|  |
| --- |
| Řešení pro zpracování a analýzu dat o výkonech VŠE verze 2. |

|  |
| --- |
| Technická specifikace |

|  |
| --- |
|  |
| Tento dokument obsahuje 97 stran |
|  |

Kontrola a schválení dokumentu

Provedené revize

|  |  |  |  |
| --- | --- | --- | --- |
| Verze | Autor | Datum | Revize |
| 1 | Tým 7 | 30.06.2023 | Popis dokumentace v rámci datového projektu 4IT500 |
| 2 | Jakub Dvoran,  Jonáš Prokop | 07.06.2023 | Aktualizace dokumentace během Praxe v datech a analytice pro business 4IT510 |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
|  |  |  |  |

Tento dokument byl zkontrolován

|  |  |  |
| --- | --- | --- |
|  | Kontrolu provedl | Datum kontroly |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

Tento dokument byl schválen

|  |  |  |  |
| --- | --- | --- | --- |
|  | Jméno | Podpis | Datum schválení |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Shrnutí

Tento dokument slouží jako technická specifikace řešení pro zpracování a analýzu dat o výkonech VŠE.

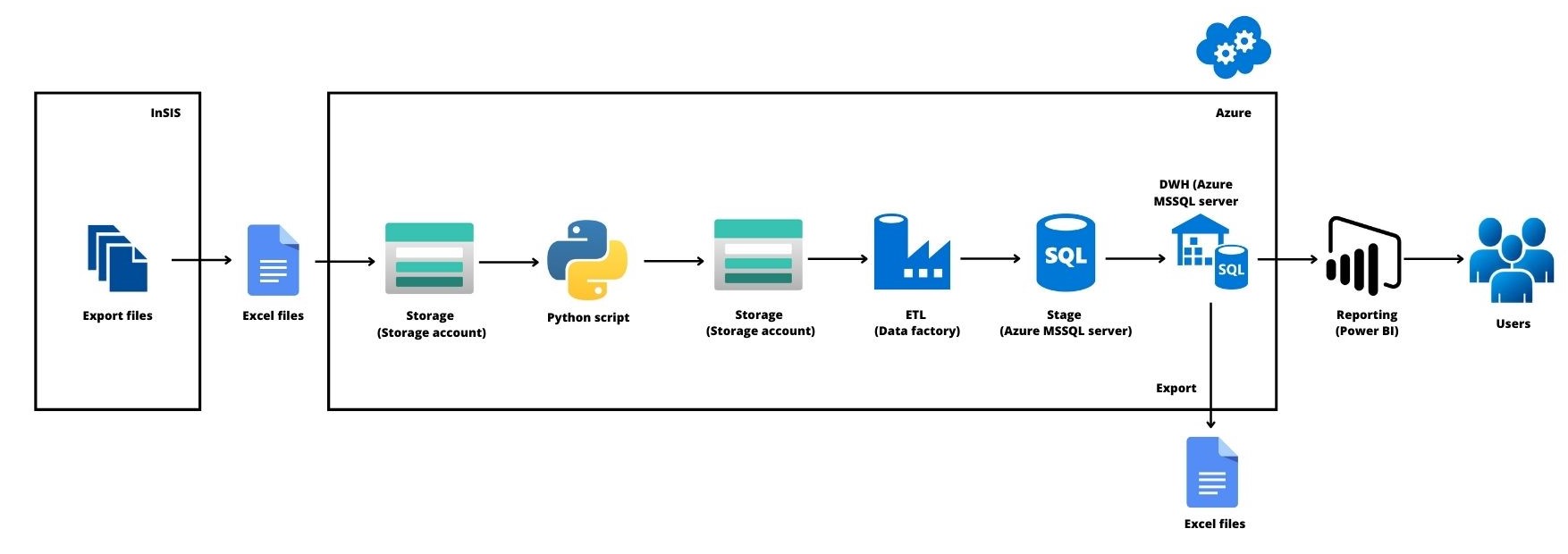
V tomto dokumentu popisujeme samotné řešení v jednotlivých vrstvách, koncepty implementace a použité technologie.

## Předpoklady a cíle

Cílem je navrhnout reportingové řešení, které umožňuje opakované automatizované zpracování dat o výkonech VŠE získaných z InSIS pro potřeby tvorby rozpočtu na úrovni jednotlivých fakult VŠE a také které slouží jako nástroj pro podporu řízení kateder a fakult VŠE.

# Architektura

Níže uvedený diagram znázorňuje pohled na architekturu řešení. Ukazuje komponenty, jejich zapojení a příklady použité technologie.



Obrázek č.1 - diagram architektury

Diagram se skládá z:

* **Systémů** – jednotlivé prostředí ve kterých jsou komponenty umístěny
* **Komponent** – jednotlivé nástroje a datové typy, které jsou použity pro tvorbu řešení
* **Činností** – představují propojení komponent (tok dat)

## Systémy architektury

**InSIS** – školní informační systém obsahuje excelové soubory, které jsou následně exponovány do prostředí azure.

**Storage –** tyto soubory jsou pak uloženy v úložišti *vsestorage* v prostředí azure. Konkrétně ve *blob services*

**Python script** –byl naprogramován ve Visual Studio Code. Jeho spuštění se provádí prostřednictvím pipeline v ADF s názvem *Python\_script*.

**ETL –** procesy ETL jsou zpracovávány prostřednictvím azure data factory.

**Stage –** jako stage databáze se používá Azure MSSQL azure.

**DWH –** datový sklad je umístěn v MS SQL server management studio.

**Reporting –** závěrečné reporty jsou zpracovávány v aplikaci Microsoft Power BI. Které jsou pak sdíleny s koncovými uživateli prostřednictvím SharePointu.

## Komponenty architektury

**Export files** – soubory exportované ze systému InSIS za určité časové období (obvykle celý rok, případně jeden semestr). Tyto soubory jsou ve formátu Excel. Následně jsou uloženy v úložišti Azure

**Storage (Storage account) –** tyto vstupní soubory jsou uloženy v azure storage account, konkrétně v blob storage kontejnery s názvem "vstupni-slozka". Azure blob storage slouží k:

* Uchování velkého objemu nestrukturovaných dat. Úložiště Blob Storage je optimalizováno pro ukládání velkého množství nestrukturovaných dat. Nestrukturovaná data jsou data, která se neřídí určitým datovým modelem nebo definicí, například textová nebo binární data.
* Kontejner organizuje sadu blobů. Účet úložiště může obsahovat neomezený počet kontejnerů a kontejner může ukládat neomezený počet blobů.

## Činnosti

**Python script –** skript v jazyce Python provádí nad vstupními daty operaci unpivot. Tento skript je již považován za operaci ETL. Poté tyto upravené tabulky uloží do úložiště blob storage kontejneru s názvem "vystupni-excel-data". Podrobnější fungování tohoto skriptu je popsáno v kapitole 4 a 8.

**ETL (Data factory) -** následně jsou tato výstupní data z předchozího kroku dále zpracována v azure data factory. Azure data factory je:

* spravovaná cloudová služba pro integraci dat. Usnadňuje vytváření, plánování a monitorování datových pump a pracovních postupů ETL/ELT.

Data factory byla použita pro intuitivní uživatelské rozhraní, transformaci dat bez použití kódu a finančně výhodné řešení.

**Stage (Azure MSSQL Server) -** pak jsou data procesována do podoby SQL tabulek a uložena ve Stage databázi v původní struktuře.

**DWH (Azure MSSQL Server)** - po tom jsou data modelována do podoby relačních tabulek a nahrána do datawarehousu (datového skladu) odkud je můžeme exportovat.

**Power BI** – data z datového skladu jsou následně propojena s aplikací Power BI, kde jsou vytvořeny finální dashboardy pro koncové uživatele.

# Datové zdroje

Tato kapitola popisuje zdroj dat potřebný k vytvoření řešení.

## InSIS

Jediným zdrojem pro naše řešení je InSIS. Jedná se o informační systém Vysoké školy ekonomické v Praze, ze kterého jsou extrahovány soubory ve formátu MS Excel.

Soubory obsahují data týkající se:

* Výpočtu pedagogických výkonů za přímou výuku předmětů,
* výpočtu pedagogických výkonů za nepřímou výuku, ukončení předmětu a praxe,
* výpočtu pedagogických výkonů za nepřímou výuku, státní a souborné zkoušky,
* výpočtu pedagogických výkonů za nepřímou výuku, vedení závěrečných prací,
* výpočtu pedagogických výkonů za nepřímou výuku, obhajoby závěrečných prací.

Jejich aktuální seznam je v podkapitole *6.3 Zdrojové soubory*.

# Vlastní procedúra pro zpracovaní vstupních souborů implemenovaná pomoci Azure Functions

Azure Functions je událostmi řízená bezserverová výpočetní platforma, v které je možné vyvíjet aplikace v různých jazycích, za použití systému Linux v cloudu. Umožnuje spouštět scripty v návaznosti na další procesy v rámci integrace v Azure Data Factory.

Protože u některých vstupních souborů dochází ke změnám sloupců, byla pro jejich zpracování naprogramována procedura v jazyce Python (dále jen python skript), která je na platformě Azure implementována pomocí služby Azure Functions. Python skript je typu http-trigger function, což znamená, že je spouštěn pomocí API vytvořené v azure API managmentu (aktivace API probíhá pomocí automaticky generovaného zavolání v data factory). Skript je spravovaný pomocí Visual Studia Code, které umožňuje jeho jednoduché aktualizovaní a uploadování do úložiště function app.

Procedura je prvním krokem nahrávání dat do datového skladu, a jako taková zajištuje přenos excelových exportních dat ze vstupní složky v úložišti blob do složky na kterou jsou napojeny pumpy nahrávající data do vrstvy L0. Tyto pumpy jsou schopny nahrávat data z připojeného excelu, i pokud se tento mění v čase, za předpokladu že bude při změnách zachováno stejné pojmenování souboru. Toto vytváření generických jmen je také zajištěno python skriptem. Dále skript (což je původním důvodem jeho existence) zajišťuje operaci unpivot u tabulek s potenciálně proměnným počtem sloupců, a mimo to také zajišťuje minoritní úpravy tabulek pro usnadnění následujících procedur (některé kroky je výrazně snazší provést vně v rámci pandas v pythonu, než uvnitř datového skladu). A zajištuje generování kódu dávky pro celou nahrávanou dávku, a kódů datasetu pro jednotlivé dílčí záznamy v tabulkách.

# Datové úložiště

Pro projekt je vytvořen základní server Azure SQL Database s názvem *dwh-vse.database.windows.net.* Připojení k serveru je realizováno pomoci *usernamu* a *hesla* vytvořených v SQL Server Management Studio (SSMS) pomoci SQL dotazu.

Následně je na portálu **Azure** vytvořená databáze s názvem *dwh,* kde jsou vytvořené všechny tabulky s potřebnými atributy, přiřazenými primárními klíči, cizími klíči a datovými typy.

V této kapitole budou také popsány principy modelování úložiště. Součástí bude datový model datového skladu.

## Azure SQL Database

*Azure SQL Database* využívá proprietární rozšíření jazyka SQL Microsoft Transact-SQL (T-SQL).

Pro řešení je využívána nejzákladnější verze databáze – Basic s kapacitou 2 GB, která muže být rozšířena

## Principy modelování

Pro potřeby řešení je vytvořen datový sklad. Vše je vytvořeno v prostředí *Azure SQL Database.*

Před začátkem implementace jsme si určily pravidla, která datový sklad maximálně zpřehlední. Pokud by se BI řešení v budoucnosti rozšiřovalo, tato pravidla zajistí, aby databáze zůstala přehledná i při velkém počtu tabulek. Byla stanovena následující pravidla:

* Názvy dimenzionálních tabulek budou v množném čísle a budou začínat velkým písmenem D. V případě víceslovných názvů budou jednotlivá slova oddělena podtržítkem, např D\_Typy\_studia
* Faktové tabulky budou označeny prefixem F, např F\_h\_cv
* Tabulky budou vždy pojmenovány podle dat, které obsahují
* Primární klíč by vždy měl začínat velkými písmeny PK, pak podtržítko a název příslušné dimenze, např PK\_Programy
* Názvy sloupců budou začínat velkým písmenem a v případě víceslovných názvů budou jednotlivá slova oddělena mezerou

## Datový model pro vybranou oblast

Model je vytvořen v nástroji *SQL Server Management Studio (SSMS)* od Microsoft. Viz. Příloha C – Datový model L1.

## SQL Server Managment Studio (SSMS)

SQL Server Management Studio (SSMS) je integrované prostředí pro správu jakékoli infrastruktury SQL, od SQL Serveru po Azure SQL Database. SSMS poskytuje nástroje pro konfiguraci, sledování a správu instancí SQL Serveru a databází. Pomocí SSMS můžeme nasazovat, monitorovat a upgradovat komponenty datové vrstvy používané aplikacemi a sestavovat dotazy a skripty.

# Datové úložiště

Pro projekt byl vytvořen základní server *Azure SQL Database* s názvem ***dwh-vse.database.windows.net****.* Připojení k serveru bylo realizováno pomocí SQL dotazu.

### Vytvoření databáze a tabulek

Následně na základě modelu byla v aplikací *SQL Server Management Studio* vytvořená databáze s názvem ***dwh***.

#### SQL scripty

Scripty naleznete v *Příloha A – SQL kódy*.

#### Tabulky v DWH

Tabulky naleznete v *Příloha B – Tabulky*.

## Datový model

Nejdříve byl vytvořen datový model. Datový model, který je v příloze: *Příloha C – Datový model L1.*

## Zdrojové soubory

Zdrojové soubory byli nahrány do úložiště *vsestorage*, kde byl vytvořen kontejner s názvem vystupni-excel-data *v kterém jsou umístěny následující soubory*.

Kontejner *vystupni-excel-data*:

*Bilance kooperací ve výuce dle fakult.xlsx*

*Bilance kooperací ve výuce dle fakult - součty za všechny poskytovatele.xlsx*

*Hodnota H cvičení dle garantujících pracovišť.xlsx*

*Hodnota H cvičení.xlsx*

*Hodnota H praxí.xlsx*

*Hodnota H přednášek.xlsx*

*metadata.xlsx*

*Parametry výpočtu.xlsx*

*Přehled bodů za obhajoby závěrečných prací dle dodavatelů výkonu.xlsx*

*Přehled bodů za obhajoby závěrečných prací dle odběratelů výkonu.xlsx*

*Přehled bodů za státní a souborné zkoušky dle dodavatelů výkonu.xlsx*

*Přehled bodů za státní a souborné zkoušky dle odběratelů výkonu.xlsx*

*Přehled bodů za vedení závěrečných prací dle dodavatelů výkonu.xlsx*

*Přehled bodů za vedení závěrečných prací dle odběratelů výkonu.xlsx*

*Přehled bodů za výuku předmětů dle dodavatelů výkonu a předmětů.xlsx*

*Přehled bodů za výuku předmětů dle odběratelů výkonu a předmětů.xlsx*

*Přehled kooperací ve výuce předmětů.xlsx*

*Přehled kooperací za obhajoby závěrečných prací.xlsx*

*Přehled kooperací za státní a souborné zkoušky.xlsx*

*Přehled kooperací za vedení závěrečných prací.xlsx*

*Přehled předmětů a procentních podílů pracovišť, které se podílejí na jejich výuce.xlsx*

*Přehled předmětů pro obhajoby závěrečných prací, zúčastněných pracovišť a počtu studentů, kteří absolvovali tyto předměty.xlsx*

*Přehled předmětů pro státní zkoušky, zúčastněných pracovišť a počtu studentů na těchto předmětech.xlsx*

*Přehled pro výkony za vedení kvalifikačních prací.xlsx*

## Propojené služby

### Služby pro připojení k nahraným souborům

Pro připojení ke zdrojovým souborům do úložiště **vsestorage** byla vytvořena služba *blobstorage.* Pro všechny excelové soubory jsou vytvořeny tabulky, s kterými služba pracuje. Tyto sady jsou ve složce **vystupni-excel-data** a jsou pojmenovány jako:

Složka **vystupni-excel-data** obsahuje vstupní data vytvořená python skriptem:

* Každá datová sada má název – *NÁZEV TABULKY V DATABÁZI* např. **bl\_koop\_f\_poskyt = L0.bl\_koop\_f\_poskyt**

### Služby pro připojení k jednotlivým tabulkám

Byla vytvořena služba *database*, která se připojuje k naší databázi *dwh* na serveru *dwh-vse.database.windows.net*. Tato služba zajišťuje nahrání dat do jednotlivých tabulek jak L0, L1stage tak i L1 vrstvy, pro toto připojení byly vytvořeny tabulky pro jednotlivé tabulky v databázi. Tyto sady jsou rozděleny do jednotlivých složek a jsou pojmenovány jako:

Složka **L0** obsahuje tabulky tabulek L0 vrstvy:

* Každá datová sada má název – *L0\_NÁZEV TABULKY V DATABÁZI* např. **L0\_bl\_koop\_f\_poskyt = L0.bl\_koop\_f\_poskyt**

Složka **L1stage** obsahuje tabulky tabulek L1stage vrstvy:

* Každá datová sada má název – *L1stage\_D/F\_NÁZEV TABULKY V DATABÁZI* např. **L1stage\_F\_bl\_koop\_f\_poskyt = L1stage.F\_bl\_koop\_f\_poskyt**

Složka **L1** obsahuje tabulky tabulek L1 vrstvy:

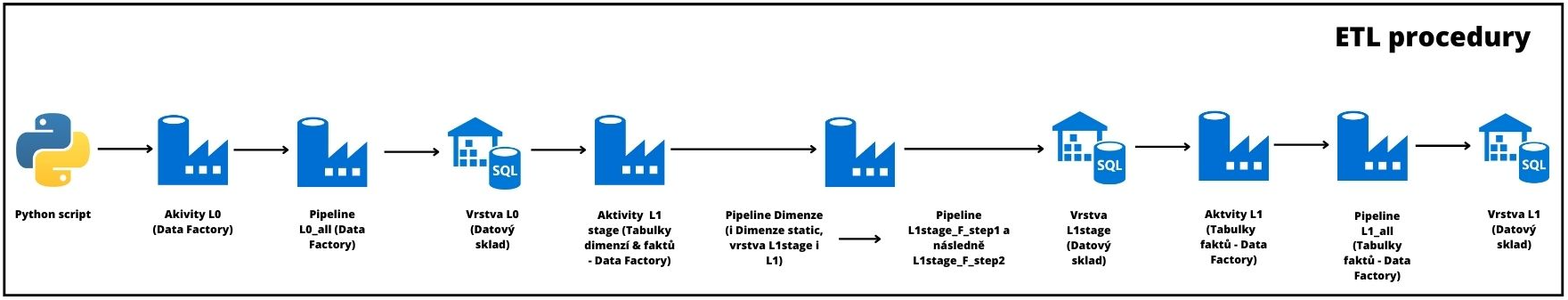
* Každá datová sada má název – *L1\_D/F\_NÁZEV TABULKY V DATABÁZI* např. **L1\_F\_bl\_koop\_f\_poskyt = L1.F\_bl\_koop\_f\_poskyt**

# ETL

Pomocí vrstvy ETL se zajišťuje nahrání zdrojových dat do stage databáze a následně do DWH.

Data jsou nejprve zpracována aplikací v Azure Functions, poté jsou nahrána z excelových souborů do vrstvy L0, a následně jsou převedena do vrstvy L1\_stage, a poté do finální L1 k dlouhodobému uložení. Jako první se nahrávají tabulky dimenzí. Ty jsou nahrávány dvěma způsoby, staticky – dimenze ve kterých se data téměř vůbec nebo alespoň minimálně nemění tyto jsou spouštěny mimo proceduru (fakulty, kategorie výkonu, typy studia.) A dynamicky – tj. data v dimenzích jsou jiná při každém nahrání, a nahrávání probíhá na začátku ETL procesu z L0 do L1\_stage (dataset, období, předměty, pracoviště, programy). Po nich jsou do vrstvy L1\_stage ve dvou krocích nahrány tabulky faktů (tabulky vytvořené v druhém kroku obsahují sloupce odvozené z tabulek vzniklých v kroku prvním). Do tabulek D-Fakulty, D-Typy\_studia a D-Kategorie\_vykonu jsou vkládány pouze nové a unikátní záznamy, a tyto tabulky jsou nahrávány mimo proceduru. Zbytek tabulek je aktualizován při každém nahrávání, procedura pro tabulky D-Období, D-Programy obsahuje podmínku *exists* a jsou do nich tak vkládány pouze záznamy, které už nejsou přítomny. Do tabulek D-Pracoviště a D-Předměty jsou vkládány nové neunikátní hodnoty spojené přímo s nahrávanou dávkou, a do tabulky D-Dataset jsou vkládány unikátní hodnoty generované aplikací v Azure Functions.

ETL procesy jsou tvořeny pomocí Azure Data Factory V2. Pro potřeby řešení vznikl datový sklad ***dwh***. Pro práci bylo také potřeba vytvořit storage ***vsestorage***, kam byly nahrány zdrojové Excel soubory a Azure Function ***unpivot-prestage***.

Soubory jsou umístěny v kontejnerů Azure blob pod názvem **vstupni-slozka**.

Obrázek č.2 – ETL procedura (Zdroj: Autoři)

## ETL procesy

ETL procesy byly realizovaný pomocí Data Factory V2. Pro potřeby našeho řešení vznikla databáze dwh. Pomocí vrstvy ETL se zajišťuje nahrání zdrojových dat do stage databáze a následně do DWH.

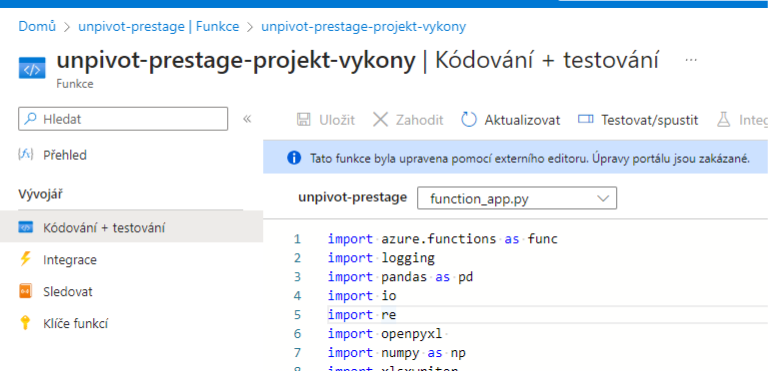
Pro potřeby řešení vznikl datový sklad dwh. Pro práci bylo také potřeba vytvořit storage vsestorage, kam byly nahrány zdrojové Excel soubory. Byla také vytvořena aplikace v azure functions, pro zajištění lepší automatizace etl operace. Nahrávaná dávka souborů je vkládána do kontejneru v Azure blob pod názvem vstupni-slozka. Kapitola ETL popisuje část procedury realizovanou pomocí azure functions, přípravu zdrojových souborů pro nahrání pomocí ETL v azure data factory, popisuje použité služby, samotný skript a jeho zapojení do procedury. ETL transformace v ADF a nahrávaní dat do jednotlivých vrstev v datovém skladu.

## Azure Functions

Byla vytvořena standartní instance Azure Functions s virtuálním linuxem, spouštěná on-demand, s volbou nahrání function pomocí Visual Studia Code. Spouští script napsaný v jazyce python. Procedura slouží jako první krok nahrávání dat do datového skladu, a jako taková zajištuje přenos excelových exportních dat ze vstupní složky v úložišti blob do složky na kterou jsou napojeny pumpy nahrávající data do vrstvy L0. Tyto pumpy jsou schopny nahrávat data vždy z připojeného excelu, i pokud se tento mění v čase, za předpokladu že bude zachováno stejné pojmenování tohoto souboru, toto generování generických jmen je také zajištěno python skriptem. Dále skript také (což je původním důvodem jeho existence) zajišťuje operaci unpivot u tabulek s potenciálně proměnným počtem sloupců, a mimo to také zajišťuje minoritní úpravy tabulek pro usnadnění následujících procedur (některé kroky je výrazně snazší provést vně v rámci pandas v pythonu, než uvnitř datového skladu). A zajištuje generování kódu dávky pro celou nahrávanou dávku, a kódů datasetu pro jednotlivé dílčí záznamy v tabulkách.

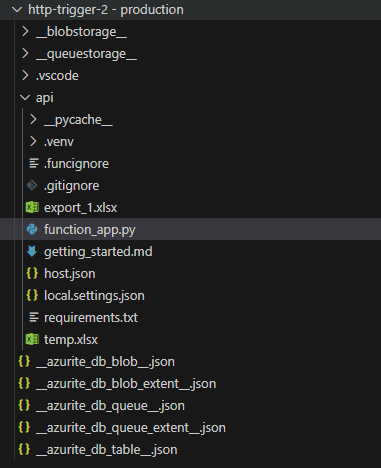
### Azure function

Jednotlivá funkce v prostředí Azure Functions. Zde máme možnost otestovat zkušebně serverless funkci, zobrazit si celkový log spuštěné procedury, a také se podívat na historii logů

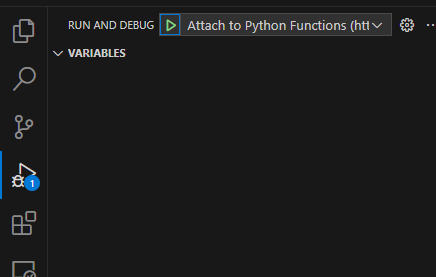
Obrázek č.3 – Azure function (Zdroj: Autoři)

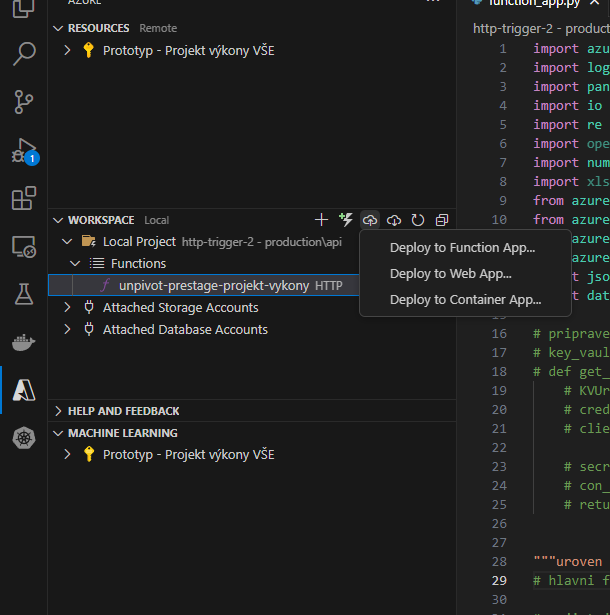
### Visual Studio Code

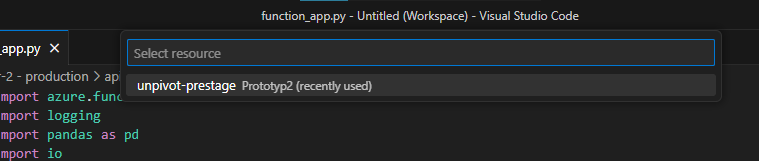
Pro vývoj a nasazení python scriptu do Azure Functions byl použit externí editor, a to Visual Studio Code, VS Code. Po stažení a instalaci VS Code se je nejprve nutné připojit k předplatnému azure, následně zvolit správnou službu App, a poté spustit zkušební instanci zdrojových souborů. Ta se nakonec nahrává ve formě zip balíčku do Azure Functions, kde může být spuštěna on-demand. Samotná synchronizace API a Functions s novou verzí obrazu function probíhá automaticky.



Obrázek č.4 - Všechny soubory nasazení aplikace, zde otevřené jako složka, se nahrávají jako zip balíček (Zdroj: Autoři)

  
Obrázek č.5 - Pro nahrání je nejprve nutné spustit debug mode (Zdroj: Autoři)

  
Obrázek č.6 - Následně se staženým modulem pro Azure a Azure CLI přejít do Azure workspace, a vybrat možnost “Deploy to Function App...” (Zdroj: Autoři)



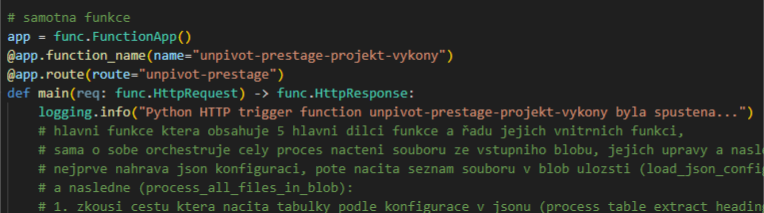
Obrázek č.7 - Poté zvolit správnou aplikaci pro nahrání (Zdroj: Autoři)

### Python script

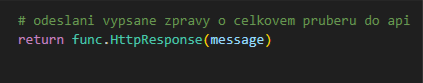
Pro účely využití Azure Functions byl vytvořen python skript. Python skript je typu http-trigger function, což znamená, že je spouštěn pomocí API vytvořené v Azure API Managmentu (aktivace API probíhá pomocí automaticky generovaného zavolání v Data Factory). Skript je spravovaný pomocí Visual Studia Code, které umožňuje jeho jednoduché aktualizovaní a uploadování do úložiště function App.

Hlavními funkcemi python scriptu jsou provádění operace unpivot na tabulkách načtených do knihovny pandas, vytváření generických jmen výstupních souborů, podle načtených záhlaví tabulek, a export takto pojmenovaných souborů do výstupní složky úložiště blob. Díky tomu můžeme původně různě pojmenované výstupní soubory z InSISu automaticky nahrát do úložiště blob, a navázat na ně další procesy v Azure Data Factory. Dále také zodpovídá za generování kódu dávky a datasetu a za minoritní úpravy tabulek nutné pro další operace. A nakonec generuje tabulku metadata.xlsx, která je využita při tvorbě dimenze dataset.

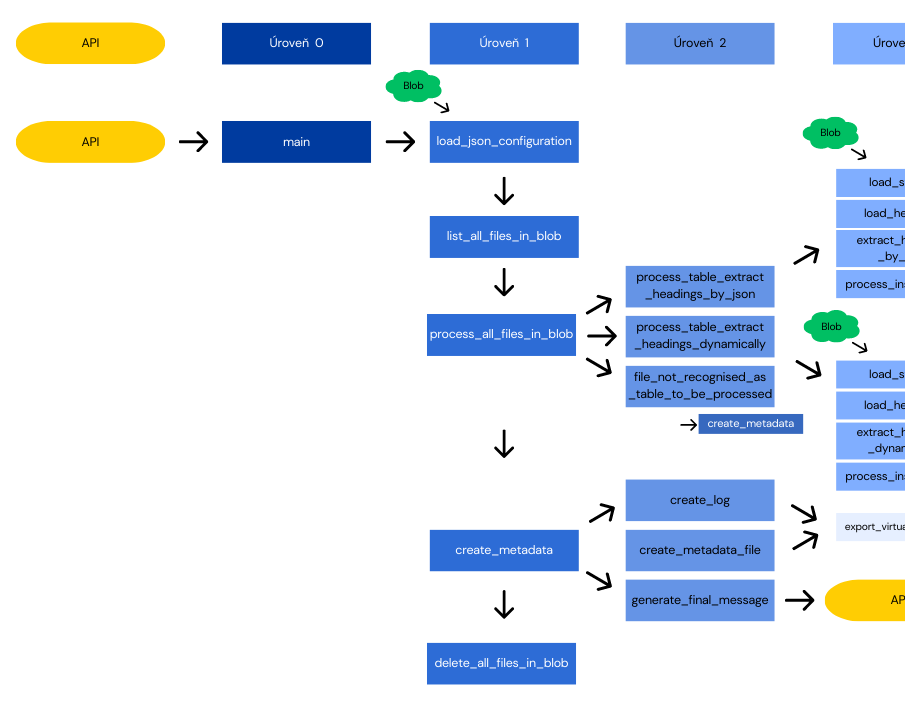
Po zavolání service API je informace o aktivaci procedury předána do function app a je spuštěna serverless aplikace.

  
Obrázek č.8 – Function app (Zdroj: Autoři)

Je spuštěna hlavní funkce main. Ta řídí celý proces. Funkce je napojením skriptu na Azure Functions a je zakončena odesláním finální zprávy do API.



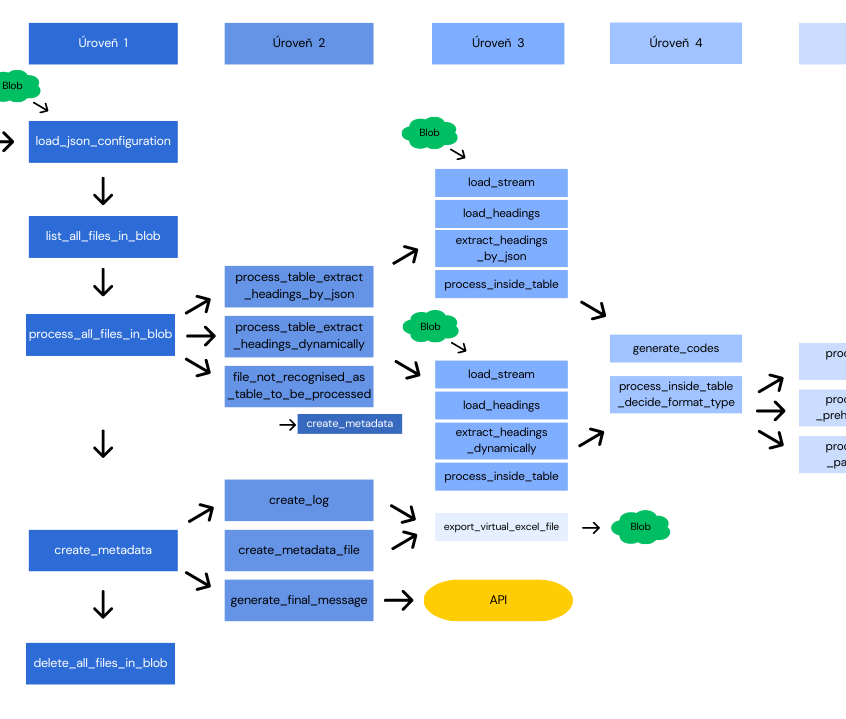
Obrázek č.9 – Finální zpráva do API (Zdroj: Autoři)

  
Obrázek č.10 – Model pythonu 1 část (Zdroj: Autoři)

Úroveň níže je vždy zabalena v úrovni vyšší, funkce zapsány v jednom souvislém sloupci vždy následují ve sledu po sobě.

Na úrovni main je skript rozdělen do 5 hlavních částí, a to load\_json\_configuration, list\_all\_files\_in\_blob, process\_all\_files\_in\_blob, create\_metadata, delete all\_files\_in\_blob.

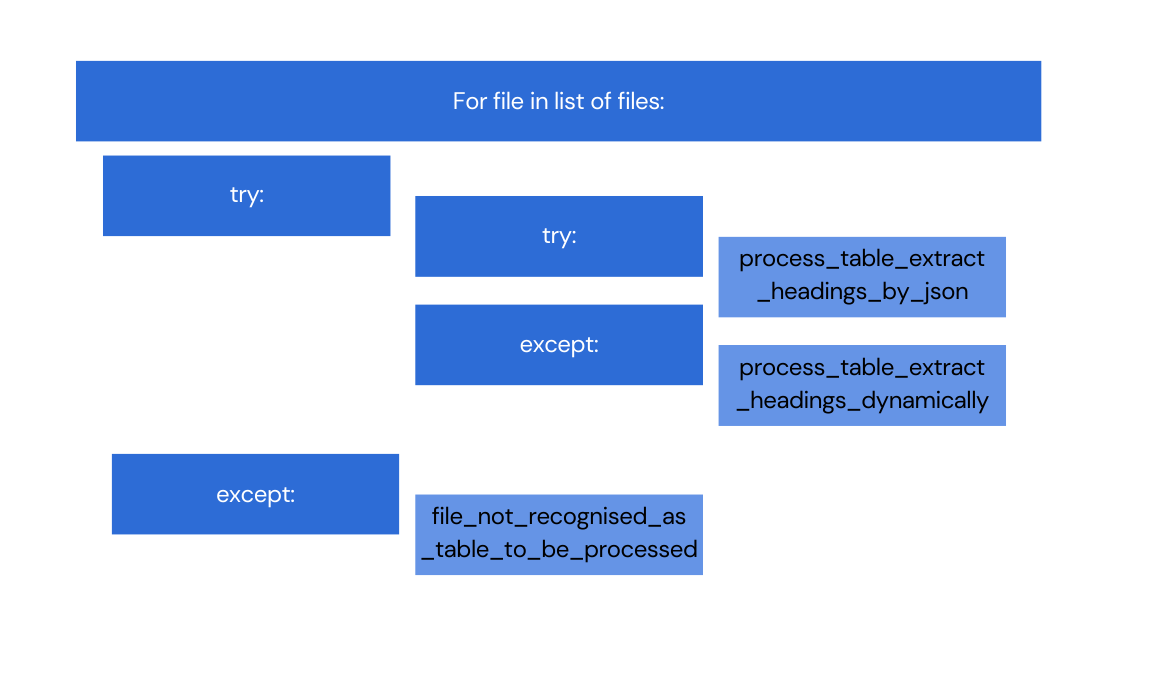
První část zodpovídá za načtení json konfigurace z jejího vstupního úložiště json-konfigurace. Druhá část načte vstupní blob vstupni-slozka, a vypíše seznam všech přítomných souborů, tím následně listuje třetí funkce a provádí tabulkové operace, které budou rozebrány později. Výstup třetí funkce má podobu zjednodušeného logu, a načítá se čtvrtou funkcí, která z něj generuje log a metadata. A nakonec je, pokud je tak specifikováno v metadatech, zavolána funkce, která smaže všechny soubory ve vstupním blobu.

  
Obrázek č.11 – Model pythonu 2 část (Zdroj: Autoři)

Funkce create\_metadata a process\_all\_files\_in\_blob se dále rozvětvují do dalších úrovní, 2 a 6 respektive.

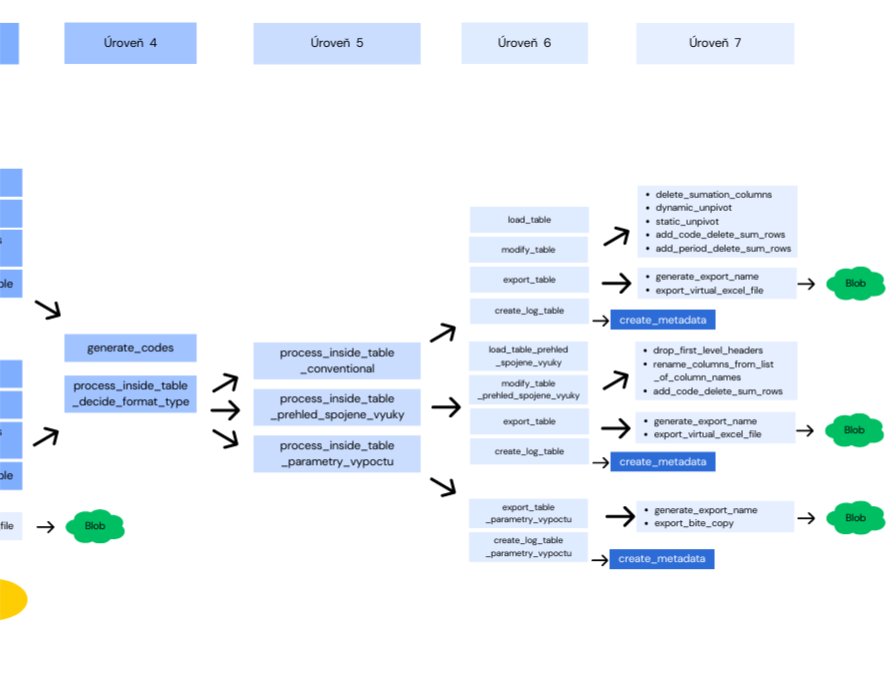
Create\_metadata v sobě volá create\_log – funkce tvořící exportní log, create metadata\_file – funkce tvořící soubor s metadaty a generate final\_message – funkce která shrnuje průběh procedury.

Process\_all\_files\_in\_blob uvnitř obsahuje for-loop, ten pročítá jednotlivé načtené soubory ve vstupním blobu, a podrobuje je vnitřnímu a vnějšímu setu try/except konstruktů. Ty ve vnitřní úrovni zkouší, zda bude tabulka procesována se statickým načtením záhlaví nebo s načtením dynamickým, a ve vnějším konstruktu, nejprve zkouší konstrukt první a pokud selžou oba pokusy o zpracování, tak je soubor z operace vyňat, a bude zaznamenána chyba.

  
Obrázek č.12 – Schéma funkce process\_all\_files\_in\_blob (Zdroj: Autoři)

Obě cesty process\_table\_extract\_headings využívají 4 vnitřní funkce, load\_stream, funkci, která zajišťuje stažení dat exportu ze vstupní složky blob úložiště, vstupni-slozka. Dále load\_headings, funkce, která načte prvních 12 řádků exportní tabulky, a načte je do pandas. Následně se obě cesty rozdělují, extract\_headings\_by\_json zkouší záhlaví načíst podle parametrů specifikovaných v jsonu, a extract\_headings\_dynamically se snaží ho nalézt podle shody klíčových slov. Obě cesty se poté spojují ve funkci process\_inside\_table.

Process\_inside\_table, společná operace, která nám sdružuje obě předešlé cesty, se skládá ze 2 částí generate\_codes a process\_inside\_table\_based\_on\_format, generate\_codes tvoří dílčí kódy a skládá z nich kód dávky. Kód dávky je složen vždy z kódu generovaného z období z pole fixace dat v záhlaví, kódu vygenerovaného z data exportu, a značky FIS/VŠE. Process\_inside\_table\_based\_on\_format, funkce, která zastřešuje finální dělení, a to podle formátu tabulky, obsahuje 2 vrstvy – první dělení zde probíhá podle značky speciální formát, na standartní cestu a cestu vymykající se normě. A poté se cesta vymykající se normě ještě dále dělí přímo podle konkrétního jména exportu, a to parametrů výpočtu, a přehledů spojené výuky.

  
Obrázek č.13 – Model pythonu 3 část (Zdroj: Autoři)

Všechny tři cesty už v sobě obsahují samotné zpracování surových tabulek, od jejich načtení, přes úpravy podle json-konfigurace, až po generaci jména, export tabulek a tvorbu logu.

Tyto funkce následně využívají až dvě další vrstvy, v první vrstvě jsou to pro standardní cestu load\_table, modify\_table, export\_table a create\_log\_table; pro parametry výpočtu export\_parametry\_vypoctu a create\_log\_table\_parametry\_vypoctu; pro přehled spojené výuky pak load\_table\_prehled\_spojene\_vyuky, modify\_table\_prehled\_spojene\_vyuky, export\_table a create\_log\_table. Funkce modify\_table a export\_table jsou navíc ještě dále rozvětveny.

Modify\_table volá pro operace specifikované v json konfiguraci celou řadu dílčích funkcí.

Jsou jimi:

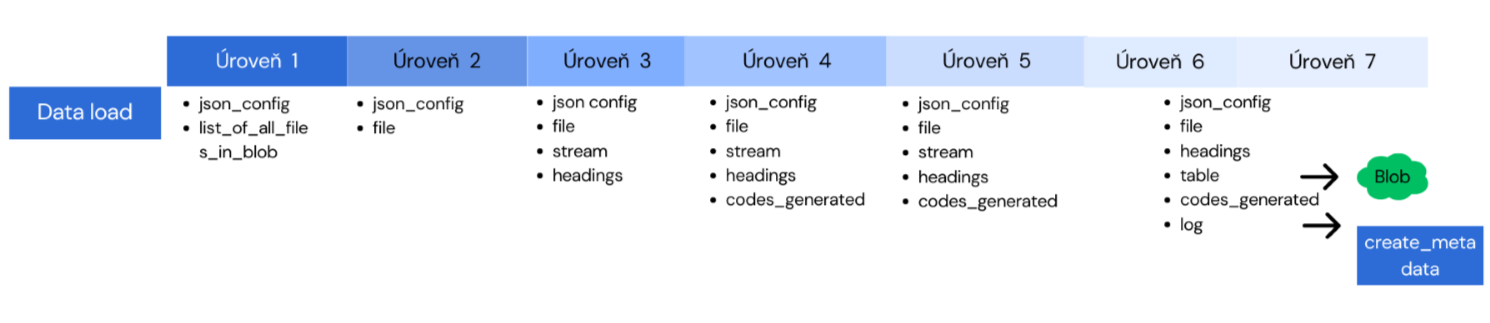
* delete\_sumation\_columns, která odstraňuje sloupce sum před unpivotem
* dynamic\_unpivot a static\_unpivot, které zajišťují unpivot s proměnným počtem sloupců a unpivot se stabilními sloupci (na úrovni kódu je operace prováděna téměř stejně, toto označení je tedy hlavně informativní, neboť jsou oba typy unpivotu prováděny pro jiné typy dat)
* add\_code\_delete\_sum\_rows, ta obsahuje for-loop, který na každém řádku generuje kód datasetu a zároveň odstraňuje řádky s hodnotami sumace; oba kroky jsou sdruženy pro velkou výpočetní náročnost pročítání tabulky
* add\_period\_delete\_sum\_rows, ta obsahuje forloop, který na každém řádku vkládá kód datasetu vygenerovaný podle fixace dat ze záhlaví, a zároveň odstraňuje řádky s hodnotami sumace; oba kroky jsou sdruženy pro velkou výpočetní náročnost pročítání tabulky

Kód datasetu je tvořen vždy složením kódu generovaného z řádku období, kódu generovaného z data exportu, a značky FIS/VŠE.

U přehledů spojené jsou dále aplikovány dvě specifické funkce: drop\_first\_level\_headers – funkce na odstranění spojeného záhlaví načtené surové tabulky v pandas, rename\_columns\_from\_list\_of\_columns\_names – funkce, která přejmenuje po prvním kroku vzniklé nesmyslné záhlaví tabulky.

Export\_table je obecně složena ze dvou funkcí, generate\_export\_name – funkce pro generování exportního jména a export\_virtual\_excel\_file – funkce exportující do výstupního kontejneru blob storage, tato funkce nejprve vytvoří bitovou kopii excelu generovaného z výstupní pandas tabulky, a tu poté nahraje do výstupní azure blob storage.

U parametrů výpočtu máme ještě export\_bite\_copy – funkce exportující pouze na začátku procedury stažený bitový obraz vstupního excelu.

Obrázek č.14 –Vizualizace načtených dat v process\_all\_files\_in\_blob (Zdroj: Autoři)

**Vysvětlivky k výše uvedenému obrázku** (*Obrázek č.14*):

* Json\_config – načtená json konfigurace,
* List\_of\_all\_files\_in\_blob – seznam souborů ve vstupním kontejneru
* Blob storage, file – informace o souboru uloženém v kontejneru vstupní
* Blob\_storage, headings – informace načtené ze záhlaví,
* Codes\_generated – kódy pro tvorbu kódu datasetu a kód dávky,
* Table – samotná tabulka načtená do pandas,
* Log – záznam o provedených operacích

Funkce create\_log, zajištuje tvorbu souhrnného logu operace, a to tak, že sdružuje výstupy předešlých operací a vytváří z nich souhrnný pandas dataframe, který pak exportuje do blob úložiště jako excel.

Funkce create\_metadata\_file na vstupu bere dataframe vytvořený předchozí funkcí, vybírá z něj unikátní kódy datasetu, a generuje pro ně soubor s dodatečnými informacemi, který je pak načítán v rámci ETL procedury v Azure Data Factory.

Funkce generate\_final\_message generuje finální zprávu shrnující průběh operací, tato zpráva je nakonec odeslána do API.

Funkce delete\_all\_files\_in\_blob je na závěr je spouštěná procedura, která podobně jako process\_all\_files\_in\_blob pročítá seznamem souborů přítomných ve vstupní složce blob a jednotlivě je maže.

### Json konfigurace

Pro účely python skriptu byla vytvořena snadno spravovatelná konfigurace ve formátu json, ta je spravována v rámci úložiště blob ve složce json-konfigurace, pod jménem “vykony\_konf.json”. Obsahuje informace, které je nutné k dávce dodat nad rámec informací již přítomných v exportech, a nastavení operací prováděných na jednotlivých tabulkách v exportech. Konfigurace se skládá z legendy, části export\_info, kde jsou obsaženy dodatečné informace o dávce nutné pro nahrání, a nakonec z části tabulky\_info, která obsahuje samotné konfigurace procedury vždy pro jeden daný export.

Část legendy obsahuje vysvětlení všech uvedených parametrů.

Parametry definované jako značka nabývají hodnot “ano” a “ne”.

Část export\_info obsahuje parametry:

* final a fis – značky pro označení verze exportu
* datum\_exportu – datum ke kterému byla dávka exportována ze systému InSIS
* containery – jména složek se soubory v rámci uložiště blob
* odstranit\_vstupni\_soubory – značka která informuje skript o tom, zda mají být soubory ve vstupním úložišti blob po úspěšném zpracování smazány

Část tabulky\_info obsahuje parametry:

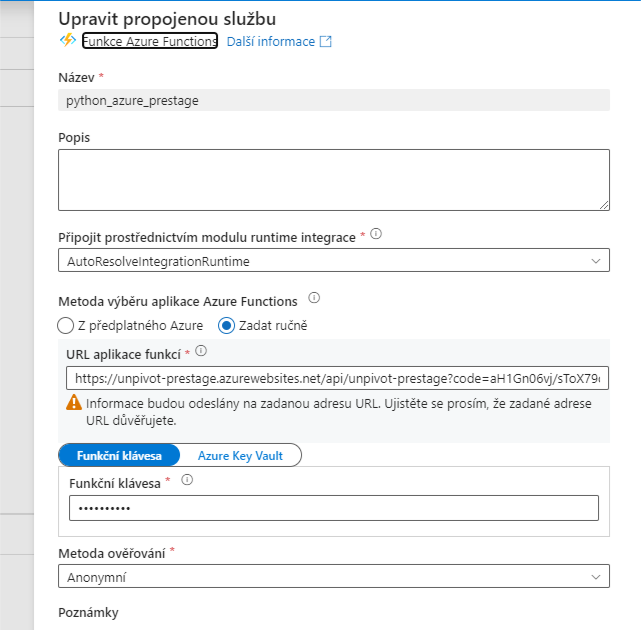
* radek\_zahlavi – číslo řádku, kde končí záhlaví a začíná samotná tabulka s hodnotami
* beh – číslo řádku, kde je vypsána informace o běhu
* kriterium – číslo řádku, kde je vypsána informace o kritériu
* specialni\_format – klasická cesta neumí načítat a přepisovat speciální formátování tabulek, například spojené sloupce, takové tabulky jsou po zatržení této možnosti zpracovány speciální cestou
* pridani\_sloupce\_obdobi – značka, která označuje tabulku pro provedeni operace přidání sloupce období
* unpivot\_dynamicky – značka, která označuje tabulku pro provedení dynamického unpivotu, pokud je zvoleno ano, je nutno dodat také hodnoty pro fixni\_sloupce\_dynamicky, sloupec\_atributu\_dynamicky a sloupec\_hodnoty\_dynamicky
* fixni\_sloupce\_dynamicky – pokud dochází u dané tabulky k dynamickému unpivotu, je nutné definovat, které sloupce budou z unpivotu vyňaty
* sloupec\_atributu\_dynamicky – pojmenování atributu (program, fakulta), u něhož provádíme unpivot dynamicky
* sloupec\_hodnoty\_dynamicky – hodnota pro daný atribut (body za výkon, pokusy u závěrečných zkoušek)
* unpivot\_staticky – značka, která označuje tabulku pro provedení statickeho unpivou, pokud je zvoleno ano, je nutno dodat také hodnoty pro fixni\_sloupce\_staticky, sloupec\_atributu\_staticky a sloupec\_hodnoty\_staticky
* fixni\_sloupce\_staticky – pokud dochází u dané tabulky k statickému unpivotu, je nutné definovat, které sloupce budou z unpivotu vyňaty
* sloupec\_atributu\_staticky – pojmenovaní atributu (program, fakulta), u něhož provádíme unpivot staticky
* sloupec\_hodnoty\_staticky – hodnota pro daný atribut (body za výkon, pokusy u závěrečných zkoušek)",
* "nahrazeni\_jmen\_sloupcu – značka, která označuje tabulku pro provedeni operace nahrazeni jmen sloupců
* slovnik\_pro\_nahrazeni – slovník (seznam pro přehled spojené výuky) obsahuje informace o tom, jak mají byt sloupce přejmenovány
* odstraneni\_sloupcu\_sumace – značka, která označuje tabulku pro provedeni operace odstranění sloupců se součty
* sloupce\_sumace – sloupce se součty, které mají byt odstraněny
* odstraneni\_radku\_sumace – značka, která označuje tabulku pro provedeni operace, která odstraní řádky se součty
* sloupec\_radku\_sumace – jméno sloupce ve kterém se vyskytuji řádky se součty

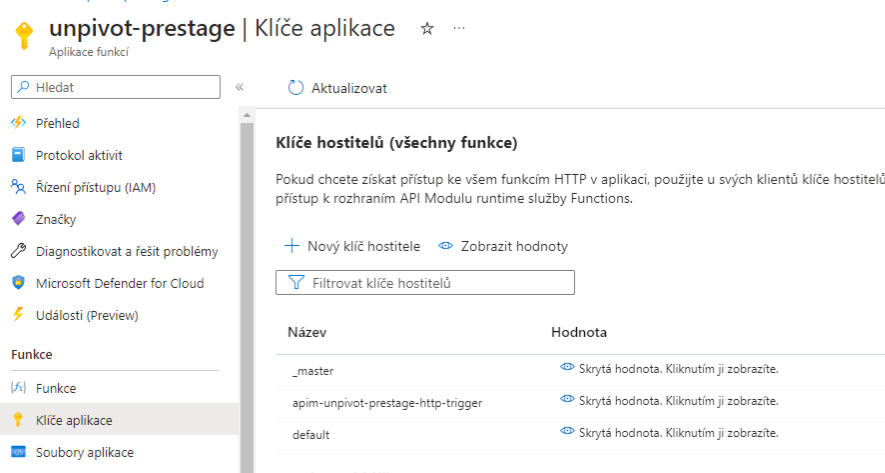
### Připojení na úložiště blob

Python skript na vstupu načítá soubory ze složky vstupni-slozka, a následně generuje a ukládá výstupní soubory do složky vystupni-excel-data. Také ukládá zjednodušený log své operace, který shrnuje všechny prováděné operace do složky log-unpivot-operace (celý log je dostupný v Azure Functions).

### Napojení na Data Factory

Integraci Functions do Data Factory provádíme za pomoci pipeline python-script, integrace je založena na automatickém generování get requestu do api. Po obdržení get requestu api aktivuje funkci (ta se spouští on-demand na virutálním linuxu), jsou provedeny operace a funkce ukládá výstupní data do dalšího úložiště blob. Exporty jsou ukládány vždy pod generickými jmény, takže na ně mohou být snadno napojeny další operace v rámci adf.

  
Obrázek č.15 – UL funkce a funkční klávesa (Zdroj: Autoři)



Obrázek č.16 – Klíče hostitelů (Zdroj: Autoři)

## Data Factory

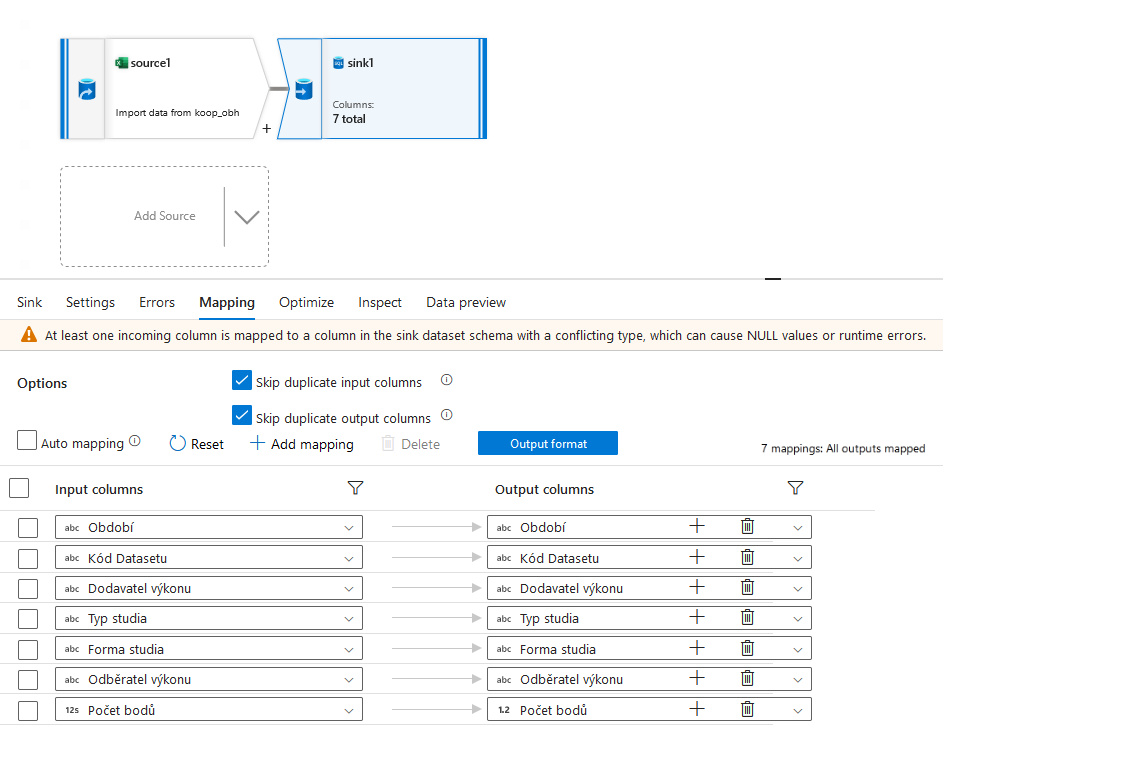
Azure Data Factory (ADF) je plně spravované řešení integrace dat bez serveru pro příjem, přípravu a transformaci jakýchkoli dat ve velkém.

Snadno vytváří ETL a ELT procesy bez kódu ve vizuálním prostředí, a umožňuje ke složitějším a komplexnějším implementacím doplnit integraci vlastního řešení v podobě kódu.

Celá námi vytvořená procedura pracuje s dvěma typy informací, na vstupu a v první úrovni jsou to nahrané excelové exportní soubory (blob storage vstupni-slozka a vystupni-excel-data), které se následně převádí do podoby relační databáze. V té se poté transformují sérií úprav (vrstvy L0 a L1\_stage), a ukládají k dlouhodobém uložení do finální vrstvy databáze (vrstva L1).

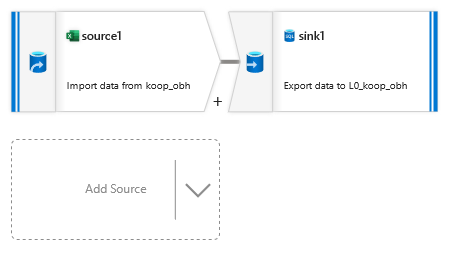
### Vrstva L0

V data flows, jsme pro každý datový soubor z " vystupni-excel-data " vytvořili složku s názvem L0, kde máme 24 data flows pro každý datový soubor z " vystupni-excel-data" (kromě dátového souboru Parametry výpočtu.xlsx). Vytvořili jsme nový datový tok jako: New dataflow > Add Source > Dataset > vybrali jsme dataset z "vystupni-excel-data" v tomto případě to byl koop\_obh. Důležité je přejít do Projection, zadat import projection a zkontrolovat, zda máme správný datový typ pro jednotlivé sloupce, a pokud ne, nastavit jej ručně.

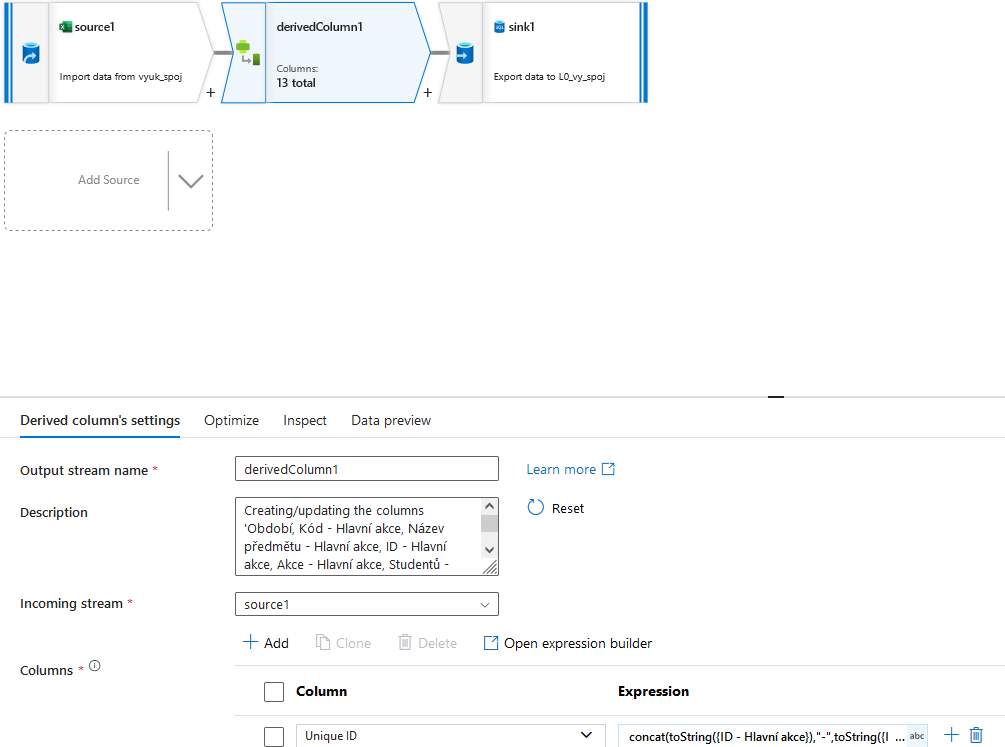


Obrázek č.17 – Sink aktivita (Mapping) – L0\_koop\_obh (Zdroj: Autoři)

Jak je vidět na obrázku výše, funkce "*Auto mapping*" je vždy automaticky zapnutá. V tomto případě, jak vidíme v pravém horním rohu, je to v pořádku, protože ze 7 sloupců jsou všechny namapovány správně, ale někdy "*Auto mapping*" nemůže namapovat všechny sloupce a může se stát, že jeden nebo více sloupců je vynecháno. Pokud taková situace nastane, je třeba vypnout "*Auto mapping*" a ručně namapovat sloupce, které nebyly namapovány správně. Tato situace obvykle nastává, když vstupní sloupce nemají shodné názvy s výstupními sloupci nebo když datové typy ve vstupních sloupcích nejsou shodné s datovými typy ve výstupních sloupcích.

Obrázek č.18 – Dataflow aktivita – L0\_koop\_obh (Zdroj: Autoři)

Na obrázku č.15 můžete vidět přiklad jedné z aktivit. Ostatní kopírovací aktivity jsou stejné, až na par výjimek. Těmito výjimkami jsou činnosti L0\_vyuk\_spoj a L0 vyuk\_hlv, kde je pro tyto 2 tabulky ještě vytvořeno jedinečné ID.

Toto ID se vytvoří pomocí funkce concat, která spojí více řetězců v následujícím případě (ID - spojená akce) - (ID - hlavní akce) - (kód tabulky).

Obrázek č.19 – Vytvoření jedinečného ID – L0\_vyuk\_spoj (Zdroj: Autoři)

### Vrstva L1 stage

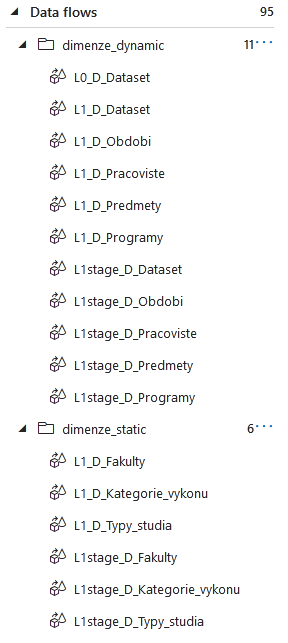
Ve vrstvě L1 stage se nacházejí tabulky dimenzí a tabulky faktů. Tabulky dimenzí jsou v ADF rozděleny na 2 části, a to: statické tabulky dimenzí, u nichž se nepředpokládá, že se během běhu dalšího řešení změní, resp. minimálně a dynamické tabulky dimenzí, které se během běhu dalšího řešení změní. Tabulky dimenzí jsou nahrávány jako první, po nich jsou do vrstvy L1\_stage ve dvou krocích nahrány tabulky faktů (tabulky vytvořené v druhém kroku obsahují sloupce odvozené z tabulek vzniklých v kroku prvním).

Do tabulek D-Fakulty, D-Typy\_studia a D-Kategorie\_vykonu jsou vkládány pouze unikátní záznamy, a tyto tabulky jsou nahrávány mimo proceduru.

Pro aktualizaci těchto tabulek je nutné upravit data ve zdrojovém excelovém souboru v úložišti blob ve složce dimenze (ten je pojmenován totožně jako cílové tabulky dimenze), a to tak, že bude soubor obsahovat jen příslušný řádek, který chceme do databáze přidat. (Anebo je možné tento přidat ke zbytku dat a vymazat obsah tabulky v dimenzi apriori.) Nakonec je potřeba v adf spustit pipeline “D-static” a provést tak aktualizaci.

Zbytek tabulek je aktualizován při každém nahrávání, procedura pro tabulky D-Období, D-Programy obsahuje podmínku *exists* a jsou do nich tak vkládány pouze záznamy, které už nejsou přítomny. Do tabulek D-Pracoviště a D-Předměty jsou vkládány nové neunikátní hodnoty spojené přímo s nahrávanou dávkou, a do tabulky D-Dataset jsou vkládány unikátní hodnoty generované aplikací v Azure Functions.

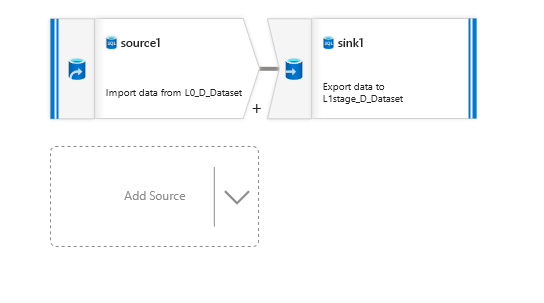
Tato vrstva se skládá z 35 transformačních aktivit.



Obrázek č.20 – Dynamické a statické dimenze (Zdroj: Autoři)

#### Transformační aktivity Dimenzí - L1 stage

Činnost **L1stage\_D**\_Dataset pouze importuje data z L0\_D\_Dataset a poté je exportuje do L1stage\_D\_Dataset v poměru 1:1. V operaci sink pak exportujeme data do tabulky L1stage\_D\_Dataset.



Obrázek č.21 – Aktivita L1 stage (Zdroj: Autoři)

**L1stage\_D\_Pracoviste** má vstupní source L0\_vy\_dod\_pred, který má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

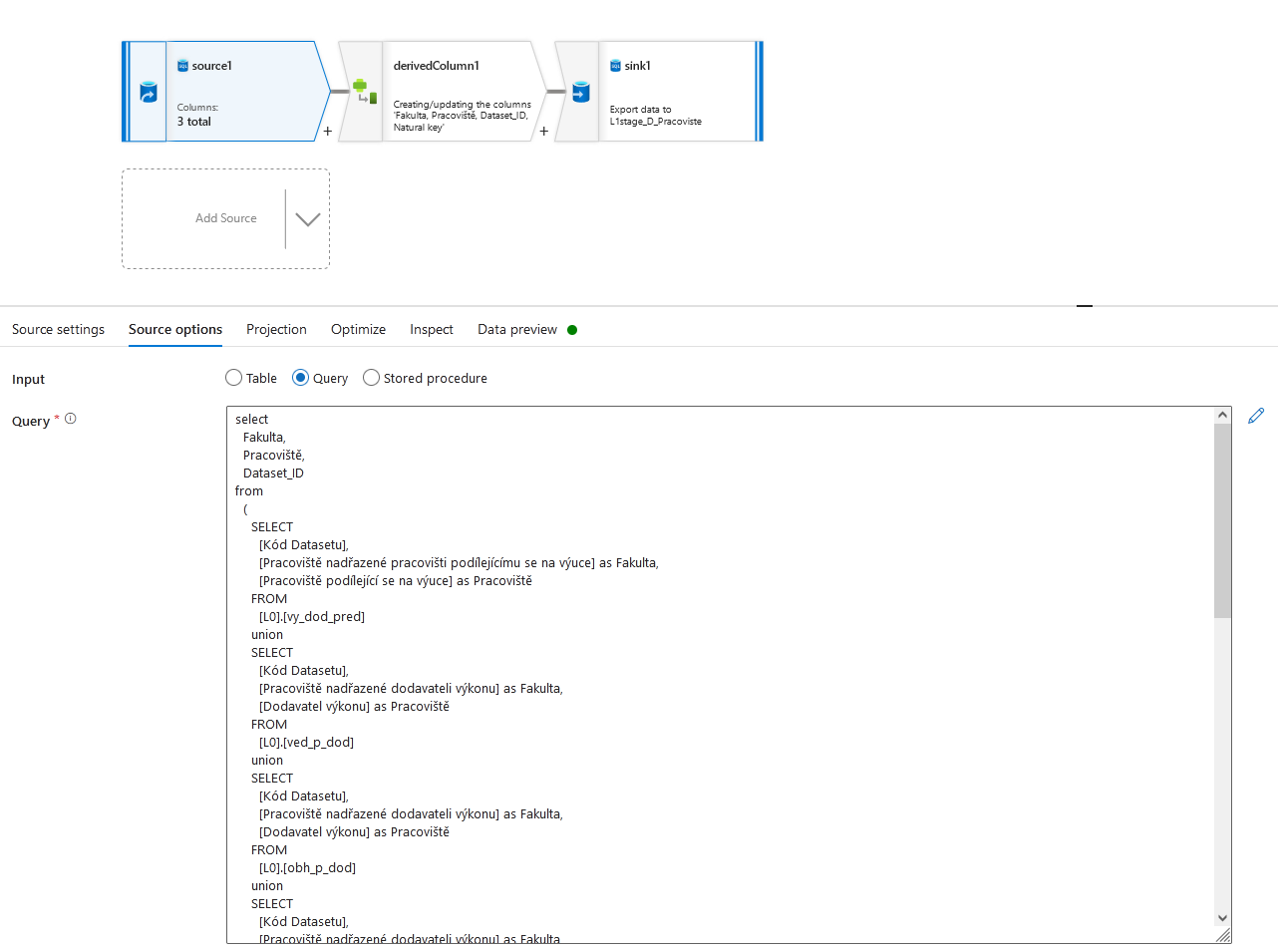
*Hlavní dotaz provede LEFT JOIN t1s tabulkou t2 na základě sloupce Kód Datasetu a pak vybere sloupce Fakulta, Pracoviště a Dataset\_ID odvozené tabulky t1.*

*Odvozená tabulka t1 vznikne spojením výsledků několika poddotazů pomocí operátoru UNION. Každý poddotaz vybírá různé sloupce z různých tabulek [L0].[vy\_dod\_pred], [L0].[ved\_p\_dod], [L0].[obh\_p\_dod], [L0].[stat\_dod], [L0].[vy\_od\_pred], [L0].[ved\_p\_prehled] a [L0].[stat\_prehled].*

*V hlavním dotazu vybíráme Fakulta a Pracoviště aby odpovídali názvům z tabulky t1. LEFT JOIN spojí odvozenou tabulku t1 s tabulkou [L1].[D\_Dataset] na základě sloupce Kód Datasetu.*

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Pracoviště", "Fakulta" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Tenhle přirozený klíč je důležitý proto, abychom do stejné tabulky nenačetli stejný řádek dat více než jednou. V operaci sink pak exportujeme data do tabulky L1stage\_D\_Pracoviste.



Obrázek č.22 – Source options v aktivitě L1stage\_D\_Pracoviste (Zdroj: Autoři)

**L1stage\_D\_Predmety** má vstupní source L0\_vy\_dod\_pred, který má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT načte sloupce: [Kód], [Název předmětu], t3.[Pracoviště\_ID] (alias [Garantující pracoviště\_ID]) a t2.[Dataset\_ID].*

*Dále obsahuje poddotaz definovaný jako t1, který kombinuje data z více tabulek pomocí operátoru UNION. Jedná se o následující tabulky: [L0].[vy\_dod\_pred], [L0].[ved\_p\_prehled], [L0].[stat\_prehled] a odvozenou tabulku s názvem t4, která je výsledkem operace JOIN mezi [L0].[vy\_spoj] a [L0].[pred\_podil].*

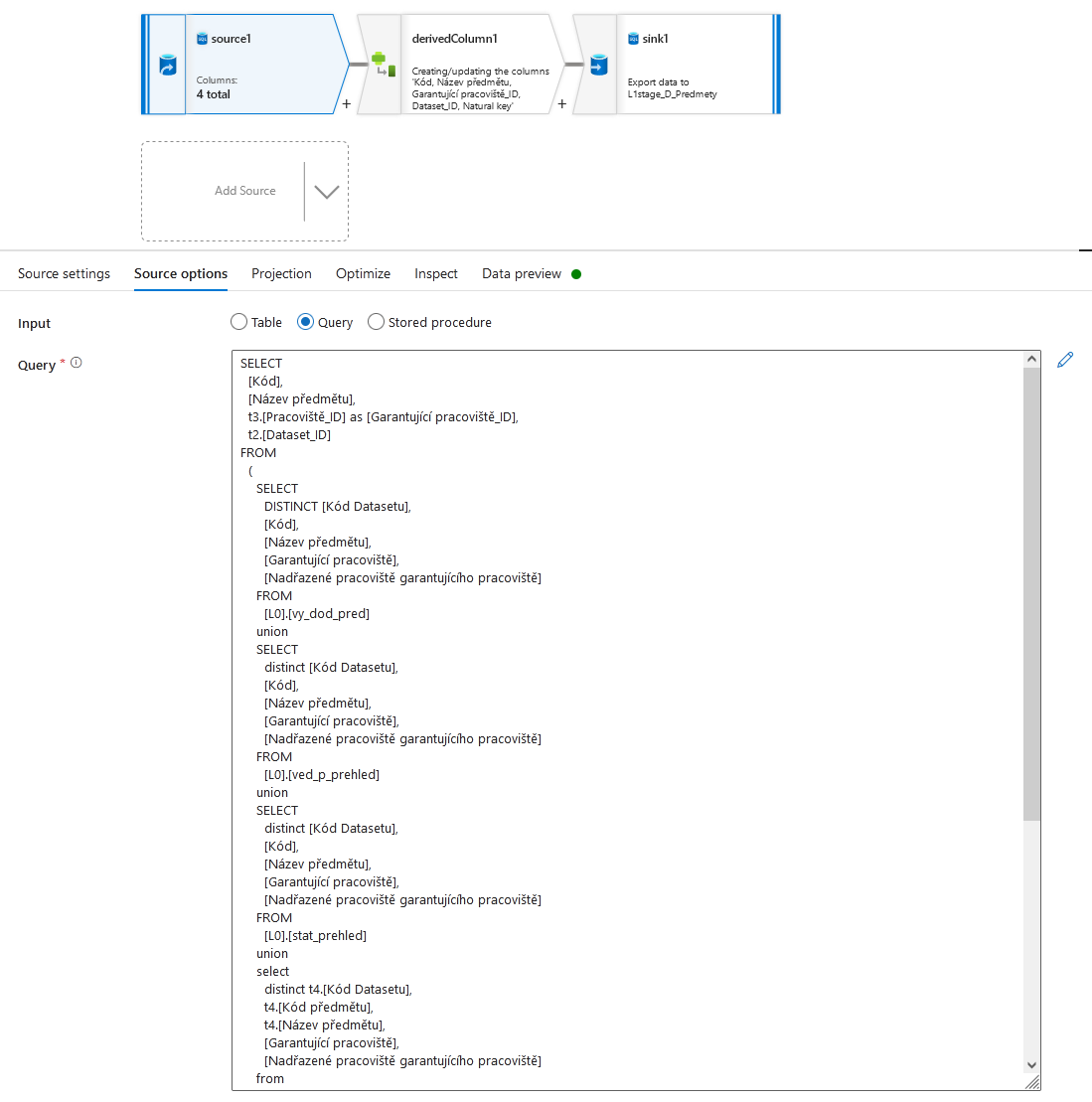
*Poddotaz t1 vybere odlišné hodnoty z uvedených sloupců: [Kód datasetu], [Kód], [Název předmětu], [Garantující pracoviště] a [Nadřazené pracoviště garantujícího pracoviště].*

*Vnější dotaz spojuje t1 s dalšími dvěma tabulkami pomocí operátoru LEFT JOIN:*

*[L1].[D\_Dataset] se připojí jako t2 pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*[L1].[D\_Pracoviště] je připojeno jako t3 pomocí podmínek t2.Dataset\_ID = t3.Dataset\_ID, t1.[Garantující pracoviště] = t3.Pracoviště a t1.[Nadřazené pracoviště garantujícího pracoviště] = t3.Fakulta.*

Pak se v odvozeném sloupci vytvoří přirozený klíč, který použitím funkce concat a spojením hodnot pomocí pomlček vytvoří přirozený klíč, který obsahuje hodnoty Kód, Garantující pracoviště\_ID a Dataset\_ID, protože poslední dva sloupce jsou číselné hodnoty, musíme je převést na řetězec. V operaci sink pak exportujeme data do tabulky L1stage\_D\_Predmety.



Obrázek č.23 – Source options v aktivitě L1stage\_D\_Predmety (Zdroj: Autoři)

**L1stage\_D\_Programy** má na vstupu 2 zdroje L0\_bl\_koop\_f\_poskyt a L1\_D\_Programy. L0\_bl\_koop\_f\_poskyt má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*První příkaz SELECT:*

*První select provádíme pro ty odběratele, které jsou zároveň i pracoviště, tj hlavni 6 fakult 'FFU','FMV','FPH','FIS','NF','FMJH' + 'OZS','PEDO' a zároveň sloupci program přiradíme sloupec fakulta.*

*Navíc do sloupce [Vykazovat samostatně] zahrne statickou hodnotu "Pouze pracoviště". Podmínka filtruje výsledky tak, aby zahrnovaly pouze řádky, ve kterých je ve sloupci [Odběratel výkonu] jedna z uvedených hodnot.*

*Druhý příkaz SELECT:*

*Tento příkaz SELECT rovněž načte odlišné hodnoty ze sloupce [Odběratel výkonu] v tabulce [L0].[bl\_koop\_f\_poskyt]. Přejmenuje sloupec [Odběratel výkonu] na Program a z hodnoty mezi závorkami vyčlení podřetězec, který přiřadí do sloupce [Fakulta]. Do sloupce [Vykazovat samostatně] zahrne statickou hodnotu "Pouze programy". Podmínka filtruje výsledky tak, aby vyloučila řádky, v nichž sloupec [Odběratel výkonu] odpovídá některé z uvedených hodnot.*

*Operátor UNION spojí výsledky obou příkazů SELECT, čímž získáme jedinou množinu výsledků, která obsahuje řádky splňující buď podmínku s operátorem IN, nebo podmínku s operátorem NOT IN. Výsledná sada bude mít tři sloupce: Program, Fakulta a Vykazovat samostatně.*



Obrázek č.24 – Source options v aktivitě L1stage\_D\_Programy (Zdroj: Autoři)

Zdrojem 1 je tabulka faktů a zdrojem 2 je dimenze programy z datového skladu. Operace exist porovnává tyto 2 zdroje a hledá, zda v tabulce faktů není nový program, který ještě nemáme v tabulce dimenzí.

Pak se v odvozeném sloupci vytvořen sloupec informatika, který, pokud Fakulta = FIS, bude hodnota ve sloupci informatika 1, pokud je hodnota ve sloupci Fakulta jiná než FIS, bude hodnota ve sloupci 0.

#### Transformační aktivity tabulky Faktů – L1 stage

**L1stage\_F\_bl\_koop\_f** má na vstupu zdroj L0\_bl\_koop\_f. L0\_bl\_koop\_f má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které budou zahrnuty do sady výsledků a to:*

*t1.[Období]*

*[Dodavatel výkonu]*

*[Typ výuky]*

*[Typ studia]*

*[Odběratel výkonu]*

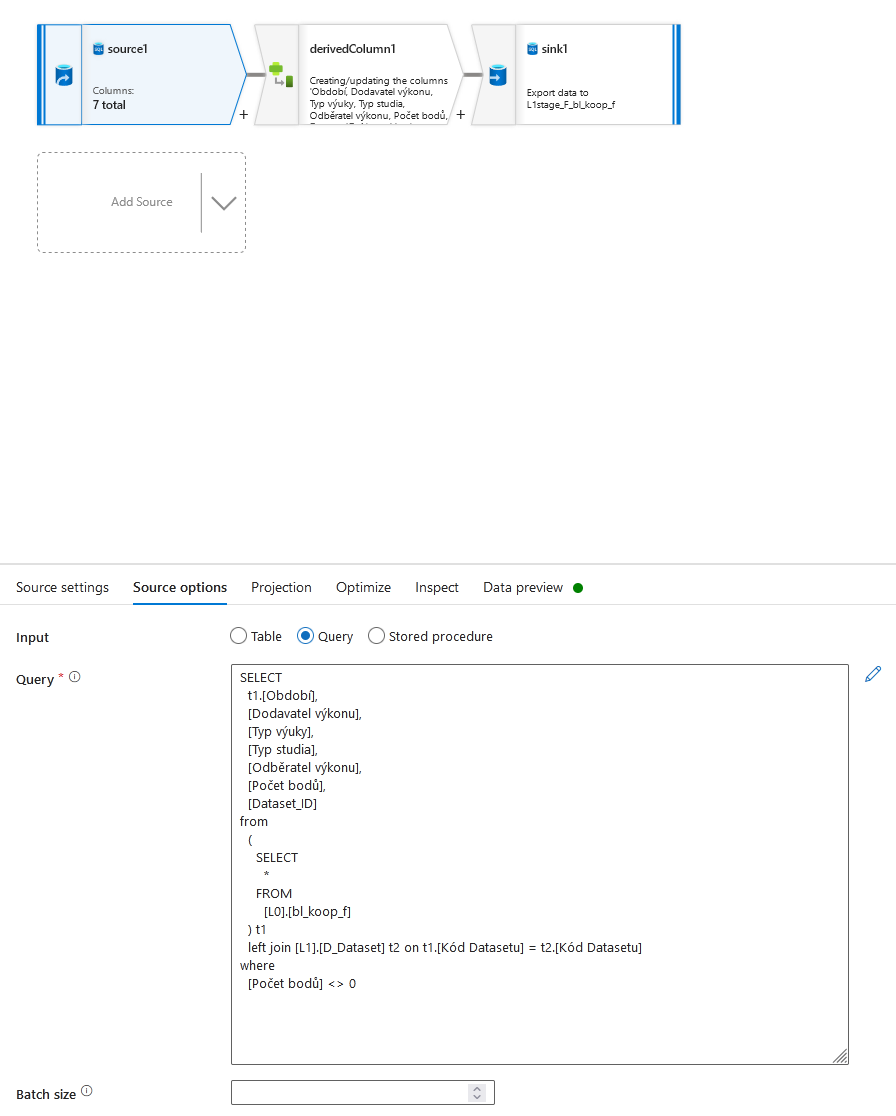
*[Počet bodů]*

*[Dataset\_ID]*

*Příkaz FROM obsahuje poddotaz, který vybere všechny sloupce z tabulky [L0].[bl\_koop\_f] a přiřadí jí jméno "t1".*

*Příkaz LEFT JOIN spojí poddotaz "t1" s tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu]. To znamená, že záznamy z "t1" budou spojeny se shodnými záznamy z "t2" na základě rovnosti uvedených sloupců.*

*Příkaz WHERE aplikuje filtrační podmínku [Počet bodů] <> 0, což znamená, že do výsledné množiny budou zahrnuty pouze řádky, kde hodnota sloupce [Počet bodů] není rovna nule.*



Obrázek č.25 – Source options v aktivitě L1stage\_F\_bl\_koop\_f (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Dodavatel výkonu", "Typ výuky", "Typ studia", "Odběratel výkonu" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Následný Přirozený klíč může vypadat následovně: "FFU-prímá výuka-B-FFU-3"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_bl\_koop\_f.

**L1stage\_F\_bl\_koop\_f\_poskyt** má na vstupu zdroj L0\_bl\_koop\_f\_poskyt. L0\_bl\_koop\_f\_poskyt má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které budou zahrnuty do sady výsledků:*

*t1.[Období]*

*[Typ výuky]*

*[Typ studia]*

*[Odběratel výkonu]*

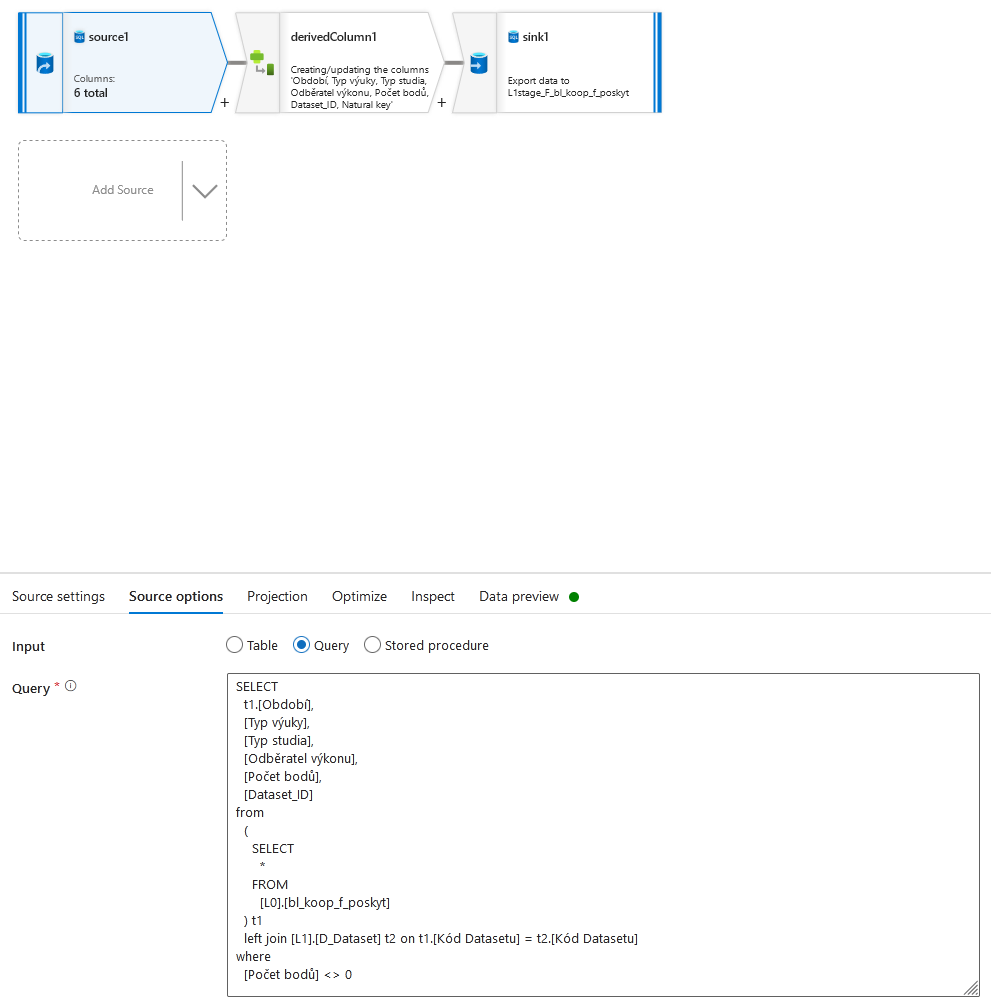
*[Počet bodů]*

*[Dataset\_ID]*

*Příkaz FROM obsahuje poddotaz, který vybere všechny sloupce z tabulky [L0].[bl\_koop\_f\_poskyt] a přiřadí jí alias "t1".*

*Příkaz LEFT JOIN spojí poddotaz "t1" s tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód datasetu] = t2.[Kód datasetu]. To znamená, že záznamy z "t1" budou spojeny se shodnými záznamy z "t2" na základě rovnosti uvedených sloupců.*

*Příkaz WHERE aplikuje filtrační podmínku [Počet bodů] <> 0, což znamená, že do výsledné množiny budou zahrnuty pouze řádky, kde hodnota sloupce [Počet bodů] není rovna nule.*



Obrázek č.26 – Source options v aktivitě L1stage\_F\_bl\_koop\_f\_poskyt (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Typ výuky", "Typ studia", "Odběratel výkonu" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Následný Přirozený klíč může vypadat následovně: "prímá výuka-B-FFU-3"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_bl\_koop\_f.\_poskyt.

**L1stage\_F\_h\_cv** má na vstupu zdroj L0\_h\_cv. L0\_h\_cv má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz e SELECT určuje sloupce, které budou zahrnuty do sady výsledků:*

*t1.[Období]*

*[Fakulta]*

*[Dataset\_ID]*

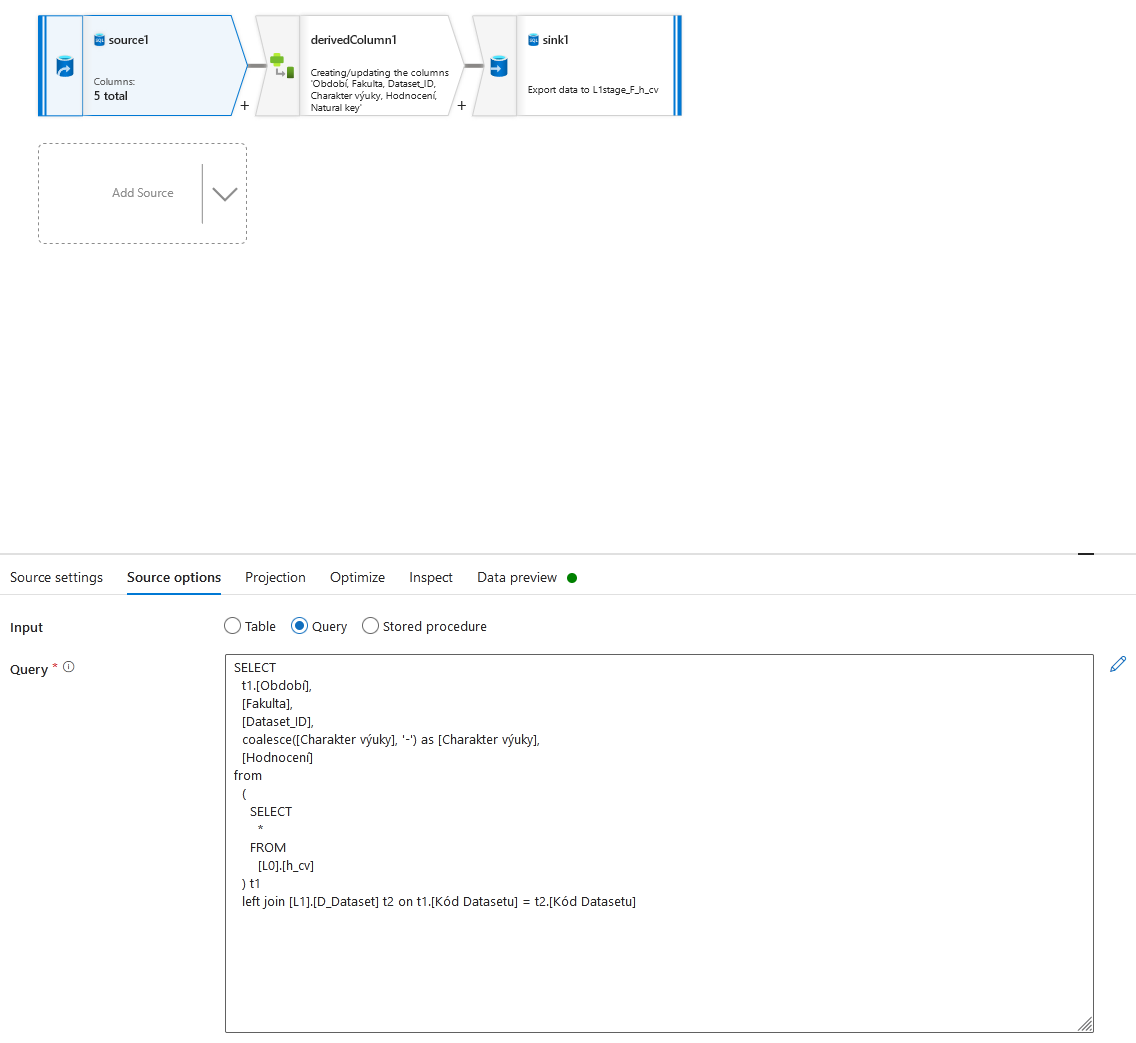
*coalesce([Charakter výuky], '-') as [Charakter výuky]*

*[Hodnocení]*

*Příkaz FROM obsahuje poddotaz, který vybere všechny sloupce z tabulky [L0].[h\_cv] a přiřadí jí alias "t1".*

*Příkaz LEFT JOIN spojí poddotaz "t1" s tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu]. To znamená, že záznamy z "t1" budou spojeny se shodnými záznamy z "t2" na základě rovnosti uvedených sloupců.*

*Příkaz COALESCE se v příkazy SELECT používá pro zpracování nulových hodnot. Vezme hodnotu sloupce [Charakter výuky] a pokud je nulová, nahradí ji pomlčkou ("-").*



Obrázek č.27 – Source options v aktivitě L1stage\_F\_h\_cv (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Typ výuky", "Typ studia", "Odběratel výkonu" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč, jestli je [Charakter výuky] Null případně v našem případě "-" může vypadat následovně: "FFU---1"

Anebo jestli je v sloupci [Charakter výuky] hodnota, klíč může vypadat následovně: FFU-pocítacová výuka-1, FMV-jazyková výuka-1.

V operaci sink pak exportujeme data do tabulky L1stage\_F\_h\_cv.

**L1stage\_F\_h\_cv\_g** má na vstupu zdroj L0\_h\_cv\_g. L0\_h\_cv\_g má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které mají být zahrnuty do sady výsledků dotazu:*

*t1.[Období]: Sloupec [Období] z tabulky t1.*

*t2.[Dataset\_ID]: Sloupec [Dataset\_ID] z tabulky t2.*

*Funkce COALESCE slouží ke zpracování nulových hodnot ve sloupci [Charakter výuky] a jejich nahrazení pomlčkou (-). Výsledný sloupec je aliasován jako [Charakter výuky].*

*[Hodnocení].*

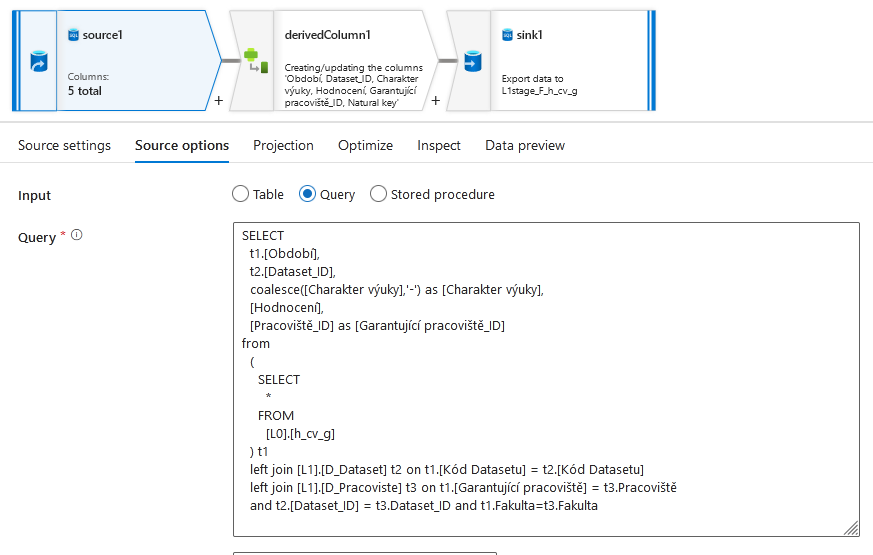
*[Pracoviště\_ID] je vybrán a pojmenován jako [Garantující pracoviště\_ID].*

*Příkaz FROM určuje tabulky použité v dotazu:*

*Začíná poddotazem (SELECT \* FROM [L0].[h\_cv\_g]) t1, který vybere všechny sloupce z tabulky [L0].[h\_cv\_g] a aliasuje ji jako t1.*

*První LEFT JOIN slouží ke spojení poddotazu t1 s tabulkou [L1].[D\_Dataset] na základě podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Další spojení LEFT JOIN se provede pro spojení předchozího výsledku spojení s tabulkou [L1].[D\_Pracoviště] na základě více podmínek: t1.[Garantující pracoviště] = t3.Pracoviště, t2.[Dataset\_ID] = t3.Dataset\_ID a t1.Fakulta = t3.Fakulta.*



Obrázek č.28 – Source options v aktivitě L1stage\_F\_h\_cv\_g (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Charakter výuky", řetězcovou reprezentaci "Garantující pracoviště\_ID " a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč, jestli je [Charakter výuky] Null případně v našem případě "-" klíč může vypadat následovně: "--5-1"

Anebo jestli je v sloupci [Charakter výuky] hodnota, klíč může vypadat následovně: "pocítacová výuka-5-1"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_h\_cv\_g.

**L1stage\_F\_h\_pred** má na vstupu zdroj L0\_h\_pred. L0\_h\_pred má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které mají být zahrnuty do sady výsledků dotazu:*

*t1.[Období]: z tabulky t1.*

*t2.[Dataset\_ID]: z tabulky t2.*

*[Forma studia]: Sloupec*

*[Mimosemestrální výuka]:*

*[Studentů]:*

*[Akcí]: Sloupec*

*Hodnocení]:*

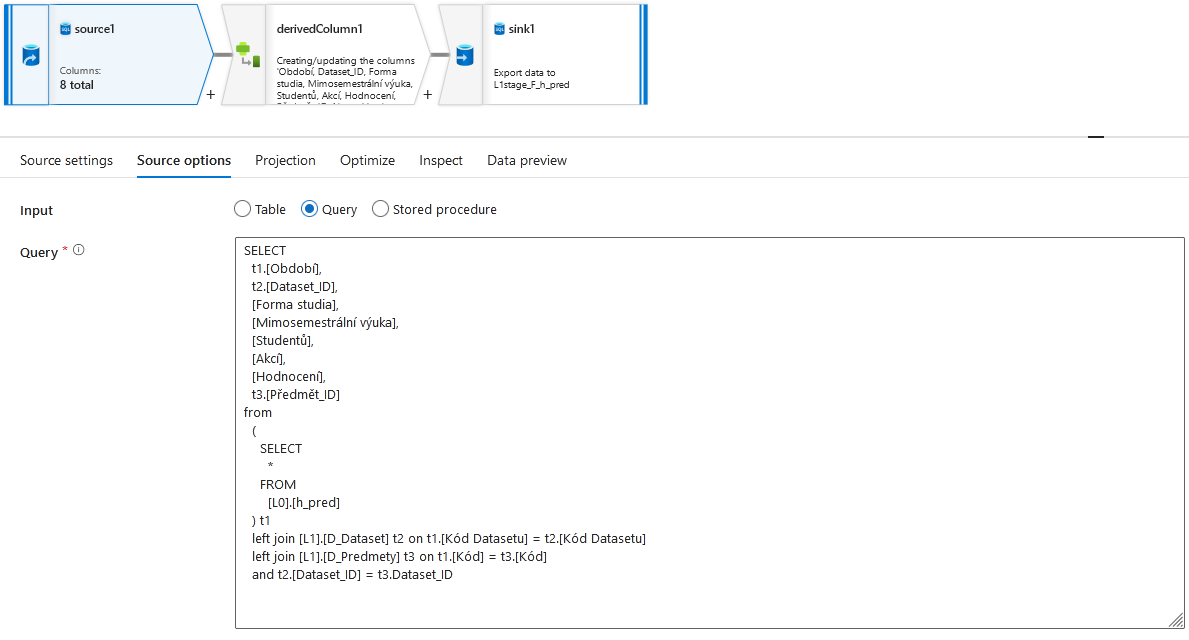
*t3. [Předmět\_ID]: z tabulky t3.*

*Příkaz FROM určuje tabulky použité v dotazu:*

*Začíná poddotazem (SELECT \* FROM [L0].[h\_pred]) t1, který vybírá všechny sloupce z tabulky [L0].[h\_pred] a označuje ji jako t1.*

*Nejprve se provede LEFT JOIN, který spojí poddotaz t1 s tabulkou [L1].[D\_Dataset] na základě podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Provede se další spojení LEFT JOIN, které spojí výsledek předchozího spojení s tabulkou [L1].[D\_Predmety] na základě podmínek t1.[Kód] = t3.[Kód] a t2.[Dataset\_ID] = t3.Dataset\_ID.*



Obrázek č.29 – Source options v aktivitě L1stage\_F\_pred (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Forma studia", "Mimosemestrální výuka", řetězcovou reprezentaci "Akcí", řetězcovou reprezentaci "Předmět\_ID" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč, jestli je [Charakter výuky] Null případně v našem případě "-" klíč může vypadat následovně: "--5-1"

Nebo jestli je v sloupci [Charakter výuky] hodnota, klíč může vypadat následovně: "pocítacová výuka-5-1"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_h\_pred.

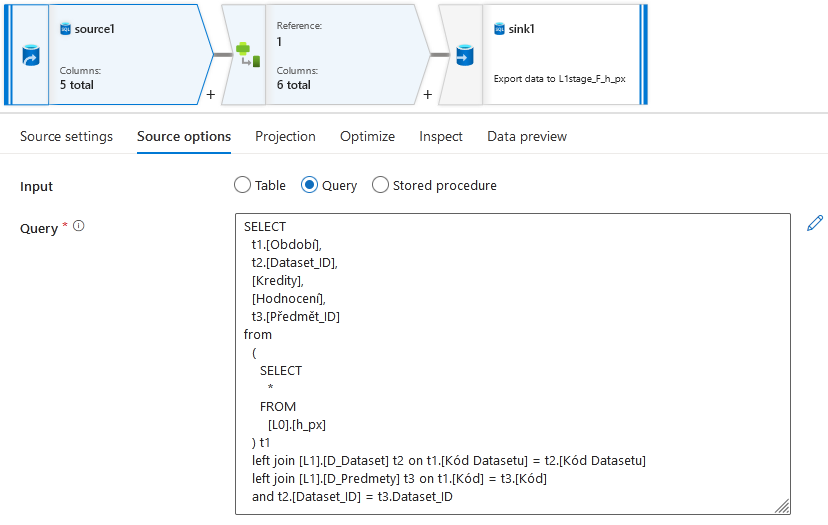
**L1stage\_F\_h\_px** má na vstupu zdroj L0\_h\_px. L0\_h\_px má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají načíst a to: [Období], t2.[Dataset\_ID], [Kredity], [Hodnocení] a t3.[Předmět\_ID].*

*Příkaz FROM obsahuje poddotaz (SELECT \* FROM [L0].[h\_px]) t1, který vybere všechny sloupce z tabulky [L0].[h\_px] a přiřadí jí alias t1.*

*První operace LEFT JOIN je provedena nad tabulkou [L1].[D\_Dataset] s aliasem t2.Spojí t1 a t2 pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu]. To znamená, že sloupce Kód Datasetu z t1 a t2 se musí shodovat, aby došlo ke spojení.*

*Druhá operace LEFT JOIN se provede nad tabulkou [L1].[D\_Predmety] alias t3. Spojí t1 a t3 pomocí dvou podmínek: t1.[Kód] = t3.[Kód] a t2.[Dataset\_ID] = t3.Dataset\_ID. Obě podmínky musí být splněny, aby spojení proběhlo.*



Obrázek č.30 – Source options v aktivitě L1stage\_F\_px (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje řetězcovou reprezentaci "Předmět\_ID" a "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "7446-7"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_h\_px.

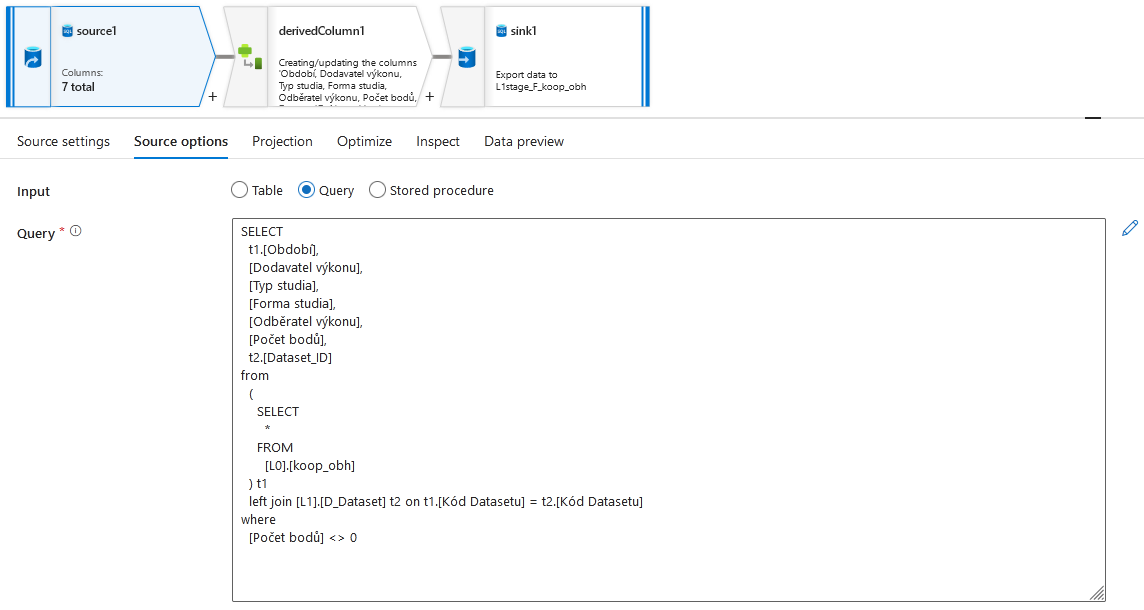
**L1stage\_F\_koop\_obh** má na vstupu zdroj L0\_koop\_obh. L0\_koop\_obh má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají načíst a to: t1.[Období], [Dodavatel výkonu], [Typ studia], [Forma studia], [Odběratel výkonu], [Počet bodů] a t2.[Dataset\_ID].*

*Příkaz FROM obsahuje poddotaz (SELECT \* FROM [L0].[koop\_obh]) t1, který vybere všechny sloupce z tabulky [L0].[koop\_obh] a přiřadí jí alias t1.*

*Operace LEFT JOIN se provede nad tabulkou [L1].[D\_Dataset] s aliasem t2. Spojí t1 a t2 pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu]. To znamená, že sloupce Kód Datasetu z t1 a t2 se musí shodovat, aby došlo ke spojení.*

*Příkaz WHERE aplikuje filtrační podmínku [Počet bodů] <> 0, což znamená, že do výsledné množiny budou zahrnuty pouze řádky, kde hodnota sloupce [Počet bodů] není rovna nule.*



Obrázek č.31 – Source options v aktivitě L1stage\_F\_koop\_obh (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Dodavatel výkonu", "Typ studia", "Forma studia", "Odběratel výkonu" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "FFU-B-prez-FFU-7".

V operaci sink pak exportujeme data do tabulky L1stage\_F\_koop\_obh.

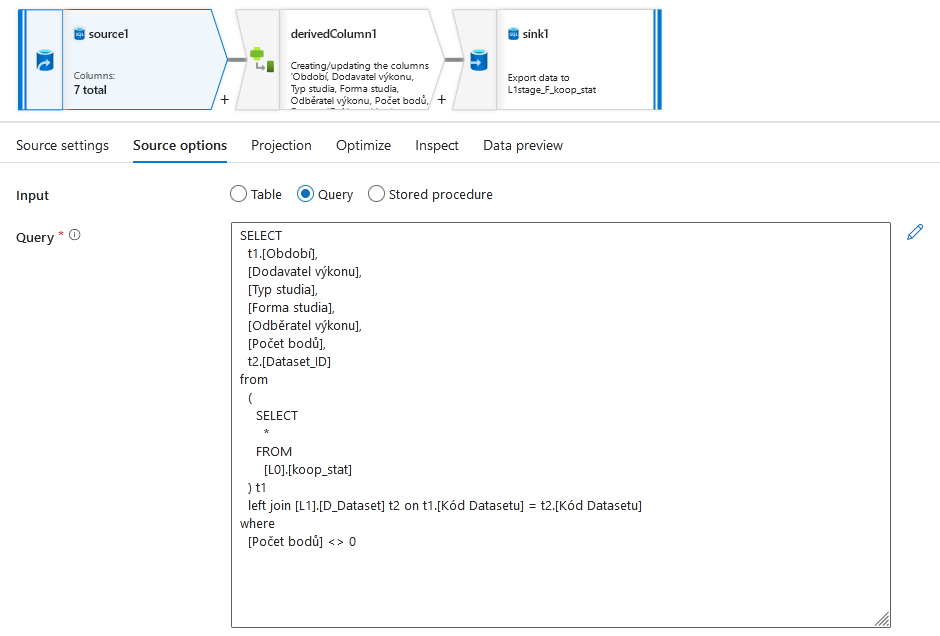
**L1stage\_F\_koop\_stat** má na vstupu zdroj L0\_koop\_stat. L0\_koop\_stat má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají získat: t1.[Období], [Dodavatel výkonu], [Typ studia], [Forma studia], [Odběratel výkonu], [Počet bodů] a t2.[Dataset\_ID].*

*Příkaz FROM obsahuje poddotaz (SELECT \* FROM [L0].[koop\_stat]) t1, který vybere všechny sloupce z tabulky [L0].[koop\_stat] a přiřadí jí alias t1.*

*Operace LEFT JOIN se provede nad tabulkou [L1].[D\_Dataset] s aliasem t2. Spojí t1 a t2 pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu]. To znamená, že sloupce Kód Datasetu z t1 a t2 se musí shodovat, aby došlo ke spojení.*

*Příkaz WHERE slouží k použití filtrační podmínky. Podmínka [Počet bodů] <> 0 určuje, že do výsledku mají být zahrnuty pouze řádky, u nichž sloupec [Počet bodů] není 0.*



Obrázek č.32 – Source options v aktivite L1stage\_F\_koop\_stat (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Dodavatel výkonu", "Typ studia", "Forma studia", "Odběratel výkonu" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "FFU-N-prez-FFU-7"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_koop\_stat.

**L1stage\_F\_koop\_total** má na vstupu 4 zdroje a to: L1stage\_F\_koop\_stat, L1stage\_F\_koop\_ved\_p, L1stage\_F\_koop\_obh, L1stage\_F\_pr\_koop\_vyuk.

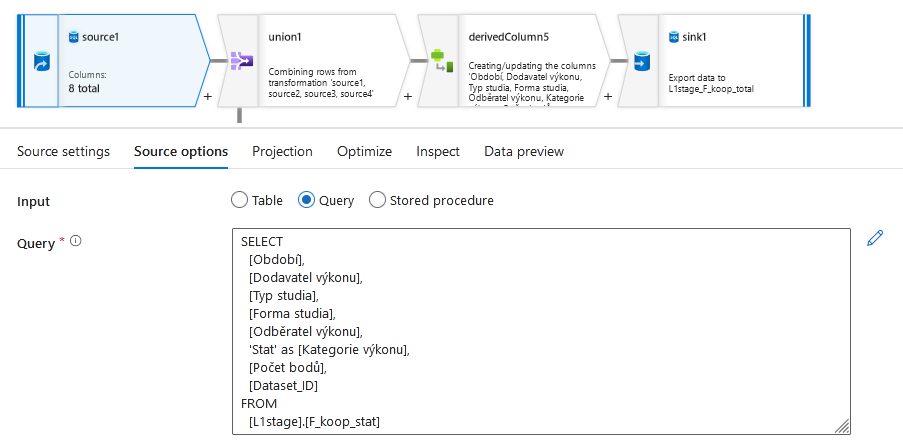
Všechny zdroje mají ve zdrojových možnostech vstupní dotaz místo tabulky. Dotaz v L1stage\_F\_koop\_stat provádí následující:

*Příkaz SELECT určuje sloupce, které se mají načíst: [Období], [Dodavatel výkonu], [Typ studia], [Forma studia], [Odběratel výkonu], 'Stat' jako [Kategorie výkonu], [Počet bodů] a [Dataset\_ID].*

*Příkaz FROM určuje tabulku [L1stage].[F\_koop\_stat], ze které se data získávají.*

*V příkazu SELECT je* '*Stat*' *as [Kategorie výkonu] použito k přidání konstantní hodnoty "Stat" jako sloupec s názvem [Kategorie výkonu] do množiny výsledků.*

"Stat" je zkratka pro Přehled bodů za státní a souborné zkoušky.



Obrázek č.33 – Source options v aktivitě L1stage\_F\_koop\_total - zdroj L1stage\_F\_koop\_stat (Zdroj: Autoři)

Dotaz v L1stage\_F\_koop\_ved\_p a L1stage\_F\_koop\_obh mají vstupní dotaz stejný jako "L1stage\_F\_koop\_stat" jediný rozdíl je počátečním příkazu SELECT kde v předchozím případě byla kategorie výkonu "stat", tak teď je to pro L1stage\_F\_koop\_ved skratka "ved" jako: "přehled bodů za vedení závěrečných prací" a pro L1stage\_F\_koop\_obh skratka "obh" jako: "obhajoby závěrečných prací".

Poslední vstupní zdroj je L1stage\_F\_pr\_koop\_vyuk, který má také dotaz na vstupu místo tabulky. Dotaz v L1stage\_F\_pr\_koop\_vyuk provádí následující:

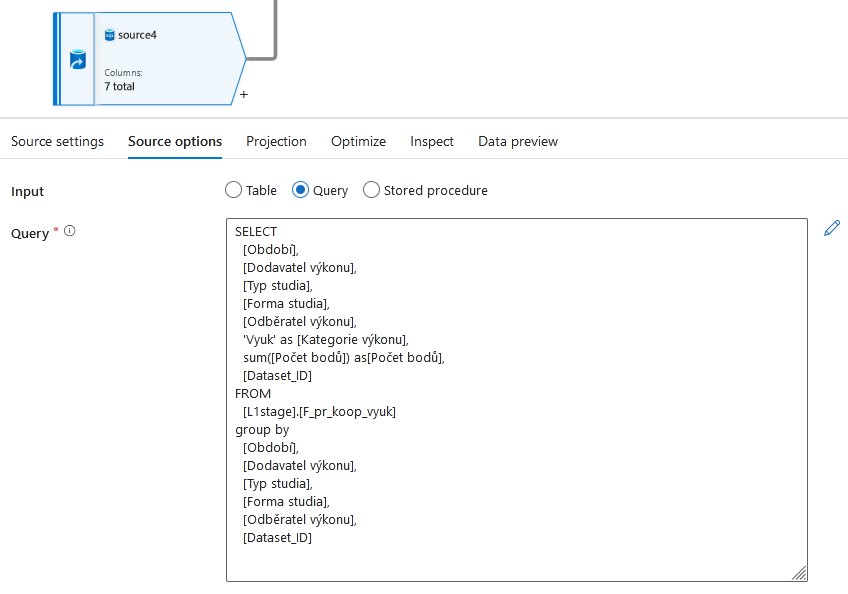
*Příkaz SELECT určuje sloupce, které se mají načíst: [Období], [Dodavatel výkonu], [Typ studia], [Forma studia], [Odběratel výkonu], 'Vyuk' as [Kategorie výkonu], sum([Počet bodů]) jako [Počet bodů] a [Dataset\_ID, Funkce sum([Počet bodů]) vypočítá součet sloupce [Počet bodů] v rámci každé skupiny.*

*Příkaz FROM určuje tabulku [L1stage].[F\_pr\_koop\_vyuk], ze které se data získávají.*

*Příkaz GROUP BY slouží ke seskupení výsledné sady na základě konkrétních sloupců a to: [Období], [Dodavatel výkonu], [Typ studia], [Forma studia], [Odběratel výkonu] a [Dataset\_ID]. To znamená, že dotaz vypočítá souhrnný součet [Počet bodů] pro každou jedinečnou kombinaci těchto sloupců.*

*'Vyuk' as [Kategorie výkonu] přidá do množiny výsledků konstantní hodnotu 'Vyuk' jako sloupec s názvem [Kategorie výkonu].*

"Vyuk" je zkratka pro Přehled bodů za výuku předmětů.



Obrázek č.34 – Source options v aktivitě L1stage\_F\_koop\_total - zdroj L1stage\_F\_pr\_koop\_vyuk (Zdroj: Autoři)

Následně za pomocí funkce Union spojíme všechny 4 datové toky do jednoho. 

Obrázek č.35 – UNION všech 4 datových toků (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Dodavatel výkonu", "Typ studia", "Forma studia", "Odběratel výkonu", "Kategorie výkonu" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "FMV-B-komb-FMV-Obh-7", podle toho, jaká je "Kategorie výkonu".

**L1stage\_F\_koop\_ved\_p** má na vstupu zdroj L0\_koop\_ved\_p. L0\_koop\_ved\_p má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Hlavní příkaz vybere následující sloupce:*

*[Období]*

*[Dodavatel výkonu]*

*[Typ studia]*

*[Forma studia]*

*[Odběratel výkonu]*

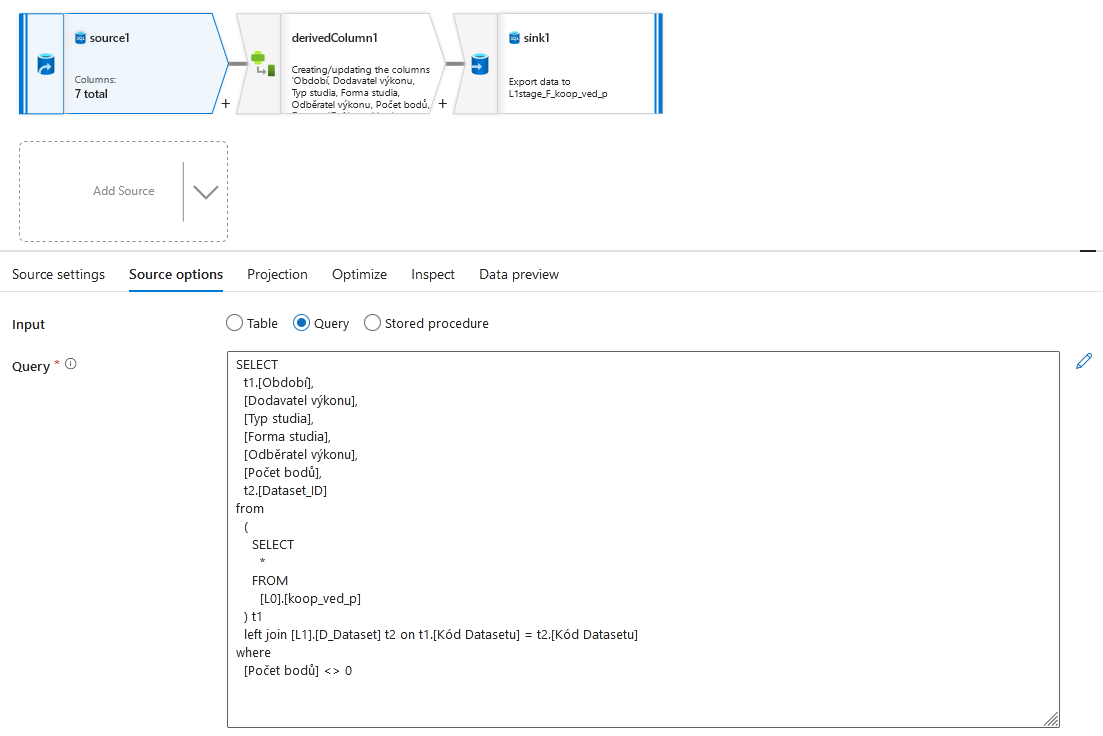
*[Počet bodů]*

*t2.[Dataset\_ID]*

*Příkaz začíná vytvořením odvozené tabulky s názvem "t1" výběrem všech sloupců z tabulky [L0].[koop\_ved\_p]. Tato odvozená tabulka slouží jako levá strana následné operace LEFT JOIN.*

*Příkaz provede operaci left join, při níž spojí odvozenou tabulku "t1" s tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Příkaz WHERE filtruje výsledek vyloučením řádků, u nichž hodnota [Počet bodů] není rovna 0.*



Obrázek č.36 – Source options v aktivitě L1stage\_F\_koop\_ved\_p(Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Dodavatel výkonu", "Typ studia", "Forma studia", "Odběratel výkonu" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "FFU-B-prez-FFU-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_koop\_ved\_p.

**L1stage\_F\_obh\_p\_dod** má na vstupu zdroj L0\_obh\_p\_dod. L0\_obh\_p\_dod má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Hlavní příkaz vybere následující sloupce:*

*[Období]*

*t2.[Dataset\_ID]*

*[Typ studia]*

*[Počet bodů]*

*t3.[Pracoviště\_ID]*

*Příkaz začíná vytvořením odvozené tabulky s názvem "t1" výběrem všech sloupců z tabulky [L0].[obh\_p\_dod]. Tato odvozená tabulka slouží jako levá strana následných operací LEFT JOIN.*

*Příkaz provede operaci left join, při níž se odvozená tabulka "t1" spojí s tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu].*

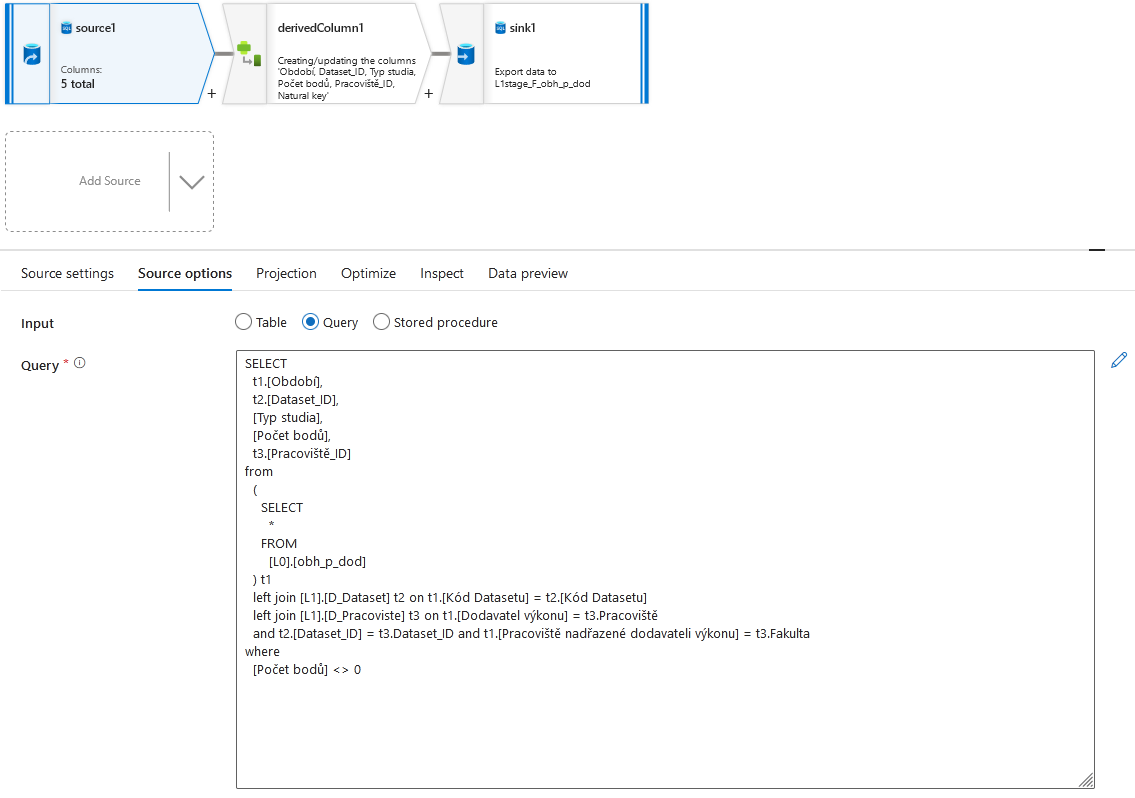
*Příkaz provede další operaci left join a spojí výsledek předchozího spojení s tabulkou [L1].[D\_Pracoviste] pomocí více podmínek:*

*t1.[Dodavatel výkonu] = t3.Pracoviště*

*t2.[Dataset\_ID] = t3.Dataset\_ID*

*t1.[Pracoviště nadřazené dodavateli výkonu] = t3.Fakulta*

*Příkaz WHERE filtruje výsledek vyloučením řádků, kde hodnota [Počet bodů] není rovna 0.*



Obrázek č.37 – Source options v aktivitě L1stage\_F\_obh\_p\_dod (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Typ studia", řetězcovou reprezentaci "Pracoviště\_ID" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "B-232-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_obh\_p\_dod.

**L1stage\_F\_obh\_p\_od** má na vstupu zdroj L0\_obh\_p\_od. L0\_obh\_p\_od má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Hlavní příkaz vybere následující sloupce:*

*[Období]*

*[Dataset\_ID]*

*[Odběratel výkonu]*

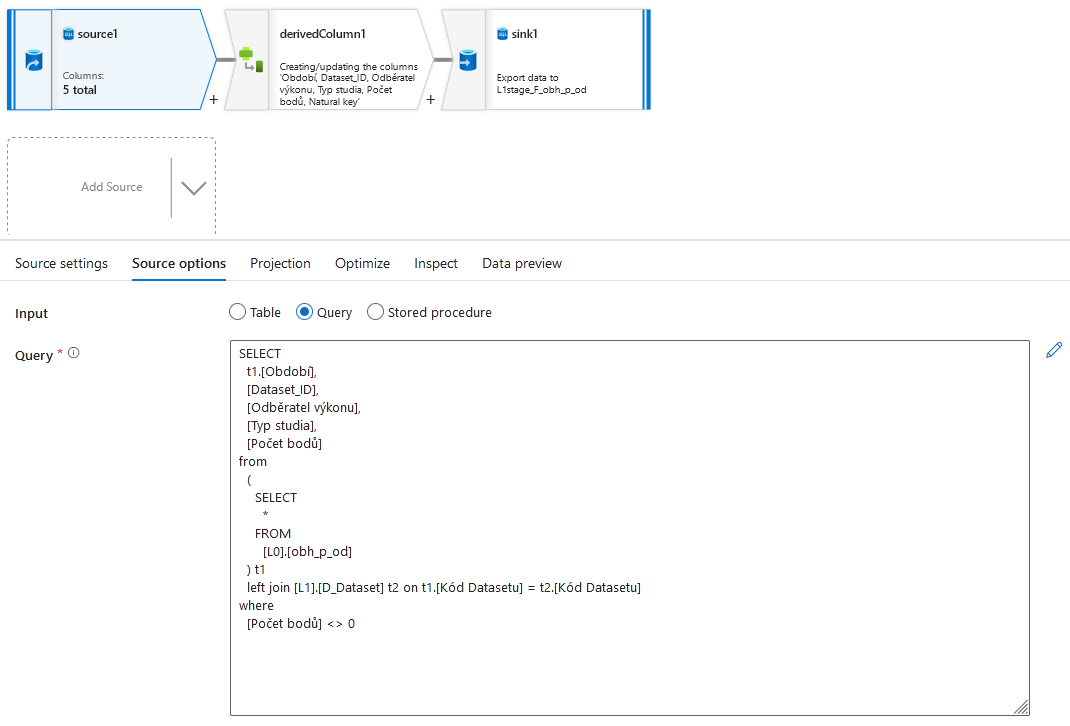
*[Typ studia]*

*[Počet bodů]*

*Příkaz začíná vytvořením odvozené tabulky s názvem "t1" výběrem všech sloupců z tabulky [L0].[obh\_p\_od]. Tato odvozená tabulka slouží jako levá strana následné operace LEFT JOIN.*

*Příkaz provede operaci left join, při níž spojí odvozenou tabulku "t1" s tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Příkaz WHERE filtruje výsledek vyloučením řádků, u nichž hodnota [Počet bodů] není rovna 0.*



Obrázek č.38 – Source options v aktivitě L1stage\_F\_obh\_p\_od (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Odběratel výkonu", " Typ studia" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "FFU-B-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_obh\_p\_od.

**L1stage\_F\_obh\_p\_prehled** má na vstupu zdroj L0\_obh\_p\_prehled. L0\_obh\_p\_prehled má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Hlavní příkaz vybere následující sloupce:*

*[Období]*

*t2.[Dataset\_ID]*

*[Typ studia]*

*[Forma studia]*

*[Odběratel výkonu]*

*[Počet pokusů] \* [Podíl - nepřímá výuka - prezenční] as [Počet bodů]*

*t3. [Předmět\_ID]*

*t4. [Pracoviště\_ID]*

*Vypočtený sloupec [Počet bodů] se získá vynásobením hodnot [Počet pokusů] a [Podíl - nepřímá výuka - prezenční].*

*Příkaz začíná vytvořením odvozené tabulky s názvem "t1" výběrem všech sloupců z tabulky [L0].[obh\_p\_prehled]. Tato odvozená tabulka slouží jako levá strana následných operací LEFT JOIN.*

*Příkaz provede operaci left join, při níž spojí odvozenou tabulku "t1" s tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu]. odpovídající řádek.*

*Příkaz provede další operaci left join a spojí výsledek předchozího spojení s tabulkou [L1].[D\_Predmety] pomocí více podmínek:*

*t1.[Kód] = t3.[Kód].*

*t2.[Dataset\_ID] = t3.Dataset\_ID*

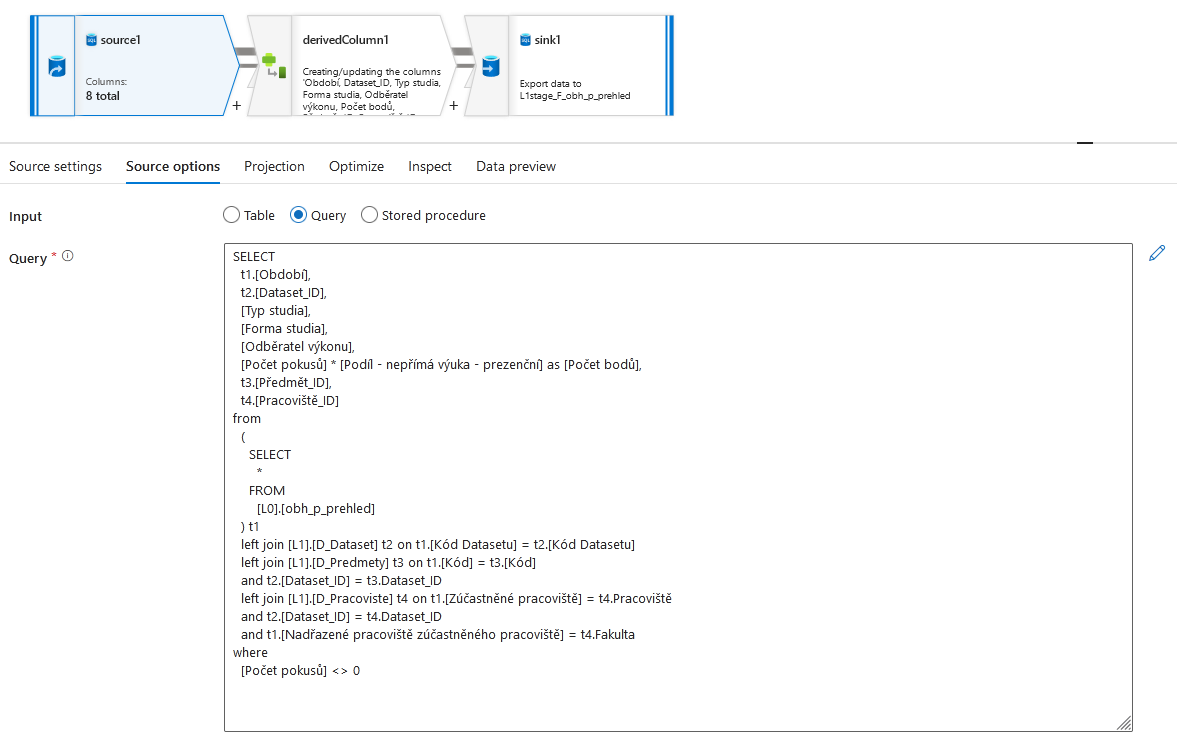
*Příkaz provede další operaci left join, která spojí výsledek předchozího spojení s tabulkou [L1].[D\_Pracoviste] pomocí více podmínek:*

*t1.[Zúčastněné pracoviště] = t4.Pracoviště*

*t2.[Dataset\_ID] = t4.Dataset\_ID*

*t1.[Nadřazené pracoviště zúčastněného pracoviště] = t4.Fakulta*

*Příkaz WHERE filtruje výsledek vyloučením řádků, kde hodnota [Počet pokusů] není rovna 0.*



Obrázek č.39 – Source options v aktivitě L1stage\_F\_obh\_p\_prehled (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Typ studia", "Forma studia", "Odběratel výkonu", řetězcovou reprezentaci "Předmět\_ID", řetězcovou reprezentaci "Pracoviště\_ID" a řetězcovou reprezentaci " Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "B-prezencní-FFU-6716-231-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_obh\_p\_prehled.

**L1stage\_F\_pr\_koop\_vyuk** má na vstupu zdroj L0\_pr\_koop\_vyuk. L0\_pr\_koop\_vyuk má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Hlavní příkaz vybere následující sloupce:*

*[Období]*

*[Dodavatel výkonu]*

*[Typ výuky]*

*[Typ studia]*

*[Forma studia]*

*[Mimosemestrální výuka]*

*[Odběratel výkonu]*

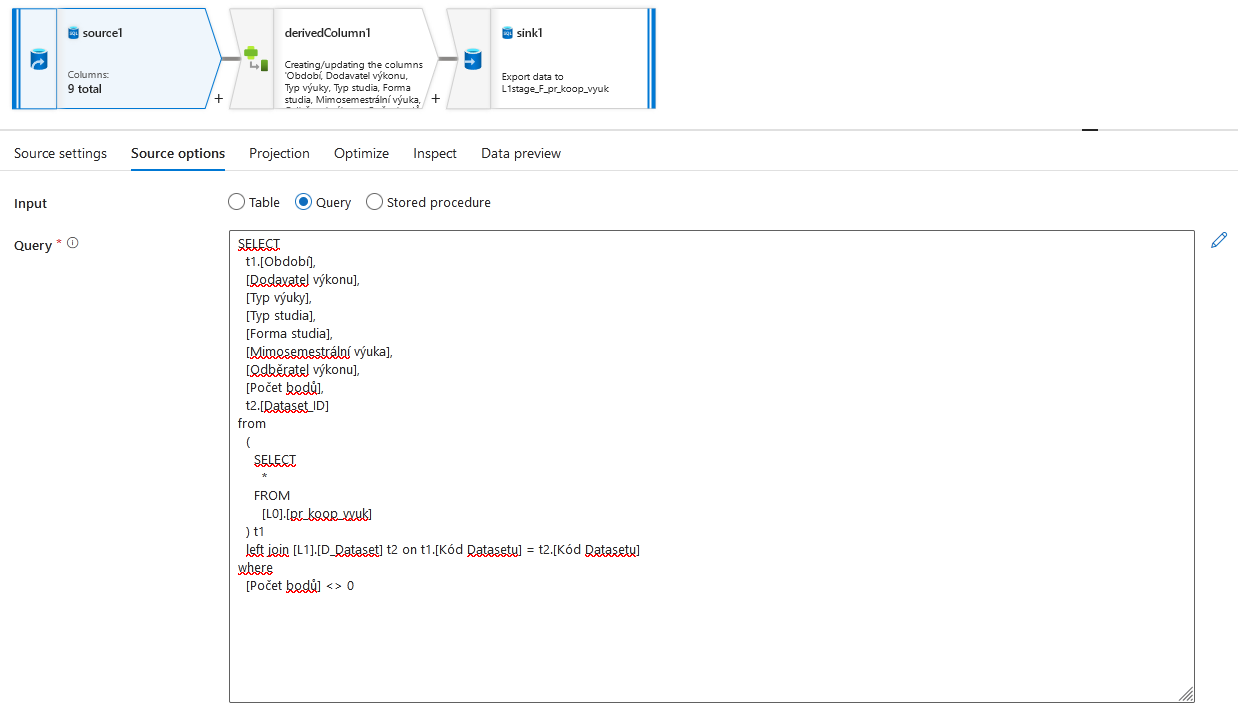
*[Počet bodů]*

*t2.[Dataset\_ID]*

*Příkaz začíná vytvořením odvozené tabulky s názvem "t1" výběrem všech sloupců z tabulky [L0].[pr\_koop\_vyuk]. Tato odvozená tabulka slouží jako levá strana následné operace LEFT JOIN.*

*Příkaz provede operaci left join, při níž spojí odvozenou tabulku "t1" s tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Příkaz WHERE filtruje výsledek vyloučením řádků, u nichž hodnota [Počet bodů] není rovna 0.*



Obrázek č.40 – Source options v aktivitě L1stage\_F\_pr\_koop\_vyuk(Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Dodavatel výkonu", "Typ výuky", "Typ studia", "Forma studia", "Mimosemestrální výuka", "Odběratel výkonu" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "FFU-prímá výuka-B-prez-ne-FFU-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_pr\_koop\_vyuk.

**L1stage\_F\_pred\_podily** má na vstupu zdroj L0\_pred\_podil. L0\_pred\_podil má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz vybere následující sloupce:*

*t1.[Období],*

*t3.[Předmět\_ID],*

*t3.[Garantující pracoviště\_ID],*

*t5.[Pracoviště\_ID] as [Pracoviště podílející se na výuce\_ID],*

*t2.[Dataset\_ID],*

*[Podíl - přednáška - prezenční],*

*[Podíl - cvičení - prezenční],*

*[Podíl - nepřímá výuka - prezenční],*

*[Podíl - přednáška - mimosemestrální výuka],*

*[Podíl - cvičení - mimosemestrální výuka],*

*[Podíl - nepřímá výuka - mimosemestrální výuka],*

*[Podíl - přednáška - kombinovaná],*

*[Podíl - cvičení - kombinovaná],*

*[Podíl - nepřímá výuka - kombinovaná],*

*[Podíl - přednáška - distanční],*

*[Podíl - cvičení - distanční],*

*[Podíl - nepřímá výuka - distanční]*

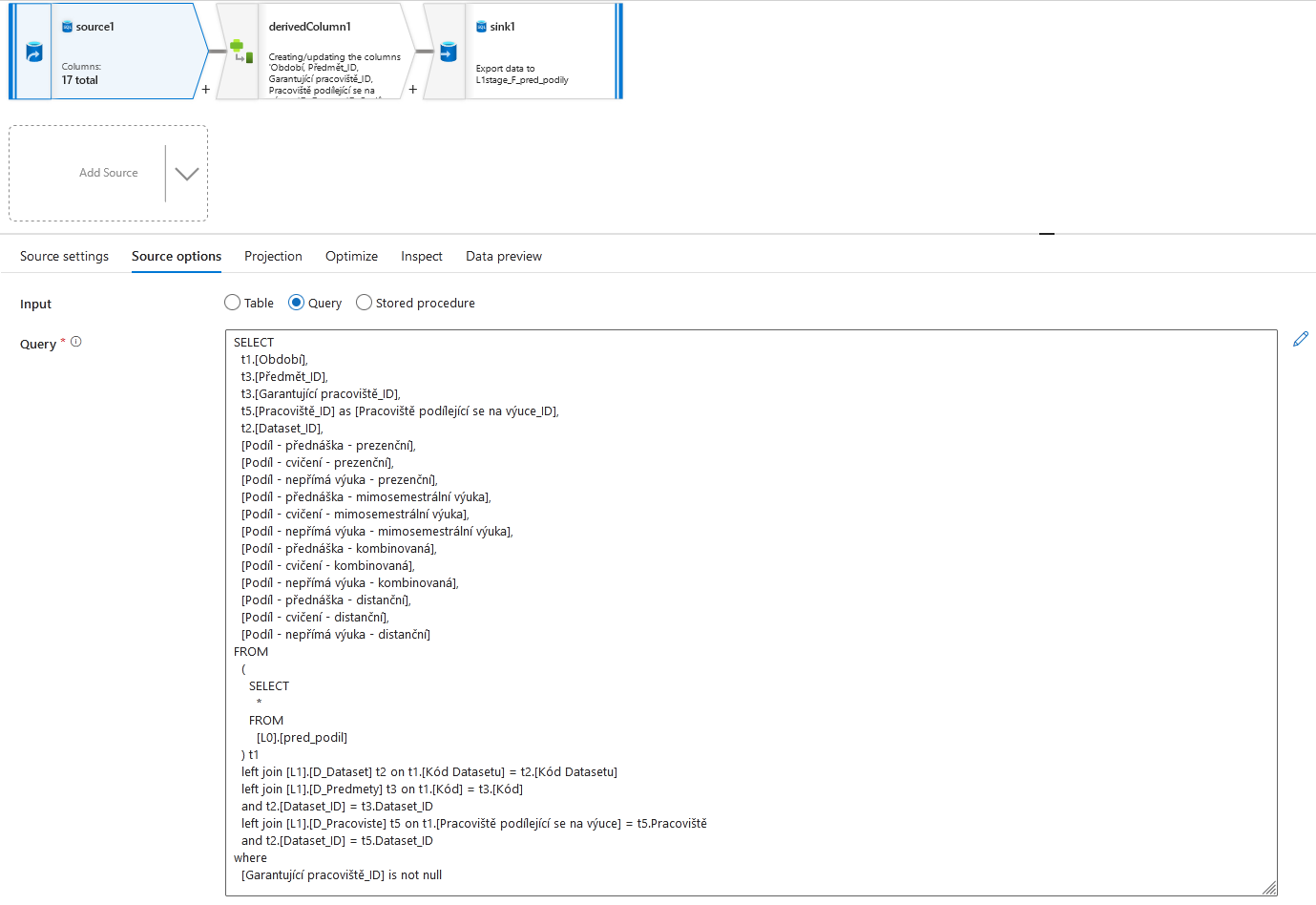
*Příkaz specifikuje zdrojovou tabulku pro první část dotazu. Vybere všechny sloupce z tabulky [L0].[pred\_podil] a přiřadí jí alias t1.*

*Příkaz provede operaci left join mezi t1 a tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód Datasetu] = t2.[Kód Datasetu]. Výsledku přiřadí alias t2.*

*Příkaz spojení left join se provede mezi t1 a tabulkou [L1].[D\_Dataset] pomocí podmínek t1.[Kód] = t3.[Kód] a t2.[Dataset\_ID] = t3.Dataset\_ID. Výsledku je přiřazen alias t3.*

*Příkaz left join se provede mezi t1 a tabulkou [L1].[D\_Pracoviste] za použití podmínek t1.[Pracoviště podílející se na výuce\_ID] = t5.Pracoviště a t2.[Dataset\_ID] = t5.Dataset\_ID. Výsledku je přiřazen alias t5.*

*Příkaz aplikuje filtrační podmínku WHERE [Garantující pracoviště\_ID] není null, která zahrnuje pouze řádky, v nichž sloupec [Garantující pracoviště\_ID] není null.*

Obrázek č.41 – Source options v aktivitě L1stage\_F\_pred\_podily (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje řetězcovou reprezentaci "Předmět\_ID", "Garantující pracoviště\_ID", "Pracoviště podílející se na výuce\_ID", "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "9532-306-306-9"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_pred\_podily.

**L1stage\_F\_pred\_prehled** má na vstupu zdroj L1stage\_F\_vy\_od\_pred. L1stage\_F\_vy\_od\_pred má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz vybere následující sloupce:*

*[Období]*

*coalesce(t1.[Charakter výuky], '-') as [Charakter výuky]*

*t1.[Forma studia]*

*t1.[Mimosemestrální výuka]*

*t1.[Odběratel výkonu]*

*t1.[Typ jednotky výuky]*

*t1.[Počet bodů]*

*t1.[Dataset\_ID]*

*t1.[Předmět\_ID]*

*t1.[Garantující pracoviště\_ID]*

*t2.[Pracoviště podílející se na výuce\_ID]*

*[Podíl - přednáška - prezenční]*

*[Podíl - cvičení - prezenční]*

*[Podíl - nepřímá výuka - prezenční]*

*[Podíl - přednáška - mimomestrální výuka]*

*[Podíl - cvičení - mimosemestrální výuka]*

*[Podíl - nepřímá výuka - mimosemestrální výuka]*

*[Podíl - přednáška - kombinovaná]*

*[Podíl - cvičení - kombinovaná]*

*[Podíl - nepřímá výuka - kombinovaná]*

*[Podíl - přednáška - distanční]*

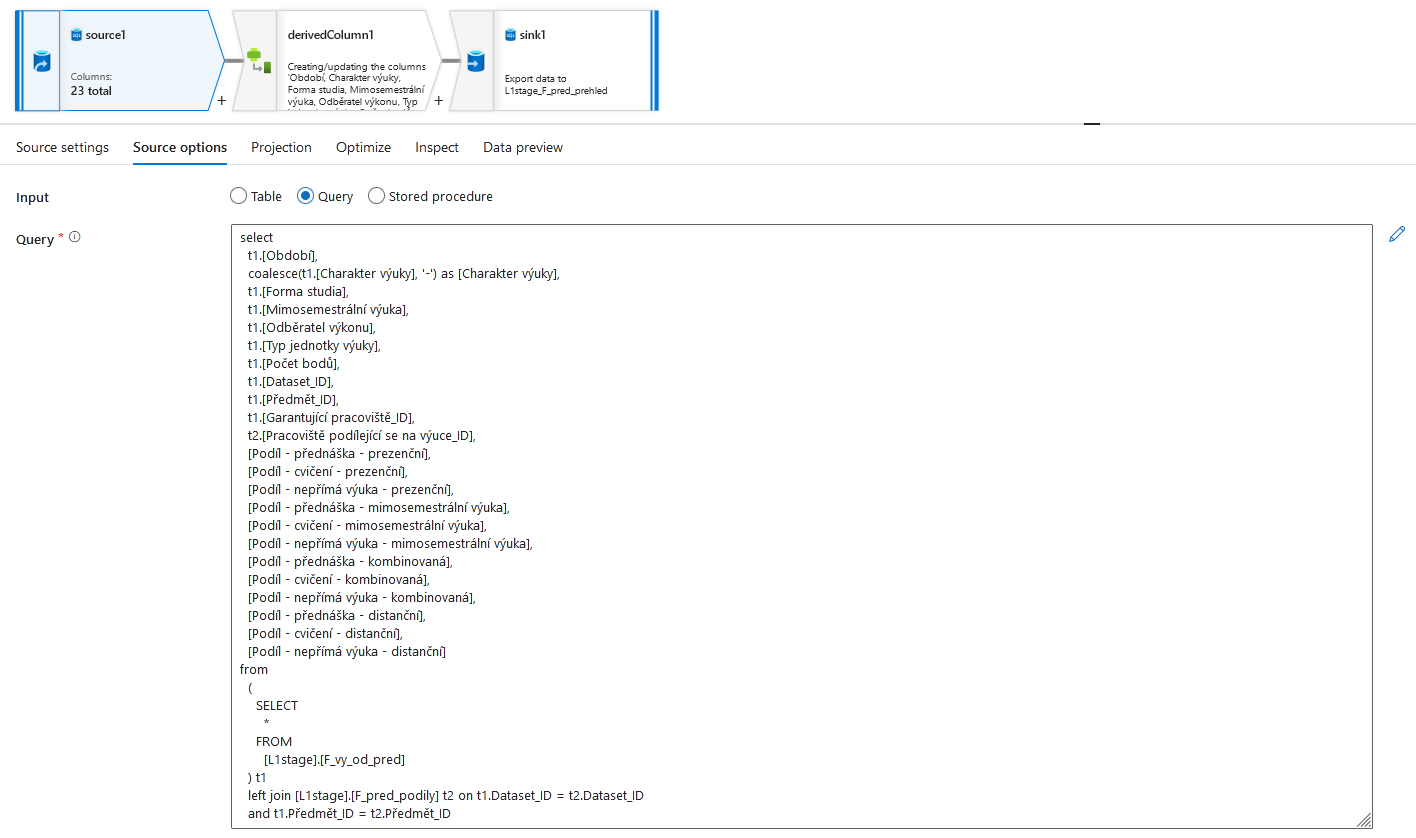
*[Podíl - cvičení - distanční]*

*[Podíl - nepřímá výuka - distanční]*

*Funkce COALESCE nahradí případné nulové hodnoty v t1.[Charakter výuky] pomlčkou ("-"). Tím je zajištěno, že i když je hodnota nulová, ve výsledku se místo ní zobrazí pomlčka.*

*Příkaz určuje zdrojovou tabulku pro první část dotazu. Vybere všechny sloupce z tabulky [L1stage].[F\_vy\_od\_pred] a přiřadí jí alias t1.*

*Příkaz provede operaci left join mezi t1 a tabulkou [L1stage].[F\_vy\_pred\_podily] pomocí podmínek t1.Dataset\_ID = t2.Dataset\_ID a t1.Předmět\_ID = t2.Předmět\_ID. Výsledku je přiřazen alias t2.*



Obrázek č.42 – Source options v aktivitě L1stage\_F\_pred\_prehled (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Charakter výuky", "Forma studia", "Mimosemestrální výuka", "Odběratel výkonu", "Typ jednotky výuky" a řetězcovou reprezentaci "Dataset\_ID", "Předmět\_ID", "Pracoviště podílející se na výuce\_ID" když je ve sloupci je NULL a "Garantující pracoviště\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "--kombinovaná-ne-FMV-Pr-9-8559-322"

Také jsou vytvořeny další 2 sloupce v odvozeném sloupci a o to:

* Počet odebraných bodů podle dodavatelů
* PracovištěID\_new

Výraz v "Počet odebraných bodů podle dodavatelů" provádí následující:

Výraz obsahuje několik vnořených příkazů iif, což jsou podmíněné výrazy, které vyhodnocují podmínku a vracejí hodnotu podle toho, zda je podmínka pravdivá nebo nepravdivá. Funkce and slouží ke kombinaci více podmínek.

Celková logika je následující:

*Příkaz* zkontroluje různé podmínky na základě hodnot polí {Typ jednotky výuky}, {Forma studia} a {Mimosemestrální výuka}.

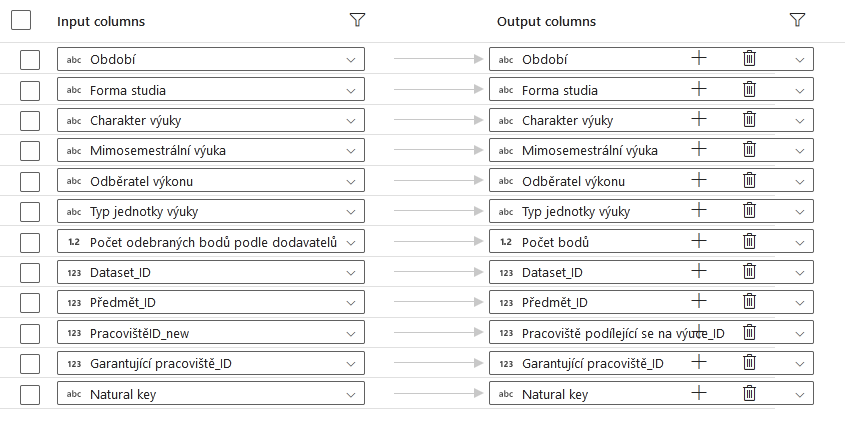
Pokud je podmínka splněna, vypočtěte hodnotu vynásobením {Počet bodů} s příslušným procentním polem (např. {Podíl - přednáška - prezenční}).

Pokud není splněna žádná z podmínek, vrátí se {Počet bodů} bez násobení.

Souhrnně řečeno, tento výraz vypočítá hodnotu na základě zadaných podmínek a zahrnuje vynásobení počtu bodů ({Počet bodů}) různými procentními poli na základě konkrétní kombinace hodnot v polích {Typ jednotky výuky}, {Forma studia} a {Mimosemestrální výuka}.

Výraz v "PracovištěID\_new" provádí nasledující:

Kontroluje, zda hodnota {Pracoviště podílející se na výuce\_ID} není nulová. Pokud není null, vrátí hodnotu {Pracoviště podílející se na výuce\_ID}. Pokud je však null, vrátí hodnotu {Garantující pracoviště\_ID}.

V operaci sink je třeba se podívat na mapování jednotlivých sloupců, protože některé sloupce nejsou použity z původního datového souboru nebo jsou namapovány na jiný název než původní. Pak exportujeme data do tabulky L1stage\_F\_pred\_prehled. 

Obrázek č.43 – Mapping v Sink aktivitěL1stage\_F\_pred\_prehled (Zdroj: Autoři)

**L1stage\_F\_prehledy\_total** má na vstupu 4 zdroje a to: L1stage\_F\_stat\_prehled, L1stage\_F\_ved\_p\_prehled, L1stage\_F\_ved\_p\_prehled, L1stage\_F\_pred\_prehled.

Všechny zdroje mají ve zdrojových možnostech vstupní dotaz místo tabulky. Dotaz v L1stage\_F\_stat\_prehled provádí následující:

Tento dotaz vybere konkrétní sloupce z tabulky [L1stage].[F\_stat\_prehled] a některým vybraným sloupcům přiřadí aliasy:

*[Období]*

*[Forma studia]*

*[Odběratel výkonu]*

*[Počet bodů] AS [Podle H fakulty] s přiděleným aliasem [Podle H fakulty]*

*[Počet bodů] AS [Podle H garant. katedry] s přiděleným aliasem [Podle H garant. katedry]*

*'Stat' AS [Kategorie výkonu]: Představuje kategorii výkonu, která je staticky nastavena jako "Stat".*

*[Dataset\_ID]*

*[Předmět\_ID]*

*[Pracoviště\_ID] AS [Pracoviště podílející se na výuce\_ID]: s přiděleným aliasem [Pracoviště podílející se na výuce\_ID]*

Dotaz v L1stage\_F\_ved\_p\_prehled provádí následující:

*Příkaz SELECT určuje sloupce, které se mají z tabulky načíst.*

*[Období]*

*[Forma studia]*

*[Odběratel výkonu]*

*[Počet bodů za vedení] je vybrán dvakrát a jsou mu přiřazeny aliasy [Podle H fakulty] a [Podle H garant. katedry].*

*Řetězec "Ved" je vybrán a je mu přiřazen alias [Kategorie výkonu].*

*[Dataset\_ID]*

*[Předmět\_ID]*

*Je vybrán řetězec [Pracoviště\_ID] a je mu přiřazen alias [Pracoviště podílející se na výuce\_ID].*

*Příkaz FROM určuje tabulku, ze které budou data získána: [L1stage].[F\_ved\_p\_prehled].*

Dotaz v opakujícím se zdroji L1stage\_F\_ved\_p\_prehled je téměř totožný. Jediné rozdíly jsou v tom, že místo 2x po sobě opakujícího se sloupce [Počet bodů za vedení] je v tomto zdroji vybrán sloupec [Počet bodů za obhajoby] a aliasován shodnými sloupci jako v předchozím zdroji a místo řetězce "Ved" je vybrán řetězec "Obh" a aliasován řetězcem [Kategorie výkonu].

*Dotaz v posledním zdroji L1stage\_F\_pred\_prehled je spojením dvou selectů, kde první select zahrnuje kromě počtu bodů podle H fakulty zároveň i přepočet bodů podle H garantující katedry. Tenhle přepočet se provádí pouze pro cvičeni, tj. typ jednotky výuky = cv. Zatímco druhy select pak vybírá řádky pro přednášky a nepřímou výuku kde počet bodů podle H fakulty = poctu bodů podle G katedry. Tento přepočet se děla mimo jiné i kvůli následujícímu použiti v dashboardu kde by mělo byt zajištěné přepínaní mezi počtem bodu podle fakulty a podle g katedry. Dotaz:*

*První příkaz SELECT, který obsahuje seznam sloupců, které se mají z dotazu načíst.*

*t1.[Období] načte sloupec "Období" z přiřazeného poddotazu t1.*

*[Forma studia]*

*[Mimosemestrální výuka]*

*[Odběratel výkonu]*

*[Typ jednotky výuky] AS [Kategorie výkonu] načte sloupec "Typ jednotky výuky" a přiřadí mu alias "[Kategorie výkonu]".*

*[Počet bodů] AS [Podle H fakulty] načte sloupec "Počet bodů" a přiřadí mu alias "[Podle H fakulty]".*

*[Počet bodů] \* t4.Hodnocení / t3.Hodnocení as [Podle H garant. katedry] vypočítá nový sloupec vynásobením "Počet bodů" hodnotou t4.Hodnocení a následným vydělením t3.Hodnocení. Výsledné hodnotě je přiřazen alias "[Podle H garant. katedry]".*

*t1.[Dataset\_ID] načte sloupec "Dataset\_ID" z aliasovaného poddotazu t1.*

*[Předmět\_ID]*

*t1.[Pracoviště podílející se na výuce\_ID] načte sloupec "Pracoviště podílející se na výuce\_ID" z přiřazeného poddotazu t1.*

*Příkaz FROM určuje poddotaz, který vybere všechny sloupce z tabulky [L1stage].[F\_pred\_prehled] a přiřadí jí alias t1.*

*Kód provádí operace LEFT JOIN s dalšími tabulkami.*

*První LEFT JOIN:*

*Tabulka 1: [L1stage].[F\_pred\_prehled] (t1)*

*Tabulka 2: [L1].[D\_Pracoviste] (t2)*

*Podmínka spojení:*

*t1.[Garantující pracoviště\_ID] = t2.Pracoviště\_ID*

*t1.[Dataset\_ID] = t2.[Dataset\_ID]*

*Druhý LEFT JOIN:*

*Tabulka 1: t2 (z předchozího spojení)*

*Tabulka 2: [L1stage].[F\_h\_cv] (t3)*

*Podmínka spojení:*

*t2.[Fakulta] = t3.[Fakulta].*

*t1.[Dataset\_ID] = t3.[Dataset\_ID]*

*t1.[Charakter výuky] = t3.[Charakter výuky]*

*Třetí LEFT JOIN:*

*Tabulka 1: t1 (z prvního spojení)*

*Tabulka 2: [L1stage].[F\_h\_cv\_g] (t4)*

*Podmínka spojení:*

*t1.Garantující pracoviště\_ID] = t4.[Garantující pracoviště\_ID].*

*t1.[Dataset\_ID] = t4.[Dataset\_ID]*

*t1.[Charakter výuky] = t4.[Charakter výuky*

*Příkaz WHERE filtruje výsledky výběrem řádků, kde se hodnota [Typ jednotky výuky] rovná "Cv".*

*Druhý příkaz select vybere sloupce z tabulky [L1stage].[F\_pred\_prehled]. Vyhledá řádky, kde hodnota [Typ jednotky výuky] není rovna 'Cv'.*

*Operátor UNION spojí výsledky prvního a druhého dotazu a vrátí jedinou množinu výsledků. Výsledek bude obsahovat všechny sloupce vybrané v obou dotazech.*



Obrázek č.44 – Source v aktivitě L1stage\_F\_prehledy\_total – zdroj (Zdroj: Autoři)

Následně za pomocí funkce Union spojíme všechny 4 datové toky do jednoho.

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojí více hodnot z různých sloupců a výrazů do jednoho řetězce. Výsledný řetězec se skládá z těchto zřetězených částí oddělených znaky pomlčky (-). Funkce trim se použije k odstranění mezer z výsledného řetězce.

Přirozený klíč může pak vypadat následovně: "prezencní--FFU-8-6716-238-Ved"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_prehledy\_total.

**L1stage\_F\_stat\_dod** má na vstupu zdroj L0\_stat\_dod. L0\_stat\_dod má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají z dotazu získat:*

*t1.[Období] načte sloupec "Období" z přiřazeného poddotazu t1.*

*t2.[Dataset\_ID] načte sloupec "Dataset\_ID" z přiřazené tabulky t2.*

*[Typ studia]*

*[Počet bodů]*

*t3. [Pracoviště\_ID] načte sloupec "Pracoviště\_ID" z přiřazené tabulky t3.*

*Příkaz FROM začíná poddotazem, který vybere všechny sloupce z tabulky [L0].[stat\_dod] a přiřadí jí alias t1.*

*První LEFT JOIN se provede mezi t1 a [L1].[D\_Dataset] (t2). Podle podmínky:*

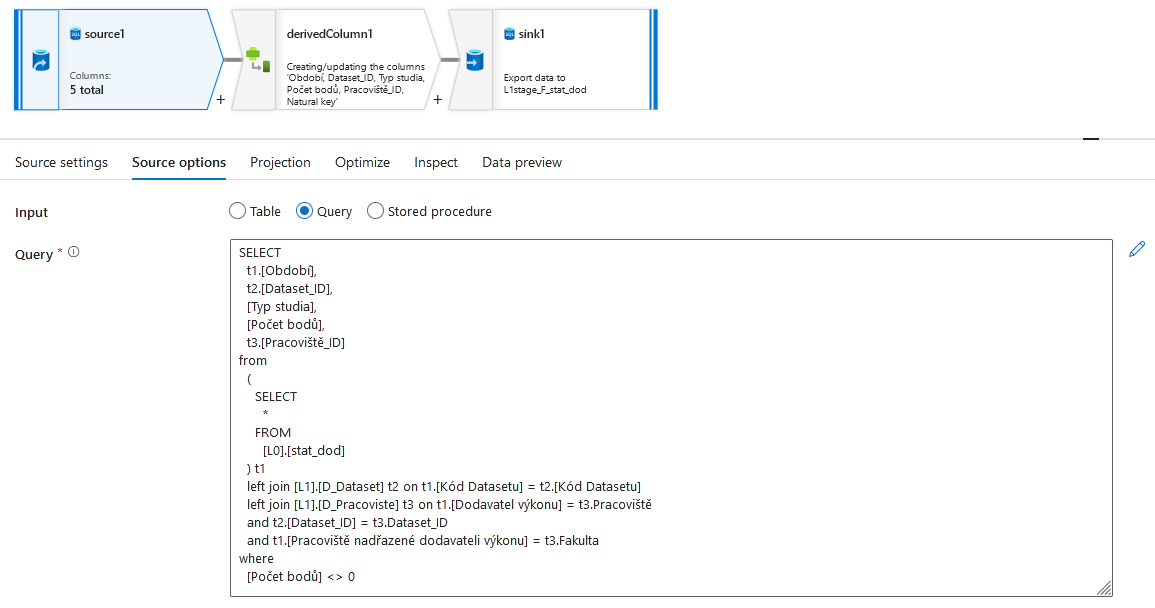
*t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Druhý LEFT JOIN se provede mezi t1 a [L1].[D\_Pracoviste] (t3). Podle podmínky:*

*t1.[Dodavatel výkonu] = t3.Pracoviště*

*t2.[Dataset\_ID] = t3.Dataset\_ID*

*t1.[Pracoviště nadřazené dodavateli výkonu] = t3.Fakulta*

*Příkaz WHERE filtruje výsledky výběrem řádků, kde hodnota [Počet bodů] není rovna 0.* 

Obrázek č.45 – Source options v aktivite L1stage\_F\_stat\_dod(Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Typ studia", řetězcovou reprezentaci "Pracoviště\_ID" a "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "B-232-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_stat\_dod**.**

**L1stage\_F\_stat\_od** má na vstupu zdroj L0\_stat\_od. L0\_stat\_od má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají z dotazu získat:*

*t1.[Období]*

*[Dataset\_ID]*

*[Odběratel výkonu]*

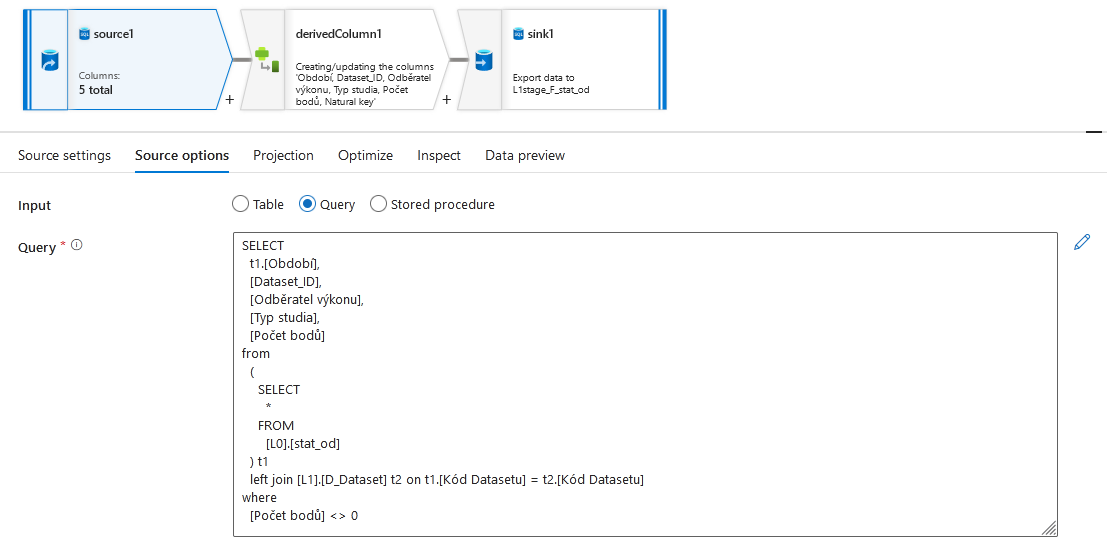
*[Typ studia]*

*[Počet bodů]*

*Příkaz FROM začíná poddotazem, který vybere všechny sloupce z tabulky [L0].[stat\_od] a přiřadí jí alias t1.*

*Kód provede operaci LEFT JOIN s tabulkou [L1].[D\_Dataset] pomocí podmínky t1.[Kód datasetu] = t2.[Kód datasetu].*

*Příkaz WHERE filtruje výsledky výběrem řádků, kde hodnota [Počet bodů] není rovna 0.*



Obrázek č.46 – Source options v aktivitě L1stage\_F\_stat\_od(Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Odběratel výkonu", "Typ studia" a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "FFU-B-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_stat\_od**.**

**L1stage\_F\_stat\_prehled** má na vstupu zdroj L0\_stat\_prehled. L0\_stat\_prehled má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT slouží k určení sloupců, které mají být zahrnuty do sady výsledků:*

*t1.[Období]*

*t2.[Dataset\_ID]*

*[Typ studia]*

*[Forma studia]*

*[Odběratel výkonu]*

*[Podíl - nepřímá výuka - prezenční] \* [Počet pokusů] (vypočtený sloupec s jménem [Počet bodů])*

*t3.[Předmět\_ID]*

*Pracoviště\_ID*

*Příkaz FROM určuje zdrojové tabulky pro dotaz. V tomto případě se použije poddotaz pro výběr všech sloupců z tabulky [L0].[stat\_prehled] a výsledek se označí aliasem t1.*

*Dotaz používá několik operací LEFT JOIN ke spojení dalších tabulek:*

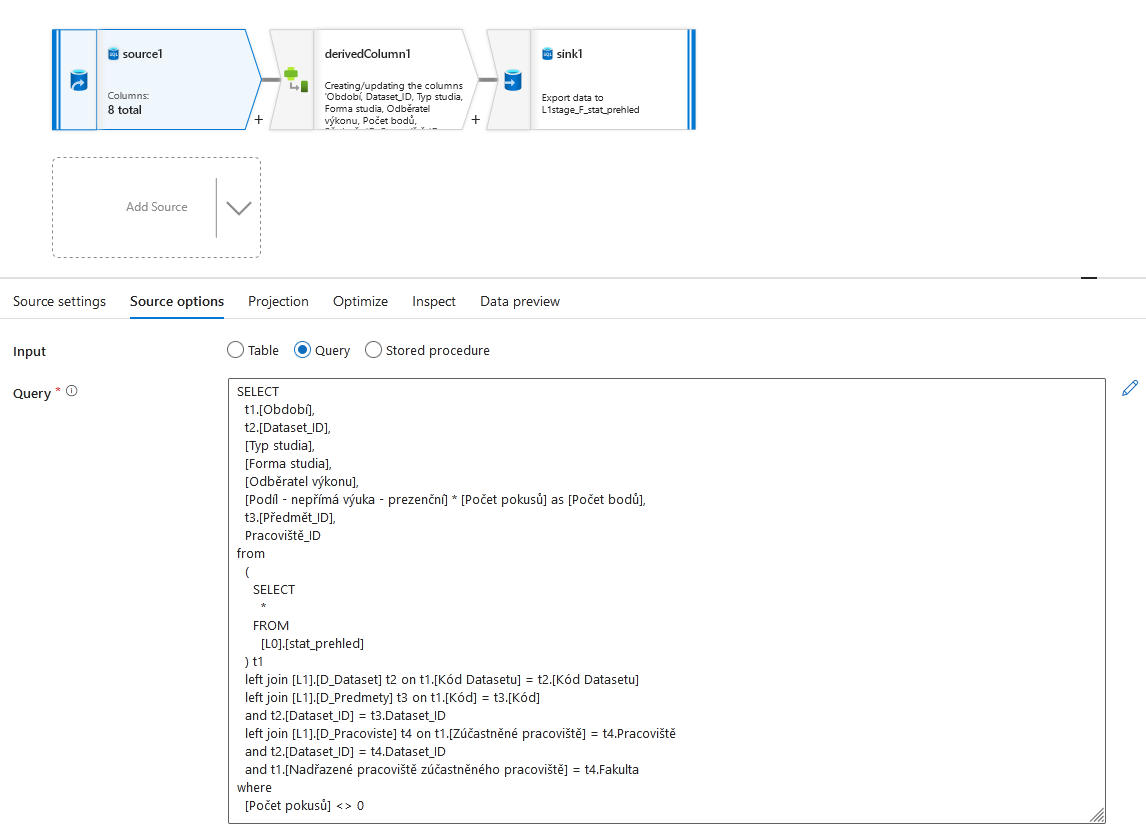
*LEFT JOIN [L1].[D\_Dataset] t2 ON t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*LEFT JOIN [L1].[D\_Predmety] t3 ON t1.[Kód] = t3.[Kód] AND t2.[Dataset\_ID] = t3.Dataset\_ID*

*LEFT JOIN [L1].[D\_Pracoviste] t4 ON t1.[Zúčastněné pracoviště] = t4.Pracoviště AND t2.[Dataset\_ID] = t4.Dataset\_ID AND t1.[Nadřazené pracoviště zúčastněného pracoviště] = t4.Fakulta*

*Tato spojení se provádějí na základě konkrétních podmínek sloupců mezi tabulkami.*

*Příkaz WHERE filtruje výsledky na základě podmínky [Počet pokusů] <> 0. Tato podmínka zajišťuje, že do výsledné množiny budou zahrnuty pouze řádky, kde hodnota sloupce [Počet pokusů] není rovna nule.*



Obrázek č.47 – Source options v aktivitě L1stage\_F\_stat\_prehled (Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Typ studia", "Forma studia", "Odběratel výkonu" a řetězcovou reprezentaci "Předmět\_ID", "Pracoviště\_ID", "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "N-prezencní-N-IB (FMV)-8427-293-9"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_stat\_prehled**.**

**L1stage\_F\_ved\_p\_dod** má na vstupu zdroj L0\_ved\_p\_dod. L0\_ved\_p\_dod má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT slouží k určení sloupců, které mají být zahrnuty do sady výsledků:*

*t1.[Období]*

*t2.[Dataset\_ID]*

*[Typ studia]*

*[Počet bodů]*

*t3.[Pracoviště\_ID]*

*Příkaz FROM určuje zdrojovou tabulku dotazu. V tomto případě se použije poddotaz pro výběr všech sloupců z tabulky [L0].[ved\_p\_dod] a výsledek se aliasuje jako t1.*

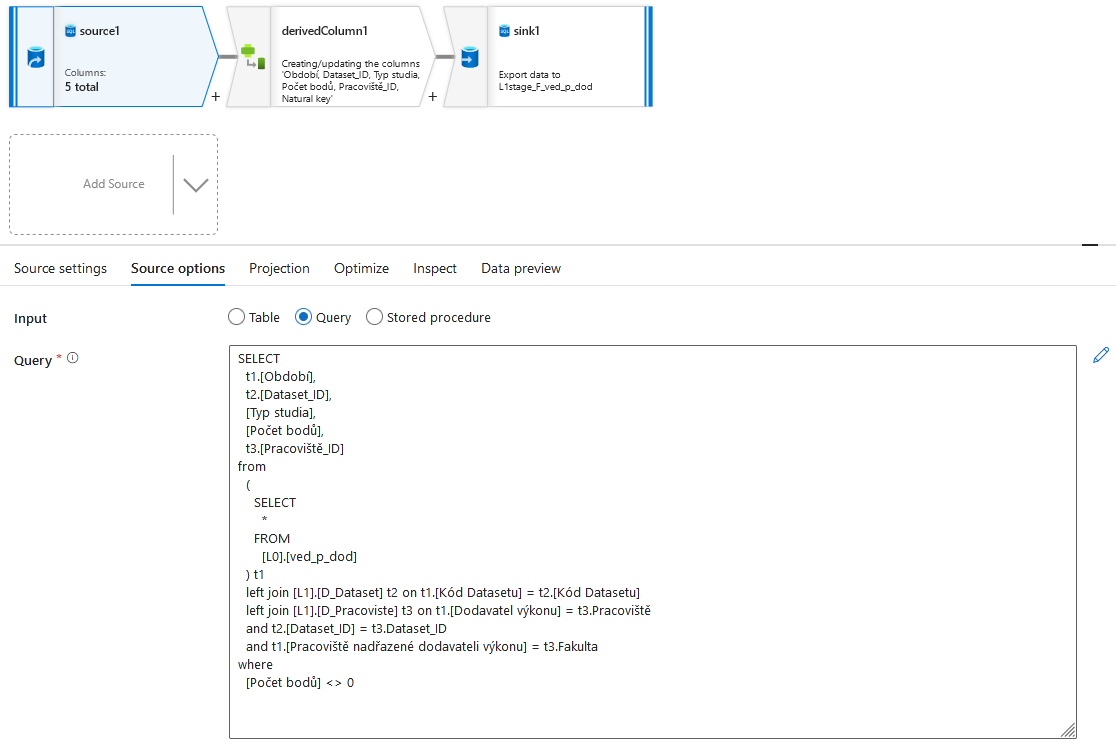
*Dotaz používá dvě operace LEFT JOIN ke spojení dalších tabulek:*

*LEFT JOIN [L1].[D\_Dataset] t2 ON t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*LEFT JOIN [L1].[D\_Pracoviste] t3 ON t1.[Dodavatel výkonu] = t3.Pracoviště AND t2.[Dataset\_ID] = t3.Dataset\_ID AND t1.[Pracoviště nadřazené dodavateli výkonu] = t3.Fakulta*

*Tato spojení se provádějí na základě konkrétních podmínek sloupců mezi tabulkami.*

*Příkaz WHERE filtruje výsledky na základě podmínky [Počet bodů] <> 0. Tato podmínka zajišťuje, že do množiny výsledků budou zahrnuty pouze řádky, u nichž hodnota sloupce [Počet bodů] není rovna nule.*



Obrázek č.48 – Source options v aktivitě L1stage\_F\_ved\_p\_dod(Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Typ studia", a řetězcovou reprezentaci "Pracoviště\_ID" a "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "B-232-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_ved\_p\_dod.

**L1stage\_F\_ved\_p\_od** má na vstupu zdroj L0\_ved\_p\_od. L0\_ved\_p\_od má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT slouží k určení sloupců, které mají být zahrnuty do sady výsledků:*

*t1.[Období]*

*[Dataset\_ID]*

*[Odběratel výkonu]*

*[Typ studia]*

*[Počet bodů]*

*Příkaz FROM určuje zdrojovou tabulku dotazu. V tomto případě se použije poddotaz pro výběr všech sloupců z tabulky [L0].[ved\_p\_od] a výsledek se aliasuje jako t1.*

*Dotaz používá operaci LEFT JOIN pro připojení k jiné tabulce:*

*LEFT JOIN [L1].[D\_Dataset] t2 ON t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Toto spojení se provádí na základě podmínky, že sloupec t1.[Kód Datasetu] odpovídá sloupci t2.[Kód Datasetu].*

*Příkaz WHERE filtruje výsledky na základě podmínky [Počet bodů] <> 0. Tato podmínka zajišťuje, že do množiny výsledků budou zahrnuty pouze řádky, kde hodnota sloupce [Počet bodů] není rovna nule.*



Obrázek č.49 – Source options v aktivitě L1stage\_F\_ved\_p\_od(Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Odběratel výkonu", "Typ studia", a řetězcovou reprezentaci "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "FFU-B-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_ved\_p\_od.

**L1stage\_F\_ved\_p\_prehled** má na vstupu zdroj L0\_ved\_p\_prehled. L0\_ved\_p\_prehled má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají načíst:*

*t1.[Období]*

*t2.[Dataset\_ID]*

*[Typ studia]*

*[Forma studia]*

*[Odběratel výkonu]*

*[Počet bodů za vedení] - kód vypočítá hodnotu [Počet bodů za vedení] na základě hodnoty [Typ studia]. Pokud je [Typ studia] - "B", hodnota [Počet pokusů] se vynásobí 6(5 bodů za vedení bakalářské práce plus 1 bod za posudek*)*. V opačném případě, pokud [Typ studia] není "B", tj. jedná se o diplomovou práci, hodnota [Počet pokusů] se vynásobí 11 (10 bodů za vedení diplomové práce plus 1 bod za posudek).*

*[Počet pokusů] jako [Počet bodů za obhajoby]*

*t3. [Předmět\_ID]*

*t4. [Pracoviště\_ID]*

*Příkaz FROM určuje tabulky použité v dotazu. Poddotaz (SELECT \* FROM [L0].[ved\_p\_prehled]) t1 vybere všechny sloupce z tabulky [L0].[ved\_p\_prehled] a označí ji jako t1.*

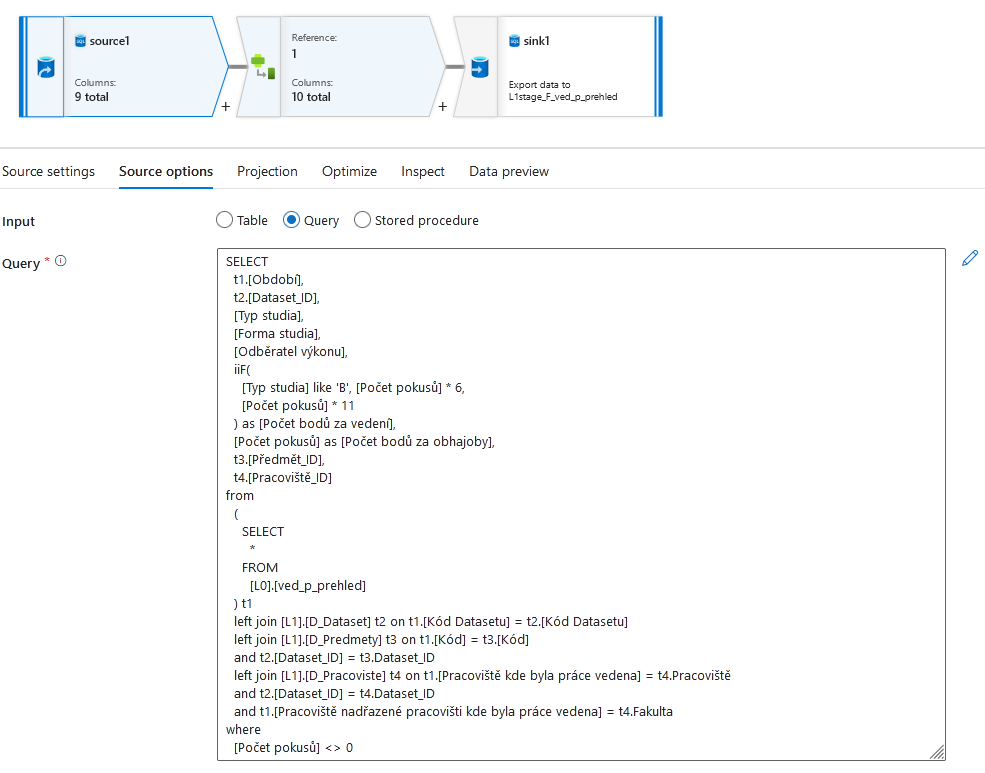
Dotaz provádí left join na více tabulek:

*Left join [L1].[D\_Dataset] t2 je založeno na podmínce t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Left join s [L1].[D\_Predmety] t3 je založeno na podmínkách t1.[Kód] = t3.[Kód] a t2.[Dataset\_ID] = t3.Dataset\_ID.*

*Left join s [L1].[D\_Pracoviste] t4 je založeno na podmínkách t1.[Pracoviště kde byla práce vedena] = t4.Pracoviště, t2.[Dataset\_ID] = t4.Dataset\_ID a t1.[Pracoviště nadřazené pracovišti kde byla práce vedena] = t4.Fakulta.*

*Příkaz WHERE filtruje řádky na základě podmínky [Počet pokusů] <> 0. Vybírá řádky, kde hodnota [Počet pokusů] není rovna 0.*



Obrázek č.50 – Source options v aktivitě L1stage\_F\_ved\_p\_prehled(Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Typ studia", "Forma studia", "Odběratel výkonu" a řetězcovou reprezentaci "Předmět\_ID", "Pracoviště\_ID" a "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "B-prezencní-FFU-6716-238-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_ved\_p\_prehled.

**L1stage\_F\_vy\_dod\_pred** má na vstupu zdroj L0\_vy\_dod\_pred. L0\_vy\_dod\_pred má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají načíst:*

*t1.[Období]*

*t2.[Dataset\_ID]*

*COALESCE([Charakter výuky], '-') as [Charakter výuky]*

*[Forma studia]*

*[Mimosemestrální výuka]*

*[Typ jednotky výuky]*

*SUM([Počet bodů]) AS [Počet bodů]*

*t3.[Předmět\_ID]*

*t4.[Pracoviště\_ID]*

*Příkaz FROM určuje tabulky použité v dotazu. Poddotaz (SELECT \* FROM [L0].[vy\_dod\_pred]) t1 vybere všechny sloupce z tabulky [L0].[vy\_dod\_pred] a označí ji jako t1.*

*Dotaz provádí spojení vlevo na více tabulek:*

*Left join s [L1].[D\_dataset] t2 je založeno na podmínce t1.[Kód datasetu] = t2.[Kód datasetu].*

*Left join s [L1].[D\_Predmety] t3 je založeno na podmínkách t1.[Kód] = t3.[Kód] a t2.[Dataset\_ID] = t3.Dataset\_ID.*

*Left join s [L1].[D\_Pracoviste] t4 je založeno na podmínkách t1.[Pracoviště podílející se na výuce] = t4.Pracoviště, t2.[Dataset\_ID] = t4.Dataset\_ID a t1.[Pracoviště nadřazené pracovišti podílejícímu se na výuce] = t4.Fakulta.*

*Příkaz WHERE filtruje řádky na základě podmínky [Počet bodů] <> 0. Vybírá řádky, kde hodnota [Počet bodů] není rovna 0.*

*Příkaz GROUP BY seskupuje množinu výsledků podle následujících sloupců:*

*t1.[Období]*

*t2.[Dataset\_ID]*

*t1.[Charakter výuky]*

*t1.[Forma studia]*

*t1.[Mimosemestrální výuka]*

*t1.[Typ jednotky výuky]*

*t3.[Předmět\_ID]*

*t4.[Pracoviště\_ID]*

*Souhrnně řečeno, kód SQL načítá konkrétní sloupce z více tabulek pomocí left join Jeho součástí je agregace pomocí funkce SUM pro výpočet součtu sloupce [Počet bodů]. Funkce COALESCE se používá k nahrazení hodnot NULL ve sloupci [Charakter výuky] pomlčkou ("-"). Výsledkem je seskupený a agregovaný soubor dat na základě zadaných sloupců.*

Obrázek č.51 – Source options v aktivitě L1stage\_F\_vy\_dod\_pred(Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Charakter výuky", "Forma studia", "Mimosemestrální výuka", "Typ jednotky výuky" a řetězcovou reprezentaci "Předmět\_ID", "Pracoviště\_ID" a "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "--distancní-ne-Cv-7494-243-8"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_vy\_dod\_pred.

**L1stage\_F\_vy\_od\_pred** má na vstupu zdroj L0\_vy\_od\_pred. L0\_vy\_od\_pred má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají načíst:*

*t1.[Období]*

*t2.[Dataset\_ID]*

*COALESCE([Charakter výuky], '-') as [Charakter výuky]*

*[Forma studia]*

*[Mimosemestrální výuka]*

*[Odběratel výkonu]*

*[Typ jednotky výuky]*

*SUM(t1.[Počet bodů]) AS [Počet bodů]*

*t3.[Předmět\_ID]*

*t3.[Garantující pracoviště\_ID]*

*Příkaz FROM určuje tabulky použité v dotazu. Poddotaz (SELECT \* FROM [L0].[vy\_od\_pred]) t1 vybere všechny sloupce z tabulky [L0].[vy\_od\_pred] a označí ji jako t1.*

*Dotaz provádí left join na více tabulek:*

*[L1].[D\_Dataset] t2 je založeno na podmínce t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Left join s [L1].[D\_Predmety] t3 je založeno na podmínkách t1.[Kód] = t3.[Kód] a t2.[Dataset\_ID] = t3.Dataset\_ID.*

*Příkaz WHERE filtruje řádky na základě podmínky [Počet bodů] <> 0. Vybírá řádky, u nichž hodnota [Počet bodů] není rovna 0.*

*Příkaz GROUP BY seskupuje množinu výsledků podle následujících sloupců:*

*t1.[Období]*

*t2.[Dataset\_ID]*

*t1.[Charakter výuky]*

*t1.[Forma studia]*

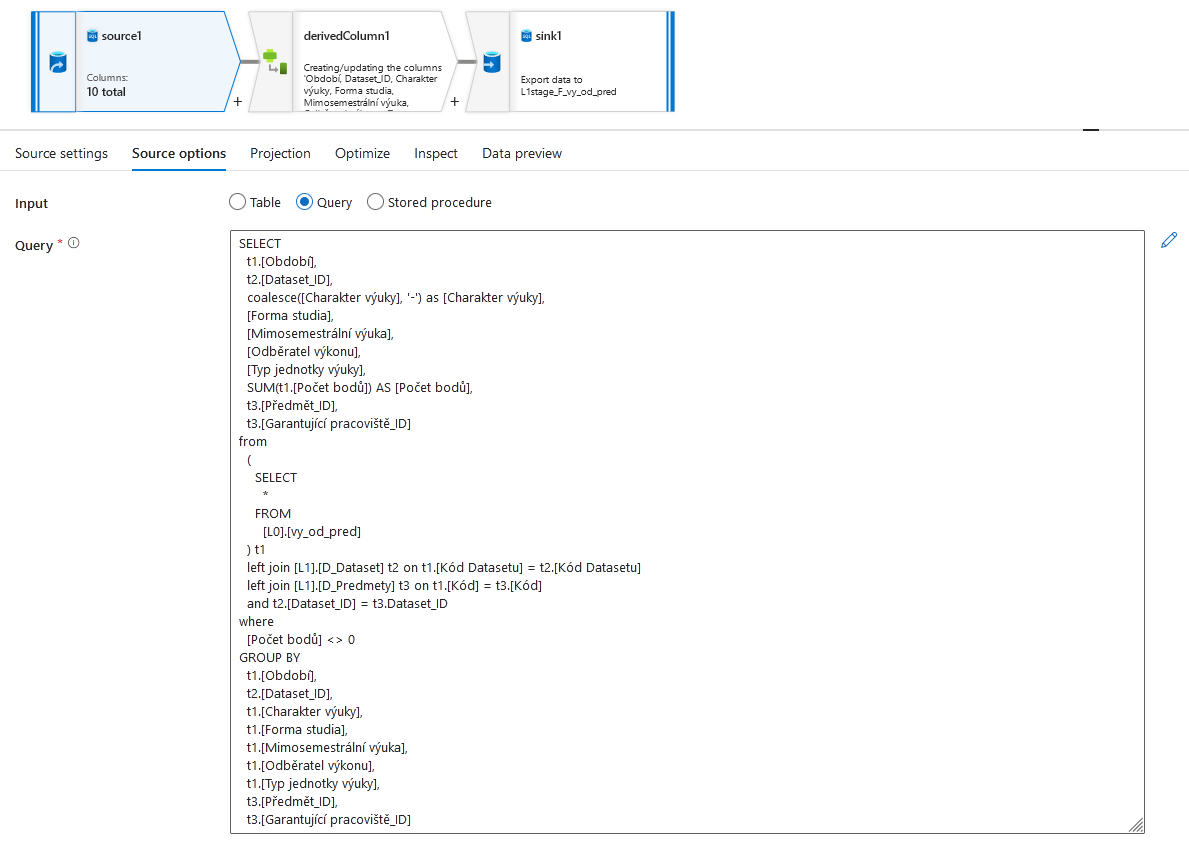
*t1.[Mimosemestrální výuka]*

*t1.[Odběratel výkonu]*

*t1.[Typ jednotky výuky]*

*t3.[Předmět\_ID]*

*t3.[Garantující pracoviště\_ID]*

*SQL načítá konkrétní sloupce z více tabulek pomocí levého spojení. Zahrnuje agregaci pomocí funkce SUM pro výpočet součtu sloupce [Počet bodů]. Funkce COALESCE se používá k nahrazení hodnot NULL ve sloupci [Charakter výuky] pomlčkou ("-"). Výsledkem je seskupený a agregovaný soubor dat na základě zadaných sloupců.*

Obrázek č.52 – Source options v aktivitě L1stage\_F\_vy\_od\_pred(Zdroj: Autoři)

Pak se v odvozeném sloupci vytvoří přirozený klíč, který spojuje hodnoty "Charakter výuky", "Forma studia", "Mimosemestrální výuka", "Odběratel výkonu", "Typ jednotky výuky" a řetězcovou reprezentaci "Předmět\_ID", "Garantující pracoviště\_ID" a "Dataset\_ID" dohromady.

Přirozený klíč může pak vypadat následovně: "--kombinovaná-ne-FMV-Cv-8562-312-9"

V operaci sink pak exportujeme data do tabulky L1stage\_F\_vy\_od\_pred.

**L1stage\_F\_vy\_hlv** má na vstupu zdroj L0\_vy\_hlv. L0\_vy\_hlv má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají načíst:*

*t1.[Období]*

*t2.[Dataset\_ID]*

*[ID]*

*[Akce]*

*[Studentů]*

*t3.[Předmět\_ID]*

*[Unikátní ID]*

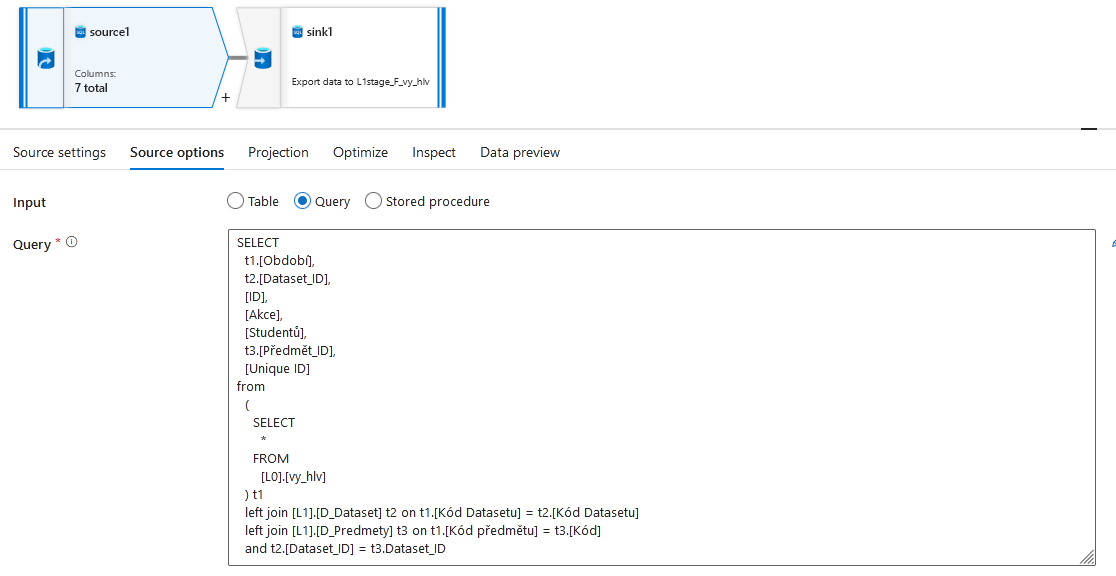
*Příkaz FROM určuje tabulky použité v dotazu. Poddotaz (SELECT \* FROM [L0].[vy\_hlv]) t1 vybere všechny sloupce z tabulky [L0].[vy\_hlv] a označí ji jako t1.*

*Dotaz provádí spojení vlevo na dvě tabulky:*

*Left join s [L1].[D\_Dataset] t2 je založeno na podmínce t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Left join s [L1].[D\_Predmety] t3 je založeno na podmínkách t1.[Kód predmětu] = t3.[Kód] a t2.[Dataset\_ID] = t3.Dataset\_ID.*

*Výsledkem je kombinace sloupců z těchto tabulek, u nichž jsou splněny podmínky spojení.*



Obrázek č.53 – Source options v aktivitě L1stage\_F\_vy\_hlv (Zdroj: Autoři)

V operaci sink pak exportujeme data do tabulky L1stage\_F\_vy\_hlv.

**L0\_vy\_spoj** má na vstupu zdroj L0\_vy\_spoj. L0\_vy\_spoj má ve zdrojových možnostech vstupní dotaz místo tabulky. Tenhle dotaz provádí následující:

*Příkaz SELECT určuje sloupce, které se mají načíst:*

*t1.[Období]*

*t2.[Dataset\_ID]*

*[ID]*

*[Akce]*

*[Studentů]*

*t3.[Předmět\_ID]*

*[Unikátní ID]*

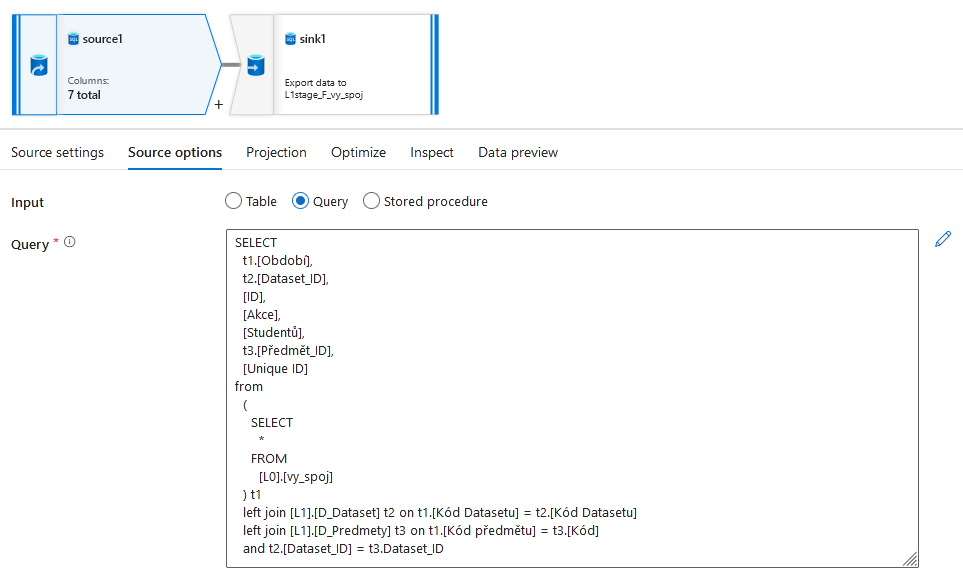
*Příkaz FROM určuje tabulky použité v dotazu. Poddotaz (SELECT \* FROM [L0].[vy\_spoj]) t1 vybere všechny sloupce z tabulky [L0].[vy\_spoj] a označí ji jako t1.*

*Dotaz provádí left join na dvě tabulky:*

*Left join s [L1].[D\_Dataset] t2 je založeno na podmínce t1.[Kód Datasetu] = t2.[Kód Datasetu].*

*Left join s [L1].[D\_Predmety] t3 je založeno na podmínkách t1.[Kód predmětu] = t3.[Kód] a t2.[Dataset\_ID] = t3.Dataset\_ID.*

*Výsledkem je kombinace sloupců z těchto tabulek, u nichž jsou splněny podmínky spojení.*

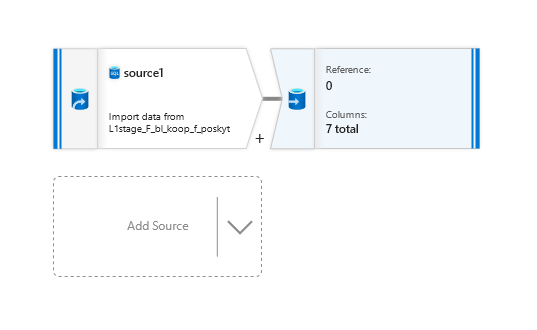


Obrázek č.54 – Source options v L0\_vy\_spoj(Zdroj: Autoři)

V operaci sink pak exportujeme data do tabulky L1stage\_F\_vy\_spoj.

### Vrstva L1

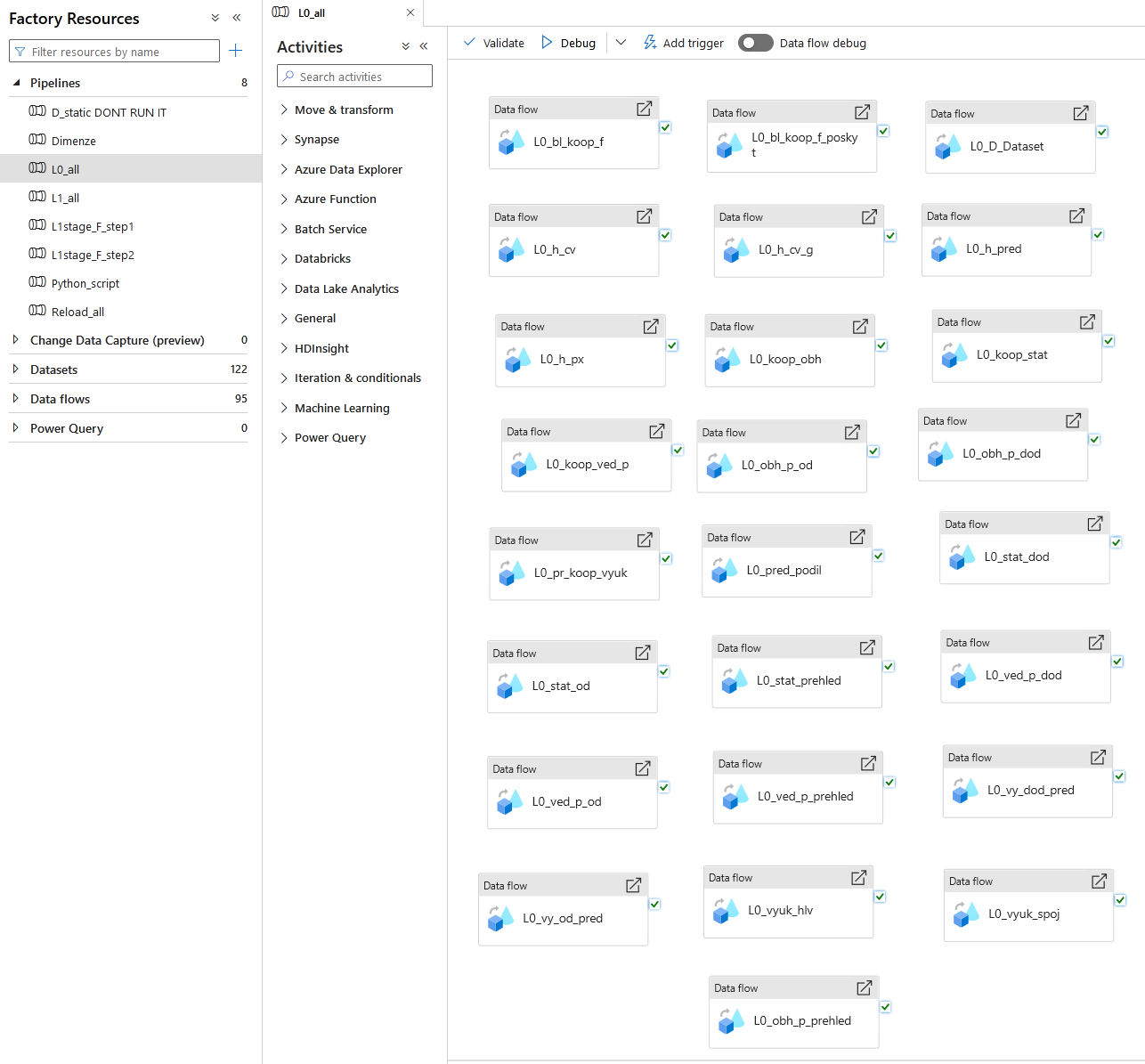
V činnostech L1 se již neprovádějí žádné transformace. Jediné, co se v aktivitách L1 děje, je kopírování dat 1:1 z tabulek L1stage do tabulek L1. Konkrétně je do L1 exportováno 35 tabulek z L1stage. Tyto tabulky jsou uvedeny v příloze B – Tabulky.

Obrázek č.55 – Export dat z L1stage\_F\_bl\_koop\_f\_poskyt do L1\_F\_bl\_koop\_f\_poskyt tabulek(Zdroj: Autoři)

Také byly vytvořeny v datovém skladu relace mezi tabulkami L1. Diagramy jednotlivých relací jsou v příloze: *Příloha C – Schémata L1*nebo kódy SQL v příloze: *Příloha A – SQL kódy*.

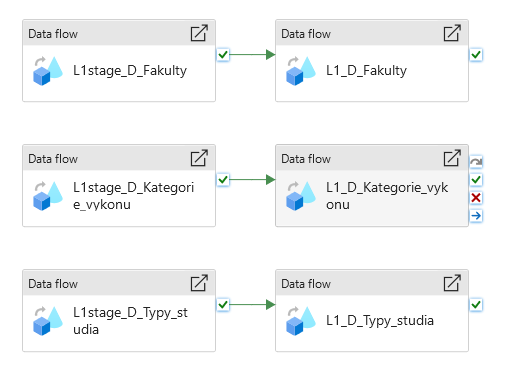
### Pipeline – datové pumpy

Pumpa *L0\_all* nahraje data do tabulek vrstvy L0. V podstatě nakopíruje data 1:1 tak jak jsou ve *vystupni-excel-data* až na výjimku L0\_vyuk\_hlv a L0\_vyuk\_spoj ve kterých se ještě vytvářejí odvozené sloupce.

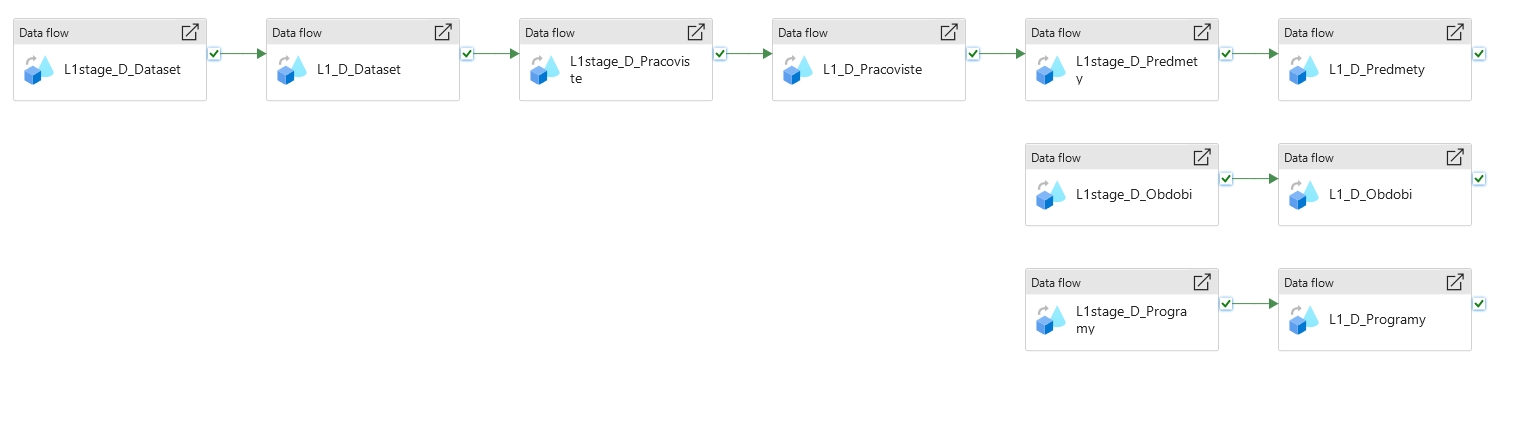


Obrázek č.56 – Datová pumpa *L0\_all*(Zdroj: Autoři)

Další datová pumpa je pumpa, která nahrává tabulky dimenzí. Tyto pumpy jsou dvě: jedna načítá tabulky dimenzí static (static v tomto případě znamenají, že se jedná o dimenze, ve kterých se data téměř vůbec nebo alespoň minimálně nemění. Na rozdíl od dimenzí nazývaných dynamic.



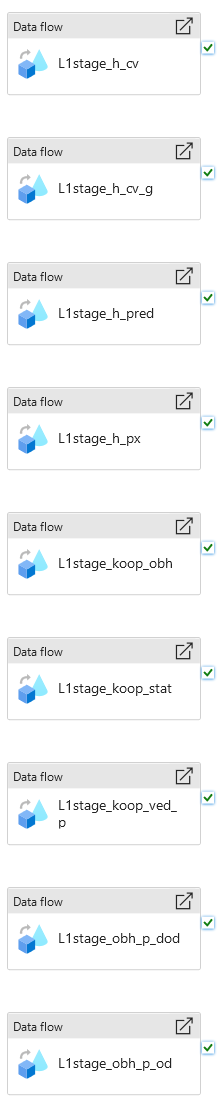
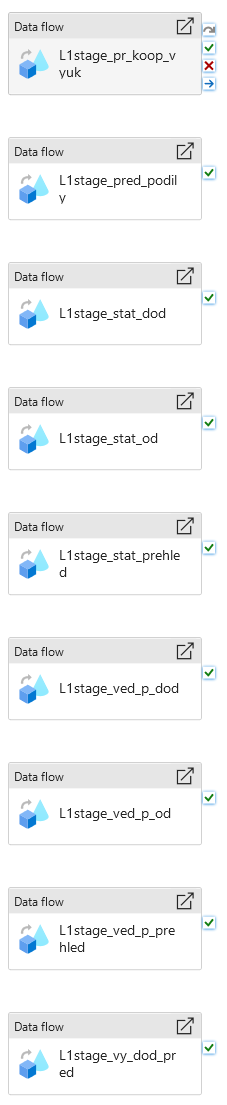
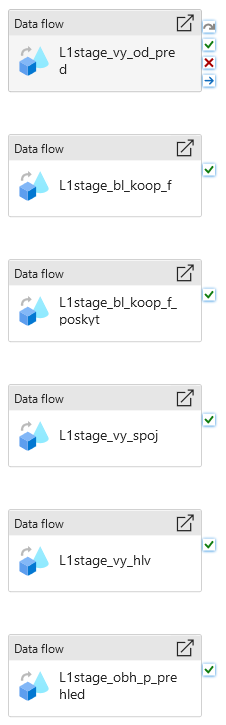
Obrázek č.57 – Datová pumpa *D\_static*(Zdroj: Autoři)

Obrázek č.58 – Datová pumpa *Dimenze*(Zdroj: Autoři)

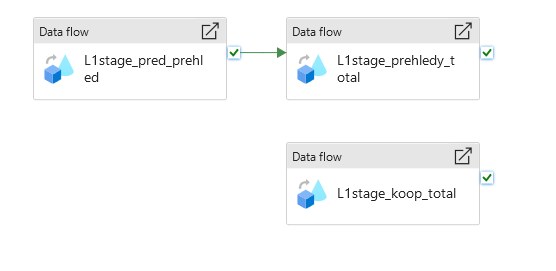
Všechny aktivity jsou spojené vazbou Úspěch, tzn., že se další aktivita spustí až po úspěchu předešlé aktivity. Například když selže aktivita *L1stage\_D\_Dataset*, aktivita *L1\_D\_Dataset* se nespustí.

Následují datové pumpy pro tabulky faktů, které jsou pro přehlednost rozděleny na 2 pumpy a to: *L1stage\_F\_step1* a *L1stage\_F\_step2.*

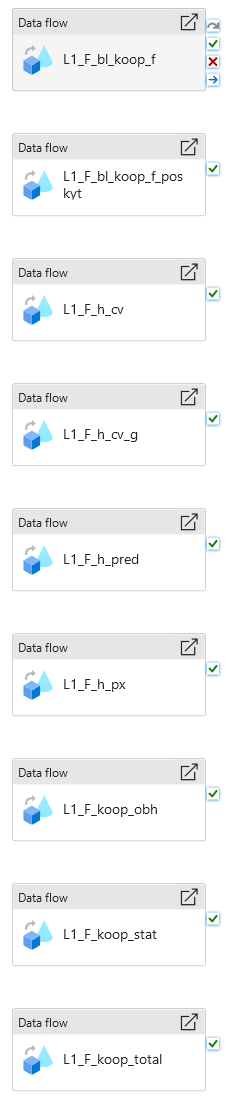
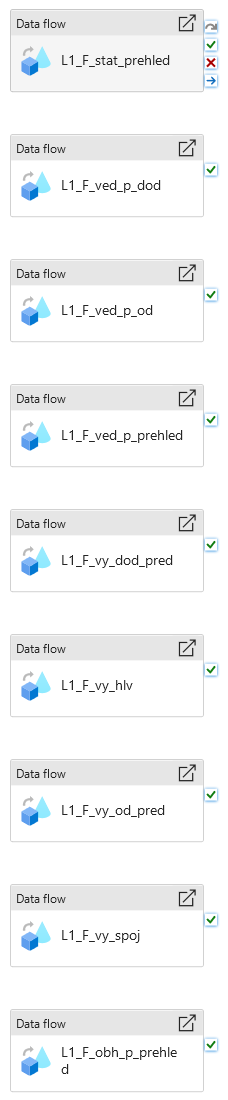
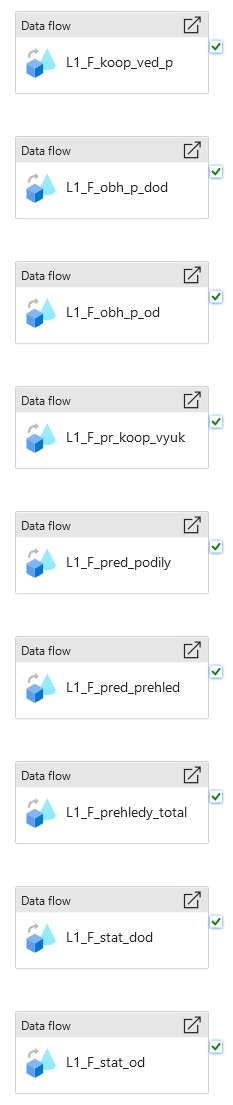
Je to rozdělené, protože tabulky v *L1stage\_F\_step2* jsou aktualizovány odvozením dat z jiných tabulek L1stage. A když selže nahrávaní dat ze zdrojových souborů již jsou ve *L1stage\_F\_step1* pak by aktualizace tabulek v L1stage\_F\_step2 neměla proběhnout.



Obrázek č.59 – Datová pumpa *L1stage\_F\_step1*(Zdroj: Autoři)

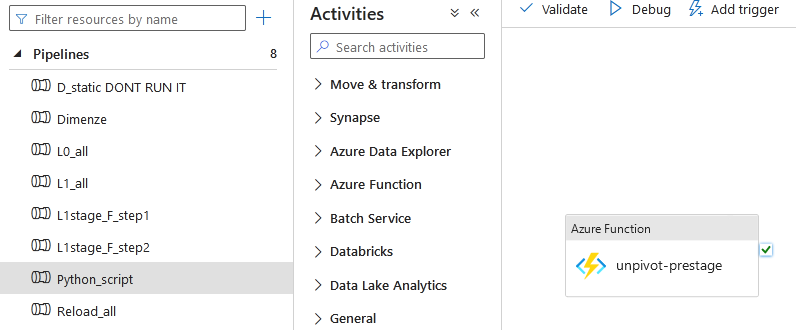


Obrázek č.60 – Datová pumpa *L1stage\_F\_step2*(Zdroj: Autoři)

Pumpa *L1\_all* naplňuje tabulky faktů ve vrstvě L1.

Obrázek č.61 – Datová pumpa *L1\_all* (Zdroj: Autoři)

Pumpa *Python\_script* pošle get request na API, kterého proces je popisován v kapitole 8.



Obrázek č.62 – Datová pumpa (Azure function) - Python *script*(Zdroj: Autoři)

Pipeline *Reload\_all* pak zapne všechna výše popsaná datové pumpy v logickém sledu. A všechny pumpy jsou spojené vazbou Úspěch, abych se další pumpa začala až po úspěchu předešlé pumpy.



Obrázek č.63 – Datová pumpa *Reload\_all*(Zdroj: Autoři)

## Manuál po nahrávaní dat do Azure a spuštění Pipeline

Pro nahrávání dat do azure a spouštění pipeline v azure data factory jsme vytvořili jednoduchý a přehledný návod, který podrobně vysvětluje jednotlivé kroky. Tenhle návod naleznete v příloze *D – Návod po nahrávaní dat do DWH.*

Přílohy

Příloha A

*Příloha A – SQL kódy*. Obsahuje všechny SQL skripty pro vytvoření tabulek v databázi.

Příloha B

*Příloha B – Tabulky*. Obsahuje všechny tabulky L0 a L1stage, jejich klíče a datové typy.

Příloha C

*Příloha C – Schémata L1*. Obsahuje schémata cizích a primárních klíčů mezi tabulkami L1.

Příloha D

*Příloha D – Návod po nahrávaní dat do DWH*. Obsahuje podobný manuál po nahraní dat do prostředí Azure a spuštěni pipeline.

Příloha E

*Příloha E – Views\_SQL kódy*. Obsahuje všechny SQL skripty pro vytvoření views v databázi.