하며, 응용 소프트웨어를 컴파일 할 때에 오디세우스/COSMOS의 오브젝트 파일을 링크시키도록 설정하여 컴파일 해야 한다. 자세한 과정은 다음과 같다.

① 오디세우스/COSMOS의 헤더 파일과 오브젝트 파일을 원하는 위치에 복사한다. 헤더 파일로서는 오디세우스/COSMOS의 소스 코드 디렉토리의 Header 디렉토리 안에 있는 cosmos_r.h, dblablib.h, param.h, trace.h 파일이 필요하고 오브젝트 파일로서는 2장에서 컴파일하여 생성한 cosmos.o 파일이 필요하다.

② 응용 소프트웨어에서 오디세우스/COSMOS의 API 함수를 사용하기 위해서는 먼저 LRDS_Init() 함수와 LRDS_AllocHandle() 함수를 호출해야 하고, 오디세우스/COSMOS의 API 함수를 더 이상 사용하지 않을 때는 LRDS_FreeHandle() 함수와 LRDS_Final() 함수를 호출해야 한다. 각 함수의 역할과 사용방법은 Reference Manual을 참조하도록 한다.

③ 응용 소프트웨어를 컴파일 할 때는 오디세우스/COSMOS를 링크시켜야 하며, 추가로 POSIX thread를 사용하기 위한 컴파일 옵션(-pthread)과 수학 함수를 사용하기 위한 컴파일 옵션(-lm)을 반드시 포함시켜야 한다. 예를 들어서 gcc 컴파일러를 사용하며, 오디세우스/COSMOS의 헤더 파일들이 /user/test/Header 디렉토리에 있고, 오디세우스/COSMOS의 오브젝트 파일(cosmos.o)이 /user/test/COSMOS 디렉토리에 있으며, test.c라는 소스 코드를 컴파일하여 test라는 응용 소프트웨어를 만들려면 다음과 같은 명령어로 컴파일 해야 한다.

```
> gcc –pthread –lm –o test –I/user/test/Header test.c /user/test/COSMOS/cosmos.o
```

4. 예제

본 장에서는 오디세우스/COSMOS를 이용하여 작성한 예제 프로그램들에 대해 설명한다. 예제 프로그램은 데이터베이스를 저장하기 위한 볼륨을 포멧하는 COSMOS_SimpleFormat, 데이터를 저장할 릴레이션을 생성하는 COSMOS_CreateRelation, 튜플을 삽입하는 COSMOS_InsertTuple, 릴레이션의 튜플들을 출력하는 COSMOS_PrintRelation으로 구성되어있다. 4개의 프로그램의 소스코드는 모두 예제 프로그램 디렉토리에 있으며 컴파일 하여 사용할 수 있다. 컴파일은 예제 프로그램 디렉토리 내에서 수행해야 하며 방법은 다음과 같다.

① Makefile 파일을 열어서 오디세우스/COMOS의 헤더 파일과 오브젝트 파일의 경로를 수정한다. 즉, Makefile 내의 INCLUDE 변수의 값과 COSMOS_OBJ 변수의 값을 수정한다. 예를 들어서 오디세우스/COSMOS의 헤더 파일들이 /user/test/Header 디렉토리에 있고, 오디세우스/COSMOS의 오브젝트 파일(cosmos.o)이 /user/test/COSMOS 디렉토리에 있으면, INCLUDE = ./Header를 INCLUDE = /user/test/Header로 수정하고, COSMOS_OBJ = ./COSMOS/cosmos.o를 COSMOS_OBJ = /user/test/COSMOS/cosmos.o로 수정한다.

② make를 수행하여 컴파일 한다. 컴파일 후에 실행 파일들(COSMOS_SimpleFormat, COSMOS_CreateRelation, COSMOS_InsertTuple, COSMOS_PrintRelation)이 생성된 것을 확인한다. 실행 파일들이 생성되어 있지 않다면 컴파일이 정상적으로 이루어지지 않은 것이므로 컴파일 오류를 확인하여 문제를 해결한 후에 다시 컴파일 해야 한다.

4.1. COSMOS_SimpleFormat

COSMOS_SimpleFormat은 데이터베이스를 저장하기 위한 볼륨을 포멧하는 프로그램이다. 사용 방법은 아래와 같다.

```
COSMOS_SimpleFormat [-volumeTitle <volume title>] [-volumeID <volume ID>]
[-device <device path> <number of page>]+
```

<volume title>은 생성할 볼륨의 이름이다. <volume ID>는 생성할 볼륨의 식별자로 0보다 큰 2byte 정수로 정해야 한다. <device path>와 <number of page>는 볼륨을 구성하는 디바이스의 이름과 그 디바이스의 페이지 수이다. <device path>와 <number of page>는 복수 개이어도 된다.

예를 들어서 test라는 이름을 갖고 볼륨의 식별자가 1000이며 500개의 페이지를 갖는 test1.vol이라는 디바이스와 400개의 페이지를 갖는 test2.vol이라는 디바이스로 구성된 볼륨을 포멧하려면 다음과 같은 명령어를 실행하면 된다.

```
> COSMOS_SimpleFormat -volumeTitle test -volumeID 1000 -device test1.vol 500 -device test2.vol 400
```

위 명령어를 실행한 후에는 test1.vol과 test2.vol이 생성되었는가를 확인한다.

4.2. COSMOS_CreateRelation

COSMOS_CreateRelation는 데이터를 저장할 릴레이션을 생성하는 프로그램이다. 사용 방법은 아래와 같다.

```
COSMOS_CreateRelation [<device path>]+
```

<device path>는 볼륨을 구성하는 디바이스의 이름으로 한 개 이상인 경우에는 모두 적어 주어야 한다.

프로그램을 실행하면 생성할 릴레이션 이름과 릴레이션을 구성하는 컬럼들의 수, 각 컬럼의 종류를 입력하라는 글이 나오며, 글에 따라서 값을 입력하면 릴레이션이 생성된다. 예제 프로그램에서는 컬럼의 종류로서 integer, float, 128글자 문자열만 사용할 수 있다.

예를 들어서 test1.vol, test2.vol이라는 디바이스로 구성된 볼륨에 integer 컬럼, float 컬럼, 문자열 컬럼으로 구성된 testRelation이라는 릴레이션을 생성하려면 다음과 같이 수행하면 된다.

```
> COSMOS_CreateRelation test1.vol test2.vol
Please type the relation name: testRelation
Please type the number of columns: 3
Please type the type of 1th column (1:integer, 2:float, 3:string(128)): 1
Please type the type of 2th column (1:integer, 2:float, 3:string(128)): 2
Please type the type of 3th column (1:integer, 2:float, 3:string(128)): 3
>
```

4.3. COSMOS_InsertTuple

COSMOS_InsertTuple은 릴레이션에 튜플을 삽입하는 프로그램이다. 사용 방법은 아래와 같다.

```
COSMOS_InsertTuple [<device path>]+
```

<device path>는 볼륨을 구성하는 디바이스의 이름으로 한 개 이상인 경우에는 모두 적어 주어야 한다.

프로그램을 실행하면 릴레이션 이름과 릴레이션에 삽입할 튜플을 구성하는 컬럼들의 값을 입력하라는 글이 나오며, 글에 따라서 값을 입력하면 튜플이 삽입된다. 이때, 각 컬럼의 종류는 글로 알려준다.

예를 들어서 test1.vol, test2.vol이라는 디바이스로 구성된 볼륨 내에 integer 컬럼, float 컬럼, 문자열 컬럼으로 구성되고 testRelation이라는 이름을 갖는 릴레이션에 <10, 3.14, “abcdefg”>라는 튜플을 삽입하려면 다음과 같이 수행하면 된다.

```
> COSMOS_InsertTuple test1.vol test2.vol
Please type the relation name: testRelation
Please type the value of 1th column (integer): 10
```

Please type the value of 2th column (float): 3.14

Please type the value of 3th column (string(128)): abcdefg

>

4.4. COSMOS_PrintRelation

COSMOS_PrintRelation은 릴레이션의 튜플들을 출력하는 프로그램이다. 사용 방법은 아래와 같다.

```
COSMOS_PrintRelation [<device path>]+
```

<device path>는 볼륨을 구성하는 디바이스의 이름으로 한 개 이상인 경우에는 모두 적어 주어야 한다.

프로그램을 실행하면 릴레이션 이름을 입력하라는 글이 나오며 릴레이션 이름을 입력하면 해당 릴레이션에 포함된 튜플들의 내용이 출력된다.

예를 들어서 test1.vol, test2.vol이라는 디바이스로 구성된 볼륨 내에 testRelation이라는 이름을 갖는 릴레이션에 <10, 3.14, “abcdefg”>, <5, 2.718, “hijklmn”>이라는 튜플이 포함되어 있을 때, 프로그램을 아래와 같이 실행할 수 있다.

> COSMOS_PrintRelation test1.vol test2.vol

Please type the relation name: testRelation

integer		float		string	
10		3.140000		abcdefg	
5		2.718000		hijklmn	

>