Cartridge API - Deferrred Error Handling

The `scanctx_cache` package

The functions/procedures of this package operate in the scope of searches started in the current database session. The `scanctx_cache` package can be used to query scan contexts, which are PL/SQL objects holding information on the context of executing a JCC operator during a SQL statement execution. They are defined as follows

```
create or replace type SCAN_CONTEXT as object (
      scanId number,                 -- a unique idenfier of the scan context being described
      baseTableSchema varchar(30),   -- the schema owning the structure table involved
      baseTableName varchar(30),     -- the structure table involved in the search
      indexedColumnName varchar(30), -- the column on which the current searched JChem index was created
      indexSchema varchar(30),       -- the schema where the JChem index was created
      indexName varchar(30),         -- name of the JChem index the search is/was executed on
      usrOpId varchar2(200),         -- currently unused
      startDate date,                -- start date of the search
      operatorName varchar(30),      -- name of the JChem search operator
      arguments SCAN_ARGUMENTS,      -- arguments of the JChem operator without the indexed column
      note varchar2(32767)           -- currently unused
)
```

The `arguments` attribute returns the `SCAN_ARGUMENTS` is defined as follows:

```
create or replace type SCAN_ARGUMENTS is varray(6) of varchar2(32767);
```

The execution of <u>each JCC operator in a given SQL statement</u> has its own scan context. A maximum of 200 scan contexts are cached by database sessions allowing for 200 JCC operators in a single SQL statement. The notion of `scan context` is extended here to mean both domain scan access and filtering/functional access by a JCC operator. The `get_scanctxs` function returns all cached scan contexts. The function `get_scan_context(scanId number)` returns a scan context having the specified scan id. The scan context cache can be cleared with `clear()`.

Examples:

1. To return all cached scan contexts:

```
select ctx.scanid,
       ctx.baseTableSchema,
       ctx.baseTableName,
       ctx.indexedColumnName,
       ctx.indexSchema,
       ctx.indexName,
       ctx.startDate,
       ctx.operatorName,
       ctx.arguments
  from table(scanctx_cache.t_get_scanctxs()) ctx;
```

2. To return one particular scan context:

```
select scanctx_cache.get_scan_context(?) from dual;
```

3. For the following SQL:

```
select count(*) from a, b
where a.some_column = b.some_column and
      jc_compare(a.structure, 'Brc1ccccc1', 't:s') = 1 and
      jc_compare(b.structure, 'Clc1ccccc1', 't:s) = 1;
```

two scan contexts will be created. They will differ in

- the `baseTableName` attribute (it will be 'A' for one and 'B' for the other)
- the `indexName` attribute (assuming that the JChem indexes on both a.structure and b.structure were created in the same schema, in which case they will necessarily have to differ)
- the first element of the arguments attribute (one will be 'Brc1ccccc1', the other will be 'Clc1ccccc1').

To retrieve the scan context id for the first operator an SQL similar to the following can be used:

```
select ctx.scanid
from table(scanctx_cache.t_get_scanctxs()) ctx
where upper(ctx.baseTableName) = 'A');
```

The `error_cache` package

The functions/procedures of this package operate in the scope of searches started in the current database session.

The `get_error_count` function can be used to query the number of cached error records. Error records are PL/SQL objects holding information about an error. They are defined as follows:

```
create or replace type ERROR_RECORD as object(
    scanId number,
    rid varchar2(100), -- The rowid of the target structure involved in the error (where applicable)
    errorMessage varchar2(32767),
    note varchar2(32767) -- Note about the error; e.g. indicates that the error message was truncated
)
```

Error records are cached until they are accessed or until `clear()` is called. Cached error records can be accessed through the `t_get_errors(scanId number := null)` function. Leaving the `scanId` parameter at its default value, all cached errors are returned (and removed from the cache). If the `scanId` parameter is specified, error records associated with the corresponding scan contexts are returned (and removed from the cache).

Example:

The following statement can be used to retrieve all attributes of all cached error records:

```
select errors.scanid,
       errors.rid,
       errors.errormessage,
       errors.note
from table(error_cache.t_get_errors());
```

Error table during index creation and rebuild

During index creation and rebuild with `haltOnError` parameter set to "n" or "nf", the error messages can be logged into an error table. To do this, set the `errorTableName` parameter to the name of the error table. The columns of the error table are the following:

```
"insert_date" date not null,
"index_name" varchar2(70 byte) not null,
"rid" rowid,
"message" varchar2(4000 byte)
```

If the error table doesn't exist in case of an error, the table is created. If there is an error during inserting the error row into the error table for some reason, the table will NOT be dropped and no data will be lost, and the error still can be searched in the log. If there is a problem with the error table inserts (e.g. different table structure than expected), it will be added to the log file, but doesn't throw exception (to be in harmony with the `haltOnError` parameter).

Example:

```
CREATE INDEX jc_idx ON jchemtest(structure_col) INDEXTYPE IS jc_idxtype PARAMETERS('haltOnError=nf,
errorTableName=error_table');
```