

SDAT - sběr dat



Říjen 2014

**© ZHOTOVITEL Czech s.r.o.**, Všechna práva vyhrazena

Říjen 2014

**© ZHOTOVITEL Czech s.r.o.**, Všechna práva vyhrazena

Říjen 2014

**© ZHOTOVITEL Czech s.r.o.**, Všechna práva vyhrazena

**Technická specifikace SDAT**

**TS-8 Popis jazyka kontrol**

**Popis jazyka kontrol a příklady zápisu**

***Verze: 1.6***

**Identifikace dokumentu**

|  |  |
| --- | --- |
|  |  |
| Název dokumentu: | **Technická specifikace SDAT, TS-8 Popis jazyka kontrol** |
| Verze dokumentu: | **1.6** |
| Projekt:  | **SDAT - sběr dat** |
| Autor: | NESS Czech s.r.o., ČNB |

**Historie dokumentu**

|  |  |  |
| --- | --- | --- |
| **Datum** | **Verze** | **Popis verze a změn oproti verzi předchozí** |
| 2018-11-29 | 0.1 | Úvodní verze dokumentu |
| 2019-04-15 | 0.2 | Doplnění gramatiky uživatelského jazyka a drobné změny |
| 2019-07-24 | 0.3 | Opravy v pravidlech Pragma, RelationalExpression, DataPoint. Pravidlo pro DataPoint bylo dále upřesněno. Z gramatiky odstraněn FilterFail a zaveden jako funkce. Doplněna funkce FILTER. |
| 2019-10-14 | 0.4 | Vložení kapitol 4.1.7 a 5.1.7 obsahující popis specifických konstrukcí pro JOIN funkce a doplnění JOIN funkcí.Dílčí upřesnění a opravy popisu jazyka, funkcí a gramatiky. |
| 2020-03-27 | 0.5 | Doplnění a upřesnění kontrol typu JOIN. Doplnění popisu hodnot NULL v zápisu kontrol.Dílčí upřesnění a opravy popisu jazyka, funkcí a gramatiky. |
| 2020-05-22 | 0.6 | Doplnění funkcí REF\_DATE a DATA\_FILING\_EXISTS |
| 2020-10-08 | 0.7 | Doplnění funkcí EXT\_EXISTS, EXT\_RETURN, ISIN\_CHECK, LEI\_CHECK, ICO\_CHECK, REF\_OSOBA\_LEI a NUMBER.Úprava definice funkce DATE. |
| 2020-02-09 | 0.8 | Doplnění funkcí CONS\_HIST, RIAD\_EXISTS, RIAD\_CHECK, RIAD\_CHECK\_OUTER, RIAD\_FETCH, RIAD\_FETCH\_OUTER |
| 2021-03-22 | 0.9 | Doplněn úvod kapitoly 6.14. Speciální funkce pro AnaCredit |
| 2021-04-09 | 1.0 | Doplnění funkce COMPLETENESS a nové pragmy. |
| 2021-04-27 | 1.1 | Doplnění funkce REF\_OSOBA\_KOD a \_ISBLANK. |
| 2021-05-07 | 1.2 | Doplnění nových funkcí ADD\_DAYS, FILTER\_UNIQUE a DOM\_CONTENT.Úprava popisu funkcí COUNT a COUNTA.  |
| 2021-06-02 | 1.3 | Doplněna nová funkce ADD\_MONTHS |
| 2021-11-29 | 1.4 | Doplněny nové funkce REF\_RCVD\_TIME, ISO\_DATETIME a EXT\_EXISTS\_2 |
| 2022-03-16 | 1.5 | Doplnění nové funkce REF\_LEI\_ALLOWED().Doplnění nové kapitoly 7.3. Předpoklad |
| 2022-10-17 | 1.6 | Doplnění nové funkce REF\_ROZSAH()Úprava popisu funkce LINREG |

**Obsah**

[1. Přílohy a odkazy na jiné dokumenty 6](#_Toc117670025)

[2. Úvod 7](#_Toc117670026)

[2.1. Účel dokumentu 7](#_Toc117670027)

[3. Kontroly v systému SDAT 8](#_Toc117670028)

[3.1. Úvod 8](#_Toc117670029)

[3.2. Jednovýkazové kontroly 8](#_Toc117670030)

[3.3. Mezivýkazové kontroly 10](#_Toc117670031)

[3.4. Sémantický a uživatelský jazyk kontrol 11](#_Toc117670032)

[3.5. Kontroly převzaté z externích specifikací 11](#_Toc117670033)

[4. Sémantický jazyk kontrol 12](#_Toc117670034)

[4.1. Popis sémantického jazyka kontrol 12](#_Toc117670035)

[4.1.1. Úvod 12](#_Toc117670036)

[4.1.2. Základní konstrukce 12](#_Toc117670037)

[4.1.3. Specifikace údajů a parametrů 14](#_Toc117670038)

[4.1.4. Hodnoty údajů a parametrů 19](#_Toc117670039)

[4.1.5. Proměnné 20](#_Toc117670040)

[4.1.6. Specifické konstrukce 21](#_Toc117670041)

[4.1.7. Typy RowSet, Record a operace dereference 27](#_Toc117670042)

[4.1.8. Kontrolní výraz 33](#_Toc117670043)

[5. Uživatelský jazyk kontrol 34](#_Toc117670044)

[5.1. Popis uživatelského jazyka kontrol 34](#_Toc117670045)

[5.1.1. Úvod 34](#_Toc117670046)

[5.1.2. Základní konstrukce 34](#_Toc117670047)

[5.1.3. Specifikace údajů a parametrů 34](#_Toc117670048)

[5.1.4. Hodnoty údajů a parametrů 37](#_Toc117670049)

[5.1.5. Proměnné 37](#_Toc117670050)

[5.1.6. Specifické konstrukce 37](#_Toc117670051)

[5.1.7. Typy RowSet, Record a operace dereference 45](#_Toc117670052)

[5.1.8. Kontrolní výraz 52](#_Toc117670053)

[5.2. Gramatika uživatelského jazyka kontrol 53](#_Toc117670054)

[6. Popis funkcí jazyka kontrol 55](#_Toc117670055)

[6.1. Úvod 55](#_Toc117670056)

[6.2. Logické funkce 56](#_Toc117670057)

[6.2.1. AND 56](#_Toc117670058)

[6.2.2. IF 56](#_Toc117670059)

[6.2.3. ISBLANK 57](#_Toc117670060)

[6.2.4. ISNUMBER 57](#_Toc117670061)

[6.2.5. NOT 58](#_Toc117670062)

[6.2.6. OR 58](#_Toc117670063)

[6.4. Matematické funkce 59](#_Toc117670064)

[6.4.1. ABS 59](#_Toc117670065)

[6.4.2. AVERAGE 59](#_Toc117670066)

[6.4.3. NUMBER 60](#_Toc117670067)

[6.4.4. ROUND 60](#_Toc117670068)

[6.4.5. ROUNDDOWN 61](#_Toc117670069)

[6.4.6. ROUNDUP 61](#_Toc117670070)

[6.4.7. SIGN 62](#_Toc117670071)

[6.4.8. SUM 62](#_Toc117670072)

[6.4.9. TRUNC 63](#_Toc117670073)

[6.4.10. DIV 63](#_Toc117670074)

[6.4.11. EXP 63](#_Toc117670075)

[6.5. Řetězcové funkce 64](#_Toc117670076)

[6.5.1. CONCATENATE 65](#_Toc117670077)

[6.5.2. LEFT 65](#_Toc117670078)

[6.5.3. LEN 65](#_Toc117670079)

[6.5.4. REGEXP\_LIKE 66](#_Toc117670080)

[6.5.6. RIGHT 67](#_Toc117670081)

[6.5.7. UPPER 67](#_Toc117670082)

[6.5.8. SUBSTR 67](#_Toc117670083)

[6.6. Datumové funkce 68](#_Toc117670084)

[6.6.1. DATE 68](#_Toc117670085)

[6.6.2. DAY 69](#_Toc117670086)

[6.6.3. MONTH 69](#_Toc117670087)

[6.6.4. YEAR 70](#_Toc117670088)

[6.6.5. YEARFRAC 70](#_Toc117670089)

[6.6.6. ADD\_DAYS 70](#_Toc117670090)

[6.6.7. ADD\_MONTHS 71](#_Toc117670091)

[6.6.8. REF\_DATE 71](#_Toc117670092)

[6.6.9. REF\_RCVD\_TIME 72](#_Toc117670093)

[6.6.10. ISO\_DATETIME 72](#_Toc117670094)

[6.7. Množinové funkce 73](#_Toc117670095)

[6.7.1. COUNT 73](#_Toc117670096)

[6.7.2. COUNTA 73](#_Toc117670097)

[6.7.3. MAX 74](#_Toc117670098)

[6.7.4. MIN 74](#_Toc117670099)

[6.8. Speciální filtrační funkce 75](#_Toc117670100)

[6.8.1. FILTER\_FAIL 75](#_Toc117670101)

[6.8.2. FILTER 75](#_Toc117670102)

[6.9. Speciální funkce s intervalovou aritmetikou 76](#_Toc117670103)

[6.9.1. IAF\_EQ 76](#_Toc117670104)

[6.9.2. IAF\_GT 77](#_Toc117670105)

[6.9.3. IAF\_GTE 77](#_Toc117670106)

[6.9.4. IAF\_LT 78](#_Toc117670107)

[6.9.5. IAF\_LTE 78](#_Toc117670108)

[6.9.6. IAF\_NEQ 79](#_Toc117670109)

[6.11. Ostatní speciální funkce 80](#_Toc117670110)

[6.11.1. LINREG 80](#_Toc117670111)

[6.11.2. CHECK\_DYNCOUNT\_MAX 80](#_Toc117670112)

[6.11.3. CHECK\_DYNCOUNT\_MIN 81](#_Toc117670113)

[6.11.4. ISIN\_CHECK 81](#_Toc117670114)

[6.11.5. LEI\_CHECK 82](#_Toc117670115)

[6.11.6. ICO\_CHECK 82](#_Toc117670116)

[6.11.7. DATA\_FILING\_EXISTS 83](#_Toc117670117)

[6.11.8. REF\_OSOBA\_LEI 84](#_Toc117670118)

[6.11.9. REF\_LEI\_ALLOWED 84](#_Toc117670119)

[6.11.10. REF\_OSOBA\_KOD 85](#_Toc117670120)

[6.11.11. DOM\_CONTENT 85](#_Toc117670121)

[6.11.12. FILTER\_UNIQUE 85](#_Toc117670122)

[6.11.13. REF\_ROZSAH 86](#_Toc117670123)

[6.12. Speciální JOIN funkce 87](#_Toc117670124)

[6.12.1. JOIN\_EXISTS 87](#_Toc117670125)

[6.12.2. JOIN\_CHECK 87](#_Toc117670126)

[6.12.3. JOIN\_CHECK\_OUTER 88](#_Toc117670127)

[6.12.4. JOIN\_FETCH 89](#_Toc117670128)

[6.12.5. JOIN\_FETCH\_OUTER 90](#_Toc117670129)

[6.12.6. CONS\_HIST 91](#_Toc117670130)

[6.12.7. \_ISBLANK 92](#_Toc117670131)

[6.14. Funkce na externí číselníky 93](#_Toc117670132)

[6.14.1. EXT\_EXISTS 93](#_Toc117670133)

[6.14.2. EXT\_EXISTS\_2 93](#_Toc117670134)

[6.14.3. EXT\_RETURN 94](#_Toc117670135)

[6.15. Speciální funkce pro AnaCredit 95](#_Toc117670136)

[6.15.1. RIAD\_EXISTS 95](#_Toc117670137)

[6.15.2. RIAD\_CHECK 95](#_Toc117670138)

[6.15.3. RIAD\_CHECK\_OUTER 96](#_Toc117670139)

[6.15.4. RIAD\_FETCH 97](#_Toc117670140)

[6.15.5. RIAD\_FETCH\_OUTER 98](#_Toc117670141)

[6.15.6. COMPLETENESS 99](#_Toc117670142)

[6.16. PRAGMY 100](#_Toc117670143)

[6.16.1. #ALG() 100](#_Toc117670144)

[6.16.2. #FIRST\_REPORTING 100](#_Toc117670145)

[6.16.3. #RESTRICT() 100](#_Toc117670146)

[6.16.4. #SQL\_TODATE(ana) 100](#_Toc117670147)

[7. Doplňující informace 101](#_Toc117670148)

[7.1. Prázdné hodnoty v kontrolách 101](#_Toc117670149)

[7.2. Hodnoty NULL v jazyce kontrol 101](#_Toc117670150)

[7.3. Předpoklad 103](#_Toc117670151)

#  Přílohy a odkazy na jiné dokumenty

Dokumentaci SDAT tvoří následující sada souborů:

|  |  |  |
| --- | --- | --- |
|  | **Název** | **Popis**  |
| 0 | Základní dokument | Popisuje strukturu dokumentace SDAT. |
| 1 | Popis systému | Popisuje základní vlastnosti a architekturu se zaměřením na funkce určené pro externí uživatele. |
| 2 | Metapopis vykazovací povinnosti | Popisuje způsoby, kterými lze získat informace o metodikách vykazování a o vykazovací povinnosti – jak webovými službami, tak přes uživatelské rozhraní webové aplikace. |
| 3 | Vykazování | Popisuje způsoby, kterými lze zaslat Vydání výskytů výkazů podle předepsané vykazovací povinnosti a možnosti získání informací o stavu jejich zpracování. |
| 4 | Vykazování – specifické případy | Rozcestník na samostatné dokumenty popisující specifické případy či způsoby vykazování (AnaCredit, ISO20022 a další). |
| 5 | Technické informace | Popisuje technické náležitosti jednotlivých komunikačních rozhraní. |
| 6 | Katalog webových služeb | Popisuje webové služby – primární rozhraní pro automatizované vykazování. |
| 7 | Modely metapopisu | Popisuje architekturu systému na úrovni vybraných entitních modelů. |
| **8** | **Popis jazyka kontrol** | **Popisuje jazyk kontrol systému SDAT (tento dokument).** |

# Úvod

Tento dokument – Popis jazyka kontrol – je jedním ze sady dokumentů, které dohromady tvoří Technickou specifikaci – kompletní popis systému pro sběr dat pro potřeby externích uživatelů.

## Účel dokumentu

Dokument popisuje jazyk kontrol systému SDAT.

# Kontroly v systému SDAT

## Úvod

Kontroly vstupní dat Výkazů ČNB členíme na:

* Formální a formátové – definují obecně platná pravidla pro vykazování dat a předepsané obory hodnot vykazovaných dat.
* Logické – definují logické vztahy mezi daty ve Výkazech.

Logické kontroly se člení podle příslušnosti kontrolovaných vztahů k Výkazům na:

* Jednovýkazové – definují vztahy mezi daty v rámci jednoho Výkazu.
* Mezivýkazové – definují vztahy mezi daty různých Výkazů nebo mezi daty Výkazů za různá období.

Formální a formátové kontroly jsou definované explicitními pravidly nebo předepsaným oborem hodnot a dalšími atributy, které jsou obsažené v příslušné metodice vykazování.

Pro zápis logických vztahů mezi daty ve výkazech a některých formátových kontrol jsou používány dva formalizmy – sémantickýa ***uživatelský***jazyk kontrol.

Vykazující subjekty dostávají popisy kontrol zapsané v obou formalizmech. Sémantické zápisy kontrol mají větší stabilitu a odrážejí význam kontroly, uživatelská forma je čitelnější a lépe algoritmizovatelná.

Sémantická forma kontroly by měla mít pro vykazující subjekty předevšíminformativní charakter, zatímco uživatelská forma je určenapro vlastní provádění kontrol.

## Jednovýkazové kontroly

Jednovýkazové kontroly definují vztahy mezi daty v rámci jednoho Výkazu a jsou prováděny v rámci vstupního zpracování Vydání výskytu výkazu.

Podle dat vstupujících do kontroly se člení na:

* Kontroly v rámci jedné Datové oblasti – kontrola se provádí nad daty jedné Datové oblasti zpracovávaného Vydání výskytu výkazu.
* Kontroly mezi různými Datovými oblastmi – kontrola se provádí nad daty více Datových oblastí zpracovávaného Vydání výskytu výkazu.
* Kontroly vůči předchozím zaslaným datům daného Výkazu – kontrola se provádí vůči předchozím zaslaným datům daného Výskytu výkazu nebo předchozím zaslaným datům Výskytů daného Výkazu. Do této skupiny patří především kontroly jednoznačných klíčů záznamů dynamických Datových oblastí.
* Kontroly vůči externím datovým zdrojům – kontrola se provádí proti externímu datovému zdroji (zpravidla číselníku nebo registru), který je vytvářen procesy mimo systém SDAT.

Z věcného hlediska nelze sice typy kontrol ohraničit, následující typy jsou příkladem logických vztahů, které jsou jazykem kontrol předepisovány:

* Vyplnění hodnoty – povinnost vyplnění hodnoty Údaje nemusí být definována příslušným atributem Údaje, ale lze ji definovat i formou zápisu v jazyce kontrol. Používá se především v případě podmíněné povinnosti, kdy se povinnost vyplnění Údaje určena vyhodnocením podmínky (zpravidla vyhodnocením hodnoty jiného Údaje nebo dynamického Parametru).
* Nevyplnění hodnoty – na základě vyhodnocení podmínky definuje, že hodnota údaje nemá být vyplněna.
* Upřesnění oboru hodnot – obor hodnot definovaný pro dynamický Parametr nebo Údaj v Datové oblasti může být definicí kontroly upřesněn. Převážně se jedná o zúžení předepsaného oboru hodnot na základě vyhodnocení definované podmínky (zpravidla vyhodnocením hodnoty jiného Údaje nebo dynamického Parametru).
* Součty podle hierarchií v Datové oblasti – kontroly vztahů mezi součtovými Údaji a jejich odpovídajícími Údaji – sčítanci, které vznikají na základě definovaných hierarchií Ukazatelů nebo hodnot Parametrů (Položek číselníků) v Datové oblasti.
* Součty v dynamické datové Oblasti – kontrola součtového a odpovídajícího dynamického Údaje v dynamické Datové oblasti, pokud tato obsahuje součtový řádek/sloupec.
* Řádkové/sloupcové kontroly ve statické Datové oblasti – kontrola vztahu mezi Údaji v rámci řádku/sloupce statické Datové oblasti, která se provádí pro jednotlivé řádky/sloupce statické Datové oblasti. Kontrola se provádí pro každý řádek/sloupec nebo množinu řádků/sloupců vymezenou filtrem.
* Řádkové/sloupcové kontroly v dynamické Datové oblasti – kontrola vztahu mezi Údaji v rámci řádku/sloupce dynamické Datové oblasti, která se provádí pro jednotlivé řádky/sloupce dynamické Datové oblasti. Kontrola se provádí pro každý vykázaný dynamický řádek/sloupec nebo množinu řádků/sloupců vymezenou filtrem.
* Počet dynamických instancí řádků/sloupců – vymezuje minimální nebo maximální počet požadovaných dynamických instancí (řádků, sloupců nebo karet).
* Kontroly jednoznačného věcného klíče záznamu dynamické Datové oblasti – dynamické Datové oblasti mohou obsahovat sloupce (nebo řádky v případě opačného dynamického rozvoje), které definují jednoznačný věcný klíč záznamu v dané Datové oblasti. Kontrolována může být jednoznačnost zaslaného klíče v rámci Vydání výskytu výkazu nebo vztah daného klíče vůči dříve zaslaným datům daného Výkazu.
* Integritní kontroly mezi dynamickými Datovými oblastmi – dynamická Datová oblast může obsahovat Údaje, které se vztahují k záznamům zaslaným v jiné Datové oblasti (master – detail vztah). Kontroly definují integritu zaslaných klíčů z obou stran (výskyt klíčů zaslaných v detail Datové oblasti vůči klíčům v master Datové oblasti nebo výskyt klíčů zaslaných v master Datové oblasti v detail Datové oblasti v případě povinnosti zaslaní detailních Údajů).
* Obecná kontrola vztahů mezi Údaji různých Datových oblastí – definuje vztah mezi Údaji různých Datových oblastí. Zpravidla se jedná o vztah statických Údajů nebo agregací za dynamické Údaje.

Výše uvedené typy kontrol se týkají statických nebo dynamických Datových oblastí nebo obou těchto typů.

Týkají se také kartových Datových oblastí, kde mohou být definovány pro všechny předepsané karty nebo všechny vykázané dynamické karty nebo mohou být definovány pro konkrétní kartu nebo množinu konkrétních karet,

V kartových Datových oblastech mohou být definovány i vzájemné logické vztahy mezi vykázanými kartami navzájem. Typicky se vyskytují následující:

* Vztah mezi součtovou kartou a detailními kartami – Datová oblast může obsahovat tzv. součtovou kartu, která obsahuje agregované Údaje za ostatní předepsané nebo vykázané instance karet.
* Vztah mezi konkrétními kartami – v definici kontrolního vztahu se používají Údaje z konkrétních instancí karet.

Jednovýkazové kontroly se zpravidla provádějí nad daty jednoho Vydání výskytu výkazu.

Do jednovýkazových kontrol jsou však zahrnuté i některé kontroly zaslaného Vydání výskytu výkazu vůči předchozím zaslaným datům daného Výkazu.

Specifickou skupinou jednovýkazových kontrol jsou kontroly vůči externím datovým zdrojům. Aktuálně se vyskytují ve Výkazech MKT a AnaCredit. Pro tyto oblasti jsou samostatnými procesy vytvářeny číselníky a registry s referenčními daty, které nejsou součástí systému SDAT, ale jsou používány v kontrolách Výkazů ČNB.

## Mezivýkazové kontroly

Mezivýkazové kontroly definují vztahy mezi daty různých Výkazů nebo mezi daty Výkazů za různá období.

Nejsou prováděny v rámci vstupního zpracování Vydání výskytu výkazu, ale v samostatném procesu, který je spouštěn v okamžiku, kdy jsou dostupná data všech Výskytů výkazů, mezi nimiž je kontrola prováděna.

Z hlediska dat, mezi nimiž jsou mezivýkazové kontroly prováděné, lze definovat následující typy:

* Kontroly mezi různými Výkazy za shodné období – kontrolují vztahy mezi daty různých Výkazů vždy za odpovídající období vykazování. Dané Výkazy mají zpravidla shodnou periodicitu vykazování a jejich Výskyty mají shodnou hodnotu Stav ke dni.
* Kontroly za stejný Výkaz a různá období – kontrolují vztah dat zaslaného Výskytu výkazu proti předchozím Výskytům daného výkazu, nejčastěji proti Výskytu výkazu za bezprostředně předchozí období nebo několik předchozích Výskytů daného výkazu.
* Kontroly mezi různými Výkazy za různá období – kontrolují vztahy mezi daty různých Výkazů, které mají zpravidla odlišnou periodicitu vykazování.
* Kontroly Výkazu za určité období proti všem vykázaným datům jiného Výkazu – kontrolují vztah dat zaslaného Výskytu výkazu proti všem dosud zaslaným datům jiného Výkazu nebo proti datům jiného Výkazu za specifikované historické období.

Pro seskupení mezivýkazových kontrol podle zdrojů dat, mezi nimiž jsou kontroly prováděny, obsahuje metodika výkaznictví ČNB objekt Skupina MVK, která definuje množinu Výkazů a atributy určující období, mezi nimiž jsou definovány určité kontrolní vztahy.

Jeden Výkaz ve Skupině MVK je vždy definován jako Vlastník.

Období dat Výkazů vstupujících do mezivýkazových kontrol je definováno:

* Relativně (relativní období) – Výkaz vlastník má definováno tzv. relativní období = 0 a ostatní Výkazy ve Skupině MVK se vymezují z hlediska období vzhledem vlastníkovi. Tyto Výkazy mohou mít definované tzv. relativní období 0 (shodné období s vlastníkem), -1, -2 ad. (předchozí období relativně k vlastníkovi), +1, +2 ad. (budoucí období vzhledem k vlastníkovi)
* Absolutně (absolutní období) - používají se všechna historická data nebo data za specifikované historické období (např. za poslední měsíc – absolutní období = -30).

Použití typu období na Výkazech ve Skupině MVK se odvíjí od typu Výkazu:

* Statistický – používá se relativní období.
* Číselníkový – používá se absolutní období bez vymezení období, protože se používají všechna platná data daného Výkazu bez ohledu na Výskyt výkazu.
* Transakční - používá se absolutní období s vymezením období (např. – 30), používají se platná data za Výskyty výkazů zpětně od aktuálního kalendářního dne.

Z věcného hlediska zahrnují mezivýkazové kontroly stejné typy kontrol jako jednovýkazové kontroly, které jsou prováděny nad daty různých Datových oblastí.

Mezivýkazové kontroly stejného Výkazu za různá období mohou být definovány i nad jednou Datovou oblastí.

## Sémantický a uživatelský jazyk kontrol

Základním rozdílem mezi sémantickým a uživatelským jazykem kontrol je způsob identifikace údajů Výkazů v zápisech kontrol.

Sémantický jazyk používá k identifikace údajů Výkazů kódy sémantických objektů systému SDAT:

* Ukazatel
* Parametr
* Datový typ
* Číselník
* Doména číselníku
* Položka číselníku

Výjimečně může být v sémantické zápisu použitý í kód Datové oblasti, pokud Výkaz obsahuje údaje se shodným metapopisem v různých Datových oblastech.

Uživatelský tvar používá k identifikaci údajů Výkazů kód jednoznačně definující umístění Údaje ve struktuře Výkazu:

* Kód datové oblasti
* Kód řádku
* Kód sloupce
* Kód karty (v případě kartové oblasti)

V uživatelském jazyku jsou používány sémantické objekty v některých konstrukcích (např. v konkretizacích dynamických parametrů).

V mezivýkazových kontrolách používají oba jazyky k identifikaci údajů také kódy Výkazů.

## Kontroly převzaté z externích specifikací

Vzhledem k tomu, že metodiky některých Vykazovacích rámců jsou vytvářeny podle specifikací externích institucí (EBA, EIOPA, ESMA), které obsahují i předepsané kontroly, budou se v těchto metodikách vyskytovat odchylky v předpisech kontrol od standardního přístupu ve Výkazech ČNB.

Některé formální a formátové kontroly, které jsou ve Výkazech ČNB standardně definovány explicitními pravidly nebo předepsanými atributy v metodice (povinnost vyplnění, obor hodnot), mohou být ve Výkazech těchto Vykazovacích rámců předepsány formou sémantického a uživatelského jazyka kontrol.

EBA a EIOPA používají odlišný způsob definice a vyhodnocování odchylek relačních výrazů od výkazů ČNB. Zatímco v kontrolách Výkazů ČNB je odchylka stanovena absolutní hodnotou, kontroly Výkazů EBA a EIOPA používají pro vyhodnocování odchylky tzv. intervalovou aritmetiku. Pro tento účel jsou používány specifické funkce.

# Sémantický jazyk kontrol

## Popis sémantického jazyka kontrol

### Úvod

V této kapitole jsou popsány konstrukce sémantického jazyka kontrol.

Popis je členěn do dílčích částí od základních konstrukcí až po konstrukce celkového kontrolního výrazu.

* Základní konstrukce
* Specifikace údajů a parametrů
* Hodnoty údajů a parametrů
* Proměnné
* Specifické konstrukce
* Kontrolní výraz

V popisu konstrukcí je použita následující syntaxe:

* V závorkách < > jsou neterminály.
* Do závorek [ ] jsou zapsány nepovinné konstrukce (0-1 opakování).
* Závorky { } slouží pouze pro strukturování textu.
* Znak | znamená alternativu (odděluje různá pravidla, podle kterých lze přepsat jeden neterminální symbol).

### Základní konstrukce

V této kapitole jsou popsány základní konstrukce jazyka kontrol.

#### Aritmetické operátory

Běžné operace sčítání, odčítání, násobení a dělení jsou realizovány jako infixové operátory +, -, \*, / s obvyklou prioritou. Další operace pak mohou být realizovány formou funkcí.

#### Relační operátory

Jedná se o běžné operace porovnání na rovnost a nerovnost. Jsou realizovány infixovými operátory <, <=, =, >=, >, <>. Mohou být doplněny specifikací odchylky klíčovým slovem MARGIN, uvedeným za pravým operandem, například

1 = 2 MARGIN 1.33

Odchylka musí být uvedena jako nezáporná číselná konstanta a nemůže se vyskytovat při porovnání nečíselných datových typů, například řetězců. Relace je splněna v případě, že existuje číslo v intervalu [0, MARGIN], po jehož přičtení k některému z operandů by byla splněna relace bez použití odchylky (výše uvedený příklad tak je splněná rovnost, neboť 1 + 1 = 2 a 1 < 1.33). Složitější porovnávání, například zohledňující intervalovou aritmetiku, může být realizováno funkcemi.

#### Kulaté závorky

Operátory levá a pravá závorka slouží k řízení priority provádění výpočtu, například

 ( 1 + 2) \* (3 + 4)

Kde bez uvedení závorek by bylo v souladu s prioritami operátorů vyhodnocováno jako 1 + (2 \* 3) + 4.

#### Operátor IN

Infixový operátor IN umožňuje testovat obsažení konkrétní hodnoty (levý operand) v množině hodnot (pravý operand).

#### Množina

Množinu hodnot lze zapsat pomocí složených závorek, například

 { 1, 3, 5 }

Jedná-li se o množinu položek číselníku, lze hodnotu uvodit znakem ^ ve smyslu negace, například

{ @VSEMEN, ^CZK }

Představuje všechny prvky domény VSEMEN, z nichž je následně odstraněn prvek CZK. Typy zápisu pro položky budou ještě dále vysvětleny podrobněji.

Pokud je použita pouze negace, například { ^CZK }, platí celá množina (například celý číselník) bez prvku uvedeného v negaci.

####  Identifikátory

Identifikátor jazyka může obsahovat znak podtržítko, znaky anglické abecedy a číslice s tím, že číslice nesmí být na prvním místě. Nemůže být použito klíčové slovo jazyka. Například

CZK, \_A, X01, A\_B\_C, ne ~~1A~~, ~~IN~~, ~~ČR~~

Je-li třeba použít identifikátor, který pravidla nesplňuje, uvede se do apostrofů, například

 ‘1A’, ‘ČR’, ‘S.01.01’, ‘IN’, ‘A B C’

Metodicky tato funkcionalita neslouží pro samoúčelnou volbu nestandardních identifikátorů, ale pro ošetření situací, kdy jsou identifikátory dány, a tedy se nevolí, typicky jestliže se jedná o kódy metadatových objektů nebo konstanty na vstupu sběru dat.

#### Konstanty

**Číselné konstanty** se zapisují vždy bez mezer a oddělovačů tisíců, oddělovačem desetinných míst je vždy znak tečka, před níž musí být uvedena číslice, tedy

1000, 1.33, 0.1, ne ~~1 000~~, 1,33, ~~.1~~

**Řetězcové konstanty** se zapisují do uvozovek (znak “), například

 “A“, “B C D“

Konstanty pro **logické hodnoty** jsou TRUE a FALSE, jedná se o klíčová slova jazyka.

Konstanta pro **prázdnou hodnotu** je NULL, jedná se o klíčové slovo jazyka.

#### Proměnné

Proměnné se zapisují jako znak $ následovaný identifikátorem. Přiřazení hodnoty do proměnné se provádí operátorem **:=**, například

 $A := 123, 1 > $A

#### Pragma

Pragma je zavedena za účelem specifikace doplňujících informací ke kontrole.

Aktuálně je zavedena pragma #RESTRICT, která obsahuje vymezení kontroly pro množinu osob.

Elementem množiny může být:

* Osoba – uveden kód osoby, například (0100,3000,0800)
* Typ osoby – uveden znak @ a kód Typu osoby, například (@BANCR)

Pro zápis množiny se používá množinová konstrukce popsaná v kapitole 4.1.2.5, ovšem nepoužívají se závorky { }, jelikož samotný výraz je v kulatých závorkách ().

Příklady:

#RESTRICT(0700)

#RESTRICT(0100,3000,0800)

#RESTRICT(@BANCR)

### Specifikace údajů a parametrů

V této kapitole jsou popsány konstrukce pro specifikace Údajů výkazů formou sémantických objektů – Ukazatelů, Parametrů, Položek číselníků, Domény číselníků, Datových typů.

**Základní konstrukce popisu Údaje**

**Údaj** je obecně definován Ukazatelem a množinou 0 až N Parametrů a jejich hodnot (tzv. konkretizací Parametrů).

Základní konstrukce je:

**“[“ <ukazatel> [ { , <parKonkr> } ] “]“**

 kde

* **[] –** specifikace Údaje je v hranatých závorkách
* **<ukazatel>** - kód Ukazatele
* **<pakKonkr> -** konkretizace hodnoty Parametru:
* <kodParametr> = <kodPolozkaCiselniku>
* <kodParametr> = <kodhodnotaDatovehoTypu>
* <kodParametr> = @<kodDatovyTyp>
* <kodParametr>:@<kodDomenaCiselniku>
* <kodParametr>:@<kodDatovyTyp>
* <kodParametr>:\*
* <kodParametr>:<mnozina>

Touto konstrukcí lze popsat jeden konkrétní Statický údaj, jeden Dynamický údaj nebo filtr Dynamického údaje.

Příklady:

1. Statický údaj

[EBD1681, P0009=CZK, P0011=120, P0013=200, P0014=001, P0019=S\_VSEMEN]

1. Dynamický údaj s použitím Domény číselníku

[EPD1030, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, **P0025:@D\_S\_VSEZEME2**]

1. Dynamický údaj s použitím \* v konkretizaci Dynamického parametru

[EPD1030, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, **P0025:\***]

1. Filtr dynamického údaje s použitím množin v konkretizaci Dynamických parametrů

[EPN0001\_001, P0009=CZK, P0011=440, P0013=100, P0014=400, **P0019:{@D\_S\_CZKEUR1, ^CZK**}, P0024:{15, 21}, P0065=01, P0186=23])

**Rozšíření o upřesnění Datové oblasti**

Sémantický popis Údaje zpravidla jednoznačně definuje vykazovaný Údaj v rámci Výkazu, takže není třeba uvádět Datovou oblast.

Pokud se však v rámci Výkazu vyskytuje vykazovaný Údaj se shodným sémantickým popisem ve více Datových oblastech, je zápis doplněn o specifikaci Datové oblasti.

**“ [“ <ukazatel>, \_DX=<datovaOblast> [ { , <parKonkr > } ] “]“**

kde

* **\_DX =** speciální identifikátor definující, že se jedná o Datovou oblast
* **<datovaOblast> -** kód Datové oblasti

Příklad:

1. Upřesnění Datové oblasti, ze které je specifikovaný Údaj.

NOT(ISBLANK([ei207, **\_DX="A\_00.01"**, BAS=x17]));

**Specifikace Výkazu a Relativního období v mezivýkazových kontrolách**

U mezivýkazových kontrol je specifikace Údaje vždy doplněna o specifikaci Výkazu a pokud se mezivýkazová kontrola týká Výkazů za různá období, je specifikováno také Relativní období.

Konstrukce pro zápis specifikace Údaje se specifikací Výkazu:

**<vykaz> [%<relObd>] “[“<ukazatel> [ { , <parKonkr > } ] “]“**

 kde

* **<vykaz>** - je kód Výkazu
* **% -** speciální znak indikace relativního období v zápisu
* **<relObd> -** je číslo určující relativní období Výskytu výkazu (např. -1, -2), pokud je období 0 (nula), tak se neuvádí

Příklad:

1. Mezivýkazová kontrola s určením Výkazů s defaultním Relativním obdobím 0

WITH(P0009=CZK, P0011=050, P0013=100, P0014=001)

**DRSIFE11**[EAR2491, P0019=XAU] = **RISIFE11**[EAR0090, P0019=S\_VSEMEN, P0034=6, P0186=17, P0236=S\_ESACNB] MARGIN 1000;

1. Mezivýkazová kontrola s určením Relativního období

WITH(P0009=CZK, P0013=100, P0014=410, P0186=23, P2125=CZ, P2236=S\_RMFI)

FOR(P0236:1100000, P0125:CZ)

SUM(SOTIFE61[ETO0019, P0011=450, P0021:@D\_S\_UVEVKL2, P0065:@D\_S\_UVVKL\_8]) = SOTIFE60[ETO0019, P0011=440, P0021=S\_UVEVKL, P0065=S\_UVVKL] + SOTIFE60**%-1**[ETO0019, P0011=440, P0021=S\_UVEVKL, P0065=S\_UVVKL] + SOTIFE60%-2[ETO0019, P0011=440, P0021=S\_UVEVKL, P0065=S\_UVVKL] MARGIN 3000;

**Specifikace Výkazu a Absolutního období v mezivýkazových kontrolách**

U mezivýkazových kontrol Výkazů s Absolutním obdobím je specifikace Údaje vždy doplněna o specifikaci Výkazu a od specifikace Absolutního období je abstrahováno.

Konstrukce pro zápis specifikace Údaje se specifikací Výkazu:

**<vykaz> “[“<ukazatel> [ { , <parKonkr > } ] “]“**

 kde

* **<vykaz>** - je kód Výkazu

Příklad:

1. Mezivýkazová kontrola s určením transakčních Výkazů s Absolutním obdobím

JOIN\_EXISTS(FILTER(TRAF11\_11, **TRAFIM11**[FIM0156, P0556=NEWT, P0854=BOOLEAN, P9902=TRAF11\_11, T0024=NEWT] = "true"),

 TRAF10\_11,

 AND(X->R0011 = Y->R0011,

 X->R0012 = Y->R0012));

1. Mezivýkazová kontrola s určením transakčního Výkazu s Absolutním obdobím a číselníkového Výkazu s Absolutním obdobím.

JOIN\_EXISTS(FILTER(TRAF10\_11, OR(NOT(ISBLANK(**TRAFIM10**[FIM0065, P0556=NEWT, P0854=STRING, P9902=TRAF10\_11, T0024=NEWT])),

 NOT(ISBLANK(**TRAFIM10**[FIM0066, P0556=NEWT, P0854=STRING, P9902=TRAF10\_11, T0024=NEWT])))),

 REFF10\_11,

 OR(X->FIM0065 = Y->R0101,

 X->FIM0066 = Y->R0101));

**WITH klausule**

Pro zvýšení čitelnosti kontrolního výrazu je zavedena klausule **WITH** obsahující společné Parametry pro všechny Údaje v kontrolním výrazu, které lze vytknout.

**WITH ( <parKonkr > [ { , <parKonkr > } ] )**

kde

* **WITH ()** – klíčové slovo WITH uvozuje seznam konkretizací Parametrů v () závorkách
* **<parKonkr > -** konkretizace vytknutého Parametru, která může být zapsána následujícím způsobem:
* <kodParametr> = <kodPolozkaCiselniku>
* <kodParametr> = <hodnotaDatovehoTypu>
* <kodParametr> = @<kodDatovyTyp>

Příklad:

1. Vytknuté společné konkretizace parametrů

**WITH(P0009=CZK, P0011=050, P0013=100, P0014=001)**

DRSIFE11[EAR2491, P0019=XAU] = RISIFE11[EAR0090, P0019=S\_VSEMEN, P0034=6, P0186=17, P0236=S\_ESACNB] MARGIN 1000;

**Dynamické parametry**

V kontrolách nad Datovými oblastmi s Dynamickými parametry lze samostatně používat ve výrazech i Dynamické parametry.

**“[[“ <parKonkr>“]]“**

kde

* **[[ ]] –** specifikace Dynamického parametru je ve dvojitých hranatých závorkách
* **<kodParametru> -** konkretizace hodnoty Parametru:

Příklad:

1. Výraz nad Dynamickými parametry

FILTER\_CHECK(SIS11\_01)

IF(AND(**[[P0019]] = CZK, [[P0024]] = 15)**, [EPN0001\_002, P0009=CZK, P0011=440, P0013=100, P0014=400, P0065=08, P0186=23] < 0 MARGIN 1000);

### Hodnoty údajů a parametrů

Hodnoty údajů a parametrů mají datové typy, odpovídající jejich konkretizaci v metapopisu.

Z pohledu kontrol mohou hodnoty nabývat typu:

* Číslo
* Řetězec
* Datum
* Položka číselníku
* Binární (pouze údaje)

Pro typ *binární* nejsou podporovány žádné specifické operace a reálně takový údaj může figurovat pouze ve funkcích testujících existenci hodnoty (ISBLANK, COUNT apod.).

Platí pravidlo, že datové typy musí odpovídat kontextu použití, tedy kontrola, která by například funkci SUM předávala údaj s typem hodnoty *řetězec*, není validní a nebude vůbec přeložena.

Datové typy nejsou na sebe automaticky převeditelné s výjimkou typu *položka*, pro který je definována implicitní konverze na typ *řetězec* takto:

* Má-li binární operace, umožňující různé typy operandů, na vstupu typy položka a řetězec, jsou všechny typy považovány za řetězec, umožňuje-li operace takový vstup.
* Vyskytuje-li se hodnota typu položka na místě, kde je očekávána hodnota typu řetězec, je přetypována na řetězec a hodnotou je kód položky.

Nejlépe budou pravidla patrná na konkrétních případech (předpokládejme, že konstrukce [UK0001] a [UK0002] představují údaje, nabývající hodnot číselníku BA0025):

1. Ukázka 1:

[UK0001] = “CZ”

Operace rovnosti je definována pro typy *položka* i *řetězec*. Neboť se zde vyskytují oba, je aplikováno pravidlo 1 a následně podle pravidla 2 je hodnota ukazatele interpretována jako kód položky.

1. Ukázka 2:

SUBSTR([UK0001], 1, 1) = “C”

Funkce SUBSTR je definována pouze pro typ *řetězec*. Podle pravidla 2 je tak hodnota ukazatele považována za *řetězec*, kterým je kód položky.

1. Ukázka 3:

 [UK0001] > “C”

Operace nerovnosti je definována pro typy *číslo*, *datum* a *řetězec* (pro řetězec se vztahuje k lexikografickému uspořádání). Neboť se zde vyskytují typy *položka* a *řetězec*, je podle pravidla 1 zvolen typ porovnání řetězců a podle pravidla 2 se použije kód položky. Například položka CZ pak testu vyhovuje, AU nikoliv.

1. Ukázka 4:

 [UK0001] > [UK0002]

Operace nerovnosti je definována pro typy *číslo*, *datum* a *řetězec* (pro řetězec se vztahuje k lexikografickému uspořádání). Neboť se zde vyskytuje pouze typ *položka,* je podle pravidla 2 zvolen typ porovnání řetězců a použijí se kódy položek. Například pro hodnoty CZ a AU je test splněn.

Poznámka: porovnávání položek s řetězci by se mělo používat primárně tam, kde kódy položek mají nějakou specifickou strukturu, založenou na pozicích apod.

### Proměnné

Proměnné slouží v jazyce zejména k omezení opakovaní stejných konstrukcí nebo i zpřehlednění zápisu. Dříve, než je proměnná použita, musí jí být přiřazena hodnota operátorem **:=**

**“$”<identifikátor> “:=” <výraz>“;“**

Hodnotou může být libovolný výraz, jako smysluplné se jeví zejména:

* Hodnota údaje, například je-li buňka z jiné datové oblasti než ostatní zúčastněné buňky.
* Filtr, například má-li být použit opakovaně.
* Konstanta, je-li použita opakovaně, nebo její pojmenování zvyšuje přehlednost.

Proměnná pak může být použita tam, kde se může vyskytovat výraz stejného typu (pozor ne tam, kde je vyžadována vždy konstanta, tedy např. pro odchylku nebo specifikaci relativního období nelze proměnnou použít). Proměnnou nelze použít jako „zkratku“ pro identifikátor (např. kód DO ve filtru apod.).

1. Ukázka a: použitím proměnné S1 se vyhneme problémům s nekompatibilní sadou parametrů buněk (S1 je ze statické oblasti) a zápis je přehlednější i po přeložení do uživatelského tvaru, neboť vystihuje, že z hlediska iterace přes dynamické řádky je hodnota konstantou.

$S1 := [EXP0032, P0009=CZK, P0019=CZK];

WITH(P0019=S\_ISOMENY)

FILTER\_CHECK(JIS43\_32)

IF($S1 <> 0, [EXP0038, P0009=CZK] / $S1 \* 100 = [EXP0039, P0009=950] MARGIN 2);

$S1 := JIS43\_31[1,1];

FILTER\_CHECK(JIS43\_32)

IF($S1 <> 0, JIS43\_32[1,6] / $S1 \* 100 = JIS43\_32[1,7] MARGIN 2);

1. Ukázka b: použitím proměnné pro filtr se vyhneme jeho opakovanému výpisu, což je přehlednější a nehrozí například pozdější přehlédnutí při úpravách.

$F1 := FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = CZK);

WITH (P0011=050, P0013=100, P0014=001)

SUM(FILTER\_APPLY([BCP2194],$F1)) =

SUM(FILTER\_APPLY([BCP2007, P0009=CZK],$F1)) MARGIN 2;

1. Ukázka c: možná poněkud umělý příklad

$EULER := 0.577;

[BCP2194, P0011=050] > [BCP2194, P0011=060] / $EULER;

### Specifické konstrukce

V této kapitole jsou popsány specifické konstrukce, které jsou v jazyce kontrol navrženy pro zápis některých konkrétních typů kontrol.

**FILTER**

Specifická klausule FILTER je zavedena pro definování filtru, který je tvořen množinou podmínek, který lze použít v kontrolním výrazu k vymezení množiny Dynamických instancí (řádků, sloupců, karet), pro které se má příslušná část kontrolního výrazu (funkce) provést.

Základní konstrukce je:

**FILTER ( [ <datovaOblast>] <vyraz> [ { , vyraz> } ] )**

kde

* **<datovaOblast> -** kód Datové oblasti
* **<vyraz>**  - je výraz definující filtrační podmínku Dynamických instancí, která může být definována konkretizací Dynamických parametrů nebo libovolným výrazem.

Filtr může být definován přímo v kontrolním výrazu nebo může být přiřazen proměnné.

**<promenna>:=<filtr>“;“**

Pro použití filtru v kontrolním výrazu je zavedena klausule (funkce) FILTER\_APPLY.

**FILTER\_APPLY (“ [“ <ukazatel> [ { , <parKonkr > } ] “]“, <filtr>)**

kde

* **“[“ <ukazatel> [ { , <parKonkr > } ] “]“ -** metapopis Údaje
* **<filtr>**  - filtr, která může být definován
* přímo
* použitím proměnné

V následujících příkladech jsou ukázky různých způsobů definic a jejich použití.

Příklady:

1. Filtr je definován formou proměnné a je použit pro filtraci Dynamických instancí, které vstupují do agregační funkce SUM.

**$F1:= FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = CZK);**

WITH(P0011=050, P0013=100, P0014=001)

SUM(**FILTER\_APPLY([BCP2194],$F1**)) > 0 MARGIN 2;

1. Filtr je definován formou proměnné a je použit pro filtraci Dynamických instancí, které vstupují do dvou agregačních funkcí SUM (stejný filtr je tedy použit dvakrát).

**$F1:= FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = CZK);**

WITH (P0011=050, P0013=100, P0014=001)

SUM(**FILTER\_APPLY([BCP2194],$F1)**) = SUM(**FILTER\_APPLY([BCP2007, P0009 = CZK],$F1)**) MARGIN 2;

1. Filtr je definován přímo ve výrazu, kde je použit.

WITH (P0011=050, P0013=100, P0014=001)

SUM(**FILTER\_APPLY([BCP2194],FILTER(JIS51\_01,[BCP2195, P0011=050, P0013=100, P0014=001] = CZK**)))

=

SUM(**FILTER\_APPLY([BCP2007, P0009 = CZK],FILTER(JIS51\_01,[BCP2195, P0011=050, P0013=100, P0014=001] = CZK**))) MARGIN 2;

1. Definovány jsou dva různé filtry formou proměnných.

**$F1:= FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = CZK);**

**$F2:= FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = EUR);**

WITH (P0011=050, P0013=100, P0014=001)

SUM(**FILTER\_APPLY([BCP2184],$F1)**) > SUM(**FILTER\_APPLY([BCP2184],$F2)**) MARGIN 2;

**FOR – statická iterace**

Pro opakované provedení kontrolního výrazu pro různé množiny Statických údajů, které jsou definovány kombinacemi hodnot Parametrů, je zavedena klausule FOR.

**FOR (<parKonkr > [ { , <parKonkr > } ] )**

kde

* **<pakKonkr> -** konkretizace hodnoty Parametru:
* <kodParametr> = <kodPolozkaCiselniku>
* <kodParametr> = <kodhodnotaDatovehoTypu>
* <kodParametr> = @<kodDatovyTyp>
* <kodParametr>:<mnozina>

Příklady:

1. Opakované provedení kontrolního výrazu pro jednotlivé Položky domény číselníku

WITH(P0009=CZK, P0011=280, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0063=S\_POSP1C, P0082=S\_STKLAS, P0186=95)

**FOR(P0286=@Z\_ZNEHOD)**

[ABD0569, P0236=S\_RNUVINMRB] + [ABD0577, P0236=S\_RNBUIMRB] = [ABD0568, P0236=S\_RNBCB] MARGIN 2000;

1. Opakované provedení kontrolního výrazu pro jednotlivé Položky číselníku definované výčtem

WITH(P0009=CZK, P0011=050, P0013=100, P0014=001, P0019 = CZK, P0034 = 6, P0186=23)

**FOR (P0236 = { 1000000, 2000000} )**

[EAR1398] = 0 MARGIN 1000 ;

1. Opakované provedení kontrolního výrazu pro jednotlivé Položky číselníku definované množinou s použitím Domény číselníku a negativního výčtu Položek číselníku

WITH(P0009=CZK, P0011=050, P0013=100, P0014=001, P0019 = CZK, P0034 = 6, P0186=23)

**FOR (P0236 = {@Z\_CRZNRZ,^S\_ESACNB})**

[EAR1398] = 0 MARGIN 1000 ;

**FOR – dynamická iterace**

Klausule FOR se používá i pro opakované provedení kontrolního výrazu pro kombinace hodnot Dynamických parametrů.

**FOR (<parKonkr > [ { , <parKonkr > } ] )**

* **<pakKonkr> -** konkretizace hodnoty Parametru:
* <kodParametr> = <kodPolozkaCiselniku>
* <kodParametr> = <kodhodnotaDatovehoTypu>
* <kodParametr>:@<kodDomenaCiselniku>
* <kodParametr>:@<kodDatovyTyp>
* <kodParametr>:\*
* <kodParametr>:<mnozina>

Příklady:

1. Opakované provedení kontrolního výrazu pro vykázané Dynamické řádky v rámci Datové oblasti

WITH(P0009=CZK, P0011=440, P0013=100, P0014=400, P0186=23, P0912=N)

**FOR(P0019:@D\_S\_3MENY1, P0021:@Z\_POTUV2, P0901:@D\_S\_AN1)**

SUM([EAN0001\_001, P0117:@p3\_2\_I1, P0236=S\_RZOBY]) = SUM([EAN0002\_009, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=S\_FLDO3, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_010, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=07, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_011, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=67, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_012, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=68, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_013, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=44, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_014, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=S\_NAD10LET, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) MARGIN 5000;

1. Opakované provedení kontrolního výrazu pro množinu vykázaných kombinací hodnot Dynamických parametrů v rámci různých Datových oblastí (RIS61\_01 a SIS01\_01)

WITH(P0009=CZK, P0011=050, P0013=100, P0014=001, P0065=98, P0186=23)

**FOR(P0019:CZK, P0024:@D\_S\_VKLADUV2, P0236:@Z\_NEDOS)**

SUM(SISIFE01[EPR1383\_001, P0125=CZ, P0151=S\_NACE, P1117:@pS3\_2\_I1]) = SUM(RISIFE61[EPR1383\_001, P0125:CZ, P0151:@D\_S\_NACE\_1, P1117=@pS3\_2\_I1]) MARGIN 5000;

**FILTER\_CHECK**

Pro opakované provedení kontrolního výrazu nad jednotlivými dynamickými instancemi (řádky, sloupci, kartami) jedné Datové oblasti je zavedena klausule FILTER\_CHECK.

Obecný tvar konstrukce FILTER\_CHECK je následující:

**FILTER\_CHECK ( <DatovaOblast> [ { , <filtr> } ])**

kde

* **<DatovaOblast>** - kód Datové oblasti
* **<filtr>** - je výraz definující filtrační podmínku Dynamických instancí, která může být definována konkretizací Dynamických parametrů nebo libovolným výrazem.

Dále jsou popsány jednotlivé případy.

Pokud se kontrolní výraz provádí pro všechny vykázané dynamické instance, má klausule tvar:

**FILTER\_CHECK (<DatovaOblast>)**

kde

* **<DatovaOblast>** - je kód Datové oblasti

Příklad:

1. Opakované provedení kontrolního výrazu pro všechny zaslané Dynamické instance (řádky, sloupce, karty)

WITH(P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0186=21)

**FILTER\_CHECK(DIS85\_03)**

[PBD0203] = [PBD0204] + [PBD0205] + [PBD0206] MARGIN 3000;

Pokud se kontrolní výraz provádí pro podmnožinu dynamických instancí (řádků, sloupců, karet) vymezenou konkretizací Dynamických parametrů, má klausule tvar:

**FILTER\_CHECK (<DatovaOblast> <parKonkr > [ { , <parKonkr > } ]**

kde

* **<DatovaOblast>** - je kód Datové oblasti
* **<parKonkr> -** konkretizace Dynamických parametrů dané Datové oblasti, podle kterých se provádí filtrace dynamických záznamů
* <kodParametr> = <kodPolozkaCiselniku>
* <kodParametr> = <kodhodnotaDatovehoTypu>
* <kodParametr>:@<kodDomenaCiselniku>
* <kodParametr>:@<kodDatovyTyp>
* <kodParametr>:\*
* <kodParametr>:<mnozina>

Příklad:

1. Opakované provedení kontrolního výrazu pouze pro všechny zaslané Dynamické instance (řádky, sloupce, karty), které obsahují definovanou konkretizací Dynamického parametru

WITH(P0011=120, P0013=100, P0014=001)

**FILTER\_CHECK(DIS60\_07, P0025:CZ)**

AND(ISBLANK([EVD0131]),

 ISBLANK([EVD0132]),

 ISBLANK([EVD0133]),

 ISBLANK([EVD0134]),

 ISBLANK([EVD0135]));

Vymezení dynamických instancí (řádků, sloupců, karet) nemusí být dáno pouze konkretizace Dynamických parametrů, ale složitějším výrazem nebo množinou výrazů

**FILTER\_CHECK (<DatovaOblast> <vyraz > [ { , <vyraz > } ]**

kde

* **<DatovaOblast>** - je kód Datové oblasti
* **<vyraz>** - jakýkoliv výraz definující vymezení Dynamických instancí (řádků, sloupců, karet)

Příklady:

1. Opakované provedení kontrolního výrazu pouze pro všechny zaslané Dynamické instance (řádky, sloupce, karty), které vyhovují definovanému výrazu

WITH(P0011=050, P0013=100, P0014=001)

**FILTER\_CHECK(JIS51\_01, [BCP2195] = CZK)**

[BCP2007, P0009=CZK] = [BCP2194];

**FILTER\_FAIL**

V dynamických Datových oblastech se často vyskytují kontroly definující tzv. nepovolené kombinace Dynamických parametrů, tzn. kombinace hodnot, které nemohou být zaslány.

Pro zápis nepovolených kombinací je zavedena specifická konstrukce FILTER\_FAIL,

**FILTER\_FAIL( <DatovaOblast> , <parKonkr > <parKonkr > [ { , <parKonkr > } ] )**

kde

* **<DatovaOblast>**- kód Datové oblasti
* **<parKonkr > <parKonkr > [ { , <parKonkr > } ] -** specifikace nepovolených kombinací hodnot Dynamických parametrů
* <kodParametr> = <kodPolozkaCiselniku>
* <kodParametr> = <kodhodnotaDatovehoTypu>
* <kodParametr> = @<kodDatovyTyp>
* <kodParametr>:<mnozina>

Příklad:

1. Nepovolené kombinace hodnot Dynamických parametrů

FILTER\_FAIL(RIS22\_01, P0236:@Z\_REZBVD, P0125:@ZBEZCZ4);

### Typy RowSet, Record a operace dereference

#### Interní typ Record

Typ Record představuje strukturu, obsahující pojmenované atributy. Může jím být například řádek dynamického výkazu, ale i nějaký externí zdroj, případně může být z takových zdrojů odvozen tak, že odpovídá jednomu záznamu v nějakém (odvozeném) RowSet typu. Typ je interní v tom smyslu, že není používán explicitně a nabývají ho pouze specifické konstrukce, související s typem RowSet.

#### Typ RowSet

Typ RowSet představuje množinu záznamů typu Record. Může jím být například množina řádků dynamické datové oblasti (bez součtového řádku), externí registr, nebo může být z takových zdrojů odvozen použitím odpovídajících funkcí. Tohoto typu nabývají specifické funkce, popsané dále.

#### Operace dereference

Operace dereference „->“ slouží k odkazování atributů v typu Record. Jedná se o binární infix operátor, na jehož levé straně se vyskytuje identifikátor, odkazující k typu RowSet, a na pravé identifikátor atributu, například

X -> EPR1383\_003

může, při použití ve správném kontextu, kde X je definováno jako datová oblast SIS01\_01 odkazovat na hodnotu ukazatele EPR1383\_003 v řádku této dynamické datové oblasti. Levou a pravou stranu operátoru nadále označujme jako *zdroj* a *atribut*, nebude-li uvedeno jinak. Tuto operaci lze použít pouze ve funkcích typu JOIN.

#### Identifikace zdroje a atributu ve funkcích typu JOIN

Pro funkce typu JOIN, popsané detailněji níže, platí následující pravidla pro specifikaci zdroje a atributu operace dereference

* Zdroje jsou specifikovány jako první dva parametry funkce tak, že:
* Je-li zdrojem datová oblast, je uveden její kód, který může být v případě mezivýkazové kontroly doplněn o vymezení relativního období.
* Je-li zdrojem filtr nad datovou oblastí nebo odvozený RowSet, je uvedena proměnná, která ho obsahuje.
* V dereferenci jsou zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.
* Atributy jsou identifikovány:
* Kódem parametru nebo ukazatele v případě, že zdrojem je datová oblast nebo nad ní definovaný filtr a kódy jsou v této oblasti unikátním identifikátorem (tedy není v řádku datové oblasti použit stejný ukazatel dvakrát a nedochází ke shodě kódů ukazatelů a parametrů).
* Základní konstrukcí popisu Údaje v případě, že kódy nejsou unikátním identifikátorem.
* Identifikátorem specifickým pro externí registr, je-li zdrojem externí registr.
* Poziční identifikací \_1, \_2 atd. v případě, že zdrojem je odvozený RowSet (např. výstup funkce JOIN\_FETCH).

#### Funkce pracující s RowSet typy

**JOIN-CHECK**

Funkce JOIN\_CHECK provádí opakovaně kontrolní výraz mezi odpovídajícími si záznamy dvou dynamických datových zdrojů.

Základní konstrukce je:

**JOIN\_CHECK ( <rowSet1>, <rowSet2>, <vyraz>, <vyraz2>);**

kde

* **<rowSet1t> -** kód prvního datového zdroje (např. datové oblasti, externího číselníku)
* **<rowSet2t> -** kód druhého datového zdroje (např. datové oblasti, externího číselníku)
* **<vyraz>**  - kontrolní výraz včetně definice relačního vztahu mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj za účelem zkrácení zápisu.
* **<vyraz2>** - kontrolní výraz na záznamy splňujícími podmínky definované ve <vyraz>.

Příklady:

**Kontrola EIOPA**

Popis kontroly od EIOPA

{c0160} = empty

FILTER: {c0290} like '##7#' or {c0290} like '##8#'

Sémantický zápis

JOIN\_CHECK('SE.06.02.16.01',

 'SE.06.02.16.02',

 AND(REGEXP\_LIKE( Y->si1554, "..7.|..8."),

 Y->UI = X->UI),

 ISBLANK(X->mi1987));

Kontrola se provádí mezi záznamy dvou datových oblastí SE.06.02.16.01 a SE.06.02.16.02, které jsou relačně propojeny přes dynamický parametr UI.

Kontrola se provádí pro záznamy, pokud údaj si1554 z SE.06.02.16.02 vyhovuje definované podmínce podle regulárního výrazu.

Vlastní kontrola vyhodnocuje nevyplnění hodnoty údaje mi1987 v SE.06.02.16.01.

**Kontrola MKT**

Popis kontroly

Je-li ve výkazu TRAFIM10 údaj Identifikační kód nástroje 1 (TRAF10\_11.FIM0065) nebo údaj Identifikační kód nástroje 2 (TRAF10\_11.FIM0066) vyplněný, potom musí být údaje Identifikace investičního nástroje (TRAF11\_11.FIM0151, TRAF11\_11.FIM0152, TRAF11\_11.FIM0153) prázdné.

Sémantický zápis

JOIN\_CHECK(TRAF10\_11,

 TRAF11\_11,

 AND(OR(NOT(ISBLANK( X->FIM0065)), NOT(ISBLANK( X->FIM0066))),

 X->R0011 = Y->R0011,

 X->R0012 = Y->R0012),

 AND( ISBLANK(Y->FIM0151),

 ISBLANK(Y->FIM0152),

 ISBLANK(Y->FIM0153)));

Kontrola se provádí mezi záznamy dvou datových oblastí TRAF10\_11 a TRAF11\_11, které jsou relačně propojeny přes dva dynamické parametry R0011 a R0012.

Kontroluje se vyplnění/nevyplnění hodnot údajů vzájemně propojených záznamů daných datových oblastí dle definované podmínky.

**Kontrola AnaCredit**

Popis kontroly

IF [PANACR02.ANA0001] = '06' THEN [PANACR03.ANA0039] = 'NTAP'

Sémantický zápis

JOIN\_CHECK(PAN02\_11,

 PAN03\_11,

 AND(X->ANA0001="06",

 X->R0033 = Y->R0033,

 X->R0030 = Y->R0030,

 X->R0031 = Y->R0031),

 REGEXP\_LIKE(Y->ANA0039, "NTAP"));

Kontrola se provádí mezi záznamy dvou datových oblastí PAN02\_11 a PAN03\_11, které jsou relačně propojeny přes tři dynamické parametry R0033, R0030 a R0031.

Kontroluje se správná kombinace vyplnění hodnot údajů ANA0001 a ANA0039 vzájemně propojených záznamů daných datových oblastí.

**JOIN\_FETCH**

Funkce JOIN\_FETCH definuje odvozený datový zdroj typu RowSet, který je dále používán v kontrolním výrazu. Nad tímto odvozeným datovým zdrojem mohou být prováděny agregační funkce nebo mohou být používány jednotlivé záznamy.

Základní konstrukce je:

**<promenna> “:=” JOIN\_FETCH ( <rowSet1>, <rowSet2>, <vyraz>,<vyraz2> [ { , vyraz2> } ]);**

**<telo>;**

kde

* <promenna> - kód proměnné
* **<rowSet1t> -** kód prvního datového zdroje (např. datové oblasti, externího číselníku)
* **<rowSet2t> -** kód druhého datového zdroje (např. datové oblasti, externího číselníku)
* **<vyraz>**  - je výraz definující odvozený RowSet včetně definice relačního vztahu mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.
* **<vyraz2>** - definice odvozeného datového zdroje typu RowSet pomocí dereference.
* **<telo> -** vlastní kontrolní výraz, který používá RowSet definovaný proměnnou.

Příklady:

**Kontrola EIOPA**

Popis kontroly

{S.02.01, r0080,c0010}=sum({S.06.02, c0170,(sNNN)})

FILTER:

({S.06.02, c0290} like ‘##91’ OR {S.06.02, c0290} like ‘##92’ OR {S.06.02, c0290} like ‘##94’ OR {S.06.02, c0290} like ‘##99’) AND ({S.06.02, c0090}=[s2c\_LB:x91])

Sémantický zápis

$XX:=JOIN\_FETCH('S.06.02.01.01',

 'S.06.02.01.02',

 AND(AND(REGEXP\_LIKE( Y->si1554, "..91|..92|..94|..99"), X->ei1070 = "x91"),

 X->UI = Y->UI),

 X->mi1131);

IAF\_EQ (SUM(XX->\_1), [mi265, \_DX="S.02.01.01.01", BL=x91, VG=x80]);

Proměnná definuje odvozený RowSet obsahující údaj mi1131 z datové oblasti S.06.02.01.01, přičemž množina záznamů je vybrána na základě filtrační podmínky údaje si1554 z 'S.06.02.01.02 nad množinou propojených záznamů daných datových oblastí přes dynamický parametr UI.

Za tento RowSet obsahující množinu hodnot údaje mi1131 se provádí suma, která se porovnává vůči hodnotě statického údaje z jiné datové oblasti.

**Kontrola MKT**

Popis kontroly

Obsahuje-li hodnota údaje Kupující - LEI kód nabyvatele anebo ústřední protistrany ve výkazu TRAFIM20 (TRAF20\_21.FIM0004) předepsaný dummy LEI kód, potom údaj Kupující - IČ nabyvatele (TRAF11\_11.FIM0410) musí být vyplněný a musí být nalezen v rámci množiny osob zaslaných výkazem PERFIM30 - kontrola oproti hodnotám údaje (PERF30\_11.RFD0003).

Sémantický zápis

$XX:=JOIN\_FETCH(TRAF20\_21,

 TRAF11\_11,

 AND(REGEXP\_LIKE(X->FIM0004, "DUMMY……."),

 X->R0011 = Y->R0011,

 X->R0012 = Y->R0012),

 Y->FIM0410);

JOIN\_EXISTS($XX,

 PERF30\_11,

 AND(NOT(ISBLANK(X->\_1)),

 X->\_1 = Y->RFD0003));

Proměnná definuje odvozený RowSet obsahující údaj FIM0410 z TRAF11\_11, přičemž množina záznamů je vybrána na základě filtrační podmínky nad množinou propojených záznamů daných datových oblastí TRAF20\_21 a TRAF11\_11 přes dva dynamické parametry R0011, R0012.

Záznamy tohoto RowSetu se dále používají v kontrolním výrazu proti výkazu PERFIM30.

**Kontrola AnaCredit**

Popis kontroly

[PANACR07.ANA0063] DOES NOT EXIST IN {[PANACR04.R0034] | ([PANACR08.R0033], [PANACR08.R0030], [PANACR08.R0031] = [PANACR04.R0033], [PANACR04.R0030], [PANACR04.R0031]) AND [PANACR04.P1350] = '1'}

Sémantický zápis

$XX:=JOIN\_FETCH(PAN04\_11,

 PAN08\_11,

 AND(X->P1350="1",

 X->R0033 = Y->R0033,

 X->R0030 = Y->R0030,

 X->R0031 = Y->R0031),

 X->R0033,

 X->R0034);

JOIN\_CHECK($XX,

 PAN07\_11,

 AND(X->\_1 = Y->R0033),

 NOT(Y->ANA0063 = X->\_2));

Proměnná definuje odvozený RowSet obsahující údaje R0033 a R0034 z  datové oblasti PAN04\_11, přičemž množina záznamů je vybrána na základě filtrační podmínky nad množinou propojených záznamů dvou datových oblastí přes tři dynamické parametry R0033, R0030, R0031.

Záznamy tohoto RowSetu se dále používají v kontrolním výrazu.

**JOIN\_EXISTS**

Funkce JOIN\_EXISTS ověřuje existenci vzájemně si odpovídajících záznamů ze dvou datových zdrojů, splňujících definovanou podmínku.

Základní konstrukce je:

**JOIN\_EXISTS ( <rowSet1>, <rowSet2>, <vyraz> );**

kde

* **<rowSet1t> -** kód prvního datového zdroje (např. datové oblasti, externího číselníku)
* **<rowSet2t> -** kód druhého datového zdroje (např. datové oblasti, externího číselníku)
* **<vyraz>**  - je výraz definující kontrolní výraz včetně definice relačního vztahu mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.

Příklady:

**Kontrola MKT**

Popis kontroly

Množina hodnot údajů Identifikace investičního nástroje (TRAF30\_11.FIM0255, TRAF30\_11.FIM0307, TRAF30\_11.FIM0308, TRAF30\_11.FIM0309) musí být nalezena v rámci množiny nástrojů zaslaných výkazem REFFIM10 - kontrola oproti množině hodnot údajů (REFF10\_11.RFD0033, REFF10\_11.RFD0034, REFF10\_11.RFD0035, REFF10\_11.RFD0036).

Sémantický zápis

JOIN\_EXISTS(TRAF30\_11, REFF10\_11,

 AND(X->FIM0255 = Y->RFD0033,

 X->FIM0307 = Y->RFD0034,

 X->FIM0308 = Y->RFD0035,

 X->FIM0309 = Y->RFD0036));

Kontrola se provádí mezi záznamy dvou datových oblastí TRAF30\_11 a REFF10\_11, které jsou relačně propojeny přes čtyři prvky klíče, kterými jsou dynamické parametry a údaje.

Kontroluje se existence vzájemně si odpovídajících záznamů.

**Kontrola AnaCredit**

Popis kontroly

([PANACR02.R0033], [PANACR02.R0030], [PANACR02.R0031]) EXISTS IN {([PANACR03.R0033], [PANACR03.R0030], [PANACR03.R0031])}

Sémantický zápis

JOIN\_EXISTS(PAN02\_11,

 PAN03\_11,

 AND(X->R0033 = Y->R0033,

 X->R0030 = Y->R0030,

 X->R0031 = Y->R0031));

Kontrola se provádí mezi záznamy dvou datových oblastí PAN02\_11 a PAN03\_11, které jsou relačně propojeny přes tři dynamické parametry R0033, R0030 a R0031.

Kontroluje se existence vzájemně si odpovídajících záznamů, v tomto případě bez další podmínky.

### Kontrolní výraz

Kontrolní výraz má následující hlavní strukturu

**[ { <pragma> } ]**

**[ { <promenna> } ]**

**[ <with-klausule> ]**

**<telo>**

kde

* **<pragma>** - obsahuje doplňující informace ke kontrole
* **<promenna> -** definice proměnné použité v kontrolním výrazu
* **<with-klausule>** - obsahuje vytknuté společné Parametry Údajů
* **<telo>** - obsahuje vlastní kontrolní výraz, který může být definovaný následujícími způsoby:
* <vyraz>
* FILTER\_CHECK <vyraz> – provádí se pro množinu dynamických záznamů vyhovujících definovanému filtru.
* FOR <vyraz> – provádí se opakovaně pro množiny Údajů, které jsou sestaveny z kombinací konkretizací specifikovaných ve FOR klausuli a těle kontrolního výrazu.

Příklady:

1. Kontrola s pragmou RESTRICT

**#RESTRICT(@BANCR)**

WITH(P0011=120, P0013=200, P0014=001, P0019=S\_VSEMEN)

IF(AND([EBD1681, P0009=CZK] > 0, [EBD1682, P0009=CZK] > 0), [EBD1688, P0009=950] = 0);

# Uživatelský jazyk kontrol

## Popis uživatelského jazyka kontrol

V této kapitole jsou popsány hlavní konstrukce uživatelského jazyka kontrol.

### Úvod

V této kapitole jsou popsány konstrukce uživatelského jazyka kontrol.

Popis je členěn do dílčích částí od základních konstrukcí až po konstrukce celkového kontrolního výrazu.

* Základní konstrukce
* Specifikace údajů a parametrů
* Hodnoty údajů a parametrů
* Proměnné
* Specifické konstrukce
* Kontrolní výraz

### Základní konstrukce

Základní konstrukce jazyka kontrol jsou popsány ve stejnojmenné kapitole sémantického jazyka kontrol.

Tyto konstrukce platí i pro uživatelský jazyk kontrol.

### Specifikace údajů a parametrů

V této kapitole jsou popsány konstrukce pro specifikace Údajů výkazů formou poziční identifikátorů řádků, sloupců, karet Datových oblastí.

**Základní konstrukce popisu Údaje**

**Údaj** je v poziční formě obecně definován Datovou oblastí, řádkem, sloupcem a případně kartou.

Základní konstrukce je:

 **<DatovaOblast> “[“ <radek>,<sloupec> [ <karta> ] “]“**

 kde

* **<DatovaOblast>** - kód Ukazatele
* **[] –** specifikace Údaje je v hranatých závorkách
* **<radek> -** kód řádku
* **<sloupec> -** kód sloupce
* **<karta> -** kód karty

Příklady:

1. Statický údaj

DIS23\_21[1,1]

1. Dynamický údaj

DIS16\_03[2,5]

1. Údaj karty

C\_07.00.a[010,040,001]

**Specifikace Výkazu a Relativního období v mezivýkazových kontrolách**

U mezivýkazových kontrol je specifikace Údaje vždy doplněna o specifikaci Výkazu a pokud se mezivýkazové kontrola týká Výkazů za různá období, je specifikováno také Relativní období.

Konstrukce pro zápis specifikace Údaje se specifikací Výkazu:

**<vykaz> [%<relObd>]: <DatovaOblast> “[“ <radek>,<sloupec> [ <karta> ] “]“**

 kde

* **<vykaz>** - je kód Výkazu
* **% -** speciální znak indikace relativního období v zápisu
* **<relObd> -** je číslo určující relativní období Výskytu výkazu (např. -1, -2), pokud je období 0 (nula), tak se neuvádí

Příklad:

1. Mezivýkazová kontrola s určením Výkazů s defaultním Relativním obdobím 0

**DRSIFE11:**RIS11\_30[1,1] = **RISIFE11:**RIS11\_21[53,1] MARGIN 1000;

1. Mezivýkazová kontrola s určením Relativního období

FOR(P0236:1100000, P0125:CZ)

SUM(SOTIFE61:SOT61\_01[1,5]) = SOTIFE60:SOT60\_01[1,3] + SOTIFE60**%-1**:SOT60\_01[1,3] + SOTIFE60%-2:SOT60\_01[1,3] MARGIN 3000;

**Specifikace Výkazu a Absolutního období v mezivýkazových kontrolách**

U mezivýkazových kontrol Výkazů s Absolutním obdobím je specifikace Údaje vždy doplněna o specifikaci Výkazu a od specifikace Absolutního období je abstrahováno.

Konstrukce pro zápis specifikace Údaje se specifikací Výkazu:

**<vykaz>:<DatovaOblast> “[“ <radek>,<sloupec> [ <karta> ] “]“**

 kde

* **<vykaz>** - je kód Výkazu

Příklad:

1. Mezivýkazová kontrola s určením Výkazů s Absolutním obdobím.

JOIN\_EXISTS(FILTER(TRAF11\_11, **TRAFIM11:**TRAF11\_11[1,24] = "true"),

 TRAF10\_11,

 AND(X->2 = Y->2,

 X->3 = Y->3));

1. Mezivýkazová kontrola s určením transakčního Výkazu s Absolutním obdobím a číselníkového výkazu s Absolutním obdobím.

JOIN\_EXISTS(FILTER(TRAF10\_11, OR(NOT(ISBLANK(**TRAFIM10:**TRAF10\_11[1,33])), NOT(ISBLANK(**TRAFIM10:**TRAF10\_11[1,34])))),

 REFF10\_11,

 OR(X->33 = Y->2,

 X->34 = Y->2));

**Dynamické parametry**

V kontrolách nad Datovými oblastmi s Dynamickými parametry lze samostatně používat ve výrazech i Dynamické parametry.

Základní konstrukce je stejná jako v případě specifikace Údaje:

 **<DatovaOblast> “[“ <radek>,<sloupec> [ <karta> ] “]“**

 kde

* **<DatovaOblast>** - kód Ukazatele
* **[] –** specifikace Údaje je v hranatých závorkách
* **<radek> -** kód řádku
* **<sloupec> -** kód sloupce
* **<karta> -** kód karty

Příklad:

1. Výraz nad Dynamickými parametry

FILTER\_CHECK(SIS11\_01)

IF(AND(**SIS11\_01[2,1]** = CZK, **SIS11\_01[2,2]** = 15),

 SIS11\_01[2,6] < 0 MARGIN 1000);

### Hodnoty údajů a parametrů

Hodnoty údajů a parametrů jsou popsány ve stejnojmenné kapitole sémantického jazyka kontrol.

Tyto konstrukce platí i pro uživatelský jazyk kontrol.

#### Implicitní konverze

Pokud je v relaci testu na rovnost/nerovnost (tedy operátory >, <, >=, <=, =, <>) řetězec a číslo, číslo se převede na řetězec v dekadickém zápisu a porovná se řetězcově.

Poznámka: Tato implicitní konverze se používá v dosavadním systému MtS, existují kontroly, které s ní počítají, a je tedy převzata do SDAT.

### Proměnné

Proměnné jazyka kontrol jsou popsány ve stejnojmenné kapitole sémantického jazyka kontrol.

Tyto konstrukce platí i pro uživatelský jazyk kontrol.

### Specifické konstrukce

V této kapitole je popsáno, jak jsou specifické konstrukce sémantického jazyka kontrol transformovány do uživatelského jazyka kontrol.

**FILTER**

Specifická klausule FILTER je zavedena pro definování filtru, který je tvořen množinou podmínek, které lze použít v kontrolním výrazu k vymezení množiny Dynamických instancí (řádků, sloupců, karet), pro které se má příslušná část kontrolního výrazu (funkce) provést.

Základní konstrukce je:

**FILTER ( [ <datovaOblast>] <vyraz> [ { , vyraz> } ] )**

kde

* **<datovaOblast> -** kód Datové oblasti
* **<vyraz>** - je výraz definující filtrační podmínku Dynamických instancí, která může být definována konkretizací Dynamických parametrů nebo libovolným výrazem.

Filtr může být definován přímo v kontrolním výrazu nebo může být přiřazen proměnné.

**<promenna>:=<filtr>“;“**

Pro použití filtru v kontrolním výrazu je v sémantickém jazyku zavedena klausule (funkce) FILTER\_APPLY, která je z uživatelského zápisu vypuštěna.

**FILTER\_APPLY (“ [“ <ukazatel> [ { , <parKonkr > } ] “]“, <filtr>)**

kde

* **“[“ <ukazatel> [ { , <parKonkr > } ] “]“ -** metapopis Údaje
* **<filtr>**  - filtr, která může být definován
* přímo
* použitím proměnné

V následujících příkladech jsou ukázky různých způsobů definic a jejich použití.

Příklady:

1. Filtr je definován formou proměnné a je použit pro filtraci Dynamických instancí, které vstupují do agregační funkce SUM.

Sémantický zápis

**$F1:= FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = CZK);**

WITH(P0011=050, P0013=100, P0014=001)

SUM(**FILTER\_APPLY([BCP2194],$F1**)) > 0 MARGIN 2;

Uživatelský zápis

**$F1 := FILTER(JIS51\_01[1,34] = CZK);**

SUM(**JIS51\_01[1,33, $F1]**) > 0 MARGIN 2;

1. Filtr je definován formou proměnné a je použit pro filtraci Dynamických instancí, které vstupují do dvou agregačních funkcí SUM (stejný filtr je tedy použit dvakrát).

Sémantický zápis

**$F1:= FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = CZK);**

WITH (P0011=050, P0013=100, P0014=001)

SUM(**FILTER\_APPLY([BCP2194],$F1)**) = SUM(**FILTER\_APPLY([BCP2007, P0009 = CZK],$F1)**) MARGIN 2;

Uživatelský zápis

**$F1 := FILTER(JIS51\_01[1,34] = CZK);**

SUM(**JIS51\_01[1,33, $F1]**) = SUM(**JIS51\_01[1,32, $F1]**) MARGIN 2;

1. Filtr je definován přímo ve výrazu, kde je použit.

Sémantický zápis

WITH (P0011=050, P0013=100, P0014=001)

SUM(**FILTER\_APPLY([BCP2194],FILTER(JIS51\_01,[BCP2195, P0011=050, P0013=100, P0014=001] = CZK**)))

=

SUM(**FILTER\_APPLY([BCP2007, P0009 = CZK],FILTER(JIS51\_01,[BCP2195, P0011=050, P0013=100, P0014=001] = CZK**))) MARGIN 2;

Uživatelský zápis

SUM(**JIS51\_01[1,33, FILTER(JIS51\_01, JIS51\_01[1,34] = CZK)]**) = SUM(**JIS51\_01[1,32, FILTER(JIS51\_01, JIS51\_01[1,34] = CZK)]**) MARGIN 2;

1. Definovány jsou dva různé filtry formou proměnných.

Sémantický zápis

**$F1:= FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = CZK);**

**$F2:= FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = EUR);**

WITH (P0011=050, P0013=100, P0014=001)

SUM(**FILTER\_APPLY([BCP2184],$F1)**) > SUM(**FILTER\_APPLY([BCP2184],$F2)**) MARGIN 2;

Uživatelský zápis

**$F1 := FILTER(JIS51\_01[1,34] = CZK);**

**$F2 := FILTER(JIS51\_01[1,34] = EUR);**

SUM(J**IS51\_01[1,30, $F1]**) > SUM(**JIS51\_01[1,30, $F2]**) MARGIN 2;

**FOR – statická iterace**

Opakované provedení kontrolního výrazu pro různé množiny Statických údajů pomocí klausule FOR je v uživatelském jazyku zapsáno logickou funkcí AND.

Příklady:

1. Opakované provedení kontrolního výrazu pro jednotlivé Položky domény číselníku

Sémantický zápis

WITH(P0009=CZK, P0011=280, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0063=S\_POSP1C, P0082=S\_STKLAS, P0186=95)

**FOR(P0286=@Z\_ZNEHOD)**

[ABD0569, P0236=S\_RNUVINMRB] + [ABD0577, P0236=S\_RNBUIMRB] = [ABD0568, P0236=S\_RNBCB] MARGIN 2000;

Uživatelský zápis

**AND**(DIS40\_06[2,1] + DIS40\_06[10,1] = DIS40\_06[1,1] MARGIN 2000,

 DIS40\_06[2,2] + DIS40\_06[10,2] = DIS40\_06[1,2] MARGIN 2000,

 DIS40\_06[2,3] + DIS40\_06[10,3] = DIS40\_06[1,3] MARGIN 2000,

 DIS40\_06[2,4] + DIS40\_06[10,4] = DIS40\_06[1,4] MARGIN 2000,

 DIS40\_06[2,5] + DIS40\_06[10,5] = DIS40\_06[1,5] MARGIN 2000,

 DIS40\_06[2,6] + DIS40\_06[10,6] = DIS40\_06[1,6] MARGIN 2000);

1. Opakované provedení kontrolního výrazu pro jednotlivé Položky číselníku definované výčtem

Sémantický zápis

WITH(P0009=CZK, P0011=050, P0013=100, P0014=001, P0019 = CZK, P0034 = 6, P0186=23)

**FOR (P0236 = { 1000000, 2000000} )**

[EAR1398] = 0 MARGIN 1000 ;

Uživatelský zápis

**AND**(RIS11\_21[5,2] = 0 MARGIN 1000,

 RIS11\_21[5,3] = 0 MARGIN 1000);

1. Opakované provedení kontrolního výrazu pro jednotlivé Položky číselníku definované množinou s použitím Domény číselníku a negativního výčtu Položek číselníku

Sémantický zápis

WITH(P0009=CZK, P0011=050, P0013=100, P0014=001, P0019 = CZK, P0034 = 6, P0186=23)

**FOR (P0236 = {@Z\_CRZNRZ,^S\_ESACNB})**

[EAR1398] = 0 MARGIN 1000 ;

Uživatelský zápis

**AND**(RIS11\_21[5,2] = 0 MARGIN 1000,

 RIS11\_21[5,3] = 0 MARGIN 1000);

**FOR – dynamická iterace**

Pro opakované provedení kontrolního výrazu pro hodnoty Dynamických parametrů je klausule FOR v uživatelském zápisu shodná se sémantickým zápisem.

**FOR (<parKonkr > [ { , <parKonkr > } ] )**

* **<pakKonkr> -** konkretizace hodnoty Parametru:
* <kodParametr> = <kodPolozkaCiselniku>
* <kodParametr> = <kodhodnotaDatovehoTypu>
* <kodParametr>:@<kodDomenaCiselniku>
* <kodParametr>:@<kodDatovyTyp>
* <kodParametr>:\*
* <kodParametr>:<mnozina>

Příklady:

1. Opakované provedení kontrolního výrazu pro vykázané Dynamické řádky v rámci Datové oblasti

Sémantický zápis

WITH(P0009=CZK, P0011=440, P0013=100, P0014=400, P0186=23, P0912=N)

**FOR(P0019:@D\_S\_3MENY1, P0021:@Z\_POTUV2, P0901:@D\_S\_AN1)**

SUM([EAN0001\_001, P0117:@p3\_2\_I1, P0236=S\_RZOBY]) = SUM([EAN0002\_009, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=S\_FLDO3, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_010, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=07, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_011, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=67, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_012, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=68, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_013, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=44, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) + SUM([EAN0002\_014, P0065:@D\_S\_UVVKL\_7, P0083:@D\_S\_PRAVOS\_2, P0122=S\_NAD10LET, P0236:S\_RZOBY, P0251:@D\_S\_KATUVOB\_1, P0913:@D\_S\_AN1, P1117:@pS3\_2\_I1]) MARGIN 5000;

Uživatelský zápis

**FOR(P0019:@D\_S\_3MENY1, P0021:@Z\_POTUV2, P0901:@D\_S\_AN1)**

SUM(SIS12\_02[2,5]) = SUM(SIS12\_01[2,10, P0236:S\_RZOBY]) + SUM(SIS12\_01[2,11, P0236:S\_RZOBY]) + SUM(SIS12\_01[2,12, P0236:S\_RZOBY]) + SUM(SIS12\_01[2,13, P0236:S\_RZOBY]) + SUM(SIS12\_01[2,14, P0236:S\_RZOBY]) + SUM(SIS12\_01[2,15, P0236:S\_RZOBY]) MARGIN 5000;

1. Opakované provedení kontrolního výrazu pro množinu vykázaných kombinací hodnot Dynamických parametrů v rámci různých Datových oblastí (RIS61\_01 a SIS01\_01)

Sémantický zápis

WITH(P0009=CZK, P0011=050, P0013=100, P0014=001, P0065=98, P0186=23)

**FOR(P0019:CZK, P0024:@D\_S\_VKLADUV2, P0236:@Z\_NEDOS)**

SUM(SISIFE01[EPR1383\_001, P0125=CZ, P0151=S\_NACE, P1117:@pS3\_2\_I1]) = SUM(RISIFE61[EPR1383\_001, P0125:CZ, P0151:@D\_S\_NACE\_1, P1117=@pS3\_2\_I1]) MARGIN 5000;

Uživatelský zápis

**FOR(P0019:CZK, P0024:@D\_S\_VKLADUV2, P0236:@Z\_NEDOS)**

SUM(SISIFE01:SIS01\_01[2,5]) = SUM(RISIFE61:RIS61\_01[2,6, P0125:CZ]) MARGIN 5000;

**FILTER\_CHECK**

Pro opakované provedení kontrolního výrazu nad jednotlivými dynamickými instancemi (řádky, sloupci, kartami) jedné Datové oblasti je klausule FILTER\_CHECK v uživatelském zápisu shodná se sémantickým zápisem.

Obecný tvar konstrukce FILTER\_CHECK je následující:

**FILTER\_CHECK ( <DatovaOblast> [ { , <filtr > } ])**

kde

* **<DatovaOblast>** - kód Datové oblasti
* **<filtr>** - je výraz definující filtrační podmínku Dynamických instancí, která může být definována konkretizací Dynamických parametrů nebo libovolným výrazem.

Dále jsou popsány jednotlivé případy.

Pokud se kontrolní výraz provádí pro všechny vykázané dynamické instance, má klausule tvar:

**FILTER\_CHECK (<DatovaOblast>)**

kde

* **<DatovaOblast>** - je kód Datové oblasti

Příklad:

1. Opakované provedení kontrolního výrazu pro všechny zaslané Dynamické instance (řádky, sloupce, karty)

Sémantický zápis

WITH(P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0186=21)

**FILTER\_CHECK(DIS85\_03)**

[PBD0203] = [PBD0204] + [PBD0205] + [PBD0206] MARGIN 3000;

Uživatelský zápis

**FILTER\_CHECK(DIS85\_03)**

DIS85\_03[2,5] = DIS85\_03[2,6] + DIS85\_03[2,7] + DIS85\_03[2,8] MARGIN 3000;

Pokud se kontrolní výraz provádí pro podmnožinu dynamických instancí (řádků, sloupců, karet) vymezenou konkretizací Dynamických parametrů, má klausule tvar:

**FILTER\_CHECK (<DatovaOblast> <parKonkr > [ { , <parKonkr > } ]**

kde

* **<DatovaOblast>** - je kód Datové oblasti
* **<parKonkr>** - konkretizace Dynamických parametrů dané Datové oblasti, podle kterých se provádí filtrace dynamických záznamů
* <kodParametr> = <kodPolozkaCiselniku>
* <kodParametr> = <kodhodnotaDatovehoTypu>
* <kodParametr>:@<kodDomenaCiselniku>
* <kodParametr>:@<kodDatovyTyp>
* <kodParametr>:\*
* <kodParametr>:<mnozina>

Příklad:

1. Opakované provedení kontrolního výrazu pouze pro všechny zaslané Dynamické instance (řádky, sloupce, karty), které obsahují definovanou konkretizací Dynamického parametru

Sémantický zápis

WITH(P0011=120, P0013=100, P0014=001)

**FILTER\_CHECK(DIS60\_07, P0025:CZ)**

AND(ISBLANK([EVD0131]),

 ISBLANK([EVD0132]),

 ISBLANK([EVD0133]),

 ISBLANK([EVD0134]),

 ISBLANK([EVD0135]));

Uživatelský zápis

**FILTER\_CHECK(DIS60\_07, P0025:CZ)**

AND(ISBLANK(DIS60\_07[1,8]),

 ISBLANK(DIS60\_07[1,9]),

 ISBLANK(DIS60\_07[1,10]),

 ISBLANK(DIS60\_07[1,11]),

 ISBLANK(DIS60\_07[1,12]));

Vymezení dynamických instancí (řádků, sloupců, karet) nemusí být dáno pouze konkretizace Dynamických parametrů, ale složitějším výrazem nebo množinou výrazů

**FILTER\_CHECK (<DatovaOblast> <vyraz > [ { , <vyraz > } ]**

kde

* **<DatovaOblast>** - je kód Datové oblasti
* **<vyraz>** - jakýkoliv výraz definující vymezení Dynamických instancí (řádků, sloupců, karet)

Příklady:

1. Opakované provedení kontrolního výrazu pouze pro všechny zaslané Dynamické instance (řádky, sloupce, karty), které vyhovují definovanému výrazu

Sémantický zápis

WITH(P0011=050, P0013=100, P0014=001)

**FILTER\_CHECK(JIS51\_01, [BCP2195] = CZK)**

[BCP2007, P0009=CZK] = [BCP2194];

Uživatelský zápis

**FILTER\_CHECK(JIS51\_01, JIS51\_01[1,34] = CZK)**

JIS51\_01[1,32] = JIS51\_01[1,33] MARGIN 2;

**FILTER\_FAIL**

Pro zápis nepovolených kombinací je konstrukce FILTER\_FAIL v uživatelském zápisu shodná se sémantickým zápisem.

**FILTER\_FAIL( <DatovaOblast> , <parKonkr > <parKonkr > [ { , <parKonkr > } ] )**

kde

* **<DatovaOblast>**- kód Datové oblasti
* **<parKonkr > <parKonkr > [ { , <parKonkr > } ] -** specifikace nepovolených kombinací hodnot Dynamických parametrů
* <kodParametr> = <kodPolozkaCiselniku>
* <kodParametr> = <kodhodnotaDatovehoTypu>
* <kodParametr> = @<kodDatovyTyp>
* <kodParametr>:<mnozina>

Příklad:

1. Nepovolené kombinace hodnot Dynamických parametrů

Sémantický zápis i Uživatelský zápis (jsou shodné)

**FILTER\_FAIL(RIS22\_01, P0236:@Z\_REZBVD, P0125:@ZBEZCZ4);**

### Typy RowSet, Record a operace dereference

#### Identifikace zdroje a atributu ve funkcích typu JOIN

Pro funkce typu JOIN, popsané detailněji níže, platí následující pravidla pro specifikaci zdroje a atributu operace dereference

* Zdroje jsou specifikovány jako první dva parametry funkce tak, že:
* Je-li zdrojem datová oblast, je uveden její kód, který může být v případě mezivýkazové kontroly doplněn o vymezení relativního období.
* Je-li zdrojem filtr nad datovou oblastí nebo odvozený RowSet, je uvedena proměnná, která ho obsahuje.
* V dereferenci jsou zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.
* Atributy jsou identifikovány:
* Identifikátorem dynamické dimenze (typicky sloupec), je-li zdrojem datová oblast nebo nad ní definovaný filtr.
* Identifikátorem specifickým pro externí registr, je-li zdrojem externí registr.
* Poziční identifikací \_1, \_2 atd. v případě, že zdrojem je odvozený RowSet (např. výstup funkce JOIN\_FETCH).

#### Funkce pracující s RowSet typy

**JOIN-CHECK**

Funkce JOIN\_CHECK provádí opakovaně kontrolní výraz mezi odpovídajícími si záznamy dvou dynamických datových zdrojů.

Základní konstrukce je:

**JOIN\_CHECK ( <rowSet1>, <rowSet2>, <vyraz>, <vyraz2> );**

kde

* **<rowSet1t> -** kód prvního datového zdroje (např. datové oblasti, externího číselníku)
* **<rowSet2t> -** kód druhého datového zdroje (např. datové oblasti, externího číselníku)
* **<vyraz>**  - je výraz definující kontrolní výraz včetně definice relačního vztahu mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.
* **<vyraz2>** - kontrolní výraz na záznamy splňujícími podmínky definované ve <vyraz>.

Příklady:

**Kontrola EIOPA**

Popis kontroly

{c0160} = empty

FILTER: {c0290} like '##7#' or {c0290} like '##8#'

Sémantický zápis

JOIN\_CHECK('SE.06.02.16.01',

 'SE.06.02.16.02',

 AND(REGEXP\_LIKE( Y->si1554, "..7.|.8."),

 Y->UI = X->UI),

 ISBLANK(X->mi1987));

Uživatelský zápis

JOIN\_CHECK('SE.06.02.16.01',

 'SE.06.02.16.02',

 AND(REGEXP\_LIKE(Y->C0290, "..7.|..8."),

 Y->C0040 = X->C0040),

 ISBLANK(X->C0160));

Kontrola se provádí mezi záznamy dvou datových oblastí SE.06.02.16.01 a SE.06.02.16.02, které jsou relačně propojeny přes dynamický parametr UI.

Kontrola se provádí pro záznamy, pokud údaj C0290 z SE.06.02.16.02 vyhovuje definované podmínce podle regulárního výrazu.

Vlastní kontrola vyhodnocuje nevyplnění hodnoty údaje C0160 v SE.06.02.16.01.

**Kontrola MKT**

Popis kontroly

Je-li ve výkazu TRAFIM10 údaj Identifikační kód nástroje 1 (TRAF10\_11.FIM0065) nebo údaj Identifikační kód nástroje 2 (TRAF10\_11.FIM0066) vyplněný, potom musí být údaje Identifikace investičního nástroje (TRAF11\_11.FIM0151, TRAF11\_11.FIM0152, TRAF11\_11.FIM0153) prázdné.

Sémantický zápis

JOIN\_CHECK(TRAF10\_11,

 TRAF11\_11,

 AND(OR(NOT(ISBLANK( X->FIM0065)), NOT(ISBLANK( X->FIM0066))),

 X->R0011 = Y->R0011,

 X->R0012 = Y->R0012),

 AND( ISBLANK(Y->FIM0151), ISBLANK(Y->FIM0152), ISBLANK(Y->FIM0153)));

Uživatelský zápis

JOIN\_CHECK(TRAF10\_11,

 TRAF11\_11,

 AND(OR(NOT(ISBLANK( X->33)), NOT(ISBLANK( X->34))),

 X->2 = Y->1,

 X->3 = Y->2),

 AND( ISBLANK(Y->18), ISBLANK(Y->19), ISBLANK(Y->20)));

Kontrola se provádí mezi záznamy dvou datových oblastí TRAF10\_11 a TRAF11\_11, které jsou relačně propojeny přes dva dynamické parametry R0011 a R0012, které jsou v datových oblastech v příslušných sloupcích.

Kontroluje se vyplnění/nevyplnění hodnot údajů vzájemně propojených záznamů daných datových oblastí dle definované podmínky.

**Kontrola AnaCredit**

Popis kontroly

IF [PANACR02.ANA0001] = '06' THEN [PANACR03.ANA0039] = 'NTAP'

Sémantický zápis

JOIN\_CHECK(PAN02\_11,

 PAN03\_11,

 AND(X->ANA0001="06",

 X->R0033 = Y->R0033,

 X->R0030 = Y->R0030,

 X->R0031 = Y->R0031),

 REGEXP\_LIKE(Y->ANA0039, "NTAP"));

Uživatelský zápis

JOIN\_CHECK(PAN02\_11,

 PAN03\_11,

 AND(X->4="06",

 X->1 = Y->1,

 X->2 = Y->2,

 X->3 = Y->3),

 REGEXP\_LIKE(Y->15, "NTAP"));

Kontrola se provádí mezi záznamy dvou datových oblastí PAN02\_11 a PAN03\_11, které jsou relačně propojeny přes tři dynamické parametry R0033, R0030 a R0031.

Kontroluje se správná kombinace vyplnění hodnot údajů PAN02\_11[1,4] a PAN03\_11[1,15] vzájemně propojených záznamů daných datových oblastí.

**JOIN\_FETCH**

Funkce JOIN\_FETCH definuje odvozený datový zdroj typu RowSet, který je dále používán v kontrolním výrazu. Nad tímto odvozeným datovým zdrojem mohou být prováděny agregační funkce nebo mohou být používány jednotlivé záznamy.

Základní konstrukce je:

**<promenna> “:=” JOIN\_FETCH ( <rowSet1>, <rowSet2>, <vyraz>, <vyraz2> [ { , vyraz2> } ] );**

**<telo>;**

kde

* <promenna> - kód proměnné
* **<rowSet1t> -** kód prvního datového zdroje (např. datové oblasti, externího číselníku)
* **<rowSet2t> -** kód druhého datového zdroje (např. datové oblasti, externího číselníku)
* **<vyraz>**  - je výraz definující výraz definující odvozený RowSet včetně definice relačního vztahu mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.
* **<vyraz2>** - definice odvozeného datového zdroje typu RowSet pomocí dereference.
* **<telo> -** vlastní kontrolní výraz, který používá RowSet definovaný proměnnou.

Příklady:

**Kontrola EIOPA**

Popis kontroly

{S.02.01, r0080,c0010}=sum({S.06.02, c0170,(sNNN)})

FILTER:

({S.06.02, c0290} like ‘##91’ OR {S.06.02, c0290} like ‘##92’ OR {S.06.02, c0290} like ‘##94’ OR {S.06.02, c0290} like ‘##99’) AND ({S.06.02, c0090}=[s2c\_LB:x91])

Sémantický zápis

$XX:=JOIN\_FETCH('S.06.02.01.01',

 'S.06.02.01.02',

 AND(AND(REGEXP\_LIKE( Y->si1554, "..91|..92|..94|..99"), X->ei1070 = "x91"),

 X->UI = Y->UI),

 X->mi1131);

IAF\_EQ (SUM(XX->\_1), [mi265, \_DX="S.02.01.01.01", BL=x91, VG=x80]);

Uživatelský zápis

$XX := JOIN\_FETCH('S.06.02.01.01',

 'S.06.02.01.02',

 AND(AND(REGEXP\_LIKE(Y->C0290, "..91|..92|..94|..99"), X->C0090 = "x91"),

 X->C0040 = Y->C0040),

 X->C0170);

IAF\_EQ(SUM(XX->\_1), 'S.02.01.01.01'[R0080,C0010]);

Proměnná definuje odvozený RowSet obsahující údaj sloupce C0170 z datové oblasti S.06.02.01.01, přičemž množina záznamů je vybrána na základě filtrační podmínky sloupců C0290 a C0090 nad množinou propojených záznamů dvou datových oblastí přes dynamický parametr ve sloupci C0040.

Za tento RowSet obsahující množinu hodnot údaje C0170 se provádí suma, která se porovnává vůči hodnotě statického údaje z jiné datové oblasti.

**Kontrola MKT**

Popis kontroly

Obsahuje-li hodnota údaje Kupující - LEI kód nabyvatele anebo ústřední protistrany ve výkazu TRAFIM20 (TRAF20\_21.FIM0004) předepsaný dummy LEI kód, potom údaj Kupující - IČ nabyvatele (TRAF11\_11.FIM0410) musí být vyplněný a musí být nalezen v rámci množiny osob zaslaných výkazem PERFIM30 - kontrola oproti hodnotám údaje (PERF30\_11.RFD0003).

Sémantický zápis

$XX:=JOIN\_FETCH(TRAF20\_21,

 TRAF11\_11,

 AND(REGEXP\_LIKE(X->FIM0004, "DUMMY……."),

 X->R0011 = Y->R0011,

 X->R0012 = Y->R0012),

 Y->FIM0410);

JOIN\_EXISTS($XX,

 PERF30\_11,

 AND(NOT(ISBLANK(X->\_1)),

 X->\_1 = Y->RFD0003));

Uživatelský zápis

$XX:=JOIN\_FETCH(TRAF20\_21,

 TRAF11\_11,

 AND(REGEXP\_LIKE(X->5, "DUMMY……."),

 X->3 = Y->1,

 X->4 = Y->2),

 Y->4);

JOIN\_EXISTS($XX,

 PERF30\_11,

 AND(NOT(ISBLANK(X->\_1)),

 X->\_1 = Y->4));

Proměnná definuje odvozený RowSet obsahující údaj FIM0410 z TRAF11\_11, přičemž množina záznamů je vybrána na základě filtrační podmínky nad množinou propojených záznamů daných datových oblastí TRAF20\_21 a TRAF11\_11 přes dva dynamické parametry R0011, R0012.

Záznamy tohoto RowSetu se dále používají v kontrolním výrazu proti výkazu PERFIM30.

Proměnná definuje odvozený RowSet obsahující údaj TRAF11\_11[1,4], přičemž množina záznamů je vybrána na základě filtrační podmínky TRAF20\_21[1,5] nad množinou propojených záznamů daných datových oblastí přes dva dynamické parametry R0011, R0012, které se vyskytují v obou datových oblastech v příslušných sloupcích.

Záznamy tohoto RowSetu se dále používají v kontrolním výrazu proti výkazu PERFIM30.

**Kontrola AnaCredit**

Popis kontroly

PANACR07.ANA0063] DOES NOT EXIST IN {[PANACR04.R0034] | ([PANACR08.R0033], [PANACR08.R0030], [PANACR08.R0031] = [PANACR04.R0033], [PANACR04.R0030], [PANACR04.R0031]) AND [PANACR04.P1350] = '1'}

Sémantický zápis

$XX:=JOIN\_FETCH(PAN04\_11,

 PAN08\_11,

 AND(X->P1350="1",

 X->R0033 = Y->R0033,

 X->R0030 = Y->R0030,

 X->R0031 = Y->R0031),

 X->R0033,

 X->R0034);

JOIN\_CHECK($XX,

 PAN07\_11,

 AND(X->\_1 = Y->R0033),

 NOT(Y->ANA0063 = X->\_2));

Uživatelský zápis

$XX:=JOIN\_FETCH(PAN04\_11,

 PAN08\_11,

 AND(X->5="1",

 X->1 = Y->1,

 X->3 = Y->2,

 X->4 = Y->3),

 X->1,

 X->6);

JOIN\_CHECK($XX,

 PAN07\_11,

 AND(X->\_1 = Y->1),

 NOT(Y->3 = X->\_2));

Proměnná definuje odvozený RowSet obsahující údaje PAN04\_11 [1,1] a PAN04\_11 [1,6], přičemž množina záznamů je vybrána na základě filtrační podmínky nad množinou propojených záznamů dvou datových oblastí přes tři dynamické parametry R0033, R0030, R0031.

Záznamy tohoto RowSetu se dále používají v kontrolním výrazu.

**JOIN\_EXISTS**

Funkce JOIN\_EXISTS Funkce JOIN\_EXISTS ověřuje existenci definovaného klíče z jednoho datového zdroje v druhém datovém zdroji podle definované podmínky.

Základní konstrukce je:

**JOIN\_EXISTS ( <rowSet1>, <rowSet2>, <vyraz> );**

kde

* **<rowSet1t> -** kód prvního datového zdroje (např. datové oblasti, externího číselníku)
* **<rowSet2t> -** kód druhého datového zdroje (např. datové oblasti, externího číselníku)
* **<vyraz>**  - je výraz definující kontrolní výraz včetně definice relačního vztahu mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.

Příklady:

**Kontrola MKT**

Popis kontroly

Množina hodnot údajů Identifikace investičního nástroje (TRAF30\_11.FIM0255, TRAF30\_11.FIM0307, TRAF30\_11.FIM0308, TRAF30\_11.FIM0309) musí být nalezena v rámci množiny nástrojů zaslaných výkazem REFFIM10 - kontrola oproti množině hodnot údajů (REFF10\_11.RFD0033, REFF10\_11.RFD0034, REFF10\_11.RFD0035, REFF10\_11.RFD0036).

Sémantický zápis

JOIN\_EXISTS(TRAF30\_11,

 REFF10\_11,

 AND(X->FIM0255 = Y->RFD0033,

 X->FIM0307 = Y->RFD0034,

 X->FIM0308 = Y->RFD0035,

 X->FIM0309 = Y->RFD0036));

Kontrola se provádí mezi záznamy dvou datových oblastí TRAF30\_11 a REFF10\_11, které jsou relačně propojeny přes čtyři prvky klíče, kterými jsou dynamické parametry a údaje.

Kontroluje se existence vzájemně si odpovídajících záznamů.

Uživatelský zápis

JOIN\_EXISTS(TRAF30\_11,

 REFF10\_11,

 AND(X->28 = Y->2,

 X->29 = Y->3,

 X->30 = Y->4,

 X->31 = Y->5));

Kontrola se provádí mezi záznamy dvou datových oblastí TRAF30\_11 a REFF10\_11, které jsou relačně propojeny přes čtyři prvky klíče, kterými jsou dynamické parametry a údaje.

Kontroluje se existence vzájemně si odpovídajících záznamů.

**Kontrola AnaCredit**

Popis kontroly

([PANACR02.R0033], [PANACR02.R0030], [PANACR02.R0031]) EXISTS IN {([PANACR03.R0033], [PANACR03.R0030], [PANACR03.R0031])}

Sémantický zápis

JOIN\_EXISTS(PAN02\_11,

 PAN03\_11,

 AND(X->R0033 = Y->R0033,

 X->R0030 = Y->R0030,

 X->R0031 = Y->R0031));

Uživatelský zápis

JOIN\_EXISTS(PAN02\_11,

 PAN03\_11,

 AND(X->1 = Y->1,

 X->2 = Y->2,

 X->3 = Y->3));

Kontrola se provádí mezi záznamy dvou datových oblastí PAN02\_11 a PAN03\_11, které jsou relačně propojeny přes tři dynamické parametry R0033, R0030 a R0031, které se vyskytují v příslušných sloupcích daných datových oblastí.

Kontroluje se existence vzájemně si odpovídajících záznamů, v tomto případě bez další podmínky.

### Kontrolní výraz

Základní struktura kontrolního výrazu v uživatelském jazyku je shodná se sémantickým jazykem.

Kontrolní výraz má následující hlavní strukturu

**[ { <pragma> } ]**

**[ { <promenna> } ]**

**<telo>**

kde

* **<pragma>** - obsahuje doplňující informace ke kontrole
* **<promenna> -** definice proměnné použité v kontrolním výrazu
* **<telo>** - obsahuje vlastní kontrolní výraz, který může být definovaný následujícími způsoby:
* <vyraz>
* FILTER\_CHECK <vyraz> – provádí se pro množinu dynamických záznamů vyhovujících definovanému filtru.
* FOR <vyraz> – provádí se opakovaně pro množiny Údajů, které jsou sestaveny z kombinací konkretizací specifikovaných ve FOR klausuli a těle kontrolního výrazu.

**Pragma**

Aktuálně je zavedena pragma #RESTRICT, která obsahuje vymezení kontroly pro množinu osob.

Množina osob je definována standardní konstrukcí pro množinu, která je popsána v kapitole 4.1.2.5

Příklady:

1. Kontrola s pragmou RESTRICT

**#RESTRICT(@BANCR)**

WITH(P0011=120, P0013=200, P0014=001, P0019=S\_VSEMEN)

IF(AND([EBD1681, P0009=CZK] > 0, [EBD1682, P0009=CZK] > 0), [EBD1688, P0009=950] = 0);

## Gramatika uživatelského jazyka kontrol

V této kapitole je popsána gramatika uživatelského jazyka kontrol.

V popisu pravidel jsou použity následující formalizmy:

* “ “ – jednoduché textové terminály jsou v uvozovkách (jsou uváděny jako text pro zvýšení čitelnosti)
* <> - pravidla, neterminály a další terminály jsou ve špičatých závorkách
* ( ) – sloučení do skupiny
* { } – opakování skupiny libovolněkrát
* [ ] - nepovinná skupina
* | - varianty

 /\*Pravidla základních elementů\*/

<NULL> ::= "NULL" ;

<TRUE> ::= "TRUE" ;

<FALSE> ::= "FALSE" ;

<LETTER> ::= "\_" | "A"-"Z" ;

<STRING> ::= "\"" (~["\""])\* "\"" ;

<DIGIT> ::= "0"-"9" ;

<NUMINT> ::= <DIGIT> {<DIGIT>} ;

<NUMDEC> ::= <DIGIT> {<DIGIT>} "." <DIGIT> {<DIGIT>} ;

<IDENT> ::= ([<LETTER>]+ {<LETTER> | <DIGIT>}) | "'" (~["'"])\* "'" ;

<KOD> ::= (<LETTER> | <DIGIT>) {<LETTER> | <DIGIT>} ;

<VAR> ::= "$"<IDENT> ;

<DOM> ::= "@"<IDENT> ;

<PRAGMA> ::= "#" <IDENT> ["(" (~[")"])\* ")"] ;

<AnyInt> ::= ["-"] <NUMINT> ;

<AnyNum> ::= <AnyInt> | (["-"] <NUMDEC>) ;

/\*Pravidla zápisu definice proměnných, množin a funkcí \*/

<Assign> ::= <VAR> ":=" <Expression>“;“ ;

<SetElm> ::= ["^"] <Expression> ;

<Set> ::= ("{" "}") | ("{" <SetElm> {", " <SetElm>} "}") ;

<FunctionCall> ::= (<IDENT> "(" ")") | (<IDENT> "(" <Expression> {", " <Expression>} ")") ;

/\*Pravidla zápisu určení datového zdroje výkazů (údajů, parametrů)\*/

<Coord> ::= <IDENT> | <AnyInt> ;

<RelObd> ::= "%" <AnyInt> ;

<IdentRelObd> ::= <IDENT> <RelObd> ;

<DataPointVykPart> ::= (<IdentRelObd> | <IDENT>) ":" ;

<DataPoint> ::= [<DataPointVykPart>] <IDENT> "[" <Coord> "," <Coord>

[ "," ( (<VAR> | <FunctionCall> | <ParConcrList>) | ([( <Coord> ", ") ] (<VAR> | <FunctionCall> | <ParConcrList>)) ) ] "]" ;

/\*Pravidla zápisu konkretizace parametrů metapopisu pro určení datového zdroje výkazů\*/

<ParConcr> ::= <IDENT> (":" | "=") (<IDENT> | <KOD> | <NUMINT> | <NUMDEC> | <STRING> | <DOM> | "\*" | <NULL> | <Set>) ;

<ParConcrList> ::= <ParConcr> {"," <ParConcr>} ;

/\*Pravidla zápisu výrazů\*/

<Expression> ::= <InExpression> ;

<InExpression> ::= <RelationalExpression> ["IN" <RelationalExpression>] ;

<RelationalExpression> ::= <AdditiveExpression> [("<" | ">" | "<=" | ">=" | "=" | "<>") <AdditiveExpression>] ;

<AdditiveExpression> ::= <MultiplicativeExpression> [("+" | "-") <MultiplicativeExpression>] ;

<MultiplicativeExpression> ::= <UnaryExpression> [("\*" | "/") <UnaryExpression>] ;

<UnaryExpression> ::= ["-"] <PrimaryExpression> ;

<Bracketed> ::= "(" <Expression> ")" ;

<PrimaryExpression> ::= <STRING> | <VAR> | <DOM> | <NULL> | <TRUE> | <FALSE> | <IdentRelObd> | <IDENT> | <AnyNum> | <Set> | <DataPoint> | <ParConcr> | <FunctionCall> | <Bracketed> | <Expression> ;

/\*Pravidla zápisu kontrolního výrazu\*/

<PragmaBlock> ::= <PRAGMA> {"," <PRAGMA>} /\*pragma\*/ ;

<ForClause> ::= "FOR" "(" <ParConcrList> ")" <Expression> ; /\*definice opakování kontroly\*/

<FilterCheck> ::= "FILTER\_CHECK" "(" <Expression> {"," <Expression>} ")" <Expression> ; /\*definice opakování kontroly na dynamickou množinou\*/

<Script> ::= [<PragmaBlock>] {<Assign>} (<Expression> | <ForClause> | <FilterCheck>)“;“ ; /\* definice kontroly\*/

# Popis funkcí jazyka kontrol

V této kapitole jsou popsány jednotlivé funkce jazyka kontrol. Daná funkce je vždy obecně popsána, je popsána její konstrukce, vstupy a výstupy a u každé funkce je uveden konkrétní příklad.

## Úvod

Seznam použitých terminálů a jejich význam.

* <logVyraz> - libovolný argument, jehož výsledkem je logická hodnota true nebo false
* výraz, jehož výstupem je logická hodnota true nebo false
* konstanta – true, false
* hodnota Údaje, nebo Dynamického parametru.
* funkce navracející true, false
* <konstanta> - význam je popsán v příslušné funkci
* <datovaOblast> - kód Datové oblasti
* <regVyraz> - regulární výraz
* <číslo> - libovolné číslo
* konstanta
* hodnota Údaje nebo Dynamického parametru s číselným datovým typem.
* funkce nebo výraz navracející číselnou hodnotu
* <řetězec> - libovolný řetězec
* řetězec
* hodnota Údaje nebo Dynamického parametru
* funkce navracející řetězec
* <datum> - libovolné datum
* datum
* hodnota Údaje nebo Dynamického parametru.
* funkce navracející datum
* <RowSet> - množina záznamů
* <množina>
* <extCiselnik> - kód externího číselníku

## Logické funkce

|  |  |
| --- | --- |
| Funkce | Popis |
| AND | Vyhodnocuje, zda jsou všechny argumenty pravdivé. |
| IF | Na základě vyhodnocení definované podmínky vrátí hodnotu definovanou pro výsledek vyhodnocení true nebo hodnotu definovanou pro výsledek vyhodnocení false. |
| ISBLANK | Vyhodnocuje, zda je vstupní hodnota prázdná. |
| ISNUMBER | Vyhodnocuje, zda je vstupní hodnota číslo. |
| NOT | Negace výrazu. |
| OR | Vyhodnocuje, zda je alespoň jeden argument pravdivý. |

### AND

Funkce AND Vyhodnocuje, zda jsou všechny argumenty pravdivé.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | AND (<logVyraz>   [ { , <logVyraz> } ] )   |
| Vstupní argumenty | <logVyraz> |
| Návratová hodnota | Funkce navrací false nebo true.* TRUE – v případě, že jsou pravdivé všechny argumenty ve výrazu
* FALSE – v případě, že alespoň jeden výraz není pravda.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce je TRUE v případě, že jsou oba argumenty > 0. |
| Zápis – Sémantický tvar | AND([EBD1681, P0009=CZK] > 0, [EBD1682, P0009=CZK] > 0) |
| Zápis – Uživatelský tvar | AND(DIS23\_21[1,1] > 0, DIS23\_21[2,1] > 0) |

### IF

Funkce IF na základě vyhodnocení definované podmínky vrátí hodnotu definovanou pro výsledek vyhodnocení true nebo hodnotu definovanou pro výsledek vyhodnocení false.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | IF(<logVyraz>, <any> [ , <any>]) |
| Vstupní argumenty | <logVyraz> - podmínka<any> - výraz je vyhodnocen v případě splnění podmínky.[ , <any>]) – výraz je vyhodnocen v případě nesplnění podmínky |
| Návratová hodnota | Návratová hodnota výstupem výrazu na základě vyhodnocení podmínky. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Pokud je první argument pravdivý, tak platí druhý argument, jinak třetí. |
| Zápis – Sémantický tvar | IF([ABD0559, P0236=S\_ESACNB] > 0, [ABD0560, P0236=S\_RNUVINMRB] > 0, [ABD0560, P0236=S\_RNUVINMRB] < 0) |
| Zápis – Uživatelský tvar | IF(DIS16\_01[1, 2]>0, DIS16\_01[1, 3]>0, DIS16\_01[1, 3]<0) |

### ISBLANK

Funkce ISBLANK vyhodnocuje, zda je vstupní hodnota prázdná. Vstupem může být i množina hodnot.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ISBLANK(<any>) |
| Vstupní argumenty | <any> |
| Návratová hodnota | Funkce navrací true nebo false.* TRUE – v případě, že je argument prázdný.
* FALSE – v případě, že není argument prázdný (stačí jeden z parametrů neprázdný).
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola buňky, zdali je prázdná. |
| Zápis – Sémantický tvar | ISBLANK([ABD0562, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0186=25, P0236=S\_RNOSTKZ]) |
| Zápis – Uživatelský tvar | ISBLANK(DIS16\_01[1, 5]) |

### ISNUMBER

Funkce ISNUMBER vyhodnocuje, zda je vstupní hodnota číslo.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ISNUMBER(<any>) |
| Vstupní argumenty | <any> |
| Návratová hodnota | Funkce navrací true nebo false.* TRUE – v případě, že je argument číslo.
* FALSE – v případě, že argument není číslo.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, že buňka obsahuje číslo. |
| Zápis – Sémantický tvar | ISNUMBER([ABD0562, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0186=25, P0236=S\_RNOSTKZ]) |
| Zápis – Uživatelský tvar | ISNUMBER(DIS16\_01[1, 2]) |

### NOT

Funkce NOT slouží k negaci výrazu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | NOT(<logVyraz>) |
| Vstupní argumenty | <logVyraz> |
| Návratová hodnota | Funkce navrací negaci hodnoty true nebo false. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, že buňka neobsahuje číslo. |
| Zápis – Sémantický tvar | NOT(ISNUMBER([ABD0562, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0186=25, P0236=S\_RNOSTKZ])) |
| Zápis – Uživatelský tvar | NOT(ISNUMBER(DIS16\_01[1, 2])) |

### OR

Funkce OR vyhodnocuje, zda je alespoň jeden argument pravdivý.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | OR(<logVyraz>   [ { , <logVyraz> } ]) |
| Vstupní argumenty | <logVyraz> |
| Návratová hodnota | Funkce navrací true nebo false.* TRUE – v případě, že je alespoň jeden argument pravdivý.
* FALSE – v případě, že ani jeden argument není pravdivý.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce vyhodnotí platnost obou výrazů. |
| Zápis – Sémantický tvar | OR([ABD0560, P0236=S\_RNUVINMRB] > 0, [ABD0561, P0236=S\_RNVLAOMI] > 0); |
| Zápis – Uživatelský tvar | OR(DIS16\_01[1, 3]>0, DIS16\_01[1, 4]>0) |

## Matematické funkce

Pro všechny matematické funkce platí, že vstupní argumenty a výstupy funkcí jsou datového typu číslo.

|  |  |
| --- | --- |
| Funkce | Popis |
| ABS | Absolutní hodnota argumentu. |
| AVERAGE | Průměrná hodnota ze zadaných argumentů. |
| ROUND | Zaokrouhlení. |
| ROUNDDOWN | Zaokrouhlení dolů. |
| ROUNDUP | Zaokrouhlení nahoru. |
| SIGN | Navrací znaménko argumentu. |
| SUM | Suma ze zadaných argumentů. |
| TRUNC | Oříznutí čísla na jeho celou část. |
| DIV | Celočíselný podíl dvou čísel. |
| EXP | Exponenciální funkce. |

### ABS

Funkce ABS slouží k zjištění absolutní hodnoty ze zadaných argumentů.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ABS(<číslo>) |
| Vstupní argumenty | <číslo> |
| Návratová hodnota | Funkce navrací absolutní číselnou hodnotu ze vstupního argumentu. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Absolutní hodnota vybrané buňky musí být větší než 0. |
| Zápis – Sémantický tvar | ABS([EBD1681, P0009=CZK, P0011=120, P0013=200, P0014=001, P0019=S\_VSEMEN]) > 0 |
| Zápis – Uživatelský tvar | ABS(DIS23\_21[1,1])>0 |

### AVERAGE

Funkce AVERAGE slouží k výpočtu průměru ze zadaných argumentů.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | AVERAGE(<číslo>   [ { , <číslo> } ]) |
| Vstupní argumenty | <číslo> |
| Návratová hodnota | Funkce navrací průměrnou číselnou hodnotu ze vstupních argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Průměrná hodnota vybraných buněk musí být větší než 0. |
| Zápis – Sémantický tvar | AVERAGE([PBD0383, P0236=S\_ESACNB], [PBD0384, P0236=S\_RNUVINMRB], [PBD0385, P0236=S\_RNVLAOMI])>0; |
| Zápis – Uživatelský tvar | AVERAGE(DIS16\_02[1,2], DIS16\_02[1,3], DIS16\_02[1,4])>0; |

### NUMBER

Funkce NUMBER slouží vytvoření čísla z řetězce (datový typ TEXT) pro potřeby jiných funkcí, které požadují na vstupu datový typ CISLO.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | NUMBER(<řetězec>) |
| Vstupní argumenty | <řetězec> - musí to být číslo bez mezer, oddělovač pouze desetinných míst "." a může mít znaménko. |
| Návratová hodnota | Funkce navrací číslo jako datový typ CISLO. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, že podíl PAN03\_11[1,9] a PAN03\_11[1,10] je větší než 0. |
| Zápis – Sémantický tvar | NUMBER([ANA0033]) / NUMBER([ANA0034] > 0; |
| Zápis – Uživatelský tvar | NUMBER(PAN03\_11[1,9]) /(PAN03\_11[1,10]) > 0; |

### ROUND

Funkce ROUND slouží k zaokrouhlení čísel.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ROUND(<číslo>, <konstanta>) |
| Vstupní argumenty | <číslo><konstanta>* Pokud je konstanta větší než 0 (nula), bude číslo zaokrouhleno na zadaný počet desetinných míst.
* Pokud je konstanta rovna 0, bude číslo zaokrouhleno na nejbližší celé číslo.
* Pokud je konstanta menší než 0, bude číslo zaokrouhleno směrem doleva od desetinné čárky.
 |
| Návratová hodnota | Funkce navrací zaokrouhlenou číselnou hodnotu ze vstupního argumentu. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Zaokrouhlení na 4 desetinná místa. |
| Zápis – Sémantický tvar | ROUND ([PBD0383, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0236=S\_ESACNB], 4) |
| Zápis – Uživatelský tvar | ROUND(DIS16\_02[1,2], 4) |

### ROUNDDOWN

Funkce ROUNDDOWN slouží k zaokrouhlení čísel vždy směrem dolu (blíže k nule).

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ROUNDDOWN(<číslo><konstanta>) |
| Vstupní argumenty | <číslo><konstanta>* Pokud je konstanta větší než 0, zaokrouhlí se číslo dolů na počet desetinných míst daný touto číslicí.
* Jestliže je konstanta rovna 0 (nula), je číslo zaokrouhleno na nejbližší nižší celé číslo.
* Jestliže je konstanta menší než 0, zaokrouhlí se číslo dolů doleva od desetinné čárky.
 |
| Návratová hodnota | Funkce navrací zaokrouhlenou číselnou hodnotu ze vstupního argumentu. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Zaokrouhlení dolů na 4 desetinná místa. |
| Zápis – Sémantický tvar | ROUNDDOWN([PBD0383, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0236=S\_ESACNB], 4) |
| Zápis – Uživatelský tvar | ROUNDDOWN(DIS16\_02[1,2], 4) |

### ROUNDUP

Funkce ROUNDUP slouží k zaokrouhlení čísel vždy směrem nahoru (dále od nuly).

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ROUNDUP(<číslo>, <konstanta>) |
| Vstupní argumenty | <číslo><konstanta>* Je-li konstanta větší než 0, zaokrouhlí se číslo nahoru na počet desetinných míst daný touto číslicí.
* Jestliže se konstanta rovná 0 (nula), je číslo zaokrouhleno na nejbližší celé číslo.
* Je-li konstanta menší než 0, zaokrouhlí se číslo nahoru doleva od desetinné čárky.
 |
| Návratová hodnota | Funkce navrací zaokrouhlenou číselnou hodnotu ze vstupního argumentu. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Zaokrouhlení nahoru na 4 desetinná místa. |
| Zápis – Sémantický tvar | ROUNDUP([PBD0383, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0236=S\_ESACNB], 4) |
| Zápis – Uživatelský tvar | ROUNDUP(DIS16\_02[1,2], 4) |

### SIGN

Funkce SIGN slouží k navrácení znaménka daného argumentu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | SIGN(<číslo>) |
| Vstupní argumenty | <číslo> |
| Návratová hodnota | Funkce vrátí hodnotu 1, pokud je číslo kladné, hodnotu 0, pokud je číslo 0 a hodnotu -1, pokud je číslo záporné. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Navrácení hodnoty -1, 0 nebo 1. |
| Zápis – Sémantický tvar | SIGN([PBD0383, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0236=S\_ESACNB]) |
| Zápis – Uživatelský tvar | SIGN(DIS16\_02[1, 2]) |

### SUM

Funkce SUM vytvoří sumu ze zadaných argumentů.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | SUM(<číslo>|<množina> [ { , <číslo>|<množina> } ]) |
| Vstupní argumenty | <číslo><množina> |
| Návratová hodnota | Funkce navrátí číselnou hodnotu, která je součtem hodnot vstupních argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Porovnání součtu hodnot dynamického Údaje a hodnoty statického Údaje. |
| Zápis – Sémantický tvar | SUM([EPD1030, P0025:@D\_S\_VSEZEME2]) = [EPD1030, P0025=S\_VSEZEME] MARGIN 5000; |
| Zápis – Uživatelský tvar | SUM(DIS16\_03[2,5]) = DIS16\_03[1,5] MARGIN 5000; |

### TRUNC

Funkce TRUNC ořízne zadané číslo na celé číslo odstraněním desetinných míst, nebo jejich části.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | TRUNC(<číslo>, <konstanta>) |
| Vstupní argumenty | <číslo><konstanta>* Pokud není vyplněna – jsou odříznuta všechna desetinná místa a je ponechána pouze celá část.
* Pokud je vyplněna – určí počet desetinných míst, na která je oříznuto vstupní číslo.
 |
| Návratová hodnota | Funkce vrátí oříznutou hodnotu čísla na zadaný počet desetinných míst. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Oříznutí hodnoty Údaje na 2 desetinná místa. |
| Zápis – Sémantický tvar | TRUNC([PBD0383, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME, P0236=S\_ESACNB], 2) |
| Zápis – Uživatelský tvar | TRUNC(DIS16\_02[1, 2], 2) |

### DIV

Funkce DIV slouží pro celočíselný podíl dvou čísel.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | DIV(<číslo>, <číslo>) |
| Vstupní argumenty | <číslo> |
| Návratová hodnota | Funkce vrátí celočíselný výsledek po dělení vstupních argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Celočíselný podíl musí být větší než 0. |
| Zápis – Sémantický tvar | DIV([ABD0559, P0236=S\_ESACNB], [ABD0560, P0236=S\_RNUVINMRB]) > 0 MARGIN 2000; |
| Zápis – Uživatelský tvar | DIV(DIS16\_01[1, 2], DIS16\_01[1, 3])>0; |

### EXP

Funkce EXP je exponenciální funkce. Slouží k umocnění základu. První argument funkce je základ, druhý argument je čitatel exponentu a třetí atrigut je jmenovatel exponentu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Popis kontroly | EXP(<číslo>, <konstanta1> [, <konstanta2>]) |
| Vstupní argumenty | <číslo> - základ k umocnění<konstanta1> - čitatel exponentu<konstanta2> - jmenovatel exponentu – nepovinný, defaultně je nastaven na hodnotu 1. |
| Návratová hodnota | Funkce navrací hodnotu, která je výsledkem dané funkce. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Umocnění buňky [1, 4] na druhou musí být větší než 1. |
| Zápis – Sémantický tvar | FILTER\_CHECK(DIS25\_04)EXP([PBD0388, P0009=CZK, P0011=120, P0013=100, P0014=001, P0186=14, P0236=S\_RNBUIMRB], 2, 1)>1; |
| Zápis – Uživatelský tvar | FILTER\_CHECK(DIS25\_04)EXP(DIS25\_04[1,4], 2, 1) > 1; |

## Řetězcové funkce

|  |  |
| --- | --- |
| Funkce | Popis |
| CONCATENATE | Spojení více řetězců do jednoho. |
| LEFT | Oříznutí řetězce z leva. |
| LEN | Navrací délku řetězce. |
| REGEXP\_LIKE | Vyhledání v řetězci dle zadaného regulárního výrazu. |
| RIGHT | Oříznutí řetězce zprava. |
| UPPER | Změna celého řetězce na velká písmena. |
| SUBSTR | Navrácení části řetězce dle zadaných parametrů. |

### CONCATENATE

Funkce CONCATENATE slouží pro spojení dvou a více řetězců do jednoho řetězce.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | CONCATENATE(<řetězec>  [ { , <řetězec>} ]) |
| Vstupní argumenty | <řetězec> |
| Návratová hodnota | Funkce navrací řetězec dle vstupních argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce spojí více řetězců do jednoho |
| Zápis – Sémantický tvar | CONCATENATE("Séman", "tický") |
| Zápis – Uživatelský tvar | CONCATENATE("Séman", "tický") |

### LEFT

Funkce LEFT slouží k extrahování znaků zleva dle vstupních argumentů.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | LEFT(<řetězec>, <konstanta>) |
| Vstupní argumenty | <řetězec><konstanta> - počet extrahovaných znaků. |
| Návratová hodnota | Funkce navrací řetězec dle vstupních argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola správnosti ořezu řetězce. |
| Zápis – Sémantický tvar | LEFT(“Sémantický tvar“, 10) = “Sémantický“ |
| Zápis – Uživatelský tvar | LEFT(“Sémantický tvar“, 10) = “Sémantický“ |

### LEN

Funkce LEN slouží k navrácení počtu znaků řetězce.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | LEN(<řetězec>) |
| Vstupní argumenty | <řetězec> |
| Návratová hodnota | Funkce navrací číslo počtu znaků vstupního řetězce. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda se délka řetězce rovná 20. |
| Zápis – Sémantický tvar | LEN([si288, \_DX="C\_27.00"]) = 20;" |
| Zápis – Uživatelský tvar | LEN('C\_27.00'[1,030]) = 20;" |

### REGEXP\_LIKE

Funkce REGEXP\_LIKE slouží k nalezení části řetězce podle regulárního výrazu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | REGEXP\_LIKE(<řetězec>, <regVyraz>) |
| Vstupní argumenty | <řetězec><regVyraz> - hledaný výraz. |
| Návratová hodnota | Funkce navrací true nebo false.* TRUE – v případě, že je regulární výraz nalezen.
* FALSE – v případě, že regulární výraz není nalezen.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce nalezne část řetězce podle regulárního výrazu. |
| Zápis – Sémantický tvar | REGEXP\_LIKE([[R0012]], "D.DB.."); |
| Zápis – Uživatelský tvar | REGEXP\_LIKE(TRAF10\_11[1, 3], "D.DB.."); |

### RIGHT

Funkce RIGHT slouží k extrahování znaků zprava dle zadaných argumentů

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | RIGHT(<řetězec>, <konstanta>) |
| Vstupní argumenty | <řetězec><konstanta> - počet navrácených znaků. |
| Návratová hodnota | Funkce navrací řetězec dle vstupních argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce oříznutí řetězce na 4 znaky zprava. |
| Zápis – Sémantický tvar | RIGHT(“Sémantický tvar“, 4) = “tvar“ |
| Zápis – Uživatelský tvar | RIGHT(“Sémantický tvar“, 4) = “tvar“ |

### UPPER

Funkce UPPER slouží k převedení řetězce na velká písmena.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | UPPER(<řetězec>) |
| Vstupní argumenty | <řetězec> |
| Návratová hodnota | Funkce navrací řetězec dle vstupních argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce převede řetězec na velké znaky. |
| Zápis – Sémantický tvar | UPPER(“Sémantická kontrola“) = “SÉMANTICKÁ KONTROLA“; |
| Zápis – Uživatelský tvar | UPPER(“Sémantická kontrola“) = “SÉMANTICKÁ KONTROLA“; |

### SUBSTR

Funkce SUBSTR slouží k vyhledání a navrácení řetězce podle zadaných argumentů.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | SUBSTR(<řetězec>, <konstanta>, <konstanta>) |
| Vstupní argumenty | <řetězec><konstanta> - počátek řetězce<konstanta> - počet navrácených znaků – nepovinný argument. |
| Návratová hodnota | Funkce navrací vyhledaný řetězec dle vstupních argumentů, pokud není zadán počet znaků, vrací zbytek řetězce od pozice „počátek“. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zdali se navrácený řetězec rovná požadovanému řetězci. |
| Zápis – Sémantický tvar | SUBSTR([RFD0003, P0556=PERSON, P0854=STRING], 1, 4) = "CCPT"; |
| Zápis – Uživatelský tvar | SUBSTR(PERF30\_11[1,7],1,4)=“CCPT“ |

## Datumové funkce

|  |  |
| --- | --- |
| Funkce | Popis |
| DATE | Funkce sestaví datum dle argumentů |
| DAY | Funkce navrátí pořadí dne. |
| MONTH | Funkce navrátí pořadí měsíce. |
| YEAR | Funkce navrátí letopočet. |
| YEARFRAC | Funkce vypočítá část roku mezi dvěma letopočty. |
| ADD\_DAYS | Funkce přičte k datumu definovaný počet dní. |
| ADD\_MONTHS | Funkce přičte k datumu definovaný počet měsíců. |

### DATE

Funkce DATE navrací sestavené datum dle zadaných argumentů. Funkce je implementována ve dvou variantách. První varianta požaduje tři argumenty skládající se ze tří číselných vstupů. Druhá varianta požaduje v prvním argumentu řetězec, ze kterého je datum sestaven. Druhým argumentem je řetězec(maska), podle kterého se sestaví datum v definovaném formátu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | DATE((<číslo>, <číslo>, <číslo>)|(<řetězec1>, <řetězec2>) |
| Vstupní argumenty | <číslo> - rok, měsíc, den<řetězec1><řetězec2> - maska, podle kterého se sestaví výsledné datum. Maska m.že obsahovat yyyy (povinné), mm, dd s volitelnými oddělovači ./-. Např. “ddmmyyyy“, “dd.mm.yyyy“, “yyyy/mm/dd“. |
| Návratová hodnota | Funkce navrací sestavené datum dle zadaných argumentů. |

|  |
| --- |
| Příklad zápisu funkce(První varianta) |
| Popis kontroly | Funkce sestaví datum ze vstupních argumentů. |
| Zápis – Sémantický tvar | DATE(2018, 02, 08) |
| Zápis – Uživatelský tvar | DATE(2018, 02, 08) |

|  |
| --- |
| Příklad zápisu funkce(Druhá varianta) |
| Popis kontroly | Kontrola, že zaslané datum se rovná, nebo je menší než datum Stav ke dni. |
| Zápis – Sémantický tvar | WITH(P0008=430, P0011=050, P0013=100, P0014=001, P9902=PAN10\_11)FILTER\_CHECK(PAN10\_11)REF\_DATE() >= DATE([ANA0084], "dd.mm.yyyy"); |
| Zápis – Uživatelský tvar | FILTER\_CHECK(PAN10\_11)REF\_DATE() >= DATE(PAN10\_11[1,5], "dd.mm.yyyy"); |

### DAY

Funkce DAY navrací pořadové číslo dne ze zadaného argumentu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | DAY(<datum>) |
| Vstupní argumenty | <datum> |
| Návratová hodnota | Funkce navrací pořadí dne v měsíci dle vstupního data. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce navrátí pořadí dne. |
| Zápis – Sémantický tvar | DAY([BCP2002, P0011=050, P0013=100, P0014=001]) |
| Zápis – Uživatelský tvar | DAY(JIS51\_01[1,27]) |

### MONTH

Funkce MONTH navrací pořadové číslo měsíce ze zadaného argumentu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | MONTH(<datum>) |
| Vstupní argumenty | <datum> |
| Návratová hodnota | Funkce navrací pořadí měsíce v roce dle vstupního data. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce navrátí pořadí měsíce. |
| Zápis – Sémantický tvar | MONTH([BCP2002, P0011=050, P0013=100, P0014=001]) |
| Zápis – Uživatelský tvar | MONTH(JIS51\_01[1,27]) |

### YEAR

Funkce YEAR navrací letopočet ze zadaného argumentu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | YEAR(<datum>) |
| Vstupní argumenty | <datum> |
| Návratová hodnota | Funkce navrací rok dle vstupního data. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce navrátí letopočet. |
| Zápis – Sémantický tvar | YEAR([BCP2002, P0011=050, P0013=100, P0014=001]) |
| Zápis – Uživatelský tvar | YEAR(JIS51\_01[1,27]) |

### YEARFRAC

Funkce YEARFRAC slouží k výpočtu části roku mezi dvěma letopočty.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | YEARFRAC(<datum>, <datum>) |
| Vstupní argumenty | <datum> |
| Návratová hodnota | Funkce navrací vypočítanou část roku mezi dvěma daty. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, že je rozdíl dat větší než 0. |
| Zápis – Sémantický tvar | YEARFRAC([BCP2002], [BCP2003]) > 0; |
| Zápis – Uživatelský tvar | YEARFRAC(JIS51\_01[1,27], JIS51\_01[1,28])>0; |

### ADD\_DAYS

Funkce ADD\_DAYS slouží k přičtení definovaného počtu dní k zadanému datu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ADD\_DAYS(<datum>, <číslo>) |
| Vstupní argumenty | <datum><číslo> - počet přičítaných dní. |
| Návratová hodnota | Funkce navrací vypočítané datum. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Datum vzniku musí být o 30 dní větší, než Smluvní datum. |
| Zápis – Sémantický tvar | WITH(P0008=430, P0011=050, P0013=100, P0014=001, P9902=PAN02\_11)[ANA0004] > ADD\_DAYS(DATE([ANA0006], "dd.mm.yyyy"), 30); |
| Zápis – Uživatelský tvar | PAN02\_11[1,7] > ADD\_DAYS(DATE(PAN02\_11[1,9], "dd.mm.yyyy"), 30); |

### ADD\_MONTHS

Funkce ADD\_MONTHS slouží k přičtení definovaného počtu měsíců k zadanému datu.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ADD\_DAYS(<datum>, <číslo>) |
| Vstupní argumenty | <datum><číslo> - počet přičítaných měsíců. |
| Návratová hodnota | Funkce navrací vypočítané datum. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Datum vzniku musí být o 3 měsíce větší, než Smluvní datum. |
| Zápis – Sémantický tvar | WITH(P0008=430, P0011=050, P0013=100, P0014=001, P9902=PAN02\_11)[ANA0004] > ADD\_MONTHS(DATE([ANA0006], "dd.mm.yyyy"), 3); |
| Zápis – Uživatelský tvar | PAN02\_11[1,7] > ADD\_MONTHS(DATE(PAN02\_11[1,9], "dd.mm.yyyy"), 3); |

### REF\_DATE

Funkce REF\_DATE je speciální funkce která navrací datum odpovídající stavu ke dni daného vydání.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | REF\_DATE() |
| Vstupní argumenty |  |
| Návratová hodnota | Funkce navrací datum Stav ke dni daného vydání. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda se Stav ke dni rovná datu zaslanému v zadaném údaji. |
| Zápis – Sémantický tvar | REF\_DATE() = [di1043]; |
| Zápis – Uživatelský tvar | REF\_DATE() = 'S.01.02.01.01'[R0090,C0010]; |

### REF\_RCVD\_TIME

Funkce REF\_RCVD\_TIME je speciální funkce která navrací datum a čas odpovídající datu a času přijetí vstupní zprávy.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | REF\_RCVD\_TIME() |
| Vstupní argumenty |  |
| Návratová hodnota | Funkce navrací datum a čas přijetí vstupní zprávy. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda Datum a čas obchodování je roven, nebo menší než datum přijetí vstupní zprávy. |
| Zápis – Sémantický tvar | FILTER\_CHECK(TRAF10\_11)ISO\_DATETIME([FIM0041, P0556=NEWT, P0854=STRING, P9902=TRAF10\_11, T0024=NEWT]) <= REF\_RCVD\_TIME(); |
| Zápis – Uživatelský tvar | FILTER\_CHECK(TRAF10\_11)ISO\_DATETIME(TRAF10\_11[1,9]) <= REF\_RCVD\_TIME(); |

### ISO\_DATETIME

Funkce ISO\_DATETIME je speciální funkce která navrací datetime v lokální zóně.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ISO\_DATETIME(<řetězec>) |
| Vstupní argumenty | <řetězec> - řetězec v ISO-8601 date-time formátu (offset je volitelný) např. 2021-10-19T06:29:17.0000Z. |
| Návratová hodnota | Funkce navrací datum a čas. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda Datum a čas obchodování je roven, nebo menší než datum přijetí vstupní zprávy. |
| Zápis – Sémantický tvar | FILTER\_CHECK(TRAF10\_11)ISO\_DATETIME([FIM0041, P0556=NEWT, P0854=STRING, P9902=TRAF10\_11, T0024=NEWT]) <= REF\_RCVD\_TIME(); |
| Zápis – Uživatelský tvar | FILTER\_CHECK(TRAF10\_11)ISO\_DATETIME(TRAF10\_11[1,9]) <= REF\_RCVD\_TIME(); |

## Množinové funkce

|  |  |
| --- | --- |
| Funkce | Popis |
| COUNT | Počet číselných argumentů. |
| COUNTA | Počet neprázdných argumentů. |
| MAX | Maximální hodnota ze zadaných argumentů. |
| MIN | Minimální hodnota ze zadaných argumentů. |

### COUNT

Funkce COUNT slouží pro zjištění počtu not-null hodnot ze zadaných argumentů.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | COUNT(<any>   [ { , <any> } ]) |
| Vstupní argumenty | <any> |
| Návratová hodnota | Funkce navrací počet číselných argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce navrátí počet neprázdných hodnot z definovaných Údajů. |
| Zápis – Sémantický tvar | COUNT([ABD0559, P0236=S\_ESACNB], [ABD0560, P0236=S\_RNUVINMRB], [ABD0561, P0236=S\_RNVLAOMI]) |
| Zápis – Uživatelský tvar | COUNT(DIS16\_01[1,2],DIS16\_01[1,3],DIS16\_01[1,4]) |

### COUNTA

Funkce COUNTA slouží pro zjištění počtu všech hodnot (i null) ze zadaných argumentů.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | COUNTA(<any>   [ { , <any> } ]) |
| Vstupní argumenty | <any> |
| Návratová hodnota | Funkce navrací počet argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce ověří, že bylo zasláno více jak 5 dynamických záznamů. |
| Zápis – Sémantický tvar | COUNTA([ANA0001, P0008=430, P0011=050, P0013=100, P0014=001, P9902=PAN02\_11]) > 5; |
| Zápis – Uživatelský tvar | COUNTA(PAN02\_11[1,4]) > 5; |

### MAX

Funkce MAX slouží pro zjištění maximální hodnoty ze zadaných argumentů.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | MAX(<any>   [ { , <any> } ]) |
| Vstupní argumenty | <any> |
| Návratová hodnota | Funkce navrací maximální hodnotu ze vstupních argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce vybere maximum z vybraných argumentů. |
| Zápis – Sémantický tvar | MAX([mi15, \_DX="C\_25.00"], [mi192, \_DX="C\_25.00"]) |
| Zápis – Uživatelský tvar | MAX('C\_25.00'[020,060], 'C\_25.00'[020,070]) |

### MIN

Funkce MIN slouží pro zjištění minimální hodnoty ze zadaných argumentů.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | MIN(<any>   [ { , <any> } ]) |
| Vstupní argumenty | <any> |
| Návratová hodnota | Funkce navrací minimální hodnotu ze vstupních argumentů. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Funkce vybere minimum ze zadaných argumentů. |
| Zápis – Sémantický tvar | MIN([mi435, \_DX="C\_76.00.w", BAS=x12, LIQ=x190, MCY=x615], [mi435, \_DX="C\_76.00.w", BAS=x14, MCY=x627]) |
| Zápis – Uživatelský tvar | MIN(´C\_76.00.w´[310, 010], ´C\_76.00.w´[300, 010]) |

## Speciální filtrační funkce

### FILTER\_FAIL

Specifická funkce FILTER\_FAIL slouží pro kontrolu nepovolených kombinací hodnot dynamických parametrů v rámci Datové oblasti.

Kontrola se provádí pro všechny vykázané dynamické řádky specifikované datové oblasti.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | FILTER\_FAIL( <DatovaOblast> , <parKonkr >, <parKonkr > [ { , <parKonkr > } ] ) |
| Vstupní argumenty | <datovaOblast> - identifikátor datové oblasti<parKonkr> - konkretizace dynamického parametru |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že nebyla zaslána nepovolená kombinace.
* FALSE – v případě, že byla zaslána nepovolená kombinace.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola nepovolených kombinací hodnot dynamických parametrů P0236 a P0125 v datové oblasti RIS22\_01. |
| Zápis – Sémantický tvar | FILTER\_FAIL(RIS22\_01, P0236:@Z\_REZBVD, P0125:@ZBEZCZ4); |
| Zápis – Uživatelský tvar | FILTER\_FAIL(RIS22\_01, P0236:@Z\_REZBVD, P0125:@ZBEZCZ4); |

### FILTER

Specifická klausule FILTER je zavedena pro definování filtru, který je tvořen množinou podmínek, který lze použít v kontrolním výrazu k vymezení množiny Dynamických instancí (řádků, sloupců, karet), pro které se má příslušná část kontrolního výrazu (funkce) provést.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | FILTER ( [ <datovaOblast>] <vyraz> [ { , vyraz> } ] ) |
| Vstupní argumenty | <datovaOblast> - identifikátor datové oblasti<vyraz> - je výraz definující filtrační podmínku Dynamických instancí, která může být definována konkretizací Dynamických parametrů nebo libovolným výrazem. |
| Návratová hodnota | Množina dynamických instancí řádků ( sloupců, karet). |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Filter dynamických řádků datové oblast JIS51\_01, kde hodnota definovaného údaje [1,34] = CZK.Nad danými řádky se provede suma hodnot údaje [1,33] a suma hodnot údaje [1,32], které se porovnají. |
| Zápis – Sémantický tvar | $F1:= FILTER([BCP2195, P0011=050, P0013=100, P0014=001] = CZK);WITH(P0011=050, P0013=100, P0014=001)SUM(FILTER\_APPLY([BCP2194], $F1)) > 0 MARGIN 2; |
| Zápis – Uživatelský tvar | $F1 := FILTER(JIS51\_01[1,34] = CZK);SUM(JIS51\_01[1,33, $F1]) = SUM(JIS51\_01[1,32, $F1]) MARGIN 2; |

## Speciální funkce s intervalovou aritmetikou

Skupina funkcí pro vyhodnocení relačního výrazu s použitím tzv. intervalové aritmetiky.

Tyto funkce jsou použity v kontrolách výkazů definovaných podle taxonomií EBA a EIOPA.

Intervalová aritmetika je vysvětlena např. zde <http://faq.eurofiling.info/decimals/>

|  |  |
| --- | --- |
| Funkce | Popis |
| IAF\_EQ | = |
| IAF\_GT | > |
| IAF\_GTE | >= |
| IAF\_LT | < |
| IAF\_LTE | <= |
| IAF\_NEQ | <> |

### IAF\_EQ

Funkce vyhodnocuje vztah = (rovno) podle intervalové aritmetiky.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | IAF\_EQ(<číslo>, <číslo>) |
| Vstupní argumenty | <číslo>* Konstanta
* Údaj s číselným datovým typem.
* Funkce, nebo výraz navracející číselnou hodnotu.
 |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – splněna rovnost podle intervalové aritmetiky
* FALSE – nesplněna rovnost podle intervalové aritmetiky
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda suma hodnot Údajů je rovna hodnotě Údaje podle intervalové aritmetiky. |
| Zápis – Sémantický tvar | IAF\_EQ (SUM ([mi290, \_DX="F\_36.01.b", MCG=x25], [mi290, \_DX="F\_36.02.b", MCG=x465]), [mi290, \_DX="F\_32.04.a"]); |
| Zápis – Uživatelský tvar | IAF\_EQ(SUM('F\_36.01.b'[180,180], 'F\_36.02.b'[180,190]), 'F\_32.04.a'[120,010]); |

### IAF\_GT

Funkce vyhodnocuje vztah > (větší) podle intervalové aritmetiky.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | IAF\_GT(<číslo>, <číslo>) |
| Vstupní argumenty | <číslo>* Konstanta
* Údaj s číselným datovým typem.
* Funkce, nebo výraz navracející číselnou hodnotu.
 |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – splněna relace větší podle intervalové aritmetiky
* FALSE – nesplněna relace větší podle intervalové aritmetiky.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda suma hodnot Údajů je větší než hodnota Údaje podle intervalové aritmetiky. |
| Zápis – Sémantický tvar | IAF\_GT(SUM([mi290, \_DX="F\_36.01.b", MCG=x25], [mi290, \_DX="F\_36.02.b", MCG=x465]), [mi290, \_DX="F\_32.04.a"]); |
| Zápis – Uživatelský tvar | IAF\_GT(SUM('F\_36.01.b'[180,180], 'F\_36.02.b'[180,190]), 'F\_32.04.a'[120,010]); |

### IAF\_GTE

Funkce vyhodnocuje vztah >= (větší rovno) podle intervalové aritmetiky.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | IAF\_GTE(<číslo>, <číslo>) |
| Vstupní argumenty | <číslo>* Konstanta
* Údaj s číselným datovým typem.
* Funkce, nebo výraz navracející číselnou hodnotu.
 |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – splněna relace větší rovno podle intervalové aritmetiky
* FALSE – nesplněna relace větší rovno podle intervalové aritmetiky.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda suma hodnot Údajů je větší rovno než hodnota Údaje podle intervalové aritmetiky. |
| Zápis – Sémantický tvar | IAF\_GTE(SUM([mi290, \_DX="F\_36.01.b", MCG=x25], [mi290, \_DX="F\_36.02.b", MCG=x465]), [mi290, \_DX="F\_32.04.a"]); |
| Zápis – Uživatelský tvar | IAF\_GTE(SUM('F\_36.01.b'[180,180], 'F\_36.02.b'[180,190]), 'F\_32.04.a'[120,010]); |

### IAF\_LT

Funkce vyhodnocuje vztah < (menší) podle intervalové aritmetiky.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | IAF\_LT(<číslo>, <číslo>) |
| Vstupní argumenty | <číslo>* Konstanta
* Údaj s číselným datovým typem.
* Funkce, nebo výraz navracející číselnou hodnotu.
 |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – splněna relace menší podle intervalové aritmetiky
* FALSE – nesplněna relace menší podle intervalové aritmetiky.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda suma hodnot Údajů je menší než hodnota Údaje podle intervalové aritmetiky. |
| Zápis – Sémantický tvar | IAF\_LT(SUM([mi290, \_DX="F\_36.01.b", MCG=x25], [mi290, \_DX="F\_36.02.b", MCG=x465]), [mi290, \_DX="F\_32.04.a"]); |
| Zápis – Uživatelský tvar | IAF\_LT(SUM('F\_36.01.b'[180,180], 'F\_36.02.b'[180,190]), 'F\_32.04.a'[120,010]); |

### IAF\_LTE

Funkce vyhodnocuje vztah <= (menší rovno) podle intervalové aritmetiky.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | IAF\_LTE(<číslo>, <číslo>) |
| Vstupní argumenty | <číslo>* Konstanta
* Údaj s číselným datovým typem.
* Funkce nebo výraz navracející číselnou hodnotu.
 |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – splněna relace větší rovno podle intervalové aritmetiky
* FALSE – nesplněna relace větší rovno podle intervalové aritmetiky.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda suma hodnot Údajů je menší rovno než hodnota Údaje podle intervalové aritmetiky. |
| Zápis – Sémantický tvar | IAF\_LTE(SUM([mi290, \_DX="F\_36.01.b", MCG=x25], [mi290, \_DX="F\_36.02.b", MCG=x465]), [mi290, \_DX="F\_32.04.a"]); |
| Zápis – Uživatelský tvar | IAF\_LTE(SUM('F\_36.01.b'[180,180], 'F\_36.02.b'[180,190]), 'F\_32.04.a'[120,010]); |

### IAF\_NEQ

Funkce vyhodnocuje vztah <> (nerovno) podle intervalové aritmetiky.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | IAF\_NEQ(<číslo>, <číslo>) |
| Vstupní argumenty | <číslo>* Konstanta
* Údaj s číselným datovým typem.
* Funkce, nebo výraz navracející číselnou hodnotu.
 |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – splněna nerovnost podle intervalové aritmetiky
* FALSE – nesplněna nerovnost podle intervalové aritmetiky
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda suma hodnot Údajů není rovna hodnotě Údaje podle intervalové aritmetiky. |
| Zápis – Sémantický tvar | IAF\_IAF(SUM([mi290, \_DX="F\_36.01.b", MCG=x25], [mi290, \_DX="F\_36.02.b", MCG=x465]), [mi290, \_DX="F\_32.04.a"]); |
| Zápis – Uživatelský tvar | IAF\_IAF(SUM('F\_36.01.b'[180,180], 'F\_36.02.b'[180,190]), 'F\_32.04.a'[120,010]); |

##

## Ostatní speciální funkce

|  |  |
| --- | --- |
| Funkce | Popis |
| LINREG | Kontrola odchýlení hodnoty údaje od lineární regresní přímky. |
| CHECK\_DYNCOUNT\_MAX | Maximální počet vykázaných dynamických instancí (řádků, sloupců, karet). |
| CHECK\_DYNCOUNT\_MIN | Minimální počet vykázaných dynamických instancí (řádků, sloupců, karet). |

### LINREG

Funkce pro kontroluodchýlení hodnoty údaje od lineární regresní přímky. Zjišťuje, zda Hodnota údaje ze zpracovávaného Vydání výskytu výkazu leží v intervalu spolehlivosti hodnoty predikované na základě minulých hodnot. Principem metody je proložení přímky posledními 3 až 5 hodnotami a odvození přípustné odchylky na základě metody lineární regresní analýzy. Časová řada je zde určena počtem dní mezi jednotlivými výskyty.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | LINREG(<údaj>) |
| Vstupní argumenty | <údaj> - údaj, pro který se provádí kontrola hodnot v časové řadě |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zaslaná hodnota daného údaje vyhovuje odchylce vůči předchozím zaslaným hodnotám. |
| Zápis – Sémantický tvar | LINREG([EBD0967, P0009=CZK, P0011=120, P0013=100, P0014=001, P0019=S\_VSEMEN, P0025=S\_VSEZEME]); |
| Zápis – Uživatelský tvar | LINREG(DIS60\_02[1, 1]); |

### CHECK\_DYNCOUNT\_MAX

Funkce CHECK\_DYNCOUNT\_MAX slouží pro kontrolu maximálního počtu dynamických instancí. Pokud Datová oblast obsahuje více, než jednu dynamickou osu, kontrola probíhá vždy pouze na dynamickými instancemi pouze na první dynamické ose v  pořadí: Z, Y, X.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | CHECK\_DYNCOUNT\_MAX(<datovaOblast>, <konstanta>) |
| Vstupní argumenty | <datovaOblast><konstanta> |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda je maximální počet dynamických řádků 2. |
| Zápis – Sémantický tvar | CHECK\_DYNCOUNT\_MAX(DIS16\_01, 2) |
| Zápis – Uživatelský tvar | CHECK\_DYNCOUNT\_MAX(DIS16\_01, 2) |

### CHECK\_DYNCOUNT\_MIN

Funkce CHECK\_DYNCOUNT\_MIN slouží pro kontrolu minimálního počtu dynamických instancí. Pokud Datová oblast obsahuje více, než jednu dynamickou osu, kontrola probíhá vždy pouze na první dynamické ose v  pořadí: Z, Y, X.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | CHECK\_DYNCOUNT\_MIN(<datovaOblast>, <konstanta>) |
| Vstupní argumenty | <datovaOblast><konstanta> |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda je minimální počet řádků 1. |
| Zápis – Sémantický tvar | CHECK\_DYNCOUNT\_MIN(DIS16\_01, 1) |
| Zápis – Uživatelský tvar | CHECK\_DYNCOUNT\_MIN(DIS16\_01, 1) |

### ISIN\_CHECK

Funkce ISIN\_CHECK je speciální funkce která se provádí nad kódy typu ISIN. Implementace vychází z definice v EIOPA Taxonomii.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ISIN\_CHECK(<řetězec>) |
| Vstupní argumenty | <řetězec> |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda zaslaná hodnota odpovídá kódu ISIN. |
| Zápis – Sémantický tvar | FILTER\_CHECK('S.08.01.01.01')ISIN\_CHECK(SUBSTR([[IW]], 6, 12)); |
| Zápis – Uživatelský tvar | FILTER\_CHECK('S.08.01.01.01')ISIN\_CHECK(SUBSTR('S.08.01.01.01'[1,C0090], 6, 12)); |

### LEI\_CHECK

Funkce LEI\_CHECK je speciální funkce která se provádí nad kódy typu LEI. Implementace vychází z definice v EIOPA Taxonomii.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | LEI\_CHECK(<řetězec>) |
| Vstupní argumenty | <řetězec> |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda zaslaná hodnota odpovídá kódu LEI. |
| Zápis – Sémantický tvar | FILTER\_CHECK('S.30.04.01.01')LEI\_CHECK(SUBSTR([[CV]], 5, 20)); |
| Zápis – Uživatelský tvar | FILTER\_CHECK('S.30.04.01.01')LEI\_CHECK(SUBSTR('S.30.04.01.01'[1,C0140], 5, 20)); |

### ICO\_CHECK

Funkce ICO\_CHECK je speciální funkce která se provádí nad kódy typu IČO. Tato funkce ověřuje validitu zaslaného kódu IČO.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | ICO\_CHECK(<řetězec>) |
| Vstupní argumenty | <řetězec> |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda zaslaná hodnota odpovídá kódu IČO. |
| Zápis – Sémantický tvar | WITH(P0011=050, P0013=100, P0014=001)FILTER\_CHECK(JIS51\_01)ICO\_CHECK([BCP2182]) |
| Zápis – Uživatelský tvar | FILTER\_CHECK(JIS51\_01)ICO\_CHECK(JIS51\_01[1,16]); |

### DATA\_FILING\_EXISTS

Funkce DATA\_FILING\_EXISTS je speciální funkce, která kontroluje zaslání datových oblastí podle zadaného template.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | DATA\_FILLING\_EXISTS(<řetězec>) |
| Vstupní argumenty | <řetězec> - kódy datových oblastí, nebo jejich template (v případě EIOPA) |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola zaslání datových oblastí s templatem S.03.01 v případě vyplnění hodnoty x1 pro ukazatel ei1674. |
| Zápis – Sémantický tvar | IF([ei1674] = x1, DATA\_FILING\_EXISTS("S.03.01"), TRUE); |
| Zápis – Uživatelský tvar | IF('SE.01.01.16.01'[R0060,C0010] = x1,DATA\_FILING\_EXISTS("S.03.01"), TRUE); |

### REF\_OSOBA\_LEI

Funkce REF\_OSOBA\_LEI je speciální funkce, která navrací LEI kód vykazující osoby, který je registrovaný v SDAT.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | REF\_OSOBA\_LEI() |
| Vstupní argumenty |  |
| Návratová hodnota | Funkce navrací LEI kód registrovaný v SDAT. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola ověřuje, že část řetězce z JIS51\_01[1, 19] se rovná kódu LEI vykazující osoby.  |
| Zápis – Sémantický tvar | REF\_OSOBA\_LEI()=SUBSTR([BCP2147], 1, 20); |
| Zápis – Uživatelský tvar | REF\_OSOBA\_LEI()=SUBSTR(JIS51\_01[1, 19], 1, 20); |

### REF\_LEI\_ALLOWED

Funkce REF\_LEI\_ALLOWED je speciální funkce, která ověří oprávněnost zaslání kódu LEI podle konfigurace zasílající osoby v registru osob:

* pro Osoby, které nejsou v Registru osob označeny jako ARM nebo Obchodní systém je kontrolováno, že vykázaná hodnota odpovídá LEI vykazující Osoby.
* pro Osoby označené jako ARM nebo Obchodní systém je kontrolováno, že vykázaná hodnota odpovídá LEI Osob, za které jsou oprávněni transakce zasílat.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | REF\_LEI\_ALLOWED(<řetězec>) |
| Vstupní argumenty | <řetězec> |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola ověřuje, že vykázaná hodnota v R0012 odpovídá LEI vykazující Osoby. |
| Zápis – Sémantický tvar | FILTER\_CHECK(TRAF10\_11)REF\_LEI\_ALLOWED([[R0012]]); |
| Zápis – Uživatelský tvar | FILTER\_CHECK(TRAF10\_11)REF\_LEI\_ALLOWED(TRAF10\_11[1,3]); |

### REF\_OSOBA\_KOD

Funkce REF\_OSOBA\_KOD je speciální funkce která navrací kód vykazující osoby, který je registrovaný v SDAT.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | REF\_OSOBA\_KOD() |
| Vstupní argumenty |  |
| Návratová hodnota | Funkce navrací kód registrovaný v SDAT. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola ověřuje, že část řetězce z JIS51\_01[1, 19] se rovná kódu vykazující osoby.  |
| Zápis – Sémantický tvar | REF\_OSOBA()=SUBSTR([BCP2147], 1, 4); |
| Zápis – Uživatelský tvar | REF\_OSOBA()=SUBSTR(JIS51\_01[1, 19], 1, 4); |

### DOM\_CONTENT

Funkce DOM\_CONTENT je speciální funkce, která navrací seznam položek z definované domény číselníku.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | DOM\_CONTENT (<řetězec1>, <řetězec2>) |
| Vstupní argumenty | <řetězec1> - kód číselníku, který je použitý v požadované doméně<řetězec2> - kód domény číselníku |
| Návratová hodnota | Funkce navrací množinu položek z definované domény číselníku. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola ověřuje, že část řetězce z JIS51\_01[1, 19] je v seznamu položek z domény Z\_ANCRDT\_RMST.  |
| Zápis – Sémantický tvar | SUBSTR([ANA0067], 1, 2) IN DOM\_CONTENT(BA0025, Z\_ANCRDT\_RMST); |
| Zápis – Uživatelský tvar | SUBSTR(PANA07\_11[1, 12], 1, 2) IN DOM\_CONTENT(BA0025, Z\_ANCRDT\_RMST); |

### FILTER\_UNIQUE

Funkce FILTER\_UNIQUE je speciální funkce, která ověří unikátnost definované množiny parametrů anebo ukazatelů v rámci dynamického rozvoje.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | FILTER\_UNIQUE (<kodDO>, < parKonkr >|<ukazatel> [ { , < parKonkr >|<ukazatel> } ]) |
| Vstupní argumenty | < parKonkr > - konkretizace dynamického parametru<ukazatel> - konkretizace ukazatele |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kombinace hodnot údajů Identifikace investičního nástroje (JIS51\_01.BCP2139, JIS51\_01.BCP2141, JIS51\_01.BCP2142), Měna emise cenného papíru (JIS51\_01.P0019), Identifikace zákazníka (JIS51\_01.BCP2146, JIS51\_01.BCP2147, JIS51\_01.BCP2149), Typ vztahu na zákazníka (JIS51\_01.P1505) a Měna účtu zákazníka (JIS51\_01.BCP2195) musí být unikátní v rámci výskytu. |
| Zápis – Sémantický tvar | WITH(P0011=050, P0013=100, P0014=001)FILTER\_UNIQUE(JIS51\_01, [BCP2139], [BCP2141], [BCP2142], [[P0019]], [BCP2146], [BCP2147], [BCP2149], [[P1505]], [BCP2195]); |
| Zápis – Uživatelský tvar | FILTER\_UNIQUE(JIS51\_01, JIS51\_01[1,10], JIS51\_01[1,11], JIS51\_01[1,12], JIS51\_01[1,2], JIS51\_01[1,18], JIS51\_01[1,19], JIS51\_01[1,21], JIS51\_01[1,8], JIS51\_01[1,34]); |

### REF\_ROZSAH

Funkce REF\_ROZSAH je speciální funkce, která navrací rozsah kontrolovaného vydání.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | REF\_ROZSAH() |
| Vstupní argumenty |  |
| Návratová hodnota | Funkce navrací kód ROZSAH kontrolovaného vydání. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola ověřuje, že v případě rozsahu CON je v údaji 'C\_00.01'[0020,0010] vykázána hodnota "x7". |
| Zápis – Sémantický tvar | IF(REF\_ROZSAH() = "CON", [ei207, \_DX="C\_00.01", BAS=x17] = "x7", TRUE); |
| Zápis – Uživatelský tvar | IF(REF\_ROZSAH() = "CON", 'C\_00.01'[0020,0010] = "x7", TRUE); |

## Speciální JOIN funkce

V této kapitole jsou popsány speciální funkce, které jsou využívány v případech, kdy se provádí kontrola mezi dvěma a více datovými zdroji a je nutné držet integritu mezi definovanými klíči jednotlivých záznamů těchto zdrojů. Pravidla a specifika zápisu těchto funkcí jsou popsány v kapitole 4.1.7 a 5.1.7.

### JOIN\_EXISTS

Funkce JOIN\_EXISTS slouží pro ověření existence hodnot mezi dvěma datovými zdroji při splnění definované podmínky.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | JOIN\_EXISTS(<rowSet1>,<rowSet2>,  <logVyraz> ) |
| Vstupní argumenty | <rowSet1><rowSet2><logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj. |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka a existence.
* FALSE – v případě, že není splněna podmínka a existence.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Pokud existuje záznam v tabulce "Instrument", potom musí existovat i odpovídající záznam v tabulce "Counterparty-Instrument", kde atribut "Role protistrany" nabývá hodnoty 'Dlužník'. |
| Zápis – Sémantický tvar | JOIN\_EXISTS(PAN02\_11,  PAN04\_11,  AND(Y->P1350 = "2",  X->R0033 = Y->R0033,  X->R0030 = Y->R0030,  X->R0031 = Y->R0031)); |
| Zápis – Uživatelský tvar | JOIN\_EXISTS(PAN02\_11, PAN04\_11, AND(Y->5 = "2", X->1 = Y->1, X->2 = Y->3, X->3 = Y->4)); |

### JOIN\_CHECK

Funkce JOIN\_CHECK provádí kontrolní výraz mezi odpovídajícími si záznamy definovaných datových zdrojů. Kontrolní výraz je proveden pouze v případě, že vazební klíč existuje v obou datových zdrojích. V případě, že klíč z datového zdroje X není nalezen v datovém zdroji Y, není tento záznam vybrán a kontrola se nad ním neprovede.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | JOIN\_CHECK(<rowSet1>,<rowSet2>, <logVyraz>, <logVyraz2>) |
| Vstupní argumenty | <rowSet1><rowSet2><logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.<logVyraz2> - kontrolní výraz na záznamy splňujícími podmínky definované ve <vyraz>. |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka a existence.
* FALSE – v případě, že není splněna podmínka a existence.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Je-li hodnota údaje Hlášení OCP (TRAF12\_11.FIM0160 = false), pak hodnota údaje Předpokládaný den vypořádání (TRAF12\_21.FIM0145) musí být >= datumová část údaje Datum a čas obchodování (TRAF12\_21.FIM0041) |
| Zápis – Sémantický tvar | JOIN\_CHECK(TRAF12\_11,  TRAF12\_21, AND( X->FIM0160 = "false",  X->R0011 = Y->R0011, X->R0012 = Y->R0012), Y->FIM0145>=Y->FIM0041); |
| Zápis – Uživatelský tvar | JOIN\_CHECK(TRAF12\_11, TRAF12\_21, AND(X->6 = "false", X->1 = Y->1, X->2 = Y->2), Y->3 >= Y->7); |

### JOIN\_CHECK\_OUTER

Funkce JOIN\_CHECK\_OUTER provádí kontrolní výraz mezi odpovídajícími si záznamy definovaných datových zdrojů. Oproti klasickému JOIN\_CHECK\_OU TER v případě neexistující vazby v datovém zdroji Y jsou doplněny prázdné hodnoty a následně proveden kontrolní výraz.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | JOIN\_CHECK\_OUTER(<rowSet1>,<rowSet2>,  <logVyraz> ) |
| Vstupní argumenty | <rowSet1><rowSet2><logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj. |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka a existence.
* FALSE – v případě, že není splněna podmínka a existence.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | If {S.06.03,c0060}<>empty then {S.06.02,c0290}<>empty |
| Zápis – Sémantický tvar | $F1:=FILTER('S.06.03.01.01', NOT(ISBLANK([mi228, SU=x168])));JOIN\_CHECK\_OUTER($F1, 'SE.06.02.16.02',X->UI = Y->UI,NOT(ISBLANK(Y->si1554))); |
| Zápis – Uživatelský tvar | $F1 := FILTER('S.06.03.01.01', NOT(ISBLANK('S.06.03.01.01'[1,C0060])));JOIN\_CHECK\_OUTER($F1, 'SE.06.02.16.02', X->C0010 = Y->C0040, NOT(ISBLANK(Y->C0290))); |

### JOIN\_FETCH

Funkce JOIN\_FETCH definuje odvozený datový zdroj typu RowSet, který je dále používán v kontrolním výrazu. Datový zdroj je odvozen pouze v případě, že vazební klíč existuje v obou datových zdrojích. Nad tímto odvozeným datovým zdrojem mohou být prováděny agregační funkce nebo mohou být používány jednotlivé záznamy. Tato funkce se používá v případě, kdy je nutné vzájemně kontrolovat více než dva datové zdroje.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | $AA:=JOIN\_FETCH(<rowSet1>, <rowSet2>, <logVyraz>, <logVyraz2> [ { , logVyraz2> } ]) |
| Vstupní argumenty | <rowSet1> - kód prvního datového zdroje (datové oblasti nebo filtru).<rowSet2> - kód druhého datového zdroje (datové oblasti nebo filtru).<logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.<logVyraz2> - definice odvozeného datového zdroje typu RowSet pomocí dereference. |
| Návratová hodnota | * Rowset – navrací výstupní datový zdroj typu RowSet, který splňuje podmínky a je dále použit v zápisu kontroly.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Filter: ({S.06.02, c0290} like ‘##91’ or {S.06.02, c0290} like ‘##92’ or {S.06.02, c0290} like ‘##94’ or {S.06.02, c0290} like ‘##99’) and ({S.06.02, c0090}=[s2c\_LB:x91]); Expression: {S.02.01, r0080,c0010}=sum({S.06.02, c0170,(sNNN)}) |
| Zápis – Sémantický tvar | $F1:=JOIN\_FETCH('SE.06.02.16.01','SE.06.02.16.02',AND(REGEXP\_LIKE(Y->si1554, "^..((91)|(92)|(94)|(99))$"),X->ei1070 = "x91",X->UI = Y->UI),X->mi1131);IAF\_EQ([mi265, \_DX="SE.02.01.16.01", BL=x91, VG=x80], SUM(F1->\_1 )); |
| Zápis – Uživatelský tvar | $F1 := JOIN\_FETCH('SE.06.02.16.01', 'SE.06.02.16.02', AND(REGEXP\_LIKE(Y->C0290, "^..((91)|(92)|(94)|(99))$"), X->C0090 = "x91", X->C0040 = Y->C0040), X->C0170);IAF\_EQ('SE.02.01.16.01'[R0080,C0010], SUM(F1->\_1)); |

### JOIN\_FETCH\_OUTER

Funkce JOIN\_FETCH\_OUTER definuje odvozený datový zdroj typu RowSet, který je dále používán v kontrolním výrazu. Oproti klasickému JOIN\_FETCH v případě neexistující vazby v datovém zdroji Y jsou doplněny prázdné hodnoty v datovém zdroji Y. Nad tímto odvozeným datovým zdrojem mohou být prováděny agregační funkce nebo mohou být používány jednotlivé záznamy. Tato funkce se používá v případě, kdy je nutné vzájemně kontrolovat více než dva datové zdroje.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | $AA:=JOIN\_FETCH\_OUTER(<rowSet1>, <rowSet2>, <logVyraz>, <logVyraz2> [ { , logVyraz2> } ]) |
| Vstupní argumenty | <rowSet1> - kód prvního datového zdroje (datové oblasti nebo filtru).<rowSet2> - kód druhého datového zdroje (datové oblasti nebo filtru).<logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj.<logVyraz2> - definice odvozeného datového zdroje typu RowSet pomocí dereference. |
| Návratová hodnota | * Rowset – navrací výstupní datový zdroj typu RowSet, který splňuje podmínky a je dále použit v zápisu kontroly.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Filter: {S.06.02, c0290} like '##4#';Expression: If (sum({S.06.02, c0170,(sNNN)})/({S.02.01, r0070,c0010}+{S.02.01, r0220,c0010}))>0.3 then ({S.01.01, r0150,c0010} = [s2c\_CN:x1] or {S.01.01, r0150,c0010} = [s2c\_CN:x23]) |
| Zápis – Sémantický tvar | $F1:=FILTER('SE.06.02.16.02', REGEXP\_LIKE([si1554,\_DX='SE.06.02.16.02', SU=x168], "^..4.$"));$F2:= JOIN\_FETCH\_OUTER($F1, 'SE.06.02.16.01',X->UI = Y->UI,Y->mi1131);IF((IAF\_GT(SUM(F2->\_1),((0.3 \* (SUM([mi263, \_DX="SE.02.01.16.01", BL=x91, VG=x80], [mi263, \_DX="SE.02.01.16.01", BL=x119, VG=x80])))))),OR([ei2456, \_DX="SE.01.01.16.01"] = "x1", [ei2456, \_DX="SE.01.01.16.01"] = "x23"), (TRUE));  |
| Zápis – Uživatelský tvar | $F1 := FILTER('SE.06.02.16.02', REGEXP\_LIKE('SE.06.02.16.02'[1,C0290], "^..4.$"));$F2 := JOIN\_FETCH($F1, 'SE.06.02.16.01', X->C0040 = Y->C0040, Y->C0170);IF((IAF\_GT(SUM(F2->\_1), ((0.3 \* (SUM('SE.02.01.16.01'[R0070,C0010], 'SE.02.01.16.01'[R0220,C0010])))))), OR('SE.01.01.16.01'[R0150,C0010] = "x1", 'SE.01.01.16.01'[R0150,C0010] = "x23"), (TRUE));  |

### CONS\_HIST

Funkce CONS\_HIST slouží pro možnost získání konsolidovaných dat požadované datové oblasti uvnitř jedné z funkcí JOIN. Tato funkce je použita v případě, že do kontroly nevstupují data konkrétního výskytu, ale data vymezená jiným způsobem. Data lze dále omezit ve vazební podmínce funkcí JOIN (např. omezení datem Stav ke dni, nebo hodnotou parametru/údaje).

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | CONS\_HIST(<datovaOblast> ) |
| Vstupní argumenty | <datovaOblast> |
| Návratová hodnota | Rowset – navrací výstupní datový zdroj typu RowSet, který splňuje podmínky a je dále použit v zápisu kontroly. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Pokud atribut "Stav instrumentu" nabývá hodnot 'Nová pohledávka vzniklá v bance' nebo 'Nová pohledávka vzniklá převodem od jiné banky', potom musí být složený primární klíč jedinečný v celé historii i budoucnosti (v případě zaslání revizí). |
| Zápis – Sémantický tvar | $F1:=FILTER(PAN02\_11, [ANA0026, P0008=430, P0011=050, P0013=100, P0014=001, P9902=PAN02\_11] IN {"11", "12"});JOIN\_CHECK($F1, CONS\_HIST(PAN02\_11), AND(X->R0033 = Y->R0033, X->R0030 = Y->R0030, X->R0031 = Y->R0031, OR(AND(Y->ANA0026 IN {"11", "12"}, Y->stav\_ke\_dni <> REF\_DATE()), AND(Y->ANA0026 IN {"13"}, Y->stav\_ke\_dni < REF\_DATE()))), FALSE); |
| Zápis – Uživatelský tvar | $F1 := FILTER(PAN02\_11, PAN02\_11[1,28] IN {"11", "12"});JOIN\_CHECK($F1, CONS\_HIST(PAN02\_11), AND(X->1 = Y->R0033, X->2 = Y->R0030, X->3 = Y->R0031, OR(AND(Y->ANA0026 IN {"11", "12"}, Y->stav\_ke\_dni <> REF\_DATE()), AND(Y->ANA0026 IN "13", Y->stav\_ke\_dni < REF\_DATE()))), FALSE); |

### \_ISBLANK

V tomto případě se nejedná o funkci, ale o příznak, který se používá pro zjištění, zda je pravý záznam prázdný ve funkcích JOIN\_CHECK\_OUTER a JOIN\_FETCH\_OUTER. JOIN\_CHECK\_OUTER tedy kontroluje pouze záznamy existující v levém zdroji a JOIN\_FETCH\_OUTER vrací záznamy existující pouze v levém zdroji.

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Pokud existuje záznam v tabulce "Counterparty risk", potom musí existovat buď záznam v tabulce "Counterparty-Instrument", kde je protistrana v roli "Dlužník", nebo záznam v tabulce "Protection received" v rámci atributu "Identifikátor ručitele". |
| Zápis – Sémantický tvar | JOIN\_EXISTS(JOIN\_FETCH\_OUTER(PAN09\_11, FILTER(PAN04\_11, [[P1350]]='2'), AND(X->R0033 = Y->R0033, X->R0034 = Y->R0034, X->P0379 = Y->P0379, Y->\_ISBLANK),X->R0033, X->R0034, X->P0379 ),PAN07\_11,AND(X->\_1 = Y->R0033, X->\_2 = Y->ANA0063, X->\_3 = Y->ANA0106)); |
| Zápis – Uživatelský tvar | JOIN\_EXISTS(JOIN\_FETCH\_OUTER(PAN09\_11, FILTER(PAN04\_11, PAN04\_11[1,5] = '2'), AND(X->1 = Y->1, X->2 = Y->6, X->3 = Y->2, Y->\_ISBLANK), X->1, X->2, X->3), PAN07\_11, AND(X->\_1 = Y->1, X->\_2 = Y->3, X->\_3 = Y->4)); |

## Funkce na externí číselníky

### EXT\_EXISTS

Funkce EXT\_EXISTS slouží pro ověření existence vykázané hodnoty v externím číselníku.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | EXT\_EXISTS(<extCiselnik>, <any>, <datum>) |
| Vstupní argumenty | <extCiselnik> - kód externího číselníku ve kterém se ověřuje existence dané hodnoty.<any> - hledaná hodnota v externím číselníku.<datum> - časový řez. |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že hodnota existuje v externím číselníku v definovaném časovém řezu.
* FALSE – v případě, že hodnota neexistuje v externím číselníku v definovaném časovém řezu.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola, zda část zaslaného řetězce odpovídá kódu země v externím číselníku zemí s platností odpovídající Stavu ke dni vydání. |
| Zápis – Sémantický tvar | WITH(P0011=050, P0013=100, P0014=001)FILTER\_CHECK(JIS51\_01)EXT\_EXISTS(FIRDS\_COUNTRIES, SUBSTR([BCP2149], 1, 2), REF\_DATE()) |
| Zápis – Uživatelský tvar | FILTER\_CHECK(JIS51\_01)EXT\_EXISTS(FIRDS\_COUNTRIES, SUBSTR(JIS51\_01[1,21], 1, 2), REF\_DATE()) |

### EXT\_EXISTS\_2

Funkce EXT\_EXISTS\_2 slouží pro ověření existence kombinace vykázaných hodnot v externím číselníku. Tato funkce je aktuálně implementována pouze pro ověření existence v externím číselníku FIRDS\_FININSTRUMENTS, kde se ověřuje existence kombinace kódů ISIN a MIC.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | EXT\_EXISTS\_2(<extCiselnik>, <any>, <any>, <datum>) |
| Vstupní argumenty | <extCiselnik> - kód externího číselníku ve kterém se ověřuje existence dané hodnoty – FIRDS\_FININSTRUMENTS<any> - hledaná hodnota v externím číselníku – kód ISIN<any> - hledaná hodnota v externím číselníku – kód MIC<datum> - časový řez. |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že hodnota existuje v externím číselníku v definovaném časovém řezu.
* FALSE – v případě, že hodnota neexistuje v externím číselníku v definovaném časovém řezu.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Kontrola existence kombinace kódů ISIN a MIC v externím číselníku FIRDS\_FININSTRUMENTRS v časovém bodu – Stav ke dni. |
| Zápis – Sémantický tvar | WITH(P0556=NEWT, P0854=STRING, P9902=TRAF10\_11, T0024=NEWT)EXT\_EXISTS\_2(FIRDS\_FININSTRUMENTS, [FIM0065], [FIM0058], REF\_DATE()); |
| Zápis – Uživatelský tvar | EXT\_EXISTS\_2(FIRDS\_FININSTRUMENTS, TRAF10\_11[1,33], TRAF10\_11[1,26], REF\_DATE()); |

### EXT\_RETURN

Funkce navrací požadovanou hodnotu z definovaného externího číselníku podle zadaného klíče.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | EXT\_RETURN(<extCiselnik>, <any>, <atribut>, <datum>) |
| Vstupní argumenty | <extCiselnik> - kód externího číselníku ve kterém se ověřuje existence dané hodnoty.<any> - hodnota v externím číselníku pro kterou je požadována hodnota atributu.<atribut> - požadovaný atribut v externím číselníku.<datum> - časový řez |
| Návratová hodnota | Funkce navrací požadovanou hodnotu z externího číselníku. |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Část kontroly, která pro měnu zaslanou v JIS51\_01[1,2] vrací KURS\_STRED z externího číselníku kursů s platností odpovídající Stavu ke dni vydání. |
| Zápis – Sémantický tvar | EXT\_RETURN(KURSYCNB\_ALL, [[P0019]], KURS\_STRED, REF\_DATE()) |
| Zápis – Uživatelský tvar | EXT\_RETURN(KURSYCNB\_ALL, JIS51\_01[1,2], KURS\_STRED, REF\_DATE()) |

## Speciální funkce pro AnaCredit

V této kapitole jsou popsány speciální funkce použité v kontrolách AnaCredit. Tyto funkce vycházejí z funkcí JOIN, kdy druhým datovým zdrojem je vždy RIAD, který se v zápisu funkcí neuvádí.

Pro odkazování na atributy ze zdroje RIAD slouží stejně jako ve funkcích JOIN operace dereference. Pro zdroj RIAD je vždy na levé straně použito Y a na pravé straně je použit identifikátor atributu z registru RIAD (např. Y->RIAD\_IDK). Dereference pro část RIAD je pro sémantický a uživatelský tvar stejná. Uživatelský tvar tedy obsahuje definici atributu pomocí identifikátoru (např. Y->RIAD\_IDK).

Atributy z registru RIAD mají pro potřeby SDAT mírně upravenou názvovou konvenci. Jejich názvy a mapování na RIAD jsou dostupné [v tomto souboru na listu „Mapování atributů“](https://www.cnb.cz/export/sites/cnb/cs/statistika/.galleries/anacredit/vykazovani_do_anacredit/RIAD-soubory/RIAD_Counterparty_Reference_Data.xlsx).

Druhým typem nové funkce použité výhradně pro kontroly AnaCredit je funkce COMPLETENESS.

Bližší informace ke kontrolám AnaCredit v SDAT jsou dostupné v souboru AnaCredit\_Validation\_Checks\_SDAT, který je dostupný na stránkách  [ČNB AnaCredit](https://www.cnb.cz/cs/statistika/anacredit/vykazovani-do-anacredit/) v části věnované SDAT.

### RIAD\_EXISTS

Funkce RIAD\_EXISTS slouží pro ověření existence hodnot mezi dvěma datovými zdroji při splnění definované podmínky, kdy druhým datovým zdrojem je vždy RIAD a v zápisu kontroly se neuvádí.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | RIAD\_EXISTS(<rowSet1>,  <logVyraz> ) |
| Vstupní argumenty | <rowSet1><logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj, kterým je vždy RIAD. |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka a existence.
* FALSE – v případě, že není splněna podmínka a existence.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Pokud existuje záznam v tabulce "Counterparty risk", potom musí existovat i záznam v tabulce "Counterparty reference". |
| Zápis – Sémantický tvar | RIAD\_EXISTS(PAN09\_11,  AND(X->R0034 = Y->RIAD\_IDK,  X->P0379 = CNB\_K\_OSOBA)); |
| Zápis – Uživatelský tvar | RIAD\_EXISTS(PAN09\_11, AND(X->2 = Y->RIAD\_IDK, X->3 = CNB\_K\_OSOBA)); |

### RIAD\_CHECK

Funkce RIAD\_CHECK provádí kontrolní výraz mezi odpovídajícími si záznamy definovaného datového zdroje a zdroje RIAD. Kontrolní výraz je proveden pouze v případě, že vazební klíč existuje v obou datových zdrojích. V případě, že klíč z datového zdroje X není nalezen v datovém zdroji Y, není tento záznam vybrán a kontrola se nad ním neprovede.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | RIAD\_CHECK(<rowSet1>, <logVyraz>, <logVyraz2>) |
| Vstupní argumenty | <rowSet1><logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj, kterým je vždy RIAD.<logVyraz2> - kontrolní výraz na záznamy splňujícími podmínky definované ve <vyraz>. |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka a existence.
* FALSE – v případě, že není splněna podmínka a existence.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Pokud existuje záznam v RIAD, věřitelem pro daný instrument věřitelem musí být specializovaná finanční instituce. |
| Zápis – Sémantický tvar | RIAD\_CHECK( PAN04\_11,  AND(X->R0034 = Y->RIAD\_IDK, X->P0379 = Y->CNB\_KOSOBA, X->P1350 = '1'), Y->CNB\_SEKTOR IN {"1250140", "1250240", "1250340", "2002541", "2002542", "2002543"}); |
| Zápis – Uživatelský tvar | RIAD\_CHECK(PAN04\_11, AND(X->6 = Y->RIAD\_IDK, X->2 = Y->CNB\_KOSOBA, X->5 = '1'), Y->CNB\_SEKTOR IN {"1250140", "1250240", "1250340", "2002541", "2002542", "2002543"}); |

### RIAD\_CHECK\_OUTER

Funkce RIAD\_CHECK\_OUTER provádí kontrolní výraz mezi odpovídajícími si záznamy definovaného datového zdroje a zdroje RIAD. Oproti klasickému RIAD\_CHECK\_OUTER v případě neexistující vazby v datovém zdroji Y jsou doplněny prázdné hodnoty a následně proveden kontrolní výraz.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | RIAD\_CHECK\_OUTER(<rowSet1>,  <logVyraz> ) |
| Vstupní argumenty | <rowSet1><logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj, kterým je vždy RIAD. |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka a existence.
* FALSE – v případě, že není splněna podmínka a existence.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Věřitelem pro daný instrument věřitelem musí být specializovaná finanční instituce. |
| Zápis – Sémantický tvar | RIAD\_CHECK\_OUTER( PAN04\_11,  AND(X->R0034 = Y->RIAD\_IDK, X->P0379 = Y->CNB\_KOSOBA, X->P1350 = '1'), Y->CNB\_SEKTOR IN {"1250140", "1250240", "1250340", "2002541", "2002542", "2002543"}); |
| Zápis – Uživatelský tvar | RIAD\_CHECK\_OUTER(PAN04\_11, AND(X->6 = Y->RIAD\_IDK, X->2 = Y->CNB\_KOSOBA, X->5 = '1'), Y->CNB\_SEKTOR IN {"1250140", "1250240", "1250340", "2002541", "2002542", "2002543"}); |

### RIAD\_FETCH

Funkce RIAD\_FETCH definuje odvozený datový zdroj typu RowSet, který je dále používán v kontrolním výrazu. Datový zdroj je odvozen pouze v případě, že vazební klíč existuje v obou datových zdrojích. Nad tímto odvozeným datovým zdrojem mohou být prováděny agregační funkce nebo mohou být používány jednotlivé záznamy. Tato funkce se používá v případě, kdy je nutné vzájemně kontrolovat více než dva datové zdroje.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | RIAD\_FETCH(<rowSet1>, <logVyraz>, <logVyraz2> [ { , logVyraz2> } ]) |
| Vstupní argumenty | <rowSet1> - kód prvního datového zdroje (datové oblasti nebo filtru).<logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj, kterým je vždy RIAD.<logVyraz2> - definice odvozeného datového zdroje typu RowSet pomocí dereference. |
| Návratová hodnota | * Rowset – navrací výstupní datový zdroj typu RowSet, který splňuje podmínky a je dále použit v zápisu kontroly.
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Pokud je pro daný instrument věřitelem specializovaná finanční instituce, potom je instrument součástí sekuritizace a musí být vykázána i protistrana v roli původce. |
| Zápis – Sémantický tvar | JOIN\_EXISTS(RIAD\_FETCH( PAN04\_11,  AND(X->R0034 = Y->RIAD\_IDK, X->P0379 = Y->CNB\_KOSOBA, X->P1350 = '1', Y->CNB\_SEKTOR IN {"1250140", "1250240", "1250340", "2002541", "2002542", "2002543"}),  X->R0034, X->R0030, X->R0031, X->R0033, X->P0379),PAN04\_11,AND(X->\_1 = Y->R0034, X->\_2 = Y->R0030, X->\_3 = Y->R0031, X->\_4 = Y->R0033, X->\_5 = Y->P0379, Y->P1350 = '4')); |
| Zápis – Uživatelský tvar | JOIN\_EXISTS(RIAD\_FETCH(PAN04\_11, AND(X->6 = Y->RIAD\_IDK, X->2 = Y->CNB\_KOSOBA, X->5 = '1', Y->CNB\_SEKTOR IN {"1250140", "1250240", "1250340", "2002541", "2002542", "2002543"}), X->6, X->3, X->4, X->1, X->2), PAN04\_11, AND(X->\_1 = Y->6, X->\_2 = Y->3, X->\_3 = Y->4, X->\_4 = Y->1, X->\_5 = Y->2, Y->5 = '4')); |

### RIAD\_FETCH\_OUTER

Funkce RIAD\_FETCH\_OUTER definuje odvozený datový zdroj typu RowSet, který je dále používán v kontrolním výrazu. Oproti klasickému RIAD\_FETCH v případě neexistující vazby v datovém zdroji Y jsou doplněny prázdné hodnoty v datovém zdroji Y. Nad tímto odvozeným datovým zdrojem mohou být prováděny agregační funkce nebo mohou být používány jednotlivé záznamy. Tato funkce se používá v případě, kdy je nutné vzájemně kontrolovat více než dva datové zdroje.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | RIAD\_FETCH\_OUTER(<rowSet1>, <logVyraz>, <logVyraz2> [ { , logVyraz2> } ]) |
| Vstupní argumenty | <rowSet1> - kód prvního datového zdroje (datové oblasti nebo filtru).<logVyraz> - je výraz definující relační vztah mezi datovými zdroji a případných filtračních podmínek dynamických instancí daných datových zdrojů. V definici výrazu jsou dereference na datové zdroje uváděny jako „X“ pro první zdroj, a „Y“ pro druhý zdroj, kterým je vždy RIAD<logVyraz2> - definice odvozeného datového zdroje typu RowSet pomocí dereference. |
| Návratová hodnota | * Rowset – navrací výstupní datový zdroj typu RowSet, který splňuje podmínky a je dále použit v zápisu kontroly.
 |

### COMPLETENESS

Funkce COMPLETENESS je speciální funkcí pro kontroly AnaCredit. Tato funkce je použita v kontrolách používajících podmínky Completeness. Pro zjednodušení zápisu a optimalizaci provádění těchto kontrol byla vytvořena tato funkce a její interpretace je řízena kódem podmínek. Jednotlivé kontroly jsou pak zapsány složením funkcí COMPLETENESS dle zadání kontrol. Jednotlivé podmínky a jejich vyhodnocení je detailně popsáno v souboru, na který je odkázáno v úvodu této kapitoly.

Bližší informace ke kontrolám COMPLETENESS i dalším kontrolám AnaCredit v SDAT jsou dostupné v souboru AnaCredit\_Validation\_Checks\_SDAT, který je k dispozici na stránkách  [ČNB AnaCredit](https://www.cnb.cz/cs/statistika/anacredit/vykazovani-do-anacredit/) v části SDAT - testování. Soubor obsahuje definice jednotlivých podmínek kontrol COMPLETENESS i jejich uplatnění v jednotlivých kontrolách.

|  |  |
| --- | --- |
| Vlastnosti funkce |  |
| Tvar funkce | COMPLETENESS(<řetězec>) |
| Vstupní argumenty | <řetězec> - kód požadované podmínky COMPLETENESS:* CD0020
* CD0030
* CD0039
* CD0040
* CD0041
* CD0050
* CD0060
* CD0070
 |
| Návratová hodnota | Funkce navrací hodnoty:* TRUE – v případě, že je splněna podmínka.
* FALSE – v případě, že není splněna podmínka
 |

|  |
| --- |
| Příklad zápisu funkce |
| Popis kontroly | Není-li splněna (CD0020 nebo CD0070), musí být [Instrument.Úvěr na projektové financování]<>'NTRQ'. |
| Zápis – Sémantický tvar | FILTER\_CHECK(PAN02\_11, PANACR02[ANA0002, P0008=430, P0011=050, P0013=100, P0014=001, P9902=PAN02\_11] = "NTRQ")OR(COMPLETENESS(CD0020), COMPLETENESS(CD0070)); |
| Zápis – Uživatelský tvar | FILTER\_CHECK(PAN02\_11, PANACR02:PAN02\_11[1,5] = "NTRQ")OR(COMPLETENESS(CD0020), COMPLETENESS(CD0070)); |

## PRAGMY

### #ALG()

Pragma #ALG slouží pro interpretaci algoritmických kontrol. Jejím argumentem je kód dané kontroly např. #ALG(val\_CN0010). Jako algoritmické kontroly jsou vybrány takové kontroly, které jsou složité a nelze je zapsat standardním jazykem kontrol nebo by jejich interpretace v jazyce kontrol nebyla efektivní.

### #FIRST\_REPORTING

Pragma #FIRST\_REPORTING definuje spuštění dané kontroly pouze při prvním reportingu vykazující osoby.

### #RESTRICT()

Pragma #RESTRICT slouží pro vymezení kontroly pro množinu osob.

### #SQL\_TODATE(ana)

Pragma #SQL\_TODATE slouží pouze jako interní pragma pro optimalizaci interpretace dané kontroly.

# Doplňující informace

## Prázdné hodnoty v kontrolách

V této kapitole jsou popsána pravidla vyhodnocování kontrol, pokud jsou hodnoty údajů prázdné.

Vykazované údaje mohou mít definován atribut „povinnost“, takže v některých případech je povinnost kontrolována již ve formálních kontrolách. Většina údajů je však na této úrovni definována jako nepovinné (zvláště u metodik vytvářených z externích zdrojů EBA a EIOPA, kde atribut povinnost využíván není).

Mohou tedy nastat situace, kdy je kontrola prováděna nad údaji, pro které nebyla hodnota zaslána.

Pro vyhodnocení kontrolního výrazu budou používány implicitně náhradní hodnoty podle základního datového typu údaje následujícím způsobem:

* Číselný – používá se implicitní náhradní hodnota 0
* Textový, Datumový, Položka číselníku – nepoužívá se žádná implicitní náhradní hodnota

Implicitní náhradní hodnota se nepoužívá ve funkci ISBLANK, která se může používat pro kontrolu vyplnění hodnoty.

Pro ostatní případy, kdy bude potřeba ošetřit správné provedení kontroly v případě prázdné hodnoty, je zavedena funkce je možné použít explicitní náhradní hodnotu:

* Zavedena funkce IFBLANK, která vrátí definovanou náhradní hodnotu odpovídajícího oboru hodnot údaje.

Obecně však platí, že povinnost hodnot údajů a kontrola musí být definovány tak, aby nedošlo nevyplněním hodnot a použitím náhradních hodnot k chybnému vyhodnocení věcné kontroly.

Pro provádění kontrol platí:

* Defaultně se kontrola provádí, pro prázdné údaje se aplikují náhradní hodnoty podle výše uvedených pravidel.
* Pokud nemá být kontrola provedena v případě prázdných hodnot, bude použita pragma #SKIP\_ALL\_MISSING.

## Hodnoty NULL v jazyce kontrol

V zápisu kontrol se vyskytují případy, kdy tvar kontroly obsahuje hodnotu NULL. Tento případ se vyskytuje ze tří následujících důvodů:

* Kontrola zahrnuje buňku s GAAP údajem, který je v SDAT vykřížkovaný. (v5547\_h)
* Řetězce NULL nahrazují vykřížkované buňky, které v tomto případě jsou z důvodu nerelevantního standardu GAAP.
* Takové údaje jsou v sémantickém tvaru kontroly doprovázeny komentářem, viz:



* Kontrola zahrnuje standardně vykřížkovaný údaj. (v8726\_m\_s1)
* Předpis z EBA je obecný a zní takto: [C 07.00.a (All rows, All sheets)] {c200} <= {c150}. Kontrola se tedy provádí pro všechny řádky a všechny karty ve sloupcích c200 a c150.
* Konverze kontroly proto tyto údaje označí jako NULL.
* Kontrola v8726\_m se provádí pro všechny řádky včetně řádků s NULL.
* V kontrole je buňka s NULL nahrazena nulovou hodnotou a v případě stringové buňky je nahrazena prázdným řetězcem.



                                                   

* Kontrola zahrnuje údaj z datové oblasti patřící do GAAP. (v5332\_m)
* Předpis z EBA pro kontrolu v5332\_m zní: {F 04.01, r005, c010} + {F 04.06, r005, c010} = sum({F 20.04, r010, c011, (sNNN)}).
* Kontrola obsahují řetězec „NULL“ pro datovou oblast F\_04.06, která spadá do GAAP.
* Zároveň je ve zmíněném předpisu uvedena prerekvizita pro spuštění kontroly: apply rule (only) if specified tables are reported: (F 04.01 or F 04.06) and F 20.04.
* Kontrola by tedy platí, přestože F\_04.06 není k dispozici a je nahrazena řetězcem NULL.

## Předpoklad

Atribut Předpoklad slouží k definování podmínky spuštění kontrol. Atribut Předpoklad je plněn kódy datových oblastí, nebo kódy template (EIOPA), které jsou v SDAT uloženy jako atribut Akronym.

Kódy datových oblastí, nebo template mohou být spojovány do výrazů pomocí operátorů AND, OR a s použitím závorek. Kontrola je spuštěna pouze v případě, že je splněna definovaná podmínka, tedy jsou vykázány datové oblasti definované v Předpoklad.

V případě nevyplnění atributu Předpoklad je kontrola spuštěna při zaslání alespoň jedné datové oblasti figurující v dané kontrole.