ArkCDC v2.0 Extract Spec.

Extract Config

항목	Default	설명	비고
DBCONN	default	데이터베이스 connection 설정 파일 alias	-
TRACINGALIAS	-	Multi write 위한 tracing file alias 설정 (*필수)	하나의 Tracing File Alias 세트
TABLE	-	table 추출 대상 (*필수)	
SEQUENCE	-	sequence 추출 대상	
DDL	-	DDL 추출 대상	
TRACINGFILE_DEST	$ARKCDC_HOME/trace	tracing file 저장 경로	
TRACINGFILE_SIZE	50 MB	tracing file 사이즈	
MAP	-	map 설정 파일 alias	
FAILED_FETCH_LOB_TO_EMPTY	no	LOB 데이터 FETCH에 실패 시, EMPTY로 추출할 지 여부	
GET_TRUNCATE	yes	TRUNCATE DDL 별도 추출	
WRITE_FLUSH_RECORD_COUNT	1000	한번에 파일에 작성할 레코드 개수	
WRITE_FLUSH_BUFFER_SIZE	64k	한번에 파일에 작성할 레코드 사이즈	
IN_MEMORY_SIZE	0	메모리에 올릴 데이터들의 최대 사이즈	-
IN_MEMORY_TX_SIZE	1 GB	메모리에 올릴 트랜잭션 데이터들의 최대 사이즈	
LARGE_TRANSACTION_SIZE	128 MB	메모리에 올릴 한 트랜잭션의 최대 사이즈	
CACHE_FILE_DEST	$ARKCDC_HOME/cache	cache file 저장 경로	
CONVERT_LOB_TO_CHAR	-	LOB 데이터를 치환하여 추출할 문자열	
LOB_TO_NULL	no	LOB 데이터를 모두 NULL로 추출할지 여부	
ARK_TDE_FILENAME	-	wallet key file 저장 경로	

EXCLUDE_DML_TYPE	none	특정 DML 타입 추출을 포함/제외한다.
EXCLUDE_TAG	-	특정 태그로 트랜잭션을 추출에서 제외
EXCLUDE_TRANSACTION_NAME	-	특정 트랜잭션 명으로 트랜잭션을 추출에서 제외
EXCLUDE_USER_NAME	-	데이터베이스 사용자명을 통해 트랜잭션을 필터링.
FAILED_CONVERT_TO_UTF8	Extract 중지	UTF-8 변환 실패 시 Extract 동작을 제어한다.
ARCHIVE_ONLY	no	archive에서만 추출할지 여부
ARCHIVE_DEST	-	archive 경로
DISK_USAGE_CAPACITY	95	디스크 최대 사용률을 제어한다.
CHECKPOINT_SECONDS	10	checkpoint 갱신 간격 (단위: 초)
TRANSACTION_CHECK_INTERVAL	5s	commit 된 transaction 체크 간격 (단위: 초)
LONG_TRANSACTION_CHECK_INTERVAL	5m	long transaction 체크 간격 (단위: 초)
LONG_TRANSACTION_TIMEOUT	1h	long transaction 판단 시간 (단위: 초)
PERSISTENCE_MODE	LOGSWITCH	persist file 작성 기준 • logswitch • interval • off
PERSISTENCE_INTERVAL	6H	persist file 작성 간격 • 시간 단위 작성 • 0 : off와 동일 PERSISTENCE_MODE가 interval 설정일 경우에만 유효
INITIAL_COPY_THREADS	4	초기 복제 thread 개수
INITIAL_COPY_SPLIT_TRANSACTION	100000	초기 복제 트랜잭션의 최대 레코드 개수
INITIAL_COPY_SPLIT_SIZE	LARGE_TRANSACTION_SIZE	초기 복제 트랜잭션의 최대 크기
INITIAL_COPY_PARALLELISM_DEGREE	1	초기 복제 쿼리수행 병렬도
INITIAL_COPY_SCN_ASCENDING	-	초기 복제 쿼리 scn_ascending 힌트 사용 여부 • default: --start-dsn 옵션 시 힌트 사용, 그 외 힌트 미사용 • on: 힌트 사용 • off: 힌트 미사용
LOG_LEVEL	global.conf LOG_LEVEL (INFO)	extract logging level
LOG_BACKUP	global.conf LOG_BACKUP (all)	extract log file 백업 설정

Config 추가 버전
v. 1.4 or older
v. 1.5
v. 2.0

[표 1.1.Extract Config 표 참고]

추출 대상 객체 지정

추출 대상 객체 지정 기능은 추출 또는 추출에서 제외할 객체를 지정하는 기능이다.

목적

추출 대상을 ALL, 스키마 단위 또는 테이블 단위로 지정할 수 있도록 하기 위해 필요하다.

Wildcard를 사용하여 객체 명 패턴으로 추출할 경우 특정 객체는 제외되도록 할 때도 사용할 수 있다.

추출 대상 객체 지정 기능

추출 대상 객체 지정 기능은 Extract Config의 ***TABLE***과 ***SEQUENCE*** Config를 통해 설정이 가능하다.

Source 데이터베이스에서 추출 할 객체 명을 지정하면, 해당 객체가 Post Config에 지정 된 적용 대상 객체와 매핑된다.

매핑은 동일한 명을 기반으로 수행된다. 객체 명이 다를 경우, Post Mapper 파일에서 직접 매핑을 수행해야 한다. (객체 변환 추출 참고)

다음은 추출 대상 객체 설정 방법이다.

```
TABLE="[container.]schema(*).object(*)"
SEQUENCE="[container.]schema(*).object(*)"
```

- ***TABLE*** & ***SEQUENCE***
  - 최대 지원 길이: 255자
  - 지원 키워드: 문자(A-Z | a - z), 숫자 (0-9), 특수 문자 (_ $ # * . ,)
  - 세 부분(<container>.<schema>.<object>) 또는 두 부분(<schema>.<object>)으로 지정될 수 있다.
  - Container의 경우 Wildcard(*)가 허용되지 않는다.
  - 쉼표(,)를 통해 여러 객체를 설정할 수 있다.

특정 객체를 추출에서 제외하려면 EXCLUDE 파라미터를 사용해야 한다. 다음은 사용 방법이다:

```
TABLE = "EXCLUDE <container>.<schema>.<object>";
```

TABLE과 ***SEQUENCE*** Config는 여러 개 설정이 가능하다.

객체명 변환 추출

추출 대상 Source객체 명이 Target에서 다를 경우, Extract Mapper 파일에서 추출 되는 객체 명을 변환할 수 있다.

설정 방법은 다음과 같다:

```
MAP from_schema.from_object [ TARGET to_scheam.to_object ] { };
```

객체명 변환 시 주의 사항

객체명 변환 시 DML과 메타데이터 레코드에 변환 된 객체명이 기록된다.

DDL은 변환 추출이 지원되지 않는다.

Mapper 파일 규칙

다음은 Extract Mapper 파일이다.

```
MAP from_schema.from_object [ TARGET to_scheam.to_object ] {
        KEY_COLS (column_name, ...);
        EXCLUDE_DML (INSERT | UPDATE | DELETE, ...);
        EXCLUDE_COLS (column_name, ...);
        CONVERT_TYPE (Oracle datatype, ArkCDC datatype);
        CONVERT_DATA (Oracle datatype, value);
        COLMAP (
                column_name = @STRTONUM (column_name | value),
                column_name = @NUMTOSTR (column_name | value),
                column_name = @BINARY (column_name),
                column_name = @DATE (column_name)
        );
};
```

Mapper 파일 규칙은 다음과 같다:

- 테이블 단위 매핑
    - 테이블 단위 매핑 키워드는 세미콜론 (;)으로 구분한다.
    - 키워드 뒤에는 공백이 필요하다.
        - 지원 됨: KEY_COLS (...);
        - 지원 안됨: KEY_COLS(...);
    - MAP문 내에 명시 된 옵션들은 해당 MAP문에만 유효하다.
    - MAP문 내에서 동일한 옵션을 여러번 지정하여, 옵션이 충돌할 경우 가장 마지막 옵션이 다른 동일한 옵션들을 오버라이드한다.
- 컬럼 단위 매핑
    - COLMAP 내에 기능들은 쉼표 (,)로 구분한다.
    - 키워드 뒤에는 공백이 필요하다.
        - 지원 됨: column_name = @STRTONUM (...),
        - 지원 안됨: column_name = @STRTONUM(...),
    - COLMAP 내에 기능들은 '=' 앞 뒤로 공백이 필요하다.
        - 지원 됨: column_name = @STRTONUM (...),
        - 지원 안됨: column_name=@STRTONUM (...),
    - COLMAP문 내에 명시 된 옵션들은 해당 COLMAP문에만 유효하다.
    - COLMAP 내에 동일한 컬럼이 여러 번 지정되고, 컬럼 매핑끼리 충돌할 경우 가장 마지막 컬럼 매핑이 다른 컬럼 매핑들을 오버라이드한다.

DDL 추출 지정

목적

추출할 DDL을 지정하기 위해 필요하다.

특정 객체에 대한 DDL 또는 특정 DDL 타입을 추출에서 포함 또는 제외할 수 있도록 할 때도 사용할 수 있다.

DDL 추출 기능

DDL 추출/제외 기능은 ***DDL*** Config를 통해 설정이 가능하다. 사용 방법은 다음과 같다:

```
DDL = "INCLUDE | EXCLUDE <schema>.<object>; [OBJTYPE = ALL | <Object Type> ]; [OPERATION = ALL | CREATE | ALTER
| DROP | TRUNCATE | RENAME]"
```

- ***INCLUDE | EXCLUDE:***
  특정 객체에 대한 DDL문을 추출 또는 추출에서 제외하도록 하는 옵션이다.
- ***OBJTYPE:***
  특정 객체 타입에 대한 DDL만 추출 할 수 있도록 하는 옵션이다. 지원되는 객체는 다음과 같다:
    - TABLE
    - INDEX
    - SEQUENCE
    - VIEW
    - MATERIALIZED VIEW
    - CLUSTER
    - PROCEDURE
    - FUNCTION
    - PACKAGE

  ALL을 설정하면 지원되는 모든 객체에 대한 DDL이 추출된다.
- ***OPERATION:***
  특정 DDL문을 추출 할 수 있도록 하는 옵션이다.
- <schema>.<object>:
    - 최대 지원 길이: 255자
    - 지원 키워드: 문자(A-Z | a - z), 숫자 (0-9), 특수 문자 (_ $ # * . ,)

TRUNCATE 추출

기능

TRUNCATE의 경우 ***GET_TRUNCATE*** Config를 통해 별도로 추출 할 수 있다.

GET_TRUNCATE로 인해 추출 된 Truncate 레코드는 OP Type을 Truncate로 기록하여, DDL로 인해 추출 된 Truncate와 구별을 할 수 있다.

```
GET_TRUNCATE="[ yes | no | y | n ]"
```

- ***GET_TRUNCATE :***
  - 기본 값: yes

GET_TRUNCATE Config와 DDL Config 설정이 충돌할 경우 GET_TRUNCATE가 우선 순위이다.

특정 컬럼 제외

기능

Extract *Mapper* 파일에서 ***EXCLUDE_COLS*** Config를 통해 추출 대상 테이블에서 특정 컬럼을 추출에서 제외 시킬 수 있다:

```
MAP from_schema.from_object [ TARGET to_scheam.to_object ] {
               exclude_cols (name),
};
```

- ***EXCLUDE_COLS***로 제외 된 컬럼은 Tracing 파일에서 제외되어야 한다.
- Key 컬럼은 제외하지 않도록 해야 한다. Key 컬럼이 추출에서 제외되어도 Extract는 정상 동작해야 한다.
  하지만 Post에서 적용 시 에러가 발생할 수 있기 때문에 Key 컬럼은 제외하지 않도록 권장해야 한다.

예외 케이스

존재하지 않는 컬럼 제외 시

테이블에 존재하지 않는 컬럼이 ***EXCLUDE_COLS***에 설정할 경우, 관련 된 Warning을 로그에 출력하고 해당 설정을 무시한다.

EXCLUDE_COLS를 여러 개 설정하여 존재하는 컬럼과 존재하지 않는 컬럼의 조합이 사용된다면, 존재하는 컬럼만 추출에서 제외 한다.

```
EXCLUDE_COLS (existing_col, non-existing_col)
```

Definition

EXCLUDE_COLS로 인해 컬럼이 제외되어도 Definition에서는 제외되지 않아야 한다.

모든 컬럼 제외

테이블의 모든 컬럼이 제외될 경우 Tracing 파일에 레코드가 기록되지 않아야 한다. (Skip해야 한다.)

Update 대상 컬럼이 제외 된 경우

Update문의 대상 컬럼(SET절)이 제외되어 레코드 After 데이터가 없을 경우 해당 레코드는 Tracing 파일에서 Skip되어야 한다.

키 컬럼이 제외 된 경우

EXCLUDE_COLS의 경우 키 컬럼은 제외하지 않도록 권장해야 한다.

- Update 또는 Delete문에서 키 컬럼이 제외 될 경우 Extract가 중지되어야 한다. (Before 절이 비어있는 경우)
- Composite Primary key의 경우 일부 컬럼만 추출 제외되고, 일부는 추출에 포함 된다면 정상 추출해야 한다.
  Composite Primary key는 여러 컬럼 값의 조합으로 Row를 구별하기 때문에, 일부 컬럼이 제외 될 경우, 적용 시 올바른 Row가 Update 또는 Delete되지 않을 수 있다.
- Insert에서 키 컬럼이 추출 제외 될 경우 정상 추출 해야 한다.
- Non-key 테이블의 경우 모든 컬럼이 추출에서 제외되지 않은 이상 정상 추출해야 한다.

DML 필터링

기능

Extract가 추출하는 레코드 중, 특정 DML 타입을 제외 시키려면 다음과 같은 Config를 사용해야 한다:

- 추출 대상 객체 전부 적용: ***EXCLUDE_DML_TYPE*** *(Extract Config)*
- 테이블 단위 적용: ***EXCLUDE_DML*** *(Mapper Config)*

EXCLUDE_DML_TYPE

EXCLUDE_DML_TYPE은 Extract Config 파일 Config이다.

이 Config는 모든 추출 대상 객체에 대해 INSERT, UPDATE 또는 DELETE를 추출에서 제외한다. (Truncate와 DDL은 ***GET_TRUNCATE*** 또는 ***DDL*** Config를 통해 필터링 할 수 있다.)

```
EXCLUDE_DML_TYPE="[ NONE | INSERT | UPDATE | DELETE ]"
```

- ***EXCLUDE_DML_TYPE***
  - 기본 값: NONE (제외 DML 타입 없음)
  - 여러 개를 설정 할 수 있으며, 쉼표(,)를 통해 구분한다.

EXCLUDE_DML

EXCLUDE_DML은 Extract의 Mapper 파일에서 사용되며, 테이블 단위로 DML 필터링이 가능하다.

```
EXCLUDE_DML ([ INSERT(I) | UPDATE(U) | DELETE(D) ])
```

- ***EXCLUDE_DML***
  - 여러 개를 설정 할 수 있으며, 쉼표(,)를 통해 구분한다.
  - 약자를 통해서도 지정이 가능하다.

예외 케이스

- DELETE 타입 제외 시, 동일한 Key를 이용한 Insert 또는 Update로 인해 Target에서 중복 키 에러가 발생 할 수 있다.
- 제외 된 DML 타입 레코드는 Tracing 파일에 기록되어선 안된다.
- 모든 DML 타입이 제외 될 경우 로그와 콘솔 화면에 다음 메시지를 출력한다:
  - "Ignoring all DML type record. If not intended, confirm configuration file."

추출 지원 데이터 타입

Redo supported datatypes

Extract Redo Log로부터 추출하는 데이터 타입:

Numeric Data Types

- NUMBER, DECIMAL, BINARY_FLOAT, BINARY_DOUBLE, FLOAT

Character Data Types

- CHAR, VARCHAR, VARCHAR2, NCHAR, NVARCHAR2

Binary Data Types

- RAW

Date and Timestamp Data Types

- DATE, TIMESTAMP, INTERVAL YEAR TO MONTH, DATEINTERVAL DAY TO SECOND

Large Object Data Types

- BASICFILE
  - CLOB (inline), NCLOB (inline), BLOB (inline), BFILE, CFILE,
- SECUREFILE
  - SECUREFILE에 옵션이 없을 경우 Redo. 있을 경우Fetch(옵션: De-duplication, Compression, Encryption)

User Defined or Abstract Types

- ROWID, UROWID

Spatial Data types

- SDO_GEOMETRY

BFILE, CFILE

- 외부 파일은 복제하지 않고, 저장된 경로만 저장

Database fetch supported datatypes

Extract가 Source 데이터베이스로부터 직접 Fetch하는 데이터 타입:

Character Data Types

- LONG

Binary Data Types

- LONG RAW

Date and Timestamp Data Types

- TIMEZONE, TIMESTAMP WITH TIME ZONE, TIMESTAMP WITH LOCAL TIMEZONE

Large Object Data Types

- BASICFILE
  - CLOB (outline), NCLOB (outline), BLOB (outline)
- SECUREFILE
  - 옵션이 있는 SECUREFILE LOB (De-duplication, Compression, Encryption)

추출 지원 기능

Conventional Path와 Direct Path로 로드 된 데이터 둘 다 추출을 지원한다.

프로세스 메모리 제한

목적

Extract 모듈의 메모리 과다 점유를 방지하기 위하여 최대 메모리 사용량 제한이 필요하다.

프로세스 메모리 제한 기능

프로세스 메모리 제한 기능은 두 개의 Config를 사용하여 설정 할 수 있다:

- ***IN_MEMORY_SIZE***: Extract가 Malloc한 메모리 총 사이즈 제한
- ***IN_MEMORY_TX_SIZE***: 트랜잭션 데이터 메모리 사이즈 제한

Extract가 malloc한 사이즈가 위에 설정 된 값을 초과할 경우, 해당 내용은 Cache 파일에 기록된다.

IN_MEMORY_SIZE

Extract가 Malloc한 메모리의 최대 사이즈를 설정하는 Config이다.

```
IN_MEMORY_SIZE = "n[ k | K | m | M | g | G ]"
```

- ***IN_MEMORY_SIZE :***
  - 기본 값: 0
  - 지원 범위: 0 또는 1 이상의 정수
    - 0으로 설정할 경우 메모리 제한이 없다.
    - 1 이상의 정수로 설정할 경우 설정 된 값만큼 메모리를 제한한다.
  - 지원 포맷: 정수에 이은 문자 k, K, m, M, g, G의 조합
  - 위 표기에 따른 정수 변환 값이 536,870,912-17,179,869,184 이하 (512MB ~ 16GB)
  - 이 Config 값은 ***IN_MEMORY_TX_SIZE***보다 크게 설정하도록 해야한다.

IN_MEMORY_TX_SIZE

IN_MEMORY_TX_SIZE Config는 트랜잭션 데이터의 최대 메모리 사이즈를 설정한다.

```
IN_MEMORY_TX_SIZE = "n [ k | K | m | M | g | G ]"
```

- ***IN_MEMORY_TX_SIZE***
  - 기본 값: 1G
  - 지원 범위: 0 또는 1 이상의 정수
    - 0으로 설정할 경우 메모리 제한이 없다.
    - 1 이상의 정수로 설정할 경우 설정 된 값만큼 메모리를 제한한다.
  - 지원 포맷: 정수에 이은 문자 k, K, m, M, g, G의 조합
  - 위 표기에 따른 정수 변환 값이 536,870,912-4,294,965,096 이하 (512MB ~ 4GB)

대용량 트랜잭션 관리

목적

대용량 트랜잭션으로 인한 메모리 과다 점유를 방지하기 위해 대용량 트랜잭션 캐싱 기능을 지원한다.

대용량 트랜잭션 관리 기능

사이즈가 큰 트랜잭션이 발생 할 경우 메모리 사용량을 제한하기 위해 Cache 파일을 활용한다.

초과 된 레코드는 Cache 파일로 옮겨 쓰여지며, 해당 트랜잭션 데이터가 더 들어올 경우 Cache 파일에 이어서 작성된다.

대용량 트랜잭션의 기준은 ***LARGE_TRANSACTION_SIZE*** Config를 통해 지정할 수 있다.

Cache 파일은 ***CACHE_FILE_DEST*** Config를 통해 경로를 관리할 수 있다.

사용 방법은 다음과 같다:

```
LARGE_TRANSACTION_SIZE="n[ k | K | m | M | g | G ]";
CACHE_FILE_DEST="<cache_file>";
```

- ***LARGE_TRANSACTION_SIZE***
  - 기본 값: 128MB
  - 지원 범위: 0 또는 1 이상의 정수
    - 0으로 설정할 경우 Large 트랜잭션이 cache에 기록되지 않고 바로 Tracing 파일에 기록된다.
    - 1이상의 정수로 설정할 경우 Cache 파일에 기록된다.
  - 지원 포맷: 1 이상의 정수에 이은 문자 k, K, m, M, g, G의 조합
  - 위 표기에 따른 정수 변환 값이 134,217,728-4,294,965,096이하 (128MB ~ 4GB)
- ***CACHE_FILE_DEST***
  캐시 파일 경로 지정

Cache 파일

Cache 파일은 대용량 트랜잭션 레코드를 저장하는 파일이다.
대용량 트랜잭션에 대한 정보는 CTS 파일에 저장되며, Extract는 이 파일을 통해 대용량 트랜잭션 레코드 추출 지점을 알 수 있다.

캐시 파일은 다음과 같은 포맷으로 저장된다. 캐시 파일의 구조는 Tracing 파일과 동일하다.

```
{writer alias}_{transactionID}_{tracing sequence#}.trc
```

동작

- Extract 메모리에 있는 Commit되지 않은 transaction의 누적 사이즈가 ***LARGE_TRANSACTION_SIZE*** 설정 값을 초과
- Large 트랜잭션을 cache file에 모두 내려쓴다.
- 해당 트랜잭션이 Commit 되면 Cache 파일의 내용을 모두 Tracing 파일에 옮겨쓴다.
- cache file에는 definition record가 작성되지 않는다.

CTS 파일

CTS 파일은 cache file과 함께 동작하며, cache된 transaction의 상태를 기록하는 파일이다.

Cache 파일을 Tracing 파일에 옮겨 쓰는 과정에서 Extract가 종료 될 경우, 재시작 시 CTS 파일을 통해 이어서 추출할 지점을 알 수 있다.

파일은 다음과 같은 포맷으로 저장된다.

```
{alias}_{transactionID}.cts
```

CTS 파일에는 다음과 같은 정보가 기록된다:

- Cache 파일에 내려쓴 레코드의 추출 지점 (RBA)
- Cache 파일 Sequence
- Cache 파일 내 Position
- 캐시 된 레코드 개수

CTS 파일의 기본 동작:

- Cache된 트랜잭션이 존재하고, Log switch가 발생한 경우 마지막으로 cache된 레코드 정보 및 cache file 정보를 저장한다.

재시작 시 동작:

- 캐시 된 트랜잭션이 있는지 조회한 뒤, 있을 경우 CTS 파일이 존재하는지 확인하고 이를 로드한다.
    - CTS 파일이 있는 경우 : CTS 파일 내용을 참조하여, cache file의 기준지점(pos) 이후 내용을 삭제 후 transaction list에 추가, 재시작 지점부터 다시 캐싱
    - CTS 파일이 없는 경우 : 캐시된 트랜잭션 삭제 후 다시 캐싱

데이터, 데이터 타입 변환 추출

지원되는 데이터 변환 또는 데이터 타입 변환 추출은 다음과 같다:

1. 데이터 변환
    a. LOB 변환:
        i. LOB를 NULL 또는 Char로 변환
        ii. LOB 추출 실패 시 Empty로 변환
    b. 데이터 변환:
        i. 특정 컬럼 또는 특정 데이터 타입의 데이터를 변환
2. 데이터 타입 변환:
    a. 특정 데이터 타입을 다른 데이터 타입으로 변환

LOB 변환

LOB 변환은 Extract가 데이터를 추출할 때 LOB 데이터에 대해 변환하여 추출할지 지정하는 기능이다.

LOB 변환 추출은 Config는 다음과 같다:

- ***CONVERT_LOB_TO_CHAR***
- ***LOB_TO_NULL***
- ***FAILED_FETCH_LOB_TO_EMPTY***

사용 방법은 다음과 같다:

```
CONVERT_LOB_TO_CHAR="<char_value>";
LOB_TO_NULL="[ yes | no | y | n ]";
FAILED_FETCH_LOB_TO_EMPTY="[ yes | no | y | n ]";
```

- ***CONVERT_LOB_TO_CHAR***
    - LOB 데이터를 대체하여 추출 할 문자열을 지정하는 Config
    - 설정될 경우 LOB_TO_NULL과 FAILED_FETCH_LOB_TO_EMPTY Config는 무시된다.
    - 최대 255 byte까지 설정이 가능하다.
- ***LOB_TO_NULL***
    - LOB 데이터를 NULL로 추출할지 여부를 지정하는 Config
    - yes 또는 y로 설정될 경우 FAILED_FETCH_LOB_TO_EMPTY Config는 무시된다.
    - 기본 값은 no이다.
- ***FAILED_FETCH_LOB_TO_EMPTY***
    - LOB 데이터에 대한 Fetch가 실패할 경우 Empty로 추출할지 여부를 지정하는 Config
    - 기본 값은 no이다.

데이터 변환 추출

데이터 변환 추출은, 특정 컬럼 또는 특정 데이터 타입의 데이터를 NULL, 특정 숫자 또는 문자열로 치환하여 추출하는 기능이다.

이 설정은 Extract Mapper 파일에서 수행이 가능하다:

```
MAP from_schema.from_object [ TARGET to_scheam.to_object ] {
        convert_data (oracle_datatype, value)
};
```

- oracle_datatype:
  Oracle 고유 데이터 타입만 사용 가능하다.
  지원되는 타입은 다음과 같다:
    - CHAR
    - NCHAR
    - VARCHAR2
    - NVARCHAR2
    - NUMBER
    - BINARY_FLOAT
    - BINARY_DOUBLE
    - LONG
    - DATE
    - CLOB
    - NCLOB
    - BLOB
- value:
  치환할 데이터이다. 치환 가능한 데이터는 다음과 같다:
    - NULL
    - 숫자 (e.g. 1234)
    - 문자 (e.g. 'abcd')
    - EMPTY() (oracle_datatype이 LOB가 아닐 경우, NULL로 변환)
    - 공백 (작은 따옴표 사이에 위치해야 한다. e.g. ' ')

  허용되지 않는 데이터는 다음과 같다:
    - 공백 없는 작은 따옴표 (e.g. '')

데이터 변환 주의 사항

- 지정한 데이터 타입에 올바른 데이터를 입력해야 한다.
  올바르지 않는 형식의 데이터를 지정하면 에러가 발생한다.

데이터 타입 변환 추출

데이터 타입 변환 추출은 특정 데이터 타입을 다른 데이터 타입으로 치환하여 추출하는 기능이다.

이 설정은 Extract Mapper 파일에서 수행이 가능하다:

```
MAP from_schema.from_object [ TARGET to_scheam.to_object ] {
                convert_type (oracle_datatype, convert_type)
};
```

- oracle_datatype:
  Oracle 고유 데이터 타입만 사용 가능하다.
  지원되는 타입은 다음과 같다:
    - CHAR
    - NCHAR
    - VARCHAR2
    - NVARCHAR2
    - NUMBER
    - BINARY_FLOAT
    - BINARY_DOUBLE
    - LONG
    - DATE
- convert_type:
  CDC 타입 사용
  지원되는 타입은 다음과 같다:

    - NUMERIC
    - STRING
    - DATE

지원되는 데이터 타입 변환은 다음과 같다:

- CHAR, NCHAR, VARCHAR2, NVARCHAR2 → NUMERIC
- NUMBER, BINARY_FLOAT, BINARY_DOUBLE, LONG, DATE → STRING
- CHAR, NCHAR, VARCHAR2, NVARCHAR2 → DATE

데이터타입 변환 주의 사항

- 지원되는 데이터 타입 변환 이외의 데이터 타입이 명시 될 경우 에러가 발생한다.
- Definition에 변경된 데이터타입이 반영되지 않는다.
- DDL은 변환 추출이 지원되지 않는다.
- 치환하려는 datatype에 유효한 데이터만 추출한다.
  ex) CONVERT_TYPE(varchar2, numeric); / 실데이터: '123ABC' => 추출데이터: 123

컬럼 단위 데이터 변환

다음은 컬럼 단위로 특정 컬럼 또는 특정 값을 변환하여 추출하는 기능이다.

```
MAP from_schema.from_object [ TARGET to_scheam.to_object ] {
        COLMAP (
                column_name = @STRTONUM (column_name | value),
                column_name = @NUMTOSTR (column_name | value),
                column_name = @BINARY (column_name),
                column_name = @DATE (column_name)
        );
};
```

- @STRTONUM문자 데이터를 숫자로 변환한다.
- @NUMTOSTR
  숫자 데이터를 문자로 변환한다.
- @BINARY
  특정 컬럼 데이터를 Binary 형식으로 변환한다.
- @DATE
  String 타입 컬럼 데이터를 DATE로 변환한다.
  String 타입 컬럼만 사용 가능하다. 지원되는 포맷은 다음과 같다:
    - YYYY-DD-MM 24HH:mm:SS
    - YYYY-DD-MM
      (날짜만 지정될 경우 시간은 0으로 추출된다)

TDE/TSE 추출

목적

암호화 데이터를 추출하기 위해 필요하다..

TDE 추출 기능

TDE는 암호화 컬럼의 데이터를 그대로 추출하고 Tracing 파일에 작성할 때 복호화를 수행한다.

TSE는 Vector의 모든 컬럼 데이터들을 암호화하는 기능이다. ObjectID, 컬럼 정보, 트랜잭션 ID 등 추출에 필요한 데이터들이 모두 암호화 되어있어 레코드에서 데이터를 추출하기 전에 복호화를 수행한다.

TDE 데이터는 컬럼 단위 암호화 및 테이블스페이스 단위 암호화 데이터 추출이 지원된다.

Oracle Wallet Key 파일 저장 경로는 ***ARK_TDE_FILENAME*** Config를 통해 설정한다.

```
ARK_TDE_FILENAME = "<wallet_key_file>"
```

- ***ARK_TDE_FILENAME :***
  최대 255자까지 가능하다.

아카이브 로그 추출

목적

- redolog에서 추출시 잦은 overwirte가 발생
- 작성중인 redo 접근으로 인한 성능 저하를 염려 하는경우
- RAC환경의 한쪽 노드에서 여러 node의 모든 redo에 접근이 어려운 경우

위와 경우 redo에서 추출이 불가능하거나 운영에 부담을 느낄 수 있으므로, 아카이브만 있더라도 데이터를 추출 할 수 있는 기능이 필요

아카이브 로그 추출 기능

redolog가 아닌 아카이브 로그에서 데이터를 추출하도록 하기 위해 ***ARCHIVE_DEST***, ***ARCHIVE_ONLY*** Config를 사용할 수 있다.

사용 방법은 다음과 같다:

```
ARCHIVE_ONLY="[ yes | no | y | n ]";
ARCHIVE_DEST="<archive_log_dest>";
```

- ***ARCHIVE_ONLY***
  - redo log를 사용하지 않고 archive 로그에서만 데이터를 추출하도록 설정하는 Config이다.
- ***ARCHIVE_DEST***
  - archive log 경로를 지정하는 Config
  - 최대 255자까지 가능

키 컬럼 매핑

목적

Non-key 테이블 이더라도 데이터 검색이나 관리를 위해 index를 설정하는 경우가 있다. 이 경우 적용 성능 향상을 위해 특정 컬럼을 키 컬럼으로 사용할 수 있도록 하기 위해 필요하다.

키 컬럼 매핑

키 컬럼 매핑 기능은 ***COLMAP***의 ***key_cols*** config를 통해 설정이 가능하다.

```
MAP from_schema.from_object [ TARGET to_scheam.to_object ]  {
                key_cols (column_name , .... n );
};
```

- ***key_cols*** Config 값은 Source 테이블이 Non-key 테이블일 때만 유효하다.
  Key가 존재하는 테이블에 설정되면, 로그에 WARNING이 출력되며, 기존에 있는 Key로 추출을 이어서 진행한다.
- ***key_cols***로 설정 된 컬럼은 Tracing File Definition에서 Key Column으로 기록이 된다.
  하지만, 레코드에 대한 Before 값들은 (Non-key 테이블이므로) 모든 컬럼에 대해 기록된다.
  (key_cols 값은 Post가 (향후 목적으로) 필요할 경우 사용할 수 있으나, 꼭 사용해야 하는 건 아니다)

예외 케이스

존재하지 않는 컬럼 key_cols 설정

테이블에 존재하지 않는 컬럼을 ***key_cols***에 설정할 경우, 관련 된 Warning을 로그에 출력하고 해당 설정을 무시한다.

즉, ***key_cols*** 설정을 무시하고, 기존 키 컬럼 채택 방식으로 추출을 이어서 수행한다. (PK면 PK 사용, Non-key면 All 컬럼 사용)

다음과 같이 ***key_cols***를 여러 개 설정하여 존재하는 컬럼과 존재하지 않는 컬럼의 조합이 사용된다면, 존재하는 컬럼만 키 컬럼으로 사용하도록 한다.

```
key_cols (existing_col, non-existing_col);
```

이때 존재하지 않는 컬럼은 무시한다는 로그를 출력해야 한다.

여러 컬럼이 지정될 때, 컬럼 지정 순서는 동작에 영향을 끼쳐선 안된다.

중복 설정

key_cols Config가 동일한 객체에 대해 여러 번 설정되면, 맨 마지막 설정이 적용된다.

예를 들어 다음과 같이 동일한 테이블에 대해 서로 다른 ***key_cols***가 설정되면, 마지막 설정이 적용된다.

```
MAP from_schema.from_object [ TARGET to_scheam.to_object ]  {
        key_cols (a, c);
};


MAP from_schema.from_object [ TARGET to_scheam.to_object ]  {
        key_cols (a);
};
```

테이블 정의 데이터 추출

테이블 정의 데이터 추출 기능

테이블의 생성/정의 값을 수집하여 Tracing 파일에 저장하는 기능이다.

Tracing 파일에서 Definition 레코드로 기록된다.

DDL History 관리

목적

DDL이 발생 할 경우 테이블의 메타정보가 변경될 수 있다.

레코드가 수행된 시점 기준 메타정보가 필요할 경우 DDL History 테이블을 통해 이를 알 수 있다.

DDL History 관리 기능

DDL History는 두 개의 테이블로 관리된다.

Marker Table

Marker table에는 다음과 같은 정보가 저장된다.

- DDL 발생 시간
- DDL Event (CREATE, ALTER, DROP)
- DDL Object Type (TABLE, USER, SEQUENCE...)
- DDL Object Owner
- DDL Object name
- DB version
- DDL statement
- NLS session parameter

BEFORE DDL Trigger는 아직 DDL이 실행되지 않아 DDL의 SCN이 없습니다.

그래서 Marker Table에 위 데이터를 저장하여 생성된 SCN을 History Table의 DDL SCN 값으로 활용합니다.ㅊ

History Table

History 테이블은 **"ALTER" , "DROP"** DDL 이벤트가 발생 했을 때 다음과 같은 정보를 테이블에 저장한다:

- DDL SCN
- Object ID, Data Object ID
- Object Name
- Object Owner
- Object Type
- DDL Type
- Column 메타 정보 (JSON)

Persist 파일 관리

목적

오랫동안 Commit되지 않은 트랜잭션이 존재할 때, 재시작 시, redo/archive의 보관 주기 문제와 너무 오래 전 시점부터 추출을 시작해야 하는 문제를 해소하기 위해 Persist 파일을 활용한다.

Persist 파일 관리 기능

특정 조건 발생한 시, Commit 또는 Rollback이 수행되지 않은 트랜잭션의 데이터를 Persist 파일에 저장한다.
Persist 파일이 생성 되는 특정 조건은 ***PERSISTENCE_MODE*** Config를 통해 설정 할 수 있다.

Persist 파일은 아래와 같은 포맷으로 ***CACHE_FILE_DEST*** 경로에 저장된다. Persist 파일의 구조는 Tracing 파일과 동일하다.

```
{alias}_persist_{thread#}_{log sequence#}_{tracing sequence#}.trc
```

동작

기본 동작 (Log Switch 기준):

- Log Switch가 일어난 시점에, 현재 가지고 있는 Transaction list를 돌면서 Persist 파일에 모두 내려 쓴다.
- 가장 최근의 Persist 파일이 최초 시작 이후 아직 Commit되지 않은 transaction들을 모두 가지고 있어야 한다.
- Current redo sequence 보다 2 미만인 persist 파일은 삭제한다.
- transaction list가 비어있는 경우에는 persist 파일 작성하지 않는다.
- 이미 Caching 된 Transaction인 경우에는 persist 파일 작성하지 않고, cts 파일만 작성한다.
- Persist 파일은 가장 최근의 Redo Log에 포함되는 Long Transaction 데이터만을 저장하며, Thread 당 최대 2개까지 유지된다.

재시작 시 동작:

- persist transaction이 있는지 검색한 뒤, persist transaction을 transaction list에 추가한다.
- RAC 환경에서 노드별로 persist 파일이 존재하는 경우, checkpoint의 persist timestamp를 비교하여 가장 최근의 persist 파일 하나만 로드한다.

Config

Persist File을 작성하는 기준은 ***PERSISTENCE_MODE*** Config를 통해 설정할 수 있다:

```
PERSISTENCE_MODE = "[ LOGSWITCH | INTERVAL | OFF ]"
PERSISTENCE_INTERVAL = "[ { n } H ]"
```

- ***PERSISTENCE_MODE***
    - 기본 값: LOGSWITCH
    - LOGSWITCH: 로그 스위치 발생 기준으로 Persist 파일을 작성한다.
    - INTERVAL: 특정 시간 간격으로 Persist 파일을 작성한다.
    - OFF: Persist 파일 작성을 비활성화 한다.
- ***PERSISTENCE_INTERVAL***
    - PERSISTENCE_MODE 값이 INTERVAL 일 때만 유효하다.
    - 0 설정 시 사용 안함을 의미한다.
    - 기본 값: 6
    - 최솟 값: 0
    - 최댓 값: 2400
    - 지원 포맷: 정수에 이은 H 문자의 조합

Extract 체크포인트

목적

Extract가 재시작 했을 때 마지막으로 추출한 지점부터 이어서 추출 할 수 있도록 추출 및 Tracing 파일 기록 시점을 기록한다.

또, 프로세스의 상태 및 진행 상황을 파악 할 수 있다.

체크포인트 기능

Extract 체크포인트는 크게 Read와 Write 체크포인트로 나뉜다.

Read 체크포인트는 Extract가 Redo Log에서 읽은 위치이며, Write는 Tracing 파일에서 기록하는 위치이다.

Read Checkpoint

Read Checkpoint는 다음과 같다:

- Startup Checkpoint:
  Extract가 최초 실행 된 위치. Write가 발생하기 전에 Extract가 재시작 될 경우 추출의 시작점이 되기도 한다.
- Persistent Checkpoint:
  Extract가 재시작 되기까지 길어 지거나 Long 트랜잭션으로 인해 Persistent Caching 기능이 실행 될 경우, 마지막 저장 위치.
- Recovery Checkpoint:
  메모리에 로드 되어있는 Commit되지 않은 트랜잭션 중 가장 오래 된 트랜잭션.
  Extract 재기동 시, 이전에 추출하지 못한 트랜잭션의 시작 지점인 Recovery로 이동하여 추출을 수행한다.
- Current Checkpoint:
  Extract가 Redo Log에서 가장 최근에 읽은 리두 블럭.

Write Checkpoint

Write Checkpoint는 Extract가 Tracing 파일에서 성공적으로 기록한 것을 확인한 위치이다.

작성 및 갱신 시점

작성

Checkpoint를 작성하는 시점은 다음과 같다:

1. 모듈 start (header, startup & current - thread, scn)
2. redo open 직후 (startup & current - rba, log path)

갱신

갱신 시점은 다음과 같다:

1. 10초마다 갱신. (current)
2. tracing file write (write, current)
3. log switch 발생 (persist & current)
4. 모듈 stop (current)

재시작 후 시작 지점 설정

Extract가 재시작했을 때 Checkpoint 기록 여부 및 시점에 따라 추출 시작 시점이 다르게 설정된다.

이때, 이미 추출했던 데이터들은 재추출하지 않도록 스킵해야 한다. (Current Checkpoint 참조)
Large Transaction의 경우, 이미 캐싱 된 데이터들을 다시 캐싱하지 않도록 해야 한다. (Write Checkpoint 참조)

다음은 Checkpoint 상태에 따른 Extract 시작 지점이다.

Checkpoint 상태	시작 지점
persist - recovery 모두 기록	persist와 recovery를 비교해서 가장 최근 지점부터 시작.
persist가 최근	persist 파일을 로드 한 뒤, persist 지점부터 시작.
recovery가 최근	recovery 지점부터 시작.
동일 정보	persist 지점부터 시작.
persist만 기록	persist 파일을 로드 한 뒤, persist 지점부터 시작.
recovery만 기록	recovery 지점부터 시작.
persist - recovery 모두 없음	startup 지점부터 시작.

예외 케이스

- 체크포인트 파일이 없을 경우:
    - 파일을 새로 생성하여, 현재 확인할 수 있는 정보로 체크포인트를 작성한다.
- 체크포인트 파일 권한이 없을 경우:
    - Extract 프로세스 시작 시:
        - 체크포인트 파일에 권한이 없다는 메시지를 로그에 남긴다.
        - Extract 프로세스가 시작되지 않아야 한다.
    - Extract 프로세스 실행 중 발생 시:
        - 체크포인트 갱신 시점에 권한이 없을 경우, 체크포인트 파일에 권한이 없다는 메시지를 로그에 남긴다.
        - 프로세스를 중단 시킨다.

Extract Lag

Extract 프로세스 Lag는 레코드가 발생한 시간과 Extract가 레코드를 Tracing 파일에 기록 한 시간 간에 간격이다.

Extract Lag는 Extract 체크포인트 파일에 기록되며, 체크포인트 파일이 갱신 될 때마다 같이 갱신된다.

Extract Lag는 다음 명령어로 조회할 수 있다:

```
admgr> getlag
admgr> getlag all
admgr> getlag extract
admgr> getlag extract all
```

Checkpoint Thread

Checkpoint 용 별도 스레드가 필요하며, 해당 스레드는 Monitorin API와 통신한다.

Checkpoint 갱신 간격 설정

Checkpoint 파일의 갱신 간격은 ***CHECKPOINT_SECONDS*** Config로 설정할 수 있다.

```
CHECKPOINT_SECONDS = "n"
```

- ***CHECKPOINT_SECONDS***
    - 기본 값: 10
    - 최솟 값: 2
    - 최댓 값: 30

시점 추출

목적

특정 시점부터 Extract가 시작될 수 있도록 제어하기 위해 필요한 기능이다.

시점 추출 기능

Extract가 특정 SCN 또는 Timestamp에서 시작하도록 설정하는 기능이다.

이 기능은 다음과 같이 설정할 수 있다:

```
--start-dsn
--start-time
```

초기 복제

목적

변경 데이터가 아닌 기존에 있는 데이터를 복제하여, Source와 Target 데이터베이스를 동기화 시키기 위해 필요한 기능이다.

초기 복제 기능

초기 복제 기능은 다음과 같은 Config로 설정할 수 있다

```
INITIAL_COPY_THREADS = "n"
INITIAL_COPY_SPLIT_TRANSACTION = "n"
INITIAL_COPY_SPLIT_SIZE = "n"
INITIAL_COPY_PARALLELISM_DEGREE = "n"
INITIAL_COPY_SCN_ASCENDING = "[ on | off ]"
```

- ***INITIAL_COPY_THREADS***
    - 초기 복제 시 사용할 Capture Thread 개수
    - 기본 값: 4
    - 최솟 값: 1
    - 최댓 값: 10
- ***INITIAL_COPY_SPLIT_TRANSACTION***
    - 초기 복제 트랜잭션의 최대 레코드 개수
    - 기본 값: 100,000
    - 최솟 값: 10,000
    - 최댓 값: 1,000,000
- ***INITIAL_COPY_SPLIT_SIZE***
    - 초기 복제 트랜잭션의 최대 크기
    - 기본 값: LARGE_TRANSACTION_SIZE 값
    - 최솟 값: 1
    - 지원 포맷: 정수에 이은문자 k, K, m, M, g, G의 조합
    - 지원 범위: 정수 변환 값 134,217,728-4,294,965,096이하 (128MB ~ 4GB)
- ***INITIAL_COPY_PARALLELISM_DEGREE***
    - 초기 복제 시 쿼리 수행 병렬도
    - 기본 값: 4
    - 최솟 값: 0
    - 최댓 값: 128
- ***INITIAL_COPY_SCN_ASCENDING***
    - 초기 복제 쿼리 scn_ascending 힌트 사용 여부
    - 기본 값: on
    - --start-dsn 옵션 사용시, 해당 config 무시

초기 복제 수행

Extract가 Redo log에서 변경 데이터를 추출하지 않고 Source 데이터베이스에서 테이블의 구조와 데이터를 복제하는 기능이다.

초기 복제 수행 방법은 다음과 같다:

```
--copy
--copy-data
--copy-table
--copy-all
--script
--copy-sample
```

- --copy : 초기 복제 필수 옵션
- --copy-data : 데이터만 추출 (기본 값)
- --copy-table : 테이블 구조만 추출
- --copy-all : 데이터와 테이블 구조 둘 다 추출
- --script : ddl script로 생성
- --copy-sample : 입력한 값만큼 sample 추출 (default 1000)

통계

기능

Extract 추출 데이터를 모니터링하기 위해 통계를 기록한다.

통계는 각 모듈별로 기록되며, 통계 파일명 포맷은 다음과 같다:

```
extract_[alias]_[timestamp].stat
```

다음은 통계 파일의 예시이다:

```
-- Before (~v1.5)
START SCN       <scn>
UNTIL SCN       <scn>
DML
    INSERT      n
    UPDATE      n
    DELETE      n
DDL
    CREATE      n
    ALTER       n
    DROP        n
    TRUNCATE    n
GRANT/REVOKE    n
DEFINITION      n
OUTLINE LOB     n


-- After (v.2.0)
Start of Statistics at [timestamp]
Output to [extract_trace_path]:

DDL extraction statistics:

*** Total statistics since [timestamp] ***
    Total operations                    n
    Total creates                       n
    Total alters                        n
    Total drops                         n
    Total others                        n

Extracting from [schema].[table] to [schema].[table]:

*** Total statistics since [timestamp] ***
    Total inserts                       n
```

```
    Total updates                          n
    Total deletes                          n
    Total truncates                        n
    Total operations                       n

*** Daily statistics since [timestamp] ***
    Total inserts                          n
    Total updates                          n
    Total deletes                          n
    Total truncates                        n
    Total operations                       n

*** Hourly statistics since [timestamp] ***
    Total inserts                          n
    Total updates                          n
    Total deletes                          n
    Total truncates                        n
    Total operations                       n
```

DML 통계 정보는 테이블 단위로 기록되어야 한다.

트랜잭션 필터링

트랜잭션 필터링은 특정 태그 또는 트랜잭션 명을 가지는 트랜잭션을 추출에서 제외하는 기능이다.

양방향 복제 환경에서 사용 할 경우 레코드 사이클링을 방지한다.

레코드 사이클링은, Post가 적용한 레코드를 Extract가 추출하여 동일한 레코드가 반복되어 복제되는 현상이다.

EXCLUDE_TAG

Post가 ***SET_TAG*** Config를 통해, 적용한 레코드에 대한 특정 태그를 남기면, Extract에서 해당 태그가 걸린 트랜잭션을 전부 추출에서 제외하도록 한다.

Post는 ***SET_TAG*** Config 사용 시 dbms_streams.set_tag를 사용하여 레코드에 태그를 남긴다. 태그는 다음과 같이 기록된다.

```
REPL MARKER:   <--TAG
Dump of memory from 0x00007FA6CA46904C to 0x00007FA6CA469072
7FA6CA469040                            00230006              [..#.]
7FA6CA469050 00010000 7365741D 676E6974 6D62645F  [.....testing_dbm]
7FA6CA469060 74735F73 6D616572 61705F73 67616B63  [s_streams_packag]
7FA6CA469070 00007365                             [es..]
```

Oracle 태그는 트랜잭션의 첫 번째 레코드에만 기록된다.

예를 들어 다음과 같이 여러 Insert가 수행된 트랜잭션이 발생 할 경우, 첫 번째 레코드에만 Tag가 기록된다.

```
insert into test values (2, 'From Target ext02', sysdate);
insert into test values (3, 'From Target ext02', sysdate);
insert into test values (4, 'From Target ext02', sysdate);
commit;
```

태그 추출 제외 시 Redo Log에서 해당 태그가 포함된 **트랜잭션 전체를** 추출에서 제외해야 한다.

Insert 1

```
REDO RECORD - Thread:1 RBA: 0x000101.00000013.0010 LEN: 0x0258 VLD: 0x05 CON_UID: 0
SCN: 0x00000000022f4f86 SUBSCN:  1 01/19/2021 17:23:45
(LWN RBA: 0x000101.00000013.0010 LEN: 0x00000002 NST: 0x0001 SCN: 0x00000000022f4f86)
CHANGE #1 CON_ID:0 TYP:0 CLS:35 AFN:4 DBA:0x01000110 OBJ:4294967295 SCN:0x00000000022f4e02 SEQ:1 OP:5.2 ENC:
0 RBL:0 FLG:0x0000
ktudh redo: slt: 0x0005 sqn: 0x00007d1e flg: 0x0012 siz: 136 fbi: 0
```

```
            uba: 0x0102e4ea.02f8.05    pxid:  0x0000.000.00000000
CHANGE #2 CON_ID:0 TYP:0 CLS:36 AFN:4 DBA:0x0102e4ea OBJ:4294967295 SCN:0x00000000022f4e01 SEQ:2 OP:5.1 ENC:
0 RBL:0 FLG:0x0000
ktudb redo: siz: 136 spc: 7426 flg: 0x0012 seq: 0x02f8 rec: 0x05
            xid:  0x000a.005.00007d1e
ktubl redo: slt: 5 wrp: 1 flg: 0x0c08 prev dba:  0x00000000 rci: 0 opc: 11.1 [objn: 80268 objd: 80268 tsn: 4]
[Undo type  ] Regular undo  [User undo done   ]  No  [Last buffer split]  No
[Temp object]           No  [Tablespace Undo  ]  No  [User only        ]  No
Begin trans
 prev ctl uba: 0x0102e4ea.02f8.02 prev ctl max cmt scn:  0x00000000022f2972
 prev tx cmt scn:  0x00000000022f2989
 txn start scn:  0xffffffffffffffff  logon user: 107
 prev brb:  0x0102e4dd  prev bcl:  0x00000000
BuExt idx: 0 flg2: 0
KDO undo record:
KTB Redo
op: 0x03  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: Z
KDO Op code: DRP row dependencies Disabled
  xtype: XA flags: 0x00000000  bdba: 0x01c0025d  hdba: 0x01c0025a
itli: 1  ispac: 0  maxfr: 4858
tabn: 0 slot: 0(0x0)
CHANGE #3 CON_ID:0 TYP:0 CLS:1 AFN:7 DBA:0x01c0025d OBJ:80268 SCN:0x00000000022efe38 SEQ:2 OP:11.2 ENC:0 RBL:
0 FLG:0x0000
KTB Redo
op: 0x01  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: F  xid:  0x000a.005.00007d1e    uba: 0x0102e4ea.02f8.05
KDO Op code: IRP row dependencies Disabled
  xtype: XA flags: 0x00000000  bdba: 0x01c0025d  hdba: 0x01c0025a
itli: 1  ispac: 0  maxfr: 4858
tabn: 0 slot: 0(0x0) size/delt: 32
fb: --H-FL-- lb: 0x1  cc: 3
null: ---
col  0: [ 2]  c1 02
col  1: [17]  46 72 6f 6d 20 53 6f 75 72 63 65 20 65 78 74 30 31
col  2: [ 7]  78 79 01 13 12 18 2e
CHANGE #4 MEDIA RECOVERY MARKER CON_ID:0 SCN:0x0000000000000000 SEQ:0 OP:5.20 ENC:0 FLG:0x0000
session number   = 273
serial  number   = 23303
transaction name =
version 318767104
audit sessionid 2930886
Client Id =
login   username = OGGTEST
REPL MARKER:   <--TAG
Dump of memory from 0x00007FA6CA46904C to 0x00007FA6CA469072
7FA6CA469040                            00230006             [..#.]
7FA6CA469050 00010000 7365741D 676E6974 6D62645F  [.....testing_dbm]
7FA6CA469060 74735F73 6D616572 61705F73 67616B63  [s_streams_packag]
7FA6CA469070 00007365                             [es..]
```

Insert 2 ~ 4

```
REDO RECORD - Thread:1 RBA: 0x000104.00000007.0010 LEN: 0x0258 VLD: 0x05 CON_UID: 0
SCN: 0x00000000022f52a7 SUBSCN:  1 01/19/2021 17:29:14
(LWN RBA: 0x000104.00000007.0010 LEN: 0x00000005 NST: 0x0001 SCN: 0x00000000022f52a7)
CHANGE #1 CON_ID:0 TYP:0 CLS:17 AFN:4 DBA:0x01000080 OBJ:4294967295 SCN:0x00000000022f51ab SEQ:1 OP:5.2 ENC:
0 RBL:0 FLG:0x0000
ktudh redo: slt: 0x0011 sqn: 0x00007d63 flg: 0x0012 siz: 136 fbi: 0
            uba: 0x01000548.03b2.11    pxid:  0x0000.000.00000000
CHANGE #2 CON_ID:0 TYP:0 CLS:18 AFN:4 DBA:0x01000548 OBJ:4294967295 SCN:0x00000000022f51aa SEQ:1 OP:5.1 ENC:
0 RBL:0 FLG:0x0000
ktudb redo: siz: 136 spc: 4622 flg: 0x0012 seq: 0x03b2 rec: 0x11
            xid:  0x0001.011.00007d63
ktubl redo: slt: 17 wrp: 1 flg: 0x0c08 prev dba:  0x00000000 rci: 0 opc: 11.1 [objn: 80268 objd: 80268 tsn:
4]
[Undo type  ] Regular undo  [User undo done   ]  No  [Last buffer split]  No
[Temp object]           No  [Tablespace Undo  ]  No  [User only        ]  No
```

```
Begin trans
 prev ctl uba: 0x01000548.03b2.10 prev ctl max cmt scn:  0x00000000022f296e
 prev tx cmt scn:  0x00000000022f2976
 txn start scn:  0xffffffffffffffff  logon user: 107
 prev brb:  0x01000545  prev bcl:  0x00000000
BuExt idx: 0 flg2: 0
KDO undo record:
KTB Redo
op: 0x03  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: Z
KDO Op code: DRP row dependencies Disabled
  xtype: XA flags: 0x00000000  bdba: 0x01c0025d  hdba: 0x01c0025a
itli: 2  ispac: 0  maxfr: 4858
tabn: 0 slot: 1(0x1)
CHANGE #3 CON_ID:0 TYP:2 CLS:1 AFN:7 DBA:0x01c0025d OBJ:80268 SCN:0x00000000022f4f88 SEQ:1 OP:11.2 ENC:0 RBL:
0 FLG:0x0000
KTB Redo
op: 0x01  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: F  xid:  0x0001.011.00007d63    uba: 0x01000548.03b2.11
KDO Op code: IRP row dependencies Disabled
  xtype: XA flags: 0x00000000  bdba: 0x01c0025d  hdba: 0x01c0025a
itli: 2  ispac: 0  maxfr: 4858
tabn: 0 slot: 1(0x1) size/delt: 32
fb: --H-FL-- lb: 0x2  cc: 3
null: ---
col  0: [ 2]  c1 03
col  1: [17]  46 72 6f 6d 20 54 61 72 67 65 74 20 65 78 74 30 32
col  2: [ 7]  78 79 01 13 12 1e 0f
CHANGE #4 MEDIA RECOVERY MARKER CON_ID:0 SCN:0x0000000000000000 SEQ:0 OP:5.20 ENC:0 FLG:0x0000
session number   = 273
serial  number   = 23303
transaction name =
version 318767104
audit sessionid 2930886
Client Id =
login   username = OGGTEST
REPL MARKER:  <--
Dump of memory from 0x00007FE54D68D04C to 0x00007FE54D68D072
7FE54D68D040                             00230006              [..#.]
7FE54D68D050 00010000 7365741D 676E6974 6D62645F  [.....testing_dbm]
7FE54D68D060 74735F73 6D616572 61705F73 67616B63  [s_streams_packag]
7FE54D68D070 00007365                             [es..]

REDO RECORD - Thread:1 RBA: 0x000104.00000008.01a8 LEN: 0x0138 VLD: 0x01 CON_UID: 0
SCN: 0x00000000022f52a7 SUBSCN:  3 01/19/2021 17:29:14
CHANGE #1 CON_ID:0 TYP:0 CLS:18 AFN:4 DBA:0x01000548 OBJ:4294967295 SCN:0x00000000022f52a7 SEQ:2 OP:5.1 ENC:
0 RBL:0 FLG:0x0000
ktudb redo: siz: 92 spc: 4390 flg: 0x0022 seq: 0x03b2 rec: 0x13
            xid:  0x0001.011.00007d63
ktubu redo: slt: 17 wrp: 32099 flg: 0x0000 prev dba:  0x00000000 rci: 18 opc: 11.1 [objn: 80268 objd: 80268
tsn: 4]
[Undo type  ] Regular undo  [User undo done   ]  No  [Last buffer split]  No
[Temp object]           No  [Tablespace Undo  ]  No  [User only        ]  No
KDO undo record:
KTB Redo
op: 0x02  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: C  uba: 0x01000548.03b2.11
KDO Op code: DRP row dependencies Disabled
  xtype: XA flags: 0x00000000  bdba: 0x01c0025d  hdba: 0x01c0025a
itli: 2  ispac: 0  maxfr: 4858
tabn: 0 slot: 2(0x2)
CHANGE #2 CON_ID:0 TYP:0 CLS:1 AFN:7 DBA:0x01c0025d OBJ:80268 SCN:0x00000000022f52a7 SEQ:1 OP:11.2 ENC:0 RBL:
0 FLG:0x0000
KTB Redo
op: 0x02  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: C  uba: 0x01000548.03b2.13
KDO Op code: IRP row dependencies Disabled
```

```
  xtype: XA flags: 0x00000000  bdba: 0x01c0025d  hdba: 0x01c0025a
itli: 2  ispac: 0  maxfr: 4858
tabn: 0 slot: 2(0x2) size/delt: 32
fb: --H-FL-- lb: 0x2  cc: 3
null: ---
col  0: [ 2]  c1 04
col  1: [17]  46 72 6f 6d 20 54 61 72 67 65 74 20 65 78 74 30 32
col  2: [ 7]  78 79 01 13 12 1e 0f


REDO RECORD - Thread:1 RBA: 0x000104.00000009.01d8 LEN: 0x0138 VLD: 0x01 CON_UID: 0
SCN: 0x00000000022f52a7 SUBSCN:  5 01/19/2021 17:29:14
CHANGE #1 CON_ID:0 TYP:0 CLS:18 AFN:4 DBA:0x01000548 OBJ:4294967295 SCN:0x00000000022f52a7 SEQ:4 OP:5.1 ENC:
0 RBL:0 FLG:0x0000
ktudb redo: siz: 92 spc: 4218 flg: 0x0022 seq: 0x03b2 rec: 0x15
            xid:  0x0001.011.00007d63
ktubu redo: slt: 17 wrp: 32099 flg: 0x0000 prev dba:  0x00000000 rci: 20 opc: 11.1 [objn: 80268 objd: 80268
tsn: 4]
[Undo type  ] Regular undo  [User undo done   ]  No  [Last buffer split]  No
[Temp object]           No  [Tablespace Undo  ]  No  [User only        ]  No
KDO undo record:
KTB Redo
op: 0x02  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: C  uba: 0x01000548.03b2.13
KDO Op code: DRP row dependencies Disabled
  xtype: XA flags: 0x00000000  bdba: 0x01c0025d  hdba: 0x01c0025a
itli: 2  ispac: 0  maxfr: 4858
tabn: 0 slot: 3(0x3)
CHANGE #2 CON_ID:0 TYP:0 CLS:1 AFN:7 DBA:0x01c0025d OBJ:80268 SCN:0x00000000022f52a7 SEQ:2 OP:11.2 ENC:0 RBL:
0 FLG:0x0000
KTB Redo
op: 0x02  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: C  uba: 0x01000548.03b2.15
KDO Op code: IRP row dependencies Disabled
  xtype: XA flags: 0x00000000  bdba: 0x01c0025d  hdba: 0x01c0025a
itli: 2  ispac: 0  maxfr: 4858
tabn: 0 slot: 3(0x3) size/delt: 32
fb: --H-FL-- lb: 0x2  cc: 3
null: ---
col  0: [ 2]  c1 05
col  1: [17]  46 72 6f 6d 20 54 61 72 67 65 74 20 65 78 74 30 32
col  2: [ 7]  78 79 01 13 12 1e 0f

REDO RECORD - Thread:1 RBA: 0x000104.0000000b.0018 LEN: 0x00b8 VLD: 0x01 CON_UID: 0
SCN: 0x00000000022f52a8 SUBSCN:  1 01/19/2021 17:29:14
CHANGE #1 CON_ID:0 TYP:0 CLS:17 AFN:4 DBA:0x01000080 OBJ:4294967295 SCN:0x00000000022f52a7 SEQ:1 OP:5.4 ENC:
0 RBL:0 FLG:0x0000
ktucm redo: slt: 0x0011 sqn: 0x00007d63 srt: 0 sta: 9 flg: 0x2 ktucf redo: uba: 0x01000548.03b2.16 ext: 2
spc: 4046 fbi: 0
CHANGE #2 MEDIA RECOVERY MARKER CON_ID:0 SCN:0x0000000000000000 SEQ:0 OP:24.4 ENC:0 FLG:0x0000
```

Config

태그 기반 필터링 기능은 ***EXCLUDE_TAG*** Config를 통해 설정할 수 있다.

```
EXCLUDE_TAG="[ * | 0 | <tag> ]"
```

- ***EXCLUDE_TAG***:
  - 0 (Zero): 태그 기반 필터링을 비활성화 한다. 기본 값이다.
  - * (Asterisk): 태그 값과 상관 없이 태그가 걸린 모든 트랜잭션을 추출에서 제외한다.
  - <tag> : 최소 4자리에서 최대 2000자리 Hexadecimal (0-9, A-F) 값이어야 한다.
    태그 값이 홀수로 설정 될 경우 앞에 0을 붙이고 필터링을 수행해야 한다.
    최소, 최댓값에 맞지 않는 값은 Abended

예를 들어 다음과 같이 Post에 ***SET_TAG***가 설정되면, 이를 추출에서 제외하기 위해 Extract에선 동일한 tag 값이 설정되거나, * (Asterisk)가 사용되어야 한다.

```
-- Post Config --
SET_TAG="00112233445566778899AABBCCDDEEFF"


-- 1. Extract Config using Tag value --
EXCLUDE_TAG="00112233445566778899AABBCCDDEEFF"


-- 2. Extract Config using Asterisk --
EXCLUDE_TAG="*"
```

홀수 태그 값 참고

set_tag는 hex 값으로 설정하므로 짝수이어야 한다.

예를 들어 다음과 같이 8자리 hex 값을 설정하고 DB에 저장된 태그 값을 조회하면 8자리 그대로 저장된다.

짝수

```
-- set_tag --
begin
    dbms_streams.set_tag(tag => '74657374'); -- test
    commit;
end;
/

-- get_tag --
SQL> SELECT DBMS_STREAMS.GET_TAG FROM DUAL;

GET_TAG
--------------------------------------------------
74657374

-- Redo --
REPL MARKER:
Dump of memory from 0x00007F17B539304C to 0x00007F17B5393059
7F17B5393040                            000A0006              [....]
7F17B5393050 00010000 73657404 02C10274           [.....test...]
```

하지만 홀수로 set_tag 값을 설정하면 다음과 같이 태그 값 앞에 0이 붙는다.

홀수

```
-- set_tag --
begin
    dbms_streams.set_tag(tag => '746573745');
    commit;
end;
/

-- get_tag --
SQL> SELECT DBMS_STREAMS.GET_TAG FROM DUAL;

GET_TAG
--------------------------------------------------------------------------------
0746573745

-- Redo --
REPL MARKER:
Dump of memory from 0x00007FBBEF14704C to 0x00007FBBEF14705A
7FBBEF147040                            000B0006              [....]
7FBBEF147050 00010000 57460705 00024537           [......FW7E..]
```

그렇기 때문에 Post 쪽 Config에서 SET_TAG 값이 홀수로 설정되면, 실제 태그 값은 0이 붙은 hex 값이다.

Redo에서 태그를 통해 트랜잭션을 필터링 할 경우, 값이 홀수라면, 0을 붙여서 필터링하도록 한다.

EXCLUDE_TRANSACTION_NAME

태그 기반 필터링 외에 특정 트랜잭션에 대한 추출을 필터링 할 수 있는 기능이 있어야 한다. 기능 제공이 가능하다면 소스로 운영할 예정인 모든 DATABASE에서 적용한다.(1.5 기준 Oracle, Tibero)

트랜잭션 필터링은 Post가 적용하는 트랜잭션에 트랜잭션 명을 설정하면, Extract는 해당 트랜잭션 명을 가지는 트랜잭션을 추출에서 제외하도록 해야 한다.

예를 들어 다음과 같이 트랜잭션 명을 설정하면 트랜잭션의 첫 번째 레코드에서 트랜잭션 명이 기록된다. (트랜잭션 명은 최대 255개의 문자열이 가능하다.)

TRANSACTION NAME 설정 방법

```
-- Transaction   --
SET TRANSACTION NAME 'ARKCDC';

-- SQL   --
insert into test values (10, 10, 10);
COMMIT;
```

```
CHANGE #4 MEDIA RECOVERY MARKER CON_ID:0 SCN:0x0000000000000000 SEQ:0 OP:5.20 ENC:0 FLG:0x0000
session number   = 355
serial  number   = 53129
transaction name = ARKCDC <--
version 318767104
audit sessionid 4211343
Client Id =
login   username = OGGTEST
```

```
REDO RECORD - Thread:1 RBA: 0x000ec3.000037a0.0010 LEN: 0x023c VLD: 0x05 CON_UID: 0
SCN: 0x000000000d738946 SUBSCN:  1 06/30/2021 11:13:39
(LWN RBA: 0x000ec3.000037a0.0010 LEN: 0x00000002 NST: 0x0001 SCN: 0x000000000d738946)
CHANGE #1 CON_ID:0 TYP:0 CLS:27 AFN:4 DBA:0x010000d0 OBJ:4294967295 SCN:0x000000000d7388a2 SEQ:1 OP:5.2 ENC:
0 RBL:0 FLG:0x0000
ktudh redo: slt: 0x001e sqn: 0x00001619 flg: 0x0012 siz: 160 fbi: 0
            uba: 0x01000193.0a6f.17    pxid:  0x0000.000.00000000
CHANGE #2 CON_ID:0 TYP:0 CLS:28 AFN:4 DBA:0x01000193 OBJ:4294967295 SCN:0x000000000d7388a1 SEQ:1 OP:5.1 ENC:
0 RBL:0 FLG:0x0000
ktudb redo: siz: 160 spc: 4346 flg: 0x0012 seq: 0x0a6f rec: 0x17
            xid:  0x0006.01e.00001619
ktubl redo: slt: 30 wrp: 1 flg: 0x0c08 prev dba:  0x00000000 rci: 0 opc: 11.1 [objn: 94292 objd: 94564 tsn:
4]
[Undo type  ] Regular undo  [User undo done   ]  No  [Last buffer split]  No
[Temp object]           No  [Tablespace Undo  ]  No  [User only        ]  No
Begin trans
 prev ctl uba: 0x01000193.0a6f.16 prev ctl max cmt scn:  0x000000000d7380c4
 prev tx cmt scn:  0x000000000d7380c6
 txn start scn:  0xffffffffffffffff  logon user: 107
 prev brb:  0x00000000  prev bcl:  0x00000000
BuExt idx: 0 flg2: 0
KDO undo record:
KTB Redo
op: 0x04  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: L  itl: xid:  0x0002.017.000016c7 uba: 0x01000cea.0a94.2a
                    flg: C---    lkc:  0     scn:  0x000000000d737132
KDO Op code: DRP row dependencies Disabled
  xtype: XA flags: 0x00000000  bdba: 0x01c0015b  hdba: 0x01c0015a
itli: 1  ispac: 0  maxfr: 4858
tabn: 0 slot: 6(0x6)
LOGMINER DATA 10i:
opcode: INSERT
 Number of columns supplementally logged: 0  Flag: SE kdogspare1: 0x0 [ ] kdogspare2: 0x0 [ ] Objv#: 3
 segcol# in Undo starting from 0
```

```
 segcol# in Redo starting from 1
CHANGE #3 CON_ID:0 TYP:2 CLS:1 AFN:7 DBA:0x01c0015b OBJ:94564 SCN:0x000000000d7371bd SEQ:1 OP:11.2 ENC:0 RBL:
0 FLG:0x0000
KTB Redo
op: 0x01  ver: 0x01
compat bit: 4 (post-11) padding: 1
op: F  xid:  0x0006.01e.00001619    uba: 0x01000193.0a6f.17
KDO Op code: IRP row dependencies Disabled
  xtype: XA flags: 0x00000000  bdba: 0x01c0015b  hdba: 0x01c0015a
itli: 1  ispac: 0  maxfr: 4858
tabn: 0 slot: 6(0x6) size/delt: 12
fb: --H-FL-- lb: 0x1  cc: 3
null: ---
col  0: [ 2]  c1 0b
col  1: [ 2]  c1 0b
col  2: [ 2]  c1 0b
CHANGE #4 MEDIA RECOVERY MARKER CON_ID:0 SCN:0x0000000000000000 SEQ:0 OP:5.20 ENC:0 FLG:0x0000
session number   = 355
serial  number   = 53129
transaction name = ARKCDC
version 318767104
audit sessionid 4211343
Client Id =
login   username = OGGTEST
```

Config

트랜잭션 명 기반 필터링 기능은 ***EXCLUDE_TRANSACTION_NAME*** Config를 통해 설정할 수 있다. 기본 값은 비활성화이다.

```
EXCLUDE_TRANSACTION_NAME ="[ * | 0 | <name> ]"
```

- ***EXCLUDE_TRANSACTION_NAME*** :
  - 0 (Zero): 트랜잭션 명 기반 추출을 비활성화 한다.
  - * (Asterisk): 트랜잭션 명과 상관 없이 트랜잭션 명이 있는 모든 트랜잭션을 추출에서 제외한다.
  - <name> : 영어 (A-Z), 숫자 (0-9), 언더바 (_), 바 (-) 의 조합만 지원된다. 최소 4자리에서 최대 255자까지 지원된다.

예를 들어 다음과 같이 Post에 ***SET_TRANSACTION_NAME***이 설정되면, 이를 추출에서 제외하기 위해 Extract에선 동일한 name 값이 설정되거나 아스트릭스가 사용되어야 한다.

```
-- Post Config --
SET_TRANSACTION_NAME="ARKCDC_TRANSACTION_WITH_A_COMPLICATED_NAME"

-- 1. Extract Config using name value --
EXCLUDE_TRANSACTION_NAME ="ARKCDC_TRANSACTION_WITH_A_COMPLICATED_NAME"

-- 2. Extract Config using Asterisk --
EEXCLUDE_TRANSACTION_NAME="*"
```

EXCLUDE_USER_NAME

EXCLUDE_USER_NAME는 데이터베이스 사용자명을 통해 트랜잭션을 필터링하는 Config이다.

```
EXCLUDE_USER_NAME= "user"
```

- ***EXCLUDE_USER_NAME***:
  - 유효한 데이터베이스 사용자명을 명시해야 한다.
  - Wildcard 사용은 허용되지 않는다.

Emergency 모드

기능

EMERGENCY 모드는 프로세스가 5번 이상 재시작 될 경우 해당 프로세스의 로그와 덤프를 남기기 위해 프로세스를 임시로 시작하는 기능이다.

EMERGENCY 모드는 Extract 시작 옵션인 --emergency를 통해서도 수행이 가능하다.

```
admgr> start --emergency [module_alias]
```

EMERGENCY 모드로 실행한 프로세스가 종료될 경우, 이 프로세스는 재시작되지 않는다.

Emergency 모드가 수행된 후, 로그 레벨과 상관없이 중단되기 직전 로그 1000개를 로그 파일에 덤프 형태로 남긴다.

캐릭터셋 변환 실패

Extract는 추출한 데이터를 전부 UTF-8로 변환한다.

이때 변환이 실패할 경우 (iconv error) ***FAILED_CONVERT_TO_UTF8*** Config를 통해 다음과 같이 동작하도록 설정 할 수 있다.

```
FAILED_CONVERT_TO_UTF8="[ STOP | NULL | RAW | FETCH]";
```

- ***FAILED_CONVERT_TO_UTF8***
  - 옵션:
    - STOP (Default): Extract 중단
    - NULL: 데이터를 NULL로 추출하도록 한다.
    - RAW: UTF-8로 변환하기 전 캐릭터셋으로 추출한다.
    - FETCH: 데이터를 Select를 통해 추출한다.

추출 제외 대상

추출 제외 오브젝트

오브젝트명	설명
"BIN$*"	Recycle Bin용 오브젝트
"SYS_C*"	테이블 생성 시 제약조건명을 설정하지 않았을 때 오라클에서 지정해주는 임의의 인덱스명
“SYS_SUBP*”	오라클에서 지정해주는 임의의 서브 파티션명
"ARK_MARKER_SEQ"	ArkCDC 트리거용 오브젝트
"ARK_HIST_SEQ"	
"ARKCDC_DDL_MAKER"	
"ARKCDC_DDL_HIST"	

[표 25.1.1 추출 제외 오브젝트]

추출 제외 스키마

1. ANONYMOUS
2. APEX_030200

3. AUDSYS
4. BI
5. CTXSYS
6. DBSFWUSER
7. DBSNMP
8. DIP
9. DMSYS
10. DVF
11. DVSYS
12. EXDSYS
13. EXFSYS
14. FLOWS_FILES
15. GSMADMIN_INTERNAL
16. GSMCATUSER
17. GSMUSER
18. HR
19. IX
20. LBACSYS
21. MDDATA
22. MDSYS
23. MGMT_VIEW
24. MTSSYS
25. ODM
26. ODM_MTR
27. OE
28. OJVMSYS
29. OLAPSYS
30. ORACLE_OCM
31. ORDDATA
32. ORDPLUGINS
33. ORDSYS
34. OUTLN
35. OWBSYS
36. PM
37. SCOTT
38. SH
39. SI_INFORMTN_SCHEMA
40. SPATIAL_CSW_ADMIN
41. SPATIAL_CSW_ADMIN_USR
42. SPATIAL_WFS_ADMIN
43. SPATIAL_WFS_ADMIN_USR
44. SYS
45. SYSBACKUP
46. SYSDG
47. SYSKM
48. SYSMAN
49. SYSTEM
50. TSMSYS
51. WK_TEST
52. WKPROXY
53. WKSYS
54. WMSYS
55. XDB
56. XS$NULL
57. XTISYS

Extract 모듈 상태

다음은 Extract 모듈 상태 종류이다:

- Running
  Extract가 동작 중인 상태
- Stopped
  Extract가 정상 중지된 상태.
- Error
  에러가 발생하여 Extract가 중지 된 상태

Extract는 위 모듈 상태를 admgr, Checkpoint 파일, Manager 및 미들웨어에 제공해야 한다.

필요에 따라 모듈 상태를 추가해서 사용할 수 있다. (e.g. Manager의 경우 Agent 통신 문제로 인한 상태로 UNKNOWN이 있다)

Module Interface Thread API

Module Interface Thread API는 모듈과 통신하는 스레드로, 별도 스레드가 존재한다.
Module Interface Thread API 로 얻을 수 있는 정보는 다음과 같다:

- 통계 정보
- 체크포인트
- LAG 정보
- 추출 중인 트랜잭션 정보 (트랜잭션 추출 중단 등)ㅌ호
- Alive 체크
- 시작, 종료 요청

메타데이터 관리

기능

동일한 메타데이터를 공유하는 레코드가 여러 Tracing 파일에 걸쳐 기록 될 경우, 각 Tracing 파일에 해당 레코드의 메타데이터를 기록해야 한다.

예를 들어 하나의 트랜잭션에 있는 레코드가 다음 Tracing 파일에서 이어지거나, 레코드가 여러 레코드 세그먼트로 분할 되어 다음 Tracing 파일에 이어질 경우 그 다음 Tracing에서도 메타데이터가 기록되어야 한다.

즉, 각 Tracing 파일마다 레코드에 대한 메타데이터가 기록되어야 한다.

Redo Reader

다음은 Extract의 Read 기능이다:

- redoparser on file system read
- redoparser on ASM read

Multi-write Extract

하나의 Extract가 여러 Tracing 파일에 기록 할 수 있도록 하는 기능이다.

Multi-write Extract를 구성하려면 TRACINGALIAS Config를 여러 개 설정해야 한다.

다음은 Multi-write Extract Config 예시이다:

```
TRACINGALIAS="ext01_wrt1";
TABLE = "CDCTEST.TEST";
SEQUENCE = "";
DDL = "";
TRACINGFILE_SIZE="";
TRACINGFILE_DEST="";
MAP="map1";
FAILED_FETCH_LOB_TO_EMPTY=""
GET_TRUNCATE="no"

TRACINGALIAS="ext01_wrt2";
TABLE = "CDCTEST.TEST2";
```

```
SEQUENCE = "";
DDL = "";
TRACINGFILE_SIZE="";
TRACINGFILE_DEST="";
MAP="map2";
FAILED_FETCH_LOB_TO_EMPTY=""
GET_TRUNCATE=""
```

- TRACINGALIAS Config부터 GET_TRUNCATE Config는 하나의 Tracing Alias 단위 설정이다.
  이 Config들이 하나의 세트이며, 최대 10개까지 설정 가능하다.
- Tracing Alias 설정은 다른 Tracing Alias 설정에게 영향 받지 않으며, 설정되지 않은 Config는 Default 값이 사용된다.
- Config가 중복 될 경우, 가장 마지막 Config 설정이 사용된다.

데이터베이스 체크

데이터베이스 상태 체크 주기

Extract는 데이터베이스 상태를 주기적으로 확인한다. 상태 확인 주기는 다음과 같이 설정할 수 있다.

```
DB_STATUS_CHECK_INTERVAL="n [ s | S | m | M | h | H ]"
```

- ***DB_STATUS_CHECK_INTERVAL***
  - 기본 값: 60s
  - 최솟 값: 1s
  - 최댓 값: 1h

데이터베이스 중단 시 동작

Extract 모듈은 데이터베이스 중단 시 중지되어야 한다.

RAC 환경일 경우, Extract가 수행중인 스레드의 데이터베이스가 중단 될 때 Extract 모듈도 중지된다.

Extract가 수행 중인 스레드가 아닌 그 외 스레드에서 데이터베이스가 중단 될 경우 Extract는 정상 동작한다.

트랜잭션 관리

트랜잭션 관리 기능은 다음과 같은 Config를 통해 설정 가능하다:

```
TRANSACTION_CHECK_INTERVAL = "n[S]"
LONG_TRANSACTION_CHECK_INTERVAL = "n[M]"
LONG_TRANSACTION_TIMEOUT = "n[H]"
```

- ***TRANSACTION_CHECK_INTERVAL***
  Commit 된 transaction을 체크하는 간격이다.
  이 Config의 간격으로 메모리에 올라온 데이터 사이즈를 체크하며, IN_MEMORY_TX_SIZE를 초과할 경우 해당 시점에 가장 큰 transaction을 large transaction으로 전환 시킨다.
  - 기본 값: 5s
  - 최솟값: 2
  - 최댓값: 30
  - 지원 포맷: 정수에 이은 S문자의 조합
- ***LONG_TRANSACTION_CHECK_INTERVAL***
  Commit 되지 않은 트랜잭션 중 Long 트랜잭션을 체크하는 간격이다.
  LONG_TRANSACTION_CHECK_INTERVAL간격으로 Transaction list를 탐색하여, LONG_TRANSACTION_TIMEOUT를 초과할 경우 해당 transaction을 long transaction으로 전환 시킨다.
  - 기본 값: 5m

    - 최솟값: 5
    - 최댓값: 30
    - 지원 포맷: 정수에 이은 M문자의 조합
- ***LONG_TRANSACTION_TIMEOUT***
  Long transaction으로 판단되는 시간이다.
    - 기본 값: 1h
    - 최솟값: 1
    - 최댓값: 24
    - 지원 포맷: 정수에 이은 H문자의 조합

디스크 사용량 관리

Extract 모듈이 사용할 최대 디스크 사용률을 관리하려면 ***DISK_USAGE_CAPACITY*** Config를 사용해야 한다.

DISK_USAGE_CAPACITY Config에 설정된 퍼센트만큼 디스크가 채워지면 Extract 모듈이 종료된다.

```
DISK_USAGE_CAPACITY = "n"
```

- DISK_USAGE_CAPACITY
    - 기본 값: 95
    - 최솟 값: 0
    - 최댓 값: 99
    - 최솟값 0 설정시 디스크 사용률을 확인하지 않는다.

레코드 기록 단위

레코드 기록 단위는 Extract가 추출한 레코드를 Tracing 파일에 기록하는 단위이다.

단위 설정은 ***WRITE_FLUSH_RECORD_COUNT***와 ***WRITE_FLUSH_BUFFER_SIZE*** Config를 통해 설정할 수 있다.

```
WRITE_FLUSH_RECORD_COUNT = "[ n ]"
WRITE_FLUSH_BUFFER_SIZE = "[ n ] [ k | k | m | M ]"
```

- ***WRITE_FLUSH_RECORD_COUNT***
    - Extract가 한번에 기록 할 레코드 개수 경계점이다. 이 값만큼 레코드 개수가 채워지면 해당 레코드들이 Tracing 파일에 기록된다.
    - ***WRITE_FLUSH_RECORD_COUNT*** Config에 설정 된 개수만큼 레코드가 채워지지 않아도, ***WRITE_FLUSH_BUFFER_SIZE*** Config에 설정된 크기에 도달할 경우, 레코드가 파일에 쓰여진다.
    - 기본 값: 1000
    - 최솟 값: 1
    - 최댓 값: 10,000,000
- ***WRITE_FLUSH_BUFFER_SIZE***
    - Extract가 한번에 기록 할 레코드 사이즈 경계점이다. 이 값만큼 레코드 사이즈가 채워지면 해당 레코드들이 Tracing 파일에 기록된다.
    - ***WRITE_FLUSH_BUFFER_SIZE*** Config에 설정 된 크기만큼 레코드가 채워지지 않아도, ***WRITE_FLUSH_RECORD_COUNT*** Config에 설정된 개수에 도달할 경우, 레코드가 파일에 쓰여진다.
    - 기본 값: 64k
    - 최솟 값: 1k
    - 최댓 값: 100mb

※ 레코드가 개수 또는 사이즈만큼 채워지지 않아도, 더 이상 추출할 레코드가 없다면 Tracing 파일에 기록이 수행된다.

추출 예외 사항

1. 이미 NULL인 데이터를 NULL로 Update할 때, 해당 컬럼 이후 모든 컬럼 데이터가 NULL일 경우, 해당 Update문은 추출되지 않는다.

```
CREATE TABLE test_tab ( a NUMBER, b VARCHAR2(50), c DATE );

INSERT INTO test_tab VALUES (1, 'arkcdc', NULL);
COMMIT;
INSERT INTO test_tab VALUES (2, NULL, NULL);
COMMIT;
INSERT INTO test_tab VALUES (3, NULL, SYSDATE);
COMMIT;

UPDATE test_tab SET b = 'arkcdc' WHERE a = 1; <--
COMMIT;
UPDATE test_tab SET b = null WHERE a = 2; <--
COMMIT;
UPDATE test_tab SET b = null WHERE a = 3; <--
COMMIT;
```