Reference Manual

오디세우스/COSMOS

Version 3.0

Manual Release 1

2016년 8월

Advanced Information Technology Research Center (AITrc)

KAIST

목 차

1. 시스템 관리

1.1. LRDS_Init

Syntax

Four LRDS_Init( )

Parameters

IN/OUT	이름	타입	설명
None			

Description

COSMOS 저장 시스템을 초기화한다.

Return value

eNOERROR : COSMOS를 정상적으로 시작하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four              e;

e = LRDS_Init();
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_Final();
if(e < eNOERROR) /* error 처리 */
……
```

1.2. LRDS_Final

Syntax

Four LRDS_Final()

Parameters

IN/OUT	이름	타입	설명
None			

Description

COSMOS 저장 시스템을 말기화한다.

Return value

eNOERROR : COSMOS를 정상적으로 종료하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four              e;
```

```
e = LRDS_Init();
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_Final();
if(e < eNOERROR) /* error 처리 */
……
```

1.3. LRDS_AllocHandle

Syntax

Four LRDS_AllocHandle(Four* handle)

Parameters

IN/OUT	이름	타입	설명
OUT	handle	Four*	thread 관리용 식별자

Description

프로세스 내에서 thread를 구분하기 위하여 사용하는 handle을 할당 받는다. LRDS_Init()과 LRDS_Final()을 제외한 대부분의 COSMOS API들은 handle을 첫 번째 인자로 받는다. 단, coarse granularity locking no-thread 버전은 하나의 handle만 있으므로 인자로 받지 않는다.

Return value

eNOERROR : handle을 할당 받았음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four                e;
Four                        handle;

……
e = LRDS_AllocHandle(&handle);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_FreeHandle(handle);
if(e < eNOERROR) /* error 처리 */
……
```

1.4. LRDS_FreeHandle

Syntax

Four LRDS_FreeHandle(Four handle)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자

Description

프로세스 내에서 thread를 구분하기 위하여 사용하는 handle을 반환한다.

Return value

eNOERROR : handle을 반환하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four              e;
Four                    handle;

……
e = LRDS_AllocHandle(&handle);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_FreeHandle(handle);
if(e < eNOERROR) /* error 처리 */
……
```

1.5. LRDS_SetCfgParam

Syntax

Four LRDS_SetCfgParam(Four handle, char* name, char* value)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	name	char*	설정 파라미터의 이름
IN	value	char*	설정 파라미터의 값

Description

설정 파라미터의 값을 지정한다. 설정 파라미터에는 LOG_VOLUME_DEVICE_LIST, COHERENCY_VOLUME_DEVICE, USE_DEADLOCK_AVOIDANCE, USE_BULKFLUSH가 있다.

Return value

eNOERROR : 설정 파라미터의 값을 지정함

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four              e;
Four                    handle;

……
e = LRDS_SetCfgParam(handle, “LOG_VOLUME_DEVICE_LIST”,
                     “/cosmos/log.vol”);
if(e < eNOERROR) /* error 처리 */
```

……

1.6. LRDS_GetCfgParam

Syntax

char* LRDS_GetCfgParam(Four handle, char* name)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	name	char*	설정 파라미터의 이름

Description

설정 파라미터의 값을 읽어온다. 설정 파라미터에는 LOG_VOLUME_DEVICE_LIST, COHERENCY_VOLUME_DEVICE, USE_DEADLOCK_AVOIDANCE, USE_BULKFLUSH가 있다.

Return value

value : name으로 지정된 설정 파라미터의 값

Example

```
#include "cosmos_r.h"

Four                    handle;
char*             value;

……
value = LOM_GetCfgParam(handle, “LOG_VOLUME_DEVICE_LIST”);
printf(“%s\n”, value);
……
```

1.7. LRDS_InitLocalDS
1.8. LRDS_InitSharedDS
1.9. LRDS_FinalLocalDS
1.10. LRDS_GetCfgParam

2. 볼륨 관리

2.1. LRDS_Mount

Syntax

Four LRDS_Mount(Four handle, Four numDevices, char **devNames, Four *volId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	numDevices	Four	볼륨을 구성하는 디바이스의 수
IN	devNames	char**	디바이스의 이름들의 배열
OUT	volId	Four*	마운트된 볼륨의 ID

Description

주어진 볼륨을 저장 시스템이 사용할 수 있도록 마운트한다. 하나의 볼륨은 하나이상의 디바이스에 의해 구성될 수 있기 때문에 입력으로 볼륨을 구성하는 디바이스의 개수와 디바이스의 이름들을 배열로 넘긴다. 디바이스의 이름은 UNIX 파일 시스템에서의 이름을 사용한다. 볼륨이 성공적으로 마운트되면 그 볼륨의 식별자를 반환해 준다.

Return value

eNOERROR : 볼륨을 마운트 하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
char            deviceNameStrings[2][256];
char**          deviceNames;
Four            volId;
……
strcpy(devNameStrings[0], "/device1-name")
strcpy(devNameStrings[1], "/device2-name")

devNames[0] = devNameStrings[0];
devNames[1] = devNameStrings[1];
e = LRDS_Mount(handle, 2, devNames, &volId);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_Dismount(handle, volId);
if(e < eNOERROR) /* error 처리 */
……
```

2.2. LRDS_Dismount

Syntax

Four LRDS_Dismount(Four handle, Four volId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	데이타베이스 볼륨의 ID

Description

마운트된 볼륨을 디스마운트한다. 디스마운트할 볼륨은 마운트 시 반환해준 볼륨 식별자를 통하여 지정한다.

Return value

eNOERROR : 볼륨을 디스마운트 하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
char            deviceNameStrings[2][256];
char**          deviceNames;
Four            volId;
……
strcpy(devNameStrings[0], "/device1-name")
strcpy(devNameStrings[1], "/device2-name")

devNames[0] = devNameStrings[0];
devNames[1] = devNameStrings[1];
e = LRDS_Mount(handle, 2, devNames, &volId);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_Dismount(handle, volId);
if(e < eNOERROR) /* error 처리 */
……
```

2.3. LRDS_FormatDataVolume

Syntax

Four LRDS_FormatDataVolume(Four handle, Four numDevices, char **devNames, char *title, Four volId, Four extSize, Four *numPagesInDevice, Four segmentSize)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	numDevices	Four	볼륨을 구성하는 디바이스들의 수
IN	devNames	char**	볼륨을 구성하는 디바이스들의 이름

IN	title	char*	볼륨의 이름
IN	volId	Four	데이타베이스 볼륨의 ID
IN	extSize	Four	익스텐트의 크기(coarse granularity locking 버전에서는 타입이 Two 임)
IN	numPagesInDevice	Four*	각 디바이스의 페이지 수
IN	segmentSize	Four	세그먼트의 크기

Description

주어진 디바이스들로 데이터 볼륨을 포멧한다. 하나의 볼륨은 여러 개의 디바이스로 구성된다.

Return value

eNOERROR : 데이터 볼륨을 포멧하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
char            deviceNameStrings[2][256];
char**          deviceNames;
Four                  nPages[2];
Four            volId;
……
strcpy(devNameStrings[0], "/device1-name")
strcpy(devNameStrings[1], "/device2-name")

devNames[0] = devNameStrings[0];
devNames[1] = devNameStrings[1];
nPages[0] = 3200;
nPages[1] = 4800;
e = LRDS_FormatDataVolume(handle, 2, devNames, “test_volume”, 1005, 16,
nPages, 800);
if(e < eNOERROR) /* error 처리 */
……
```

2.4. LRDS_FormatLogVolume

Syntax

Four LRDS_FormatLogVolume(Four handle, Four numDevices, char **devNames, char *title, Four volId, Four extSize, Four *numPagesInDevice)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	numDevices	Four	볼륨을 구성하는 디바이스들의 수

IN	devNames	char**	볼륨을 구성하는 디바이스들의 이름
IN	title	char*	볼륨의 이름
IN	volId	Four	로그 볼륨의 ID
IN	extSize	Four	익스텐트의 크기(coarse granularity locking 버전에서는 타입이 Two 임)
IN	numPagesInDevice	Four*	각 디바이스의 페이지 수

Description

주어진 디바이스들로 로그 볼륨을 포멧한다. 로그는 데이터베이스의 연산을 기록하는 것으로 데이터베이스 시스템이 위부적 요인 등에 의해 비정상적인 종료를 하였을 경우, 데이터베이스의 내용을 원래대로 복귀시켜주는 역할을 한다. 트랜잭션의 철회(Roll Back)연산이나 파손회복기능을 사용하기 위해서는 반드시 로그 볼륨을 생성해야 한다. 로그 볼륨이 없거나 지정되지 않은 경우에는 트랜잭션 철회나 프로그램의 비정상적인 종료로 인한 데이터베이스 파손으로부터의 회복을 수행할 수 없다.

Return value

eNOERROR : 로그 볼륨을 포멧하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
char            deviceNameStrings[2][256];
char**          deviceNames;
Four                  nPages[2];
Four            volId;
……
strcpy(devNameStrings[0], "/device1-name")
strcpy(devNameStrings[1], "/device2-name")

devNames[0] = devNameStrings[0];
devNames[1] = devNameStrings[1];
nPages[0] = 800;
nPages[1] = 400;
e = LRDS_FormatLogVolume(handle, 2, devNames, “test_volume”, 1005, 16,
nPages);
if(e < eNOERROR) /* error 처리 */
……
```

2.5. LRDS_FormatTempDataVolume

Syntax

Four LRDS_FormatTempDataVolume(Four handle, Four numDevices, char **devNames, char *title, Four volId, Four extSize, Four *numPagesInDevice, Four segmentSize)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	numDevices	Four	볼륨을 구성하는 디바이스들의 수
IN	devNames	char**	볼륨을 구성하는 디바이스들의 이름
IN	title	char*	볼륨의 이름
IN	volId	Four	로그 볼륨의 ID
IN	extSize	Four	익스텐트의 크기(coarse granularity locking 버전에서는 타입이 Two 임)
IN	numPagesInDevice	Four*	각 디바이스의 페이지 수
IN	segmentSize	Four	세그먼트의 크기

Description

주어진 디바이스들로 임시 데이터 볼륨을 포멧한다. 임시 데이터 볼륨은 정렬 등의 연산을 수행할 때 데이터를 임시로 저장하기 위하여 사용된다.

Return value

eNOERROR : 임시 데이터 볼륨을 포멧하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
char            deviceNameStrings[2][256];
char**          deviceNames;
Four                  nPages[2];
Four            volId;
……
strcpy(devNameStrings[0], "/device1-name")
strcpy(devNameStrings[1], "/device2-name")

devNames[0] = devNameStrings[0];
devNames[1] = devNameStrings[1];
nPages[0] = 800;
nPages[1] = 400;
e = LRDS_FormatTempDataVolume(handle, 2, devNames, “test_volume”, 1005,
16, nPages, 200);
if(e < eNOERROR) /* error 처리 */
……
```

2.6. LRDS_FormatCoherencyVolume

Syntax

Four LRDS_FormatCoherencyVolume(Four handle, char *devName, char *title,

Four volId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	devName	char*	볼륨을 구성하는 디바이스들의 이름
IN	title	char*	볼륨의 이름
IN	volId	Four	로그 볼륨의 ID

Description

주어진 디바이스로 coherency 볼륨을 포멧한다. Coherency 볼륨은 다중 서버 환경에서 프로세스들 간의 버퍼 일관성을 유지하기 위하여 사용되며, 공유 메모리를 사용하지 않는 coarse-granularity locking 버전에서만 필요하다.

Return value

eNOERROR : coherency 볼륨을 포멧하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
……
e = LRDS_FormatCoherencyVolume(handle, "/devName", “test_volume”,
1005);
if(e < eNOERROR) /* error 처리 */
……
```

2.7. LRDS_ExpandDataVolume

Syntax

Four LRDS_ExpandDataVolume(Four handle, Four volId, Four numAddDevices, char **addDevNames, Four *numPagesInAddDevice)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	로그 볼륨의 ID
IN	numAddDevices	Four	볼륨에 추가할 디바이스들의 수
IN	addDevNames	char**	볼륨에 추가할 디바이스들의 이름
IN	numPagesInAddDevice	Four*	각 디바이스의 페이지 수

Description

주어진 볼륨에 주어진 디바이스들을 추가하여 볼륨의 크기를 확장한다. 이때 볼륨은 마운트되어 있어야 한다.

Return value

eNOERROR : 임시 데이터 볼륨을 포멧하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
char            deviceNameStrings[2][256];
char            addDeviceNameStrings[2][256];
char*           devNames[2];
Four                  nPages[2];
Four            volId;
……
strcpy(devNameStrings[0], "/device1-name")
strcpy(devNameStrings[1], "/device2-name")

devNames[0] = devNameStrings[0];
devNames[1] = devNameStrings[1];

e = LRDS_Mount(handle, 2, devNames, &volId);
if(e < eNOERROR) /* error 처리 */

strcpy(addDevNameStrings[0], "/add_device1-name")
strcpy(addDevNameStrings[1], "/add_device2-name")

devNames[0] = addDevNameStrings[0];
devNames[1] = addDevNameStrings[1];

nPages[0] = 800;
nPages[1] = 400;

e = LRDS_ExpandDataVolume(handle, volId, 2, devNames, nPages);
if(e < eNOERROR) /* error 처리 */

e = LRDS_Dismount(handle, volId);
if(e < eNOERROR) /* error 처리 */
……
```

3. 트랜잭션 관리

3.1. LRDS_BeginTransaction

Syntax

Four LRDS_BeginTransaction(Four handle, XactID *xactId, ConcurrencyLevel ccLevel)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
OUT	xactId	XactID*	트랜잭션 식별자
IN	ccLevel	ConcurrencyLevel	트랜잭션이 사용할 동시성 제어 수준

Description

새로운 트랜잭션을 초기화하고 트랜잭션의 시작임을 선언한다. 생성된 트랜잭션은 이를 식별하기 위한 식별자가 부여되며 xactId로 반환된다.

cclevel은 주어진 트랜잭션이 사용할 동시성 수준이다. 동시성 수준은 여러 트랜잭션들이 동시에 수행될 경우, 이를 어떻게 처리할 것인가를 결정한다. cclevel은 ConcurrencyLevel 타입으로 다음과 같이 정의된다. typedef enum { X_BROWSE_BROWSE, X_CS_BROWSE, X_CS_CS, X_RR_BROWSE, X_RR_CS, X_RR_RR } ConcurrencyLevel;

현 버전의 오디세우스/COSMOS은 X_BROWSE_BROWSE, X_RR_RR의 두 가지 동시성 수준을 사용한다.

X_BROWSE_BROWSE는 no read lock, long write lock을 사용하는 수준으로 읽기를 주로 하는 트랜잭션에서 사용한다. X_BROWSE_BROWSE로 수행되는 트랜잭션은 다른 트랜잭션이 write연산을 하더라도 주어진 볼륨(데이터)에 대한 read연산을 수행할 수 있으며 다른 트랜잭션이 write연산을 안 할 때, write 연산을 수행할 수 있다.

X_RR_RR은 long read lock, long_write_lock으로 쓰기를 주로 하는 트랜잭션에서 사용한다. X_RR_RR은 다른 트랜잭션이 write연산을 수행할 경우, 주어진 볼륨(데이터)에 대한 read연산을 수행할 수 없으며, 다른 트랜잭션이 X_RR_RR수준에서 read연산을 안 할 때 write연산을 수행할 수 있다. 또한 다른 트랜잭션이 write연산을 안 할 때, write 연산을 수행할 수 있다.

Return value

eNOERROR : 트랜잭션을 성공적으로 시작하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
XactID          xactID;
……
e = LRDS_BeginTransaction(handle, &xactID, X_RR_RR);
```

```
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_CommitTransaction(handle, &xactID);
if(e < eNOERROR) /* error 처리 */
……
```

3.2. LRDS_CommitTransaction

Syntax

Four LRDS_CommitTransaction(Four handle, XactID *xactId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	xactId	XactID*	트랜잭션 식별자

Description

주어진 트랜잭션을 완료한다. 트랜잭션이 완료되면 트랜잭션간에 수행된 데이터베이스 관련 연산이 실제로 데이터베이스에 반영된다.

Return value

eNOERROR : 트랜잭션을 성공적으로 완료하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
XactID          xactID;
……
e = LRDS_BeginTransaction(handle, &xactID, X_RR_RR);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_CommitTransaction(handle, &xactID);
if(e < eNOERROR) /* error 처리 */
……
```

3.3. LRDS_AbortTransaction

Syntax

Four LRDS_AbortTransaction(Four handle, XactID *xactId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	xactId	XactID*	트랜잭션 식별자

Description

주어진 트랜잭션을 철회한다. 트랜잭션이 철회되면 트랜잭션간에 수행된 데이터베이스 관련 연산은 모두 취소되며 데이터베이스 상태는 트랜잭션 시작 이전 상태가 된다.

Return value

eNOERROR : 트랜잭션을 성공적으로 철회하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
XactID          xactID;
……
e = LRDS_BeginTransaction(handle, &xactID, X_RR_RR);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_AbortTransaction(handle, &xactID);
if(e < eNOERROR) /* error 처리 */
……
```

3.4. LRDS_SetSavepoint

Syntax

Four LRDS_SetSavepoint(Four handle, SavepointID* spID)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
OUT	spID	SavepointID*	세이브포인트 식별자

Description

Fine granularity locking 버전에만 존재하는 API로, 주어진 트랜잭션에 대한 세이브포인트를 설정한다. LRDS_RollbackSavepoint()를 호출하여 세이브포인트 이후에 수행된 데이터베이스 관련 연산을 철회시킬 수 있다.

Return value

eNOERROR : 세이브포인트를 성공적으로 설정하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
SavepointID           spID;
……
e = LRDS_SetSavepoint(handle, &spID);
if(e < eNOERROR) /* error 처리 */
```

```
……
e = LRDS_RollbackSavepoint(handle, spID);
if(e < eNOERROR) /* error 처리 */
……
```

3.5. LRDS_RollbackSavepoint

Syntax

Four LRDS_RollbackSavepoint(Four handle, SavepointID spID)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	spID	SavepointID	세이브포인트 식별자

Description

Fine granularity locking 버전에만 존재하는 API로, 세이브포인트를 설정한 시점 이후에 수행된 데이터베이스 관련 연산을 모두 취소하고 데이터베이스 상태는 세이브포인트 설정 이전 상태가 된다.

Return value

eNOERROR : 세이브포인트까지 트랜잭션을 철회하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
SavepointID           spID;
……
e = LRDS_SetSavepoint(handle, &spID);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_RollbackSavepoint(handle, spID);
if(e < eNOERROR) /* error 처리 */
……
```

4. 릴레이션 및 색인 관리

4.1. LRDS_CreateRelation

Syntax

Four LRDS_CreateRelation(Four handle, Four volId, char* relName, LRDS_IndexDesc* idesc, Four nCols, ColInfo* cinfo, Boolean tmpRelationFlag)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨의 식별자
IN	relName	char*	생성하려는 릴레이션의 이름
IN	idesc	LRDS_IndexDesc*	클러스터링 인덱스에 대한 기술자
IN	nCols	Four	컬럼의 수(coarse granularity locking 버전에서는 타입이 Two 임)
IN	cinfo	ColInfo*	컬럼들에 대한 정보
IN	tmpRelationFlag	Boolean	임시 릴레이션인지 여부를 나타냄

Description

릴레이션을 새로 하나 만든다. 릴레이션은 여러 개의 컬럼으로 구성되며, 클러스터링을 위한 클러스터링 인덱스를 가질 수 있다. 각 컬럼은 0부터 시작하는 컬럼 번호를 가지게 되고, 이 컬럼 번호를 사용해서 원하는 컬럼을 액세스할 수 있게 된다. 새로운 릴레이션을 생성하기 위해서는 릴레이션이 가지는 컬럼의 개수와 각 컬럼에 대하여 컬럼 타입과 컬럼 값의 최대길이를 지정해 주어야 한다. 클러스터링 인덱스를 지정하는 경우에는 클러스터링 인덱스에 대한 정보, 즉 인덱스의 키를 구성하는 컬럼 번호들을 알려주어야 한다. 인덱스가 composite key를 지원하므로 하나의 컬럼이 아니라 여러 개의 컬럼 번호를 줄 수 있다.

Return value

eNOERROR : 릴레이션을 성공적으로 생성하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
LRDS_IndexDesc  idesc;
ColInfo         cinfo[2];
……
idesc.indexType = SM_INDEXTYPE_BTREE;
idesc.btree.flag = KEYFLAG_CLUSTERING;
idesc.btree.nColumns = 1;
```

```
idesc.btree.columns[0].colNo = 0;
idesc.btree.columns[0].flag = KEYINFO_COL_DESC;

cinfo[0].complexType = SM_COMPLEXTYPE_BASIC;
cinfo[0].type = SM_INT;

cinfo[1].complexType = SM_COMPLEXTYPE_BASIC;
cinfo[1].type = SM_STRING;
cinfo[1].length = 10;

e = LRDS_CreateRelation(handle, volId, “new_relation”, &idesc, 2, cinfo,
SM_FALSE);
if(e < eNOERROR) /* error 처리 */
……
```

4.2. LRDS_DestroyRelation

Syntax

Four LRDS_DestroyRelation(Four handle, Four volId, char* relName)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨의 식별자
IN	relName	char*	삭제하려는 릴레이션의 이름

Description

주어진 릴레이션을 데이터베이스로부터 삭제한다. 삭제할 릴레이션은 릴레이션 이름과 그 릴레이션이 위치해 있는 볼륨에 대한 식별자를 가지고 지정한다. 릴레이션을 삭제하기 전에 그 릴레이션이 오픈되어 있는지 검사하여 오픈되어 있으면 릴레이션을 삭제하지 않고 그냥 리턴하게 된다. 오픈 되지 않은, 즉 사용 중이 아닌 파일이면 저장 시스템에서 주어진 릴레이션에 해당하는 파일(데이터 파일과 인덱스 파일 모두)을 삭제한다. 그리고 카탈로그 테이블에서 삭제되는 릴레이션에 관한 튜플들을 삭제한다.

Return value

eNOERROR : 릴레이션을 성공적으로 삭제하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
……
e = LRDS_DestroyRelation(handle, volId, “new_relation”);
if(e < eNOERROR) /* error 처리 */
……
```

4.3. LRDS_AddIndex

Syntax

Four LRDS_AddIndex(Four handle, Four volId, char* relName, LRDS_IndexDesc* idesc, IndexID* iid)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨의 식별자
IN	relName	char*	릴레이션의 이름
IN	idesc	LRDS_IndexDesc*	인덱스에 대한 기술자
OUT	iid	IndexID*	인덱스 식별자

Description

릴레이션에 새로운 인덱스를 추가한다. 인덱스를 새로 정의하기 위해서는 인덱스에 사용되는 키에 관한 정보, 즉 키를 구성하는 컬럼들의 번호를 파라미터로 전달해 주어야 한다. 저장 시스템이 멀티 키(여러 개의 컬럼으로 구성된 키)를 지원해 주므로 키를 구성하는 컬럼으로 여러 개의 컬럼 번호를 줄 수 있다.

Return value

eNOERROR : 인덱스를 성공적으로 생성하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
LRDS_IndexDesc  idesc;
IndexID        iid;
……
idesc.indexType = SM_INDEXTYPE_BTREE;
idesc.btree.flag = KEYFLAG_CLEAR;
idesc.btree.nColumns = 2;

idesc.btree.columns[0].colNo = 4;
idesc.btree.columns[0].flag = KEYINFO_COL_DESC;

idesc.btree.columns[1].colNo = 0;
idesc.btree.columns[1].flag = KEYINFO_COL_ASC;

e = LRDS_AddIndex(handle, volId, “new_relation”, &idesc, &iid);
if(e < eNOERROR) /* error 처리 */
……
```

4.4. LRDS_DropIndex

Syntax

Four LRDS_DropIndex(Four handle, Four volId, char* relName, IndexID* iid)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨의 식별자
IN	relName	char*	릴레이션의 이름
IN	iid	IndexID*	인덱스 식별자

Description

주어진 릴레이션에 대한 인덱스를 하나 제거한다. 제거할 인덱스는 인덱스 식별자로 지정한다.

Return value

eNOERROR : 인덱스를 성공적으로 제거하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
Four                 orn;
lrds_RelTableEntry  *relTableEntry;
……

orn = LRDS_OpenRelation(handle, volId, “relation”);
if(orn < eNOERROR) /* error 처리 */

relTableEntry = LRDS_GET_RELTABLE_ENTRY(handle, orn);

e = LRDS_DropIndex(handle, volId, “relation”,
&(LRDS_GET_IDXINFO_FROM_RELTABLE_ENTRY(relTableEntry))[0].iid);
if(e < eNOERROR) /* error 처리 */
……
```

4.5. LRDS_AddColumn

Syntax

Four LRDS_AddColumn(Four handle, Four volId, char* relName, ColInfo* cinfo)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자

IN	volId	Four	볼륨의 식별자
IN	relName	char*	릴레이션의 이름
IN	cinfo	ColInfo*	추가할 컬럼에 대한 정보

Description

릴레이션에 컬럼을 추가한다.

Return value

eNOERROR : 릴레이션에 컬럼을 성공적으로 추가하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
ColInfo         cinfo;
……
cinfo.complexType = SM_COMPLEXTYPE_BASIC;
cinfo.type = SM_FLOAT;

e = LRDS_AddColumn(handle, volId, “relation”, &cinfo);
if(e < eNOERROR) /* error 처리 */
……
```

4.6. LRDS_OpenRelation

Syntax

Four LRDS_OpenRelation(Four handle, Four volId, char* relName)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨의 식별자
IN	relName	char*	릴레이션의 이름

Description

주어진 릴레이션을 오픈한다. 릴레이션의 오픈은 릴레이션에 관한 정보를 오픈 릴레이션 테이블에 등록하는 작업을 의미한다.

Return value

오픈 릴레이션 번호: 릴레이션을 성공적으로 오픈하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"
```

```
Four            handle;
Four            e;
Four                 volId;
Four            orn;
……

orn = LRDS_OpenRelation(handle, volId, “relation”);
if(orn < eNOERROR) /* error 처리 */
……
e = LRDS_CloseRelation(handle, orn);
if(e < eNOERROR) /* error 처리 */
……
```

4.7. LRDS_CloseRelation

Syntax

Four LRDS_CloseRelation(Four handle, Four orn)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	orn	Four	오픈 릴레이션 번호

Description

주어진 오픈 릴레이션 번호가 가리키는 릴레이션을 닫는다.

Return value

eNOERROR : 릴레이션을 성공적으로 닫았음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                 volId;
Four            orn;
……

orn = LRDS_OpenRelation(handle, volId, “relation”);
if(orn < eNOERROR) /* error 처리 */
……
e = LRDS_CloseRelation(handle, orn);
if(e < eNOERROR) /* error 처리 */
……
```

4.8. LRDS_CloseAllRelations

Syntax

Four LRDS_CloseAllRelations(Four handle)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자

Description

현재 thread가 오픈한 릴레이션을 모두 닫는다.

Return value

eNOERROR : 모든 릴레이션을 성공적으로 닫았음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
……

e = LRDS_CloseAllRelations(handle);
if(e < eNOERROR) /* error 처리 */
……
```

4.9. LRDS_SortRelation

Syntax

Four LRDS_SortRelation(Four handle, Four volId, Four tmpVolId, char* inRelName, KeyInfo* kinfo, Boolean newRelFlag, char* outRelName, Boolean tmpRelFlag, LockParameter* lockup)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	정렬할 릴레이션이 위치한 볼륨의 식별자
IN	tmpVolId	Four	정렬에 사용할 임시 볼륨
IN	inRelName	char*	정렬시킬 릴레이션 이름
IN	kinfo	KeyInfo*	정렬 키에 대한 정보
IN	newRelFlag	Boolean	새로운 릴레이션에 정렬된 결과를 저장할지 여부를 나타내는 플래그
IN	outRelName	char*	정렬된 결과를 저장할 릴레이션 이름
IN	tmpRelFlag	Boolean	outRelName 이 임시 릴레이션인지를 나타내는 플래그

IN	lockup	LockParameter*	요청하는 lock 에 대한 정보

Description

주어진 릴레이션을 정렬한다.

Return value

eNOERROR : 릴레이션을 성공적으로 정렬하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
Four                  tmpVolId;
KeyInfo        kinfo;
LockParameter   lockup;
……

kinfo.flag = KEYFLAG_CLEAR;
kinfo.nColumns = 1;
kinfo.columns[0].colNo = 3;
kinfo.columns[0].flag = KEYINFO_COL_DESC;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

e = LRDS_SortRelation(handle, volId, tmpVolId, “in_relation”, &kinfo,
SM_TRUE, “out_relation”, SM_FALSE, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

4.10. LRDS_GetFileIdOfRelation

5. 스캔 관리

5.1. LRDS_OpenSeqScan

Syntax

Four LRDS_OpenSeqScan(Four handle, Four orn, Four scanDirection, Four nBools, BoolExp bool[], LockParameter* lockup)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	orn	Four	오픈 릴레이션 번호
IN	scanDirection	Four	스캔 방향(FORWARD 혹은 BACKWARD)
IN	nBools	Four	boolean expression 의 개수
IN	bool[]	BoolExp	boolean expression 들의 리스트
IN	lockup	LockParameter*	요청하는 lock 에 대한 정보

Description

주어진 릴레이션에 대한 순차적 스캔을 하나 오픈한다. 이 스캔은 인덱스를 사용하지 않고, 데이터 파일 안에 물리적으로 저장된 순서대로 튜플들을 액세스하기 위하여 사용된다. 파라미터 scanDirection은 스캔의 방향을 지정하기 위하여 사용한다. scanDirection의 값이 FORWARD이면 저장순서대로 액세스하고, 그 값이 BACKWARD이면 저장 순의 역순으로 튜플들을 액세스한다. 튜플을 액세스하는데 있어서 모든 튜플을 액세스하지 않고 원하는 튜플들만을 액세스하기 위하여 boolean expression을 사용할 수 있다. Boolean expression을 사용하면 특정 컬럼 값이 boolean expression을 만족하는 튜플만을 사용자에게 보여준다. Boolean expression은 몇 개의 조건이 AND로 묶여진 형태를 가진다. lockup 파라메타는 계층구조 로크를 위해서 사용된다.

Return value

스캔 식별자 : 순차적 스캔을 오픈하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                 scanId;
Four                 orn;
BoolExp       bool[2];
LockParameter   lockup;
……

bool[0].op = SM_LE;
bool[0].colNo = 1;
```

```
bool[0].data.i = 10;

bool[1].op = SM_GE;
bool[1].colNo = 1;
bool[1].data.i = 20;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

scanId = LRDS_OpenSeqScan(handle, orn, FORWARD, 2, bool, &lockup);
if(scanId < eNOERROR) /* error 처리 */
……
e = LRDS_CloseScan(handle, scanId);
if(e < eNOERROR) /* error 처리 */
……
```

5.2. LRDS_OpenIndexScan

Syntax

Four LRDS_OpenIndexScan(Four handle, Four orn, IndexID *iid, BoundCond* startBound, BoundCond* stopBound, Four nBools, BoolExp bool[], LockParameter* lockup)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	orn	Four	오픈 릴레이션 번호
IN	iid	IndexID*	스캔에 사용되는 인덱스의 식별자
IN	startBound	BoundCond*	range scan 의 start boundary(NULL 값을 가질 수 있음)
IN	stopBound	BoundCond*	range scan 의 stop boundary(NULL 값을 가질 수 있음)
IN	nBools	Four	boolean expression 의 개수
IN	bool[]	BoolExp	boolean expression 들의 리스트
IN	lockup	LockParameter*	요청하는 lock 에 대한 정보

Description

주어진 릴레이션에 대한 인덱스 스캔을 오픈한다. 인덱스 스캔은 주어진 인덱스의 키 값의 크기 순에 따라서 튜플을 액세스한다. 이때 스캔하는 영역의 start boundary와 stop booundary를 지정함으로써 원하는 영역에 있는 튜플들만 액세스 하는 것이 가능하다(range scan). Start boundary가 stop boundary 보다 작으면 키 값이 커지는 순으로 액세스하게 되고, 반대로 start boundary가 stop boundary보다 크면 키 값이 작아지는 순으로 액세스하게 된다. Start boundary 혹은 stop boundary는 NULL 값을 가질 수 있는데, 이 경우 boundary가 없어서 가장 작은 키 값, 또는 가장 큰 값을 갖는 튜플까지 액세스한다. 인덱스 스캔에 있어서 순차적 스캔과 마찬가지로 boolean expression을 줌으로

써 boolean expression을 만족하는 튜플만을 액세스할 수 있다. lockup 파라메타는 계층구조 로크를 위해서 사용된다.

Return value

스캔 식별자 : 인덱스 스캔을 오픈하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                 volId;
Four                 scanId;
Four                 orn;
BoundCond      startBound;
BoolExp        bool[1];
LockParameter   lockup;
Four                 keyValue;
lrds_RelTableEntry  *relTableEntry;
……
orn = LRDS_OpenRelation(handle, volId, “relation”);
if(orn < eNOERROR) /* error 처리 */

relTableEntry = LRDS_GET_RELTABLE_ENTRY(handle, orn);
……
startBound.op = SM_LT;
keyValue = 10;
startBound.key.len = sizeof(Four);
bcopy(&keyValue, &(startBound.key.val[0]), sizeof(Four));



bool[0].op = SM_GT;
bool[0].colNo = 1;
bool[0].data.i = 20;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

scanId = LRDS_OpenSeqScan(handle, orn,
   &(LRDS_GET_IDXINFO_FROM_RELTABLE_ENTRY(relTableEntry))[0].iid,
   &startBound, NULL, 1, bool, &lockup);
if(scanId < eNOERROR) /* error 처리 */
……
e = LRDS_CloseScan(handle, scanId);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_CloseRelation(handle, orn);
if(e < eNOERROR) /* error 처리 */
……
```

5.3. LRDS_MLGF_OpenIndexScan

Syntax

Four LRDS_MLGF_OpenIndexScan(Four handle, Four orn, IndexID *iid, MLGF_HashValue lowerBounds[], MLGF_HashValue upperBounds[], Four nBools, BoolExp bool[], LockParameter* lockup)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	orn	Four	오픈 릴레이션 번호
IN	iid	IndexID*	스캔에 사용되는 인덱스의 식별자
IN	lowerBounds[]	MLGF_HashValue	객체를 찾을 영역의 각 키에 대한 최소값
IN	upperBounds[]	MLGF_HashValue	객체를 찾을 영역의 각 키에 대한 최대값
IN	nBools	Four	boolean expression 의 개수
IN	bool[]	BoolExp	boolean expression 들의 리스트
IN	lockup	LockParameter*	요청하는 lock 에 대한 정보

Description

주어진 릴레이션에 대한 MLGF 인덱스 스캔을 오픈한다. 인덱스 스캔에 있어서 순차적 스캔과 마찬가지로 boolean expression을 줌으로써 boolean expression을 만족하는 튜플만을 액세스할 수 있다. lockup 파라메타는 계층구조 로크를 위해서 사용된다.

Return value

스캔 식별자 : 인덱스 스캔을 오픈하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                 volId;
Four                 scanId;
Four                 orn;
MLGF_HashValue   lowerBounds[2];
MLGF_HashValue   upperBounds[2];
BoolExp        bool[1];
LockParameter    lockup;
Four                 keyValue;
lrds_RelTableEntry  *relTableEntry;
……
```

```
orn = LRDS_OpenRelation(handle, volId, “relation”);
if(orn < eNOERROR) /* error 처리 */

relTableEntry = LRDS_GET_RELTABLE_ENTRY(handle, orn);
……
lowerBounds[0] = 1;
lowerBounds[1] = 1;

upperBounds[0] = 10;
upperBounds[1] = 10;

bool[0].op = SM_GT;
bool[0].colNo = 1;
bool[0].data.i = 20;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

scanId = LRDS_OpenSeqScan(handle, orn,
   &(LRDS_GET_IDXINFO_FROM_RELTABLE_ENTRY(relTableEntry))[0].iid,
   lowerBounds, upperBounds, 1, bool, &lockup);
if(scanId < eNOERROR) /* error 처리 */
……
e = LRDS_CloseScan(handle, scanId);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_CloseRelation(handle, orn);
if(e < eNOERROR) /* error 처리 */
……
```

5.4. LRDS_MLGF_SearchNearTuple

Syntax

Four LRDS_MLGF_SearchNearTuple(Four handle, Four orn, IndexID *iid, MLGF_HashValue kval[], TupleID *tid, LockParameter* lockup)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	orn	Four	오픈 릴레이션 번호
IN	iid	IndexID*	스캔에 사용되는 인덱스의 식별자
IN	kval[]	MLGF_HashValue	이 키 값에 가까이 있는 객체를 찾음
OUT	tid	TupleID*	튜플 식별자
IN	lockup	LockParameter*	요청하는 lock 에 대한 정보

Description

MLGF 색인에서 주어진 키에 가까이 있는 객체를 찾는다.

Return value

eNOERROR : 주어진 키 값에 가까이 있는 객체를 찾았음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                 volId;
Four                 scanId;
Four                 orn;
MLGF_HashValue  kval[2];
TupleID       tid;
LockParameter   lockup;
lrds_RelTableEntry  *relTableEntry;
……
orn = LRDS_OpenRelation(handle, volId, “relation”);
if(orn < eNOERROR) /* error 처리 */

relTableEntry = LRDS_GET_RELTABLE_ENTRY(handle, orn);
……
kval[0] = 5;
kval[1] = 1;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

e = LRDS_MLGF_SearchNearTuple(handle, orn,
   &(LRDS_GET_IDXINFO_FROM_RELTABLE_ENTRY(relTableEntry))[0].iid,
   kval, &tid, &lockup);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_CloseRelation(handle, orn);
if(e < eNOERROR) /* error 처리 */
……
```

5.5. LRDS_CloseScan

Syntax

Four LRDS_CloseScan(Four handle, Four scanId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	scanId	Four	스캔 식별자

Description

스캔을 닫는다. 닫을 스캔은 스캔 식별자를 가지고 지정한다.

Return value

eNOERROR : 스캔을 성공적으로 닫았음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                  scanId;
Four                  orn;
BoolExp        bool[2];
LockParameter   lockup;
……

bool[0].op = SM_LE;
bool[0].colNo = 1;
bool[0].data.i = 10;

bool[1].op = SM_GE;
bool[1].colNo = 1;
bool[1].data.i = 20;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

scanId = LRDS_OpenSeqScan(handle, orn, FORWARD, 2, bool, &lockup);
if(scanId < eNOERROR) /* error 처리 */
……
e = LRDS_CloseScan(handle, scanId);
if(e < eNOERROR) /* error 처리 */
……
```

5.6. LRDS_CloseAllScans

Syntax

Four LRDS_CloseAllScans(Four handle)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자

Description

현재 thread가 오픈한 스캔을 모두 닫는다.

Return value

eNOERROR : 모든 스캔을 성공적으로 닫았음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
```

```
……

e = LRDS_CloseAllScans(handle);
if(e < eNOERROR) /* error 처리 */
……
```

5.7. LRDS_NextTuple

Syntax

Four LRDS_NextTuple(Four handle, Four scanId, TupleID *tid, LRDS_Cursor** cursor)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	scanId	Four	스캔 식별자
OUT	tid	TupleID*	다음 튜플의 식별자
OUT	cursor	LRDS_Cursor**	스캔의 커서

Description

다음에 스캔할 튜플로 스캔 커서를 이동하고 새로운 튜플의 튜플 식별자를 리턴해 준다. 사용자는 리턴된 튜플 식별자를 직접 LRDS_FetchTuple(), LRDS_UpdateTuple(), LRDS_DestroyTuple() 함수의 파라미터로 전달하여 리턴된 튜플에 원하는 연산을 할 수 있다.

Return value

eNOERROR : 스캔의 커서를 성공적으로 이동함

EOS : 스캔의 커서가 마지막 튜플을 가리키고 있음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                 scanId;
Four                 orn;
BoolExp        bool[2];
LockParameter   lockup;
TupleID        tid;
LRDS_Cursor*    cursor;
……

bool[0].op = SM_LE;
bool[0].colNo = 1;
bool[0].data.i = 10;

bool[1].op = SM_GE;
bool[1].colNo = 1;
```

```
bool[1].data.i = 20;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

scanId = LRDS_OpenSeqScan(handle, orn, FORWARD, 2, bool, &lockup);
if(scanId < eNOERROR) /* error 처리 */
……
e = LRDS_NextTuple(handle, scanId, &tid, &cursor);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_CloseScan(handle, scanId);
if(e < eNOERROR) /* error 처리 */
……
```

6. 튜플 관리

6.1. LRDS_CreateTuple

Syntax

```
Four LRDS_CreateTuple(Four handle, Four ornOrScanId, Boolean useScanFlag,
Four nCols, ColListStruct *clist, TupleID *tid)
```

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	nCols	Four	튜플 생성시에 데이타를 저장할 컬럼의 개수(coarse granularity locking 버전에서는 타입이 Two 임)
IN	clist	ColListStruct*	튜플의 초기 컬럼 값들
OUT	tid	TupleID*	생성된 튜플의 튜플 식별자

Description

새로운 튜플을 스캔이 오픈된 릴레이션에 삽입한다. 새로운 튜플의 컬럼 값은 파라미터 clist를 통하여 전달된다. 파라미터 clist에 엔트리의 수는 파라미터 nCols를 통하여 전달된다. 컬럼은 NULL 값을 가질 수 있으므로 모든 컬럼에 대한 내용이 clist에 들어 있을 필요는 없다. 또한 가변길이를 허용하는 SM_VARSTRING 타입의 컬럼 뿐만 아니라 SM_STRING 타입의 컬럼들도 부분적으로 데이터를 저장할 수 있다. 단, 이들 SM_STRING 타입의 컬럼에 대해서는 이들 값이 완전히 주어졌을 때를 대비하여 공간이 미리 준비된다. 단지 데이터를 한번에 기록하지 않고 여러 번에 나눠서 기록할 수 있음을 의미한다. 주의할 점은 LRDS에서는 데이터가 완전히 채워졌는지 부분적으로 채워졌는지를 기억하고 있지 않으므로 사용자가 이를 기억하고 있어야 한다.

Return value

eNOERROR : 튜플을 성공적으로 생성하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                 volId;
Four                 orn;
TupleID       tid;
```

```
ColListStruct   clist[2];
char            data[10]="abcdefghij"
Four                  value;
……
orn = LRDS_OpenRelation(handle, volId, "relation");
if(orn < eNOERROR) /* error 처리 */
……
clist[0].colNo = 0;
clist[0].nullFlag = SM_FALSE;
clist[0].start = ALL_VALUE;
clist[0].dataLength = sizeof(Four);
value = 5;
memcpy(&(clist[0].data), &value, sizeof(Four));

clist[1].colNo = 3;
clist[1].nullFlag = SM_FALSE;
clist[1].start = ALL_VALUE;
clist[1].dataLength = strlen(data);
clist[1].data.ptr = data;

e = LRDS_CreateTuple(handle, orn, SM_FALSE, 2, clist, &tid);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_CloseRelation(handle, orn);
if(e < eNOERROR) /* error 처리 */
……
```

6.2. LRDS_DestroyTuple

Syntax

Four LRDS_DestroyTuple(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	삭제할 튜플의 식별자

Description

릴레이션에서 튜플 하나를 삭제한다. 삭제하는 튜플은 파라미터 tid를 통하여 지정한다. 만일 tid의 값이 NULL이면 스캔이 현재 가리키고 있는 튜플을 삭제한다. 튜플을 삭제함과 동시에 그 릴레이션에 정의되어 있는 모든 인덱스에 대하여 삭제되는 튜플에 관한 엔트리를 삭제해 준다.

Return value

eNOERROR : 튜플을 성공적으로 삭제하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                 scanId;
Four                 orn;
BoolExp       bool[1];
LockParameter   lockup;
……

bool[0].op = SM_EQ;
bool[0].colNo = 1;
bool[0].data.i = 10;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

scanId = LRDS_OpenSeqScan(handle, orn, FORWARD, 1, bool, &lockup);
if(scanId < eNOERROR) /* error 처리 */
……
e = LRDS_DestroyTuple(handle, scanId, SM_TRUE, NULL);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_CloseScan(handle, scanId);
if(e < eNOERROR) /* error 처리 */
……
```

6.3. LRDS_UpdateTuple

Syntax

Four LRDS_UpdateTuple(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID *tid, Four nCols, ColListStruct *clist)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	갱신할 튜플의 튜플 식별자
IN	nCols	Four	갱신할 컬럼의 개수(coarse granularity locking 버전에서는 타입이 Two 임)

IN	clist	ColListStruct*	갱신할 컬럼에 관한 정보

Description

현재 튜플 혹은 주어진 튜플의 몇 개의 컬럼 값을 갱신한다. 파라미터 tid의 값이 NULL이면 현재 튜플을 갱신하고, 파라미터 tid의 값이 NULL이 아니면 주어진 튜플을 갱신한다. 갱신할 컬럼들의 번호와 갱신에 관한 정보는 파라미터 clist를 통하여 전달된다. 파라미터 clist에 있는 엔트리의 수는 파라미터 nCols로 지정한다. 각 컬럼의 갱신에 대하여 4가지 정보가 필요하다. 첫째는 갱신되는 데이터의 시작위치이며, 둘째는 갱신되는 기존 데이터의 양이며, 셋째는 기존 데이터를 대치할 새로운 데이터의 양이며, 넷째는 새로운 데이터의 내용이다. 기존 데이터의 양이 새로운 데이터의 양보다 작으면 기존 데이터의 내용을 갱신함과 동시에 두 값의 차이만큼의 데이터가 더 삽입된 것을 의미한다. 기존 데이터의 양이 새로운 데이터의 양보다 크면, 그 값의 차이만큼의 데이터의 양이 줄어들게 된다. 컬럼 값의 길이 변하는 것은 SM_VARSTRING뿐이므로 두 값이 차이가 날 수 있는 컬럼은 SM_VARSTRING 타입으로 선언된 컬럼뿐이다. 편의를 위해 몇 개의 특수한 값들이 파라미터로 사용될 수 있다. clist의 start 필드 값이 ALL_VALUE이면 그 컬럼의 기존 데이타 전체를 갱신한다는 것을 의미한다. 또한 start의 값이 END이면 기존의 데이타는 갱신하지 않고 새로운 데이타를 뒤에다 추가하겠다는 뜻이다. clist의 length 필드값이 REMAINDER인 경우는 start에 지정된 위치부터 그 컬럼의 값의 끝까지를 갱신하겠다는 의미이다.

Return value

eNOERROR : 튜플을 성공적으로 갱신하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"


Four                scanId;
Four                orn;
BoolExp        bool[1];
LockParameter   lockup;
TupleID        tid;
LRDS_Cursor*    cursor;
ColListStruct   clist[1];
char            data[10]=”abcdefghij”
……

bool[0].op = SM_EQ;
bool[0].colNo = 1;
bool[0].data.i = 10;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

scanId = LRDS_OpenSeqScan(handle, orn, FORWARD, 1, bool, &lockup);
if(scanId < eNOERROR) /* error 처리 */
……
```

```
clist[0].colNo = 3;
clist[0].nullFlag = SM_FALSE;
clist[0].start = ALL_VALUE;
clist[0].dataLength = strlen(data);
clist[0].data.ptr = data;

e = LRDS_UpdateTuple(handle, scanId, SM_TRUE, NULL, 1, clist);
if(e < eNOERROR) /* error 처리 */

……
e = LRDS_CloseScan(handle, scanId);
if(e < eNOERROR) /* error 처리 */

……
```

6.4. LRDS_FetchTuple

Syntax

Four LRDS_FetchTuple(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID *tid, Four nCols, ColListStruct clist[])

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	읽어올 튜플의 튜플 식별자
IN	nCols	Four	읽어올 컬럼의 개수(coarse granularity locking 버전에서는 타입이 Two 임)
INOUT	clist[]	ColListStruct	읽어올 컬럼에 관한 정보

Description

현재 튜플이나 주어진 튜플에서 원하는 컬럼들의 값을 읽어서 리턴한다. 읽어야 할 컬럼들은 파라미터 clist를 통하여 전달되며, 읽은 데이타의 값도 같은 파라미터를 통하여 리턴한다. 파라미터 clist에 있는 컬럼의 개수는 파라미터 nCols를 통하여 전달된다. 사용자는 각 컬럼에 대하여 어디에서 어디까지 읽어야 하는지를 지정해 주어야 한다. 여기서 위치는 각각 각 컬럼에 대하여 상대적 값이다. 읽을 데이터의 시작위치는 clist의 start 필드에 기록하고, 읽어야 할 데이터의 양은 dataLength 필드에 기록한다. 만일 start의 값이 ALL_VALUE라는 특별한 값이면 모든 컬럼 값을 다 읽어 들인다. 또한 dataLength의 값이 REMAINDER라는 특별한 값이면 주어진 start에서 컬럼의 끝에까지의 데이터를 읽어서 리턴한다. 읽고자 하는 컬럼의 타입이 SM_STRING이거나 SM_VARSTRING인 경우에는 컬럼 값을 리턴할 수 있는 공간을 사용자가 확보한 뒤 이에 대한 포인터를 clist를 통하여 전달해 주어야 한다. 이 함수에

서는 dataLength만큼의 공간이 확보되어 있다고 가정한다.

Return value

eNOERROR : 튜플을 성공적으로 읽어 들였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"


Four                  scanId;
Four                  orn;
BoolExp        bool[1];
LockParameter   lockup;
TupleID        tid;
ColListStruct   clist[1];
char            data[100];
……

bool[0].op = SM_EQ;
bool[0].colNo = 1;
bool[0].data.i = 10;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

scanId = LRDS_OpenSeqScan(handle, orn, FORWARD, 1, bool, &lockup);
if(scanId < eNOERROR) /* error 처리 */
……

clist[0].colNo = 3;
clist[0].nullFlag = SM_FALSE;
clist[0].start = ALL_VALUE;
clist[0].dataLength = strlen(data);
clist[0].data.ptr = data;

e = LRDS_FetchTuple(handle, scanId, SM_TRUE, NULL, 1, clist);
if(e < eNOERROR) /* error 처리 */

……
e = LRDS_CloseScan(handle, scanId);
if(e < eNOERROR) /* error 처리 */

……
```

6.5. LRDS_FetchColLength

Syntax

Four LRDS_FetchColLength(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID *tid, Four nCols, ColLengthInfoListStruct lengthInfoList[])

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	nCols	Four	길이를 알고 싶은 컬럼의 개수(coarse granularity locking 버전에서는 타입이 Two 임)
INOUT	lengthInfoList[ ]	ColLengthInfoListStruct	컬럼의 길이 정보를 갖고 오기 위한 버퍼

Description

컬럼의 길이를 얻어온다.

Return value

eNOERROR : 컬럼의 길이를 성공적으로 얻어 왔음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"


Four                    scanId;
Four                    orn;
BoolExp         bool[1];
LockParameter    lockup;
TupleID         tid;
ColLengthInfoListStruct lengthInfo [1]
……

bool[0].op = SM_EQ;
bool[0].colNo = 1;
bool[0].data.i = 10;

lockup.mode = L_X;
lockup.duration = L_COMMIT;

scanId = LRDS_OpenSeqScan(handle, orn, FORWARD, 1, bool, &lockup);
if(scanId < eNOERROR) /* error 처리 */
……

lengthInfo[0].colNo = 3;
```

```
e = LRDS_FetchColLength(handle, scanId, SM_TRUE, NULL, 1, lengthInfo);
if(e < eNOERROR) /* error 처리 */

……
e = LRDS_CloseScan(handle, scanId);
if(e < eNOERROR) /* error 처리 */

……
```

7. 카운터 관리

7.1. LRDS_CreateCounter

Syntax

Four LRDS_CreateCounter(Four handle, Four volId, char *cntrName, Four initialValue, CounterID *cntrId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨 식별자
IN	cntrName	char*	생성할 카운터의 이름
IN	initialValue	Four	생성할 카운터의 초기값
OUT	cntrId	CounterID*	생성된 카운터의 식별자

Description

주어진 이름으로 카운터를 생성한다.

Return value

eNOERROR : 카운터를 성공적으로 생성하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
CounterID      cntrId;
……

e = LRDS_CreateCounter(handle, volId, “testCounter”, 0, &cntrId);
if(e < eNOERROR) /* error 처리 */
……
```

7.2. LRDS_DestroyCounter

Syntax

Four LRDS_DestroyCounter(Four handle, Four volId, char *cntrName)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨 식별자
IN	cntrName	char*	생성할 카운터의 이름

Description

주어진 이름의 카운터를 삭제한다.

Return value

eNOERROR : 카운터를 성공적으로 삭제하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
……

e = LRDS_DestroyCounter(handle, volId, “testCounter”);
if(e < eNOERROR) /* error 처리 */
……
```

7.3. LRDS_GetCounterId

Syntax

Four LRDS_GetCounterId(Four handle, Four volId, char *cntrName, CounterID *cntrId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨 식별자
IN	cntrName	char*	카운터의 이름
OUT	cntrId	CounterID*	카운터의 식별자

Description

주어진 이름을 갖는 카운터의 식별자를 얻어온다.

Return value

eNOERROR : 카운터 식별자를 성공적으로 얻어왔음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
CounterID     cntrId;
……
```

```
e = LRDS_GetCounterId(handle, volId, “testCounter”, &cntrId);
if(e < eNOERROR) /* error 처리 */
……
```

7.4. LRDS_SetCounter

Syntax

Four LRDS_SetCounter(Four handle, Four volId, CounterID *cntrId, Four value)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨 식별자
IN	cntrId	CounterID*	카운터의 식별자
IN	value	Four	set 할 카운터의 값

Description

카운터의 값을 새로운 값으로 set한다.

Return value

eNOERROR : 카운터의 값을 성공적으로 set하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
CounterID     cntrId;
……

e = LRDS_GetCounterId(handle, volId, “testCounter”, &cntrId);
if(e < eNOERROR) /* error 처리 */

e = LRDS_SetCounter(handle, volId, &cntrId, 4);
if(e < eNOERROR) /* error 처리 */

……
```

7.5. LRDS_ReadCounter

Syntax

Four LRDS_ReadCounter(Four handle, Four volId, CounterID *cntrId, Four* value)

Parameters

IN/OUT	이름	타입	설명

IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨 식별자
IN	cntrId	CounterID*	카운터의 식별자
OUT	value	Four*	카운터의 값

Description

카운터의 값을 읽는다.

Return value

eNOERROR : 카운터의 값을 성공적으로 읽었음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
CounterID     cntrId;
Four                  value;
……

e = LRDS_GetCounterId(handle, volId, “testCounter”, &cntrId);
if(e < eNOERROR) /* error 처리 */

e = LRDS_ReadCounter(handle, volId, &cntrId, &value);
if(e < eNOERROR) /* error 처리 */

……
```

7.6. LRDS_GetCounterValues

Syntax

Four LRDS_GetCounterValues(Four handle, Four volId, CounterID *cntrId, Four nValues, Four *startValue)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	볼륨 식별자
IN	cntrId	CounterID*	카운터의 식별자
IN	nValues	Four	카운터 값을 nValues 만큼 증가 시킴
OUT	startValue	Four*	증가시키기 전의 카운터 값

Description

카운터의 값을 읽어오고 nValues만큼 증가시킨다.

Return value

eNOERROR : 카운터의 값을 성공적으로 읽고 증가시켰음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
CounterID     cntrId;
Four                  startValue;
……

e = LRDS_GetCounterId(handle, volId, “testCounter”, &cntrId);
if(e < eNOERROR) /* error 처리 */

e = LRDS_GetCounterValues(handle, volId, &cntrId, 2, &startValue);
if(e < eNOERROR) /* error 처리 */

……
```

8. I/O 횟수 정보

8.1. LRDS_ResetNumberOfDiskIO

Syntax

Four LRDS_ResetNumberOfDiskIO(Four handle)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자

Description

디스크 I/O의 횟수를 세기 위한 변수를 초기화한다.

Return value

eNOERROR : 디스크 I/O의 횟수를 세기 위한 변수를 초기화하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            read;
Four            write;
……
e = LRDS_ResetNumberOfDiskIO(handle);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_GetNumberOfDiskIO(handle, &read, &write);
if(e < eNOERROR) /* error 처리 */
……
```

8.2. LRDS_GetNumberOfDiskIO

Syntax

Four LRDS_GetNumberOfDiskIO(Four handle, Four* read, Four* write)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
OUT	read	Four*	디스크를 읽은 횟수
OUT	write	Four*	디스크를 쓴 횟수

Description

디스크 I/O의 횟수를 읽어온다.

Return value

eNOERROR : 디스크 I/O의 횟수를 성공적으로 읽어옴

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            read;
Four            write;
……
e = LRDS_ResetNumberOfDiskIO(handle);
if(e < eNOERROR) /* error 처리 */
……
e = LRDS_GetNumberOfDiskIO(handle, &read, &write);
if(e < eNOERROR) /* error 처리 */
……
```

9. 벌크로드

9.1. LRDS_InitRelationBulkLoad

Syntax

Four LRDS_InitRelationBulkLoad(Four handle, Four volId, Four tmpVolId, char* inRelName, Boolean isNeedSort, Boolean indexBlkLdFlag, Two pff, Two eff, LockParameter* lockup)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	volId	Four	릴레이션이 포함된 데이터 볼륨의 식별자
IN	tmpVolId	Four	벌크로드 도중 생성되는 sort stream 이 저장되는 임시 볼륨의 식별자
IN	inRelName	char*	릴레이션 이름
IN	isNeedSort	Boolean	클러스터링 인덱스 키에 대해 정렬하여 벌크로드할 지 지정하는 플래그
IN	indexBlkLdFlag	Boolean	인덱스 생성시 벌크로드 루틴을 사용할 지 지정하는 플래그
IN	pff	Two	페이지 채움 지수
IN	eff	Two	익스텐트 채움 지수
IN	lockup	LockParameter*	동시성 제어 인수

Description

릴레이션 벌크로드를 위한 준비(초기화)를 한다.

Return value

벌크로드 식별자: 릴레이션 벌크로드를 성공적으로 초기화하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
Four                  tmpVolId;
Four                  blkLdId
LockParameter   lockup;
……

lockup.mode = L_X;
```

```
lockup.duration = L_COMMIT;

blkLdId = LRDS_InitRelationBulkLoad(handle, volId, tmpVolId, “test”,
SM_FALSE, SM_FALSE, 100, 100, &lockup);
if(e < blkLdId) /* error 처리 */
……
e = LRDS_FinalRelationBulkLoad (handle, blkLdId);
if(e < eNOERROR) /* error 처리 */
……
```

9.2. LRDS_FinalRelationBulkLoad

Syntax

Four LRDS_FinalRelationBulkLoad(Four handle, Four blkLdId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	blkLdId	Four	벌크로드 식별자

Description

릴레이션 벌크로드를 정리(말기화) 한다.

Return value

eNOERROR : 릴레이션 벌크로드를 성공적으로 말기화하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four            volId;
Four                 tmpVolId;
Four                 blkLdId
LockParameter   lockup;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

blkLdId = LRDS_InitRelationBulkLoad(handle, volId, tmpVolId, “test”,
SM_FALSE, SM_FALSE, 100, 100, &lockup);
if(e < blkLdId) /* error 처리 */
……
e = LRDS_FinalRelationBulkLoad (handle, blkLdId);
if(e < eNOERROR) /* error 처리 */
……
```

9.3. LRDS_NextRelationBulkLoad

Syntax

Four LRDS_NextRelationBulkLoad(Four handle, Four blkLdId, Four nCols, ColListStruct* clist, Boolean endOfTuple, TupleID* tid)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	blkLdId	Four	벌크로드 식별자
IN	nCols	Four	삽입되는 컬럼들의 개수(coarse granularity locking 버전에서는 타입이 Two 임)
IN	clist	ColListStruct*	삽입되는 컬럼들의 정보
IN	endOfTuple	Boolean	삽입되는 컬럼들이 튜플을 구성하는 마지막 컬럼인지 지정하는 플래그
OUT	tid	TupleID*	생성된 튜플의 튜플 식별자

Description

릴레이션 벌크로드를 사용하여 튜플을 구성하는 컬럼들을 릴레이션에 삽입한다.

Return value

eNOERROR : 튜플을 성공적으로 삽입하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            e;
Four                 volId;
Four                 tmpVolId;
Four                 blkLdId
LockParameter   lockup;
TupleID       tid;
ColListStruct   clist[2];
char            data[10]=”abcdefghij”
Four                 value;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

blkLdId = LRDS_InitRelationBulkLoad(handle, volId, tmpVolId, “test”,
SM_FALSE, SM_FALSE, 100, 100, &lockup);
```

```
if(e < blkLdId) /* error 처리 */
……
clist[0].colNo = 0;
clist[0].nullFlag = SM_FALSE;
clist[0].start = ALL_VALUE;
clist[0].dataLength = sizeof(Four);
value = 5;
memcpy(&(clist[0].data), &value, sizeof(Four));

clist[1].colNo = 1;
clist[1].nullFlag = SM_FALSE;
clist[1].start = ALL_VALUE;
clist[1].dataLength = strlen(data);
clist[1].data.ptr = data;

……
e = LRDS_NextRelationBulkLoad(handle, blkLdId, 2, clist, SM_TRUE,
&tid);
if(e < eNOERROR) /* error 처리 */

……
e = LRDS_FinalRelationBulkLoad(handle, blkLdId);
if(e < eNOERROR) /* error 처리 */
……
```

10. Ordered Set

10.1. LRDS_OrderedSet_Create

Syntax

Four LRDS_OrderedSet_Create(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 생성되는 컬럼의 번호 (coarse granularity locking 버전에서는 타입이 Two 임)
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

주어진 컬럼에 빈(empty) 순서를 갖는 집합(ordered set)을 생성한다. 컬럼이 속하는 릴레이션과 튜플도 입력으로 주어진다. 튜플을 지정하는 튜플 식별자의 값이 NULL이면 스캔의 현재 튜플 안에 집합이 생성된다. 컬럼에 집합이 이미 생성되어 있는 경우에는 에러를 리턴한다.

Return value

eNOERROR : 집합을 생성하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
LockParameter   lockup;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

e = LRDS_OrderedSet_Create(handle, scanId, SM_TRUE, NULL, 3, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.2. LRDS_OrderedSet_Destroy

Syntax

Four LRDS_OrderedSet_Destroy(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 삭제되는 컬럼의 번호 (coarse granularity locking 버전에서는 타입이 Two 임)
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

주어진 컬럼에 저장된 순서를 갖는 집합(ordered set)을 삭제한다. 컬럼이 속하는 튜플과 릴레이션은 입력으로 주어진다. 집합이 삭제되면 그 컬럼은 NULL 값을 갖게 된다.

Return value

eNOERROR : 집합을 삭제하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
LockParameter   lockup;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

e = LRDS_OrderedSet_Destroy(handle, scanId, SM_TRUE, NULL, 3, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.3. LRDS_OrderedSet_CreateNestedIndex

Syntax

Four LRDS_OrderedSet_CreateNestedIndex(Four handle, Four ornFour colNo)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	orn	Four	오픈 릴레이션 번호
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)

Description

릴레이션과 그 릴레이션에 속하는 순서를 갖는 집합(ordered set) 컬럼이 주어졌을 때, 튜플들을 차례로 스캔하면서 서브색인을 갖을 만큼 충분히 큰 집합이 있으면 서브색인을 갖는 집합으로 변환한다. 또한 LRDS_OrderedSet_DestroyNestedIndex() 함수를 호출하기 전까지는 집합의 크기가 충분히 커지면 자동적으로 서브색인을 갖게 된다. 서브색인을 갖을 정도의 충분한 크기라 함은 주어진 키값을 갖는 원소를 찾는 연산에 있어서 서브색인이 없을 때의 디스크 액세스 횟수보다 서브색인이 있을 때 디스크 액세스 횟수가 적어지는 크기를 말한다.

Return value

eNOERROR : 집합이 서브색인을 갖도록 설정하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            orn;
……
e = LRDS_OrderedSet_CreateNestedIndex(handle, orn, 3);
if(e < eNOERROR) /* error 처리 */
……
```

10.4. LRDS_OrderedSet_DestroyNestedIndex

Syntax

Four LRDS_OrderedSet_DestroyNestedIndex(Four handle, Four ornFour colNo)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	orn	Four	오픈 릴레이션 번호
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이

			Two 임)

Description

릴레이션과 그 릴레이션에 속하는 순서를 갖는 집합(ordered se) 컬럼이 주어졌을 때, 튜플들을 차례로 스캔하면서 서브색인을 갖는 집합이 있으면 서브색인을 갖지 않는 집합으로 변환한다. LRDS_OrderedSet_CreateNestedIndex() 함수를 호출하기 전까지는 집합의 크기가 충분히 커져도 서브색인을 갖지 않게 된다.

Return value

eNOERROR : 집합이 서브색인을 갖지 않도록 설정하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            orn;
……
e = LRDS_OrderedSet_DestroyNestedIndex(handle, orn, 3);
if(e < eNOERROR) /* error 처리 */
……
```

10.5. LRDS_OrderedSet_AppendSortedElements

Syntax

Four LRDS_OrderedSet_AppendSortedElements(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, Four nElements, Four elementsBufSize, char *elementsBuf, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
IN	nElements	Four	삽입될 원소들의 수
IN	elementsBufSize	Four	삽입될 원소들을 담고 있는 버퍼의

			크기
IN	elementsBuf	char*	삽입될 원소들을 담고 있는 버퍼
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

주어진 집합의 끝에 원소들을 삽입한다. 여러 원소들을 한번에 삽입함으로써 하나씩 삽입할 때보다 성능이 우수하다. 삽입할 원소들은 elementsBuf라는 버퍼를 통해 주어지는데, (데이터의 길이, 데이터)의 쌍이 연속해서 있는 배열 형태를 갖는다. 삽입되는 원소들은 키값의 오름차순으로 정렬되어 있어야 하며, 또한 현재 집합의 끝에 있는 원소의 키값보다 크거나 같아야 한다.

Return value

eNOERROR : 집합에 원소들을 삽입하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
LockParameter   lockup;
char                 elementBuf[100];
OrderedSet_ElementLength elementLength;
char            data1[] = “abcd”;
char            data2[] = “efg”;
Four                 offset;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

offset = 0;

elementLength = strlen(data1);
memcpy(&elementBuf[offset], &elementLength,
                            sizeof(OrderedSet_ElementLength));
offset += sizeof(OrderedSet_ElementLength);
memcpy(&elementBuf[offset], data1, elementLength);
offset += elementLength;

elementLength = strlen(data2);
memcpy(&elementBuf[offset], &elementLength,
                            sizeof(OrderedSet_ElementLength));
offset += sizeof(OrderedSet_ElementLength);
memcpy(&elementBuf[offset], data2, elementLength);
offset += elementLength;

e = LRDS_OrderedSet_AppendSortedElements(handle, scanId, SM_TRUE,
NULL, 3, 2, offset, elementBuf, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.6. LRDS_OrderedSet_InsertElement

Syntax

Four LRDS_OrderedSet_InsertElement(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, char *element, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
IN	element	char*	삽입될 원소를 담고 있는 버퍼
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

주어진 집합에 원소 하나를 키값에 따른 적당한 위치에 삽입한다. 이 함수는 집합에 원소들이 삽입된 후 부득이하게 키값이 집합 끝에 있는 원소의 키값보다 큰 원소를 삽입해야 하는 경우를 처리하기 위해 제공되는 함수이다. 이 함수를 사용하면 LRDS_OrderedSet_AppendSortedElements() 함수를 사용할 때보다 성능이 떨어진다. 삽입할 원소는 element라는 버퍼를 통해 주어지는데,(데이터의 길이, 데이터)의 쌍이 연속해서 있는 배열 형태를 갖는다.

Return value

eNOERROR : 집합에 원소를 삽입하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
LockParameter   lockup;
char                  element[100];
OrderedSet_ElementLength elementLength;
char            data[] = “abcd”;
Four                  offset;
……
```

```
lockup.mode = L_X;
lockup.duration = L_COMMIT;

offset = 0;

elementLength = strlen(data);
memcpy(&element[offset], &elementLength,
                              sizeof(OrderedSet_ElementLength));
offset += sizeof(OrderedSet_ElementLength);
memcpy(&element[offset], data, elementLength);
offset += elementLength;

e = LRDS_OrderedSet_InsertElement(handle, scanId, SM_TRUE, NULL, 3,
element, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.7. LRDS_OrderedSet_DeleteElement

Syntax

Four LRDS_OrderedSet_DeleteElement(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, KeyValue *kval, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
IN	kval	KeyValue *	삭제될 원소의 키값
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

주어진 집합에서 원소를 삭제한다. 삭제될 원소는 원소의 키값 kval을 통해 주어진다.

Return value

eNOERROR : 집합에서 원소를 삭제하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
LockParameter   lockup;
KeyValue        kval;
Four                 key;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

key = 10;
kval.len = sizeof(Four);
memcpy(&(kval.val[0]), &key, sizeof(Four));

e = LRDS_OrderedSet_DeleteElement(handle, scanId, SM_TRUE, NULL, 3,
&kval, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.8. LRDS_OrderedSet_DeleteElements

Syntax

Four LRDS_OrderedSet_DeleteElements(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, Four nElementsToDelete, KeyValue *kval, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
IN	nElementsToDelete	Four	삭제될 원소들의 수
IN	kval	KeyValue *	삭제될 원소들의 키값 (배열)
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

주어진 집합에서 원소들을 삭제한다. 삭제될 원소들은 원소의 키값의 배열인 kval을 통해 주어진다.

Return value

eNOERROR : 집합에서 원소들을 삭제하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
LockParameter   lockup;
KeyValue        kval[2];
Four                 key;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

key = 10;
kval[0].len = sizeof(Four);
memcpy(&(kval[0].val[0]), &key, sizeof(Four));

key = 12;
kval[1].len = sizeof(Four);
memcpy(&(kval[1].val[0]), &key, sizeof(Four));

e = LRDS_OrderedSet_DeleteElements(handle, scanId, SM_TRUE, NULL, 3,
2, kval, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.9. LRDS_OrderedSet_UpdateElement

Syntax

Four LRDS_OrderedSet_UpdateElement(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, KeyValue *kval, Four updateStart, Four updateLength, Four updateDataLength, void* updateData, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는

			플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
IN	kval	KeyValue *	수정될 원소의 키값
IN	updateStart	Four	원소 내에서 수정될 부분의 시작 offset
IN	updateLength	Four	수정될 부분의 길이
IN	updateDataLength	Four	수정될 부분을 대체할 데이터의 길이
IN	updateData	void*	수정될 부분을 대체할 데이터
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

주어진 집합에서 원소를 수정한다. 수정될 원소는 원소의 키값 kval을 통해 주어진다.

Return value

eNOERROR : 집합에서 원소들을 삭제하였음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
LockParameter   lockup;
KeyValue        kval;
Four                 key;
char                 data[]=”abc”;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

key = 10;
kval.len = sizeof(Four);
memcpy(&(kval.val[0]), &key, sizeof(Four));

e = LRDS_OrderedSet_UpdateElement(handle, scanId, SM_TRUE, NULL, 3,
&kval, 3, 4, strlen(data), (void*)data, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.10. LRDS_OrderedSet_Scan_Open

Syntax

Four LRDS_OrderedSet_Scan_Open(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

주어진 집합에 대한 스캔을 연다. 스캔이 행해질 집합을 지정하기 위하여 릴레이션과 튜플 및 컬럼이 입력으로 주어진다. 어떤 집합에 대하여 스캔은 한 순간에 오직 하나만 있도록 구현된다. 스캔이 여러 개가 되면 스캔 도중 스캔의 대상이 되는 집합이 변경될 때마다 각 스캔의 위치들을 변경하는 오버헤드가 있기 때문이다.

Return value

집합 스캔 식별자: 집합에 대한 스캔을 열었음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
Four                  setScanId;
LockParameter   lockup;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

setScanId = LRDS_OrderedSet_Scan_Open(handle, scanId, SM_TRUE, NULL,
3, FORWARD, &lockup);
if(setScanId < eNOERROR) /* error 처리 */
```

```
……

e = LRDS_OrderedSet_Scan_Close(handle, setScanId);
if(e < eNOERROR) /* error 처리 */
……
```

10.11. LRDS_OrderedSet_Scan_Close

Syntax

Four LRDS_OrderedSet_Scan_Close(Four handle, Four setScanId)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	setScanId	Four	집합에 대한 스캔 식별자

Description

주어진 집합에 대한 스캔을 닫는다.

Return value

eNOERROR : 집합에 대한 스캔을 닫았음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
Four                  setScanId;
LockParameter   lockup;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

setScanId = LRDS_OrderedSet_Scan_Open(handle, scanId, SM_TRUE, NULL,
3, FORWARD, &lockup);
if(setScanId < eNOERROR) /* error 처리 */
……

e = LRDS_OrderedSet_Scan_Close(handle, setScanId);
if(e < eNOERROR) /* error 처리 */
……
```

10.12. LRDS_OrderedSet_Scan_NextElements

Syntax

Four LRDS_OrderedSet_Scan_NextElements(Four handle, Four setScanId, Four bufSize, char *elementBuf)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	setScanId	Four	집합에 대한 스캔 식별자
IN	bufSize	Four	읽은 원소들을 반환하기 위해 사용되는 버퍼의 크기
OUT	elementBuf	char*	읽은 원소들를 반환할 버퍼

Description

주어진 스캔에서 현재 위치부터 주어진 버퍼 크기에 들어가는 개수만큼의 원소들을 읽는다. 스캔의 위치는 읽은 원소만큼 증가한다. 읽혀진 원소들은 버퍼를 통해 반환되는데, (데이터의 길이, 데이터)의 쌍이 연속해서 있는 배열 형태를 갖는다. 실제 읽은 원소들의 수는 함수의 리턴 값으로 반환된다.

Return value

읽은 원소들의 수: 원소들을 읽었음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
Four                  setScanId;
LockParameter   lockup;
char elementSizeBuffer[256];
char elementBuffer[1024];
Four nElementsRead;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

setScanId = LRDS_OrderedSet_Scan_Open(handle, scanId, SM_TRUE, NULL,
3, FORWARD, &lockup);
if(setScanId < eNOERROR) /* error 처리 */
……

nElementsRead = LRDS_OrderedSet_Scan_NextElements(handle,
          setScanId, 256, elementSizeBuffer, 1024, elementBuffer);
if(nElementsRead < eNOERROR) /* error 처리 */
……

e = LRDS_OrderedSet_Scan_Close(handle, setScanId);
if(e < eNOERROR) /* error 처리 */
……
```

10.13. LRDS_OrderedSet_Scan_SkipElementsUntilGivenKeyValue

Syntax

Four LRDS_OrderedSet_Scan_SkipElementsUntilGivenKeyValue(Four handle, Four setScanId, Four keyLength, char* keyValue)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	setScanId	Four	집합에 대한 스캔 식별자
IN	keyLength	Four	키 값의 길이(coarse granularity locking 버전에서는 타입이 Two 임)
IN	keyValue	char*	키 값

Description

주어진 키보다 큰 값을 갖는 원소들을 읽을 수 있도록 스캔 커서를 이동시킨다.

Return value

eNOERROR : 스캔 커서를 이동시켰음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
Four                  setScanId;
LockParameter   lockup;
Four                  keyValue;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

setScanId = LRDS_OrderedSet_Scan_Open(handle, scanId, SM_TRUE, NULL,
3, FORWARD, &lockup);
if(setScanId < eNOERROR) /* error 처리 */
……

keyValue = 10;
e = LRDS_OrderedSet_Scan_SkipElementsUntilGivenKeyValue(handle,
                      setScanId, sizeof(Four), (char*) &keyValue);
if(e < eNOERROR) /* error 처리 */
……

e = LRDS_OrderedSet_Scan_Close(handle, setScanId);
if(e < eNOERROR) /* error 처리 */
……
```

10.14. LRDS_OrderedSet_GetTotalLengthOfElements

Syntax

Four LRDS_OrderedSet_GetTotalLengthOfElements(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, Four* totalLength, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
OUT	totalLength	Four*	원소들의 총 길이
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

순서를 갖는 집합(ordered set)에 저장된 원소들의 총 길이가 얼마이지 알아온다.

Return value

eNOERROR : 원소들의 총 길이를 알아왔음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
Four                  totalLength;
LockParameter   lockup;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

e = LRDS_OrderedSet_GetTotalLengthOfElements(handle, scanId, SM_TRUE,
NULL, 3, &totalLength, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.15. LRDS_OrderedSet_GetN_Elements

Syntax

Four LRDS_OrderedSet_GetN_Elements(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, Four* nElements, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
OUT	nElements	Four*	원소들의 개수
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

순서를 갖는 집합(ordered set)에 저장된 원소들의 개수를 알아온다.

Return value

eNOERROR : 원소들의 개수를 알아왔음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
Four                  nElements;
LockParameter   lockup;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

e = LRDS_OrderedSet_GetN_Elements(handle, scanId, SM_TRUE, NULL, 3,
&nElements, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.16. LRDS_OrderedSet_IsMember

Syntax

```
Four LRDS_OrderedSet_IsMember(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, KeyValue *kval, Four bufSize, char *elementBuf, LockParameter* lockupPtr)
```

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
IN	kval	KeyValue*	찾고자 하는 원소의 키값
IN	bufSize	Four	찾고자 하는 원소의 값을 반환할 버퍼의 크기
OUT	elementBuf	char*	찾고자 하는 원소의 값을 반환할 버퍼
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

순서를 갖는 집합(ordered set)에 저장된 주어진 키값을 갖는 원소가 속하는지를 검사한다. 키값을 갖는 원소가 있으면 버퍼를 통해 (데이터의 길이, 데이터)의 쌍으로 반환된다. 만일 (데이터의 길이, 데이터)의 쌍이 주어진 버퍼의 크기보다 큰 경우에는 버퍼의 크기와 ORDEREDSET_ELEMENT_FETCH_CHUNK_SIZE 중 작은 수만큼의 바이트들만 읽힌다.

Return value

1 : 원소가 있음

0 : 원소가 없음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"
```

```
Four            handle;
Four            scanId;
Four                  key;
KeyValue       kval;
Char                  buf[256];
LockParameter   lockup;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

key = 10;
kval.len = sizeof(Four);
memcpy(&(kval.val[0]), &key, sizeof(Four));

e = LRDS_OrderedSet_IsMember(handle, scanId, SM_TRUE, NULL, 3, &kval,
256, buf, &lockup);
if(e < eNOERROR) /* error 처리 */
……
```

10.17. LRDS_OrderedSet_HasNestedIndex

Syntax

Four LRDS_OrderedSet_HasNestedIndex(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo, LockParameter* lockupPtr)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)
IN	lockupPtr	LockParameter*	동시성 제어 인수

Description

순서를 갖는 집합(ordered set)에 서브색인이 있는지 확인한다.

Return value

SM_TRUE : 서브색인이 있음

SM_FALSE : 서브색인이 없음

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
Four            isSubIndexExist;
LockParameter   lockup;
……

lockup.mode = L_X;
lockup.duration = L_COMMIT;

isSubIndexExist  =  LRDS_OrderedSet_HasNestedIndex(handle,  scanId,
SM_TRUE, NULL, 3, &lockup);
if(isSubIndexExist < eNOERROR) /* error 처리 */
……
```

10.18. LRDS_OrderedSet_IsNull

Syntax

Four LRDS_OrderedSet_IsNull(Four handle, Four ornOrScanId, Boolean useScanFlag, TupleID* tid, Four colNo)

Parameters

IN/OUT	이름	타입	설명
IN	handle	Four	thread 관리용 식별자
IN	ornOrScanId	Four	오픈 릴레이션 번호 또는 스캔 식별자
IN	useScanFlag	Boolean	스캔을 사용하는지 여부를 나타내는 플래그
IN	tid	TupleID*	튜플 식별자
IN	colNo	Four	ordered set 이 있는 컬럼의 번호(coarse granularity locking 버전에서는 타입이 Two 임)

Description

순서를 갖는 집합(ordered set)이 NULL인지 아닌지를 알아낸다.

Return value

SM_TRUE : 집합이 NULL임

SM_FALSE : 집합이 NULL이 아님

< eNOERROR : 오류 코드

Example

```
#include "cosmos_r.h"

Four            handle;
Four            scanId;
Four            isNull;
……

isNull = LRDS_OrderedSet_IsNull(handle, scanId, SM_TRUE, NULL, 3);
if(isNull < eNOERROR) /* error 처리 */
……
```

10.19. LRDS_OrderedSet_SpecifyKeyOfElement

10.20. LRDS_OrderedSet_SpecifyVolNo

10.21. LRDS_OrderedSet_GetVolNo

11. Text

11.1. LRDS_Text_AddKeywords

11.2. LRDS_Text_DeleteKeywords

11.3. LRDS_Text_GetIndexID

12. SET

13. CollectionSet

14. CollectionBag

15. CollectionList

16. Error

16.1. LRDS_Err