DVPDC - Automtated testing

To verify the DVPD usablitiy and DVPDC functionality, various scenarios are implemented as testsets. Paired with a referece resultset, these allow to check if

- DVDP syntax is able to express all variants of data vault modells and mappings
- the DVPDC is able to transform all valid DVPD into a DVPI
- the DVPDC is able to detect and report syntax and rule violations in the DVPD

Setup

test data

All testsets are stored in the repository directory "testsets_and_example" as follows

- dvdp: all dvpd files
- model_profiles: the model profiles used by the dvpd of the testset
- reference: the logfile of the compiler for every test and the expected dvpi file, for valid dvpd

test process

Execution and evaluaton of the tests is implemented in the module "test_dvpdc", wich is configured via "dvpdc.ini" section "dvpdc_test". The test_dvpdc process calls the dvpdc for every test case, and compares the result log and result dvpi with the reference data.

- With the parameter -t you can run the automated test for a specific test number.
- With the parameter -f you can run the automated test for a specific file in the test cases.
- Without explicit declaration of a file, dvpcd_test runs all tests in the test case directory.

To support an efficient verification of the automated tests, the test process logs all test results in detail and a final summary over all tests.

Example of the summary:
Test state:225 = success 223 (6TXUQLFS)+ fail 1 (QKTJQFK3)+ incorrect 1 (W6ZNWMDD)

The fingerprint after each section is calculated from the test file names. So when some tests are failing, you can determine, if the same tests failed again in the next run, since the fingerprints will not change then.

The outcome of a test can be as follows:

- Passed/success
- Result differs/difference: Compile result does not match the reference data
- failed compile / fail: Compile failed, so result can't be correct

- no reference data / no ref: Compile was successfull but there is not reference data to compare the result with
- crashed / crash: Compiler process crashed due to a fatal implementation mistake
- incorrect: Compile was sucessfull but was expected not to be

There are currently two tests in the test set, that are meant to be not successfull. These cases verify, if the test process detects the fail/incorrect situation properly.

test workflow rules

1. Every time, the commit declares to be complete or best effort. The commit log must contain the summary of a full test run.
2. When merging or checking out a branch, you should run a full test and compare the summary with the last logged summary of the merged branch.

Conventions for tests content

The content of all core testsets must follow the naming rules, described below. The goal of the rules is to express the expected result, so it can be checked by eyesight very easy.

test names / identifiers

- All tests have a unique number
- The file names follow the pattern: t<4 digit number>[c]_<essential description>.json.dvpd
    - [c]: c after the number indicates this test to be only adressing the compiler check rules. No dvpi will be created.
    - examples t1000
- There are some legacy patterns we will transform over time: "test<2 digit number>", "test_<3digit number>"

Tests are grouped by number ranges. See test list at the end of this article.

pipeline declarations

- pipeline_name: identical to the filename
- pipeline_comment: "Test for " + Description of the usecase / constellation / this tests is targeting
- data_extraction/fetch_module_name: "none - this is a pure generator test case"

Names in the data model

- the main schema is "rlvt_text_jj" with the shortcut "rtjj"
- for multi schema test cases, additional schemas are named by progressing in the alphabet (kk,ll,mm)
- table names are structured as follows: <schema shortcut><*testnumber*><identifier>_<stereotype>
    - <schema_shortcut>: rtjj, rtkk, rtll ...
    - <identifier for hubs> 3 times a letter from A to G (e.g. AAA,BBB)

    - <identifier for links> identifiers of all connected hubs separated by + optional "" <*identifier for relation*> + *optional* "" <identifier for parent paths when different to relation>
    - <identifier for sattelites> identifier of parent + part identification (p1, p2, ...) or optional <identifier for relation>
    - <stereotype> extended stereotype postfix (e.g. hub, sat, lnk, dlnk, esat,msat,...)
    - **examples: rtjj_55_aaa_hub, rtjj_22_aaa_bbb_lnk, rtjj_20_aaa_p1_sat, rtjj_55_aaa_bbb_ttt_esat**

- name for the key columns are equal to their table as follows.

    - a 2 letter prefix declares if it is a key of a hub (HK) or a key of a link (LK)
    - name of the table, without the stereotype postfix
    - **examples**: HK_rtjj_64_aaa, LK_rtjj_22_aaa_bbb

Relation specific names

- <identifier for relations> 3 times a letter from T to W (e.g. TTT,UUU)
- <identifier for parent paths when different to relation> 3 times a letter from P to R (e.g. PPP,QQQ)

Field names and mappings Fieldnames express the target of the field, to provide easy control of the correct mapping.

- format <field position>[<rename trigger>]<*target table identifier*><column class><sequence>[F][_<identifier for relation>]
    - <field position>: F + position in the field array (e.g. F1, F2). This allows immediate identification of the field
    - <rename trigger>: XX is added to the field position, when renameing in the column mapping is expected
    - <target table identifier>: see identifer above in the table names
    - <column_class>: BK= Businesskey, DC= dependent child key, C= content, UC=untracked content
    - : Sequence of the field in the target for the same columns classe
    - F: Is tagging the the field of a column class in the table (F=Finally)
- Fields that are adressing 2 to 3 targets, concatinate the table identifiers and columnns classes, separated by "_"
- Fields that are addressing more then 3 tragets, need more thinking about a good naming strategy
- Column renaming is done by removing XX from the name to the table identifier column class, sequence and F marking
- Columns that are adressed by multiple fields, will concatenate all field postions as prefix

Examples:

- F1_AAA_BK1F - Businesskey in Hub AAA. This is the final business key field for hub aaa
- F2_BBB_BK2 - Businesskey in Hub BBB. There must be a BK1 and at least one additional BK mapped to BBB
- F3_AAA_P1_C3 - Content in sattelite AAA_P1. There will be more content, and also there must be C1 and C2 in the sattelite
- F4_CCC_BK1F_DDD_BK2 - Field is mapped to CCC and DDD as business key. It is the final business key column in CCC
- F5XX_AAA_P2_C2 - Field is mapped to AAA P2 Satellite but must be renamed to AAA_BK2
- F6F7_BBB_P1_C1 - Two fields are mapped to the same column in BBB Satellite P1

test list

The following number ranges are defined (a legacy from the proof the development):

Number	Focussed Features
00 - 19	**Violation of validation rules**
00	Compiler must complain about missing essential syntax elements
01	Compiler must complain about bad model relations
02	Compiler must complain about bad fiels mapping
03	Compiler must complain about sattlite specifc violations
20 - 39	**Varations of basic data vault modelling**
40 - 49	Reference tables**
50 - 59	**multiple features in many variations, used in compiler development**
60- 89	**relation variations (might be replaced by 100-999 and 3xxx later**
90 - 99	**model profile usage**
100-999	see catalog_of_field_mappings_in_relations.md
1000-2999	**Collection without central topic**
300x-499x	**process generation variations** with x = 0 for basic tests and x>0 for violations of comiler rules
5000-5099	**DDVUG Test Usecase "Gartenscenter** willibald"
7xxx-9xxx	**Compiler rule checks, based on cases in 100-999. Using the same 3 digit numbers**

Collection without central topic

Test numbers 1000 to 2999 canbe used for singular or very small groups of tests, for any topic, not targeted by the big test fields. It is requiered to declare the test target in the file name.

Some smaller groups are listed here

1020-1039 - Tests for interpretation of "use_as_key_hash" and "is_only_structural_element"

- 1020 - Multiple sat variants in one case
- 1021 - Multiple link variants in one case
- 1022 - Business Vault sat at Hub
- 1023 - Business Vault sat on link
- 1024 - business vault link on 2 hubs

- 1025 - business vault link on 4 hubs
- 1026 - business vault link with recursive relation
- 1027 - business vault link with recursive relation and other hub
- 1028 - business vault link with dependent child key and 2 hubs
- 1029 - business vault link with dependent child key and recurive relation
- 1030 - business vault link with dependent child key, recurive relation and other hub
- 1031c - Multiple "use_as_key_hash" mappings to same target column
- 1032c - "use_as_key_hash" mapping to unknown key hash column
- 1033c - "is_only_structural_element" declared child has no "use_as_key_hash" mapping (trigger a warning that this might result in orphan entries)
- 103x - Behaviour using relations (first we need to investigate if this can be really a use case)