Nová generace datového skladu reklamního systému Sklik.cz

Bc. Miroslav Kvasnica

Diplomová práce
2017

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2016/2017

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Miroslav Kvasnica**
Osobní číslo:	**A13491**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**kombinovaná**
Téma práce:	**Nová generace datového skladu reklamního systému Sklik.cz**
Téma anglicky:	**New-Generation Data Warehouses for the Sklik.cz Advertising System**

Zásady pro vypracování:

1. **Proveďte literární rešerši tématu datových skladů.**
2. **Analyzujte současný stav datového skladu systému Sklik.cz a definujte jeho silné a slabé stránky.**
3. **Navrhněte způsob řešení jeho aktualizace a zdůvodněte ji z pohledu zvolené technologie, její funkčnosti i očekávaného přínosu.**
4. **Realizujte zvolené řešení na vhodném modelu prototypu a tento důkladně otestujte.**
5. **Proveďte migraci systému a zhodnoťte jeho přínosy pro zadavatele.**

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **GEORGE, Lars. HBase: The Definitive Guide. 1. vyd. O'Reilly Media, 2011. 556 s. ISBN 978-1-4493-9610-7**
2. **HEWITT, Eben. Cassandra: The Definitive Guide. 1. vyd. O'Reilly Media, 2010. 332 s. ISBN 978-1-4493-9041-9**
3. **PACHEV, Sasha. Understanding MySQL Internals. 1. vyd. O'Reilly Media, 2007. 258 s. ISBN 978-0-596-00957-1**
4. **RUSSELL, John. Getting Started with Impala. 1. vyd. O'Reilly Media, 2014. 110 s. ISBN 978-1-4919-0577-7**
5. **SCHWARTZ, Baron, Peter ZAITSEV a Vadim TKACHENKO. High Performance MySQL. 3. vyd. O'Reilly Media, 2012. 826 s. ISBN 978-1-4493-1428-6**
6. **STEPHENS, Rod. Beginning Database Design Solutions. 1. vyd. Wrox, 2009. 552 s. ISBN 978-1-4571-0413-8**

Vedoucí diplomové práce: **prof. Ing. František Schauer, DrSc.**
Ústav elektroniky a měření

Datum zadání diplomové práce: **3. února 2017**

Termín odevzdání diplomové práce: **16. května 2017**

Ve Zlíně dne 3. února 2017

doc. Mgr. Milan Adámek, Ph.D.
děkan

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
L.S.

prof. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomové práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně 15.5.2017 .....

podpis autora

ABSTRAKT

Tato diplomová práce se zabývá datovými sklady pro uložení statistických dat reklamního systému Sklik.cz. Popsány jsou technologie MySQL, Hadoop, Apache Impala, Apache Hive, Druid a další. Z nich byla jako nejvhodnější řešení vybrána Apache Impala a je proto rozebrána podrobněji včetně praktických postupů pro migraci dat a její provoz.

Klíčová slova: Datový sklad, OLAP, MySQL, partitioning, Hadoop, Apache Impala, Apache Parquet, Apache Hive

ABSTRACT

This thesis describes data warehouse technologies and analyses the suitable solutions for the new generation of data warehouse for the advertising system Sklik.cz. Discussed technologies are MySQL, Hadoop, Apache Impala, Apache Hive, Druid and others. Apache Impala was chosen as the most suitable for Sklik.cz and is described in-depth. Project part of this thesis focuses on data migration, compatibility with old MySQL data warehouse and practical usage of Apache Impala.

Keywords: Data warehouse, OLAP, MySQL, partitioning, Hadoop, Apache Impala, Parquet, Apache Hive

Rodině za lásku a podporu.
Firmě za prostor a důvěru.
Škole za vstřícnost a ochotu.

OBSAH

ÚVOD

Tato diplomová práce se zabývá technologiemi pro vybudování datového skladu pro globální statistiky reklamního systému Sklik.cz firmy Seznam.cz.

V teoretické části bude popsána teorie datových skladů a podrobněji budou představeny konkrétní technologie, zejména pak Apache Impala. Bude popsán současný datový sklad Sklik.cz a statistická data, která jsou v něm uchovávána.

Nedostatky současného datového skladu Sklik.cz a požadavky na nové řešení budou rozebrány v analytické části. Dále v ní budou diskutovány jednotlivé technologie z hlediska vhodnosti pro Sklik.cz a z nich bude vybrána jedna, která bude dále otestována a popsána.

V projektové části pak bude zhodnocen přínos vybraného řešení, bude popsán proces migrace dat a další úpravy související s novým datovým skladem.

I. TEORETICKÁ ČÁST

1 Datové sklady

Než začneme v následujících kapitolách popisovat jednotlivé databázové systémy, je třeba definovat, k čemu je budeme z hlediska této práce využívat. Naším cílem bude uložit větší objemy dat tak, abychom nad nimi mohli v co nejkratším čase provádět analýzu.

Jako datový sklad (*data warehouse*) se označuje systém, který umožňuje provádět analýzu v něm uložených dat. Obvykle mívá tyto vlastnosti:

- je centrálním úložištěm dat integrujícím data z různých zdrojů
- ukládá jak aktuální, tak historická data
- data se do něj dostávají procesem *ETL* (*Extract-Transform-Load*), který je upraví do formy vhodné pro analytické dotazování
- slouží k vytváření analytický reportů, podpoře rozhodování (*decision support*) a získávání znalostí z databází (*data mining*)

Historie datových skladů sahá až do počátku osmdesátých let a v současné době tvoří jádro *Bussiness Intelligence* [37]. S datovými sklady souvisí také pojem *Datové tržiště* (*Data Mart*), což je část datového skladu, která se týká jen jednoho typu subjektu. Datové tržiště se typicky používá v jednom oddělení firmy - např. pro personální oddělení může existovat datové tržiště zahrnující zaměstnance a ve výrobním oddělení zase tržiště evidující série výrobků [52].

Novým pojmem v této oblasti je *(Data Lake)*, který lze chápat jako nadmnožinu datového skladu. *Data Lake* obsahuje velké množství dat v různých formátech (strukturovaných i nestrukturovaných), která jsou určena k různým účelům, často definovaným až v průběhu samotné analýzy dat. Nejde tedy o jeden konkrétní systém, ale o soubor různorodých nástrojů a technologií. [6]

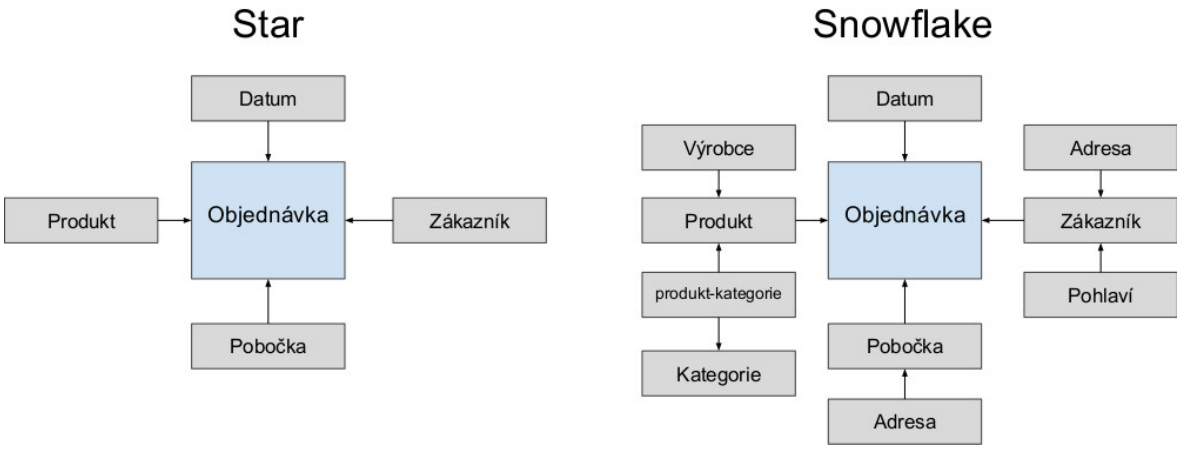
1.1 Datový sklad a forma uložení dat

Datový sklad uchovává dva typy dat: *fakta* a *dimenze*. Fakta jsou informace o subjektu, které jsou předmětem analýzy - například počet prodaných kusů zboží nebo počet zobrazení inzerátu. Jako *tabulka faktů* (*fact table*) se označuje tabulka v datovém skladu, která sdružuje fakta o daném subjektu, např. nákupu nebo zákazníkovi. [7]

Dimenze jsou pak atributy subjektu, které upřesňují fakta a podle nichž se data při dotazech filtrují a agregují. Na příkladu s prodaným zbožím by to mohla být kategorie zboží nebo datum prodeje.

1.2 Schéma datového skladu

Podle stupně normalizace dat můžeme rozdělit datové sklady na *dimenzionální* (*dimensional*) a normalizované (normalized). Dimenzionální přístup bývá reprezentován jednou tabulkou faktů a několika na ni navázanými tabulkami dimenzí, které ale netvoří hierarchii a musí splňovat pouze druhou normální formu (2NF). Takovéto schéma se označuje jako hvězdicové (*Star schema*) a jeho výhodou je přehlednost databáze a efektivita spojování tabulek při dotazování.



Obr. 1.1 Typy schémat: *hvězda* (vlevo) a *vločka*

Normalizovaný přístup má shodnou tabulku faktů jako přístup dimenzionální, ale liší se tabulky dimenzí. Ty jsou ve třetí normální formě (3NF) a mohou tedy tvořit hierarchii. Z této definice plyne, že normalizovaný přístup je podmnožinou dimenzionálního. Schéma databáze má v takovémto případě podobu sněhové vločky (*Snowflake schema*). Výhodou tohoto přístupu je úspora místa při uložení, snazší vkládání dat bez dodatečného zpracovávání a případná jednodušší změna schématu. [38]

1.3 Rozdělení datových skladů podle úložiště

Podle způsobu fyzického uložení dat rozlišujeme tři typy datových skladů.

ROLAP (*Relational Online Analytical Processing*) datový sklad fyzicky ukládá data v relačních tabulkách. Jeho výhodou je rozšířenost relačních databází a snadné vkládání i případné upravování dat v databázi, protože nevyžadují žádné dodatečné procesy při ukládání dat. Jsou také lépe škálovatelné a lépe tedy zvládají velké objemy dat. [41]

MOLAP (*Multidimensional Online Analytical Processing*) ukládá data ve specializovaném formátu optimalizovaném pro analytické operace, typicky jde o takzvané *datové kostky* (*data cubes*). Výhodou tohoto přístupu je pak vyšší rychlost při dotazování.

Mezi další výhody oproti *ROLAP* dále patří menší velikost uložených dat díky jejich kompresi. [40]

Kombinací obou typů je pak *HOLAP* (*Hybrid Online Analytical Processing*), který využívá výhod obou přístupů. Data ukládá primárně v relačním úložišti, ale nejvíce používané části z nich převádí do *MOLAP*. [39]

Existují komerční řešení pro datové sklady jako Microsoft Analysis Services nebo MicroStrategy, které kombinují všechny tři přístupy, a pak také komerční datové sklady pouze typu *ROLAP* (SAP Business Objects, Tableau Software) nebo *MOLAP* (Cognos Powerplay, Oracle Database OLAP Option, v Hadoopu pak Kyvos Insights). Zástupcem open-source *MOLAP* skladů je Palo a Apache Kylin, zatímco mezi *ROLAP* najdeme mnoho open-source řešení - Mondrian, Olaper a z Hadoopu by se tam dal zařadit např. Apache Hive nebo Apache Impala.

2 Analýza současné situace datového skladu Sklik.cz

Tato kapitola má za cíl přiblížit reklamní systém Sklik.cz a zejména jeho část pracující se statistikami.

2.1 Reklamní systém Sklik.cz

Sklik je jedním z klíčových produktů firmy Seznam.cz, a.s. Tuto firmu založil v roce 1996 v Praze Ivo Lukačovič a byla tehdy jedním z prvních katalogů a vyhledávačů webových stránek v České republice. V průběhu dalších let Seznam.cz začal postupně provozovat další služby a v roce 2006 se mezi ně přidal i samotný reklamní systém Sklik.cz. [33]

Sklik je systém typu *PPC* (*Pay Per Click*, platba za proklik), který umožňuje vlastníkům webových stránek poskytovat reklamní plochu a naopak inzerentům zadávat reklamní inzeráty. Sklik tedy tvoří prostředníka mezi těmito dvěma skupinami a jeho úkolem je propojit je tak, aby návštěvníkům webů nabízel co nejrelevantnější reklamu. [35]

Vzhledem k množství vydaných inzerátů (řádově stovky milionů za den), propojení s dalšími službami, nezbytné robustnosti a rychlosti a také vzhledem k požadované relevanci reklamy (včetně personalizace) je Sklik poměrně rozsáhlým systémem - obsahuje mnoho desítek samostatných komponent, knihoven, serverů a databází. Filozofií celého firmy Seznam.cz je využívání Open Source Software a všechny tyto komponenty tedy běží pod operačním systémem Debian GNU/Linux.

2.2 Statistiky v Skliku

2.2.1 Metriky a atributy inzerce

Při vydávání reklamy na Skliku je třeba evidovat mnoho metrik a atributů a ty je pak nutné mít uložené v různých formátech odpovídajících způsobu jejich využití.

Nejvýznamnějšími metrikami jsou počet zobrazení inzerátu a počet kliků a obě jsou také využívány k účtování zákazníkům. Účtování za proklik bývá označováno jako *CPC* (*cost per click*) a platba za zobrazení jako *CPI* (*cost per impression*), *CPT* (*cost per thousand*) nebo *CPM* (*cost per mile*). Inzerenti si mohou vybrat, který z typů plateb pro danou skupinu inzerátů chtějí použí pro své inzeráty a majitelé webů poskytující svůj reklamní prostor (na Skliku bývají označováni jako *partneři*) podle toho dostávají zaplaceno.

Mezi nejpoužívanější metriky týkající se vydávané reklamy patří na Skliku tyto:

- *počet zobrazení inzerátu* (*impression*)

- *počet zobrazení stránek* s reklamou (*page view*) - na jedné stránce bývá typicky více reklamních ploch
- *počet kliků* na inzerát (*click*) - základní metrika, také bývá označována jako prokliky
- celková *cena za inzerci* (*money*) - pro inzerenty to znamená utracenou částku, pro partnery vydělanou
- *počet konverzí* inzerátu (*conversion*) - konverze bývá definována zadavatelem reklamy, typicky jde o provedení objednávky v elektronickém obchodě, registrace uživatele nebo zobrazení nějaké významné stránky informačního webu
- *hodnota konverze* (*conversion money*) - hodnota konverze, např. cena objednávky v elektronickém obchodě
- *počet neplatných kliků* (*discarded clicks*) - počet proklikŭ, které byly odfiltrovány detektorem špatně započítaných a podvodných kliků
- *hodnota zobrazení* (*impression money*) - cena za zobrazení inzerátu

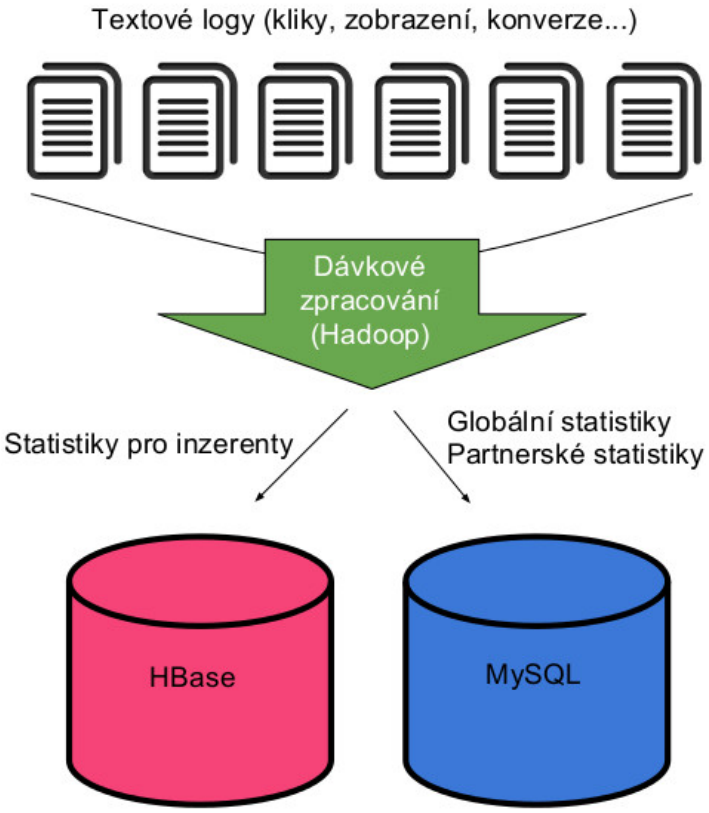
Z těchto základních statistik se vypočítávají další odvozené statistiky, jako je *CTR* (*clickthrough rate* - míra proklikovosti), průměrné *CPC* (*cena za proklik*), *PNO* (*podíl nákladů na obratu*) a další. [34]

K výše uvedeným metrikám je pak třeba uchovávat atributy, mezi něž patří především:

- *datum* zobrazení, imprese nebo kliku
- *web* na kterém se reklama zobrazila
- *zóna* ve se reklama zobrazila - partnerské weby definují v různých částech stránky různé reklamní prostory - zóny
- *pořadí inzerátu v zóně* - zóny mohou obsahovat více reklamních pozic
- *typ reklamní sítě* - z webů partnerů nebo z fulltextového vyhledávání Seznam.cz
- *typ cílení* - na základě jakých algoritmů byl inzerát vybrán - na základě klíčových slov, podle webu partnera, podle zájmů návštěvníka apod.
- *ID inzerenta*, jehož inzerát byl vybrán
- *ID kampaně a sestavy*, do níž inzerát patří
- *typ zařízení*, které návštěvník webu použil (desktop, mobil nebo tablet)

2.2.2 Zpracování a uložení statistik

Všechny statistiky o vydávání inzerátů se na Skliku primárně zapisují do textových souborů označovaných jako statlogy. Tyto statlogy jsou dále zpracovávány agregátory běžícími v Hadoopu (více o Hadoopu v kapitole 4.1), což jsou specializované programy napsané v jazyce Java nebo Scala, které využívají frameworky MapReduce nebo Spark. Tyto agregátory filtrují a spojují data a jejich výsledkem jsou buď další předzpracované statlogy (které mohou být vstupem dalších agregátorů a tvoří tak hierarchii), anebo zapisují do různých druhů databází - MySQL, HBase, Cassandra, Aerospike apod. Do každé z těchto databází přistupují další komponenty, které statistiky z ní využívají k různým účelům - od zobrazení přehledů pro inzerenty až po výpočet modelů využívaných ke zpřesnění dalšího vydávání inzerátů.



Obr. 2.1 Zpracování statistik v Skliku

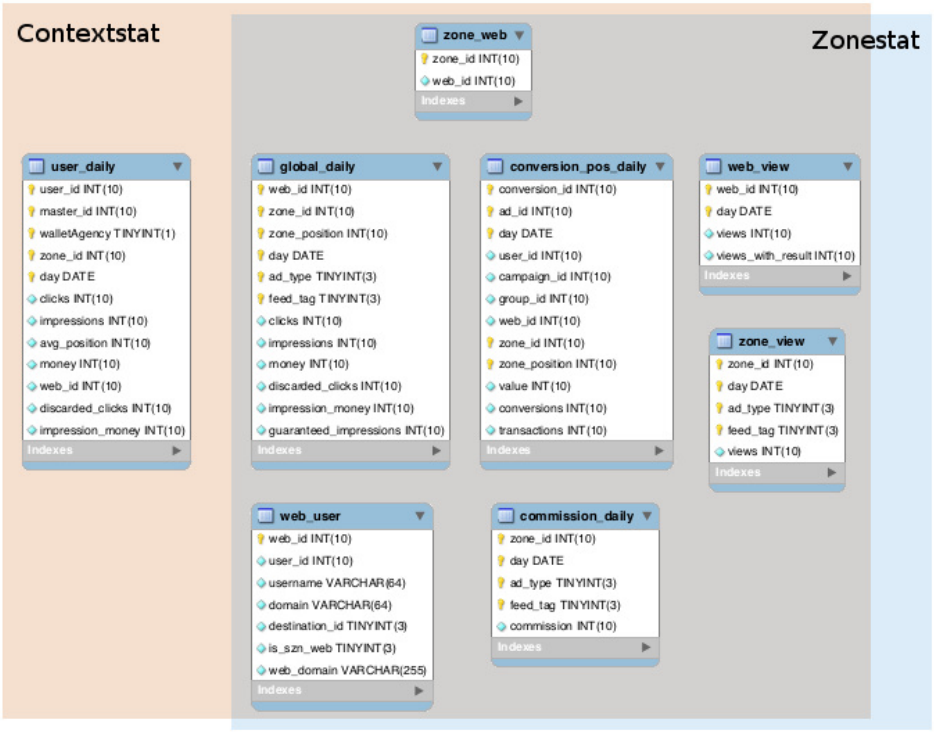
2.3 Globální statistiky

Jako globální statistiky Skliku jsou označovány souhrnné statistiky o vydávané reklamě v systému Sklik.cz, které jsou uloženy ve dvou databázích MySQL. Slouží ke kontrole celkového výkonu reklamního systému i k výpisu jednotlivých webů nebo inzerentních agentur.

Aktuální data do globálních statistik zapisuje každou půlhodinu inkrementální agregátor běžící v Hadoopu. Každou noc pak běží tento agregátor v módu, který souhrnně zpracuje celý předchozí den a v databázi opraví případná chybná data.

2.3.1 Schéma databáze pro globální statistiky

Data jsou uložena v několika tabulkách ve dvou databázích - zvlášť pro reklamu z fulltextového vyhledávače Seznam.cz (databáze *zonestat*) a zvlášť pro ostatní inzerci zobrazovanou na webech Seznam.cz a partnerů (tzv. kontextová reklama; databáze *contextstat*).



Obr. 2.2 Databázové schéma tabulek pro globální statistiky

Databáze využívají MyISAM úložiště a dohromady zaujímají téměř 1 TB dat. Zhruba třetinu z toho přitom tvoří data zkomprimovaná pomocí nástroje *myisampack* (více v 3.3). Databáze obsahují řádově desítky tabulek využívaných různými komponentami, ale pro globální statistiky dostupné přes webové rozhraní se využívají jen následující tabulky:

- *global_daily* - hlavní tabulka faktů uchovávající statistiky s dimenzemi až na úroveň pozic v zónách webů
- *commission_daily* - údaje o vyplacených provizích
- *web_view*, *zone_view* - počet zobrazení webů a zón

- *web_ user* - informace o partnerovi (poskytovateli reklamního prostoru)
- *zone_ web* - vazební tabulka mezi zónou a webem
- *conversion_ pos_ daily* - uchovává informace o provedených konverzích

Databáze *contextstat* má navíc tabulku *user_ daily*, která obsahuje statistiky rozdělené do další dimenze - podle inzerentů. Samotná tato tabulka tak tvoří zhruba 90% velikosti ze 150 GB, které zaujímají výše zmíněné tabulky využívané pro webové rozhraní globálních statistik. Do uvedené velikosti je započítána jak velikost dat v tabulce, tak velikost indexů nad tabulkou.

Vzhledem k použitému úložišti MyISAM nejsou mezi tabulkami cizí klíče.

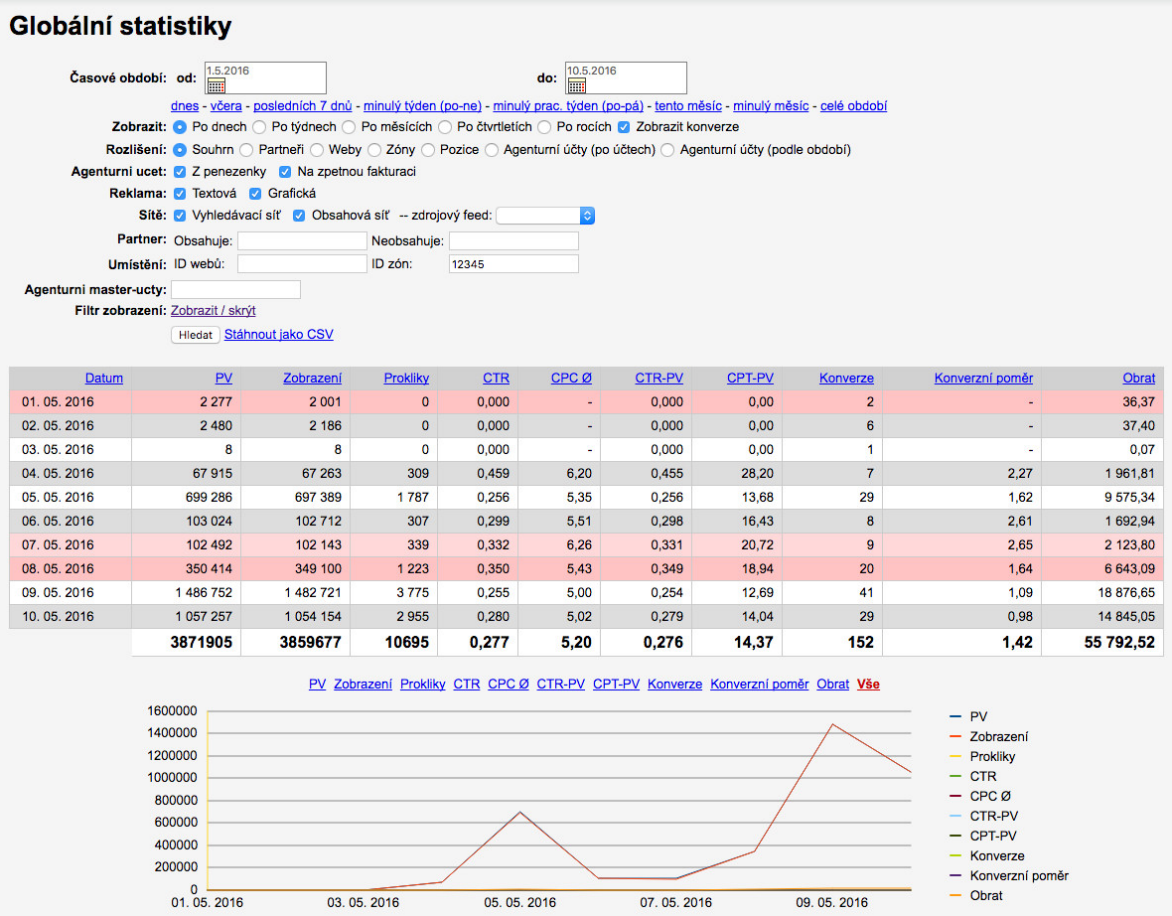
2.3.2 Rozhraní pro přístup k datům

Pro práci s globálními statistikami se využívá webové rozhraní speciálně vytvořené za tímto účelem, které je přístupné na interním administrátorském webu zaměstnancům firmy s patřičným oprávněním.

Rozhraní umožňuje zvolit požadovanou granularitu zobrazení výsledků jak na úrovni časové, tak na úrovni webů, zón a inzerentních agentur. Dále je možné definovat časový rozsah a další podmínky pro filtrování výsledků, vybírat údaje, které se mají zobrazit, a řadit podle nich.

Webové rozhraní také zobrazuje graf s vybranými metrikami a umožňuje výsledek vyexportovat do formátu CSV.

Problémem současného řešení je vzhledem k velikosti databáze doba odezvy. Pro složitější dotazy obsahující časové rozmezí delší než několik měsíců není databáze schopna vrátit výsledek.



Datum	PV	Zobrazení	Prokliky	CTR	CPC Ø	CTR-PV	CPT-PV	Konverze	Konverzní poměr	Obrat
01. 05. 2016	2 277	2 001	0	0,000	-	0,000	0,00	2	-	36,37
02. 05. 2016	2 480	2 186	0	0,000	-	0,000	0,00	6	-	37,40
03. 05. 2016	8	8	0	0,000	-	0,000	0,00	1	-	0,07
04. 05. 2016	67 915	67 263	309	0,459	6,20	0,455	28,20	7	2,27	1 961,81
05. 05. 2016	699 286	697 389	1 787	0,256	5,35	0,256	13,68	29	1,62	9 575,34
06. 05. 2016	103 024	102 712	307	0,299	5,51	0,298	16,43	8	2,61	1 692,94
07. 05. 2016	102 492	102 143	339	0,332	6,26	0,331	20,72	9	2,65	2 123,80
08. 05. 2016	350 414	349 100	1 223	0,350	5,43	0,349	18,94	20	1,64	6 643,09
09. 05. 2016	1 486 752	1 482 721	3 775	0,255	5,00	0,254	12,69	41	1,09	18 876,65
10. 05. 2016	1 057 257	1 054 154	2 955	0,280	5,02	0,279	14,04	29	0,98	14 845,05
	3871905	**3859677**	**10695**	**0,277**	**5,20**	**0,276**	**14,37**	**152**	**1,42**	**55 792,52**

Obr. 2.3 Webové rozhraní pro globální statistiky

3 Datový sklad v MySQL

Při výběru technologie pro nový datový sklad Skliku je dobré podívat se nejprve na možnosti databáze MySQL, kterou používá současný datový sklad. MySQL je databáze typu *OLTP* (*Online Transaction Procesing Tool*), která je primárně navržena jak pro čtení dat, tak pro efektivní paralelní zápis (podpora transakcí). Pro datový sklad se užívají spíše *OLAP* (*Online Analytical Processing Tool*) systémy, ale díky flexibilitě MySQL je možné při využití několika technik provozovat datový sklad i v této databází.

3.1 Indexy a jejich využití pro datový sklad

Indexy v MySQL jsou interní datové struktury (pro MyISAM engine jsou to *B stromy*, pro InnoDB *B+ stromy*), které slouží ke zrychlení dotazů, protože díky nim může odpadnout nutnost procházet při filtrování všechna data v tabulce. Indexy se vytváří a přebudovávají automaticky při každé modifikaci dat.

Pro dotazování nad datovým skladem je typické využití filtrování a řazení podle libovolné dimenze nebo metriky. Tento požadavek je právě slabinou MySQL, protože pokud bychom k tomu chtěli využívat indexy, potřebovali bychom vystavěný index pro každý z těchto sloupců, anebo ještě lépe pro každou možnou kombinaci sloupců.

Mít indexy pro každou kombinaci sloupců není pro velké tabulky únosné, protože indexy mívají velikost srovnatelnou s velikostí samotných dat v tabulce. Čtení z velkých indexů je pomalé a jejich údržba při modifikaci dat je velice náročná [49, 269]. Nejlepším řešením je tedy vytvořit indexy jen nad několika málo nejpoužívanějšími dimenzemi a místo nich využít partitioning popsaný v následující podkapitole.

3.2 Partitioning

Partitioning je technika vhodná pro velké tabulky, kdy dojde k fyzickému rozdělení dat do několika tabulek (*partitions*) se stejným schématem a k nim je pak přistupováno pomocí jedné "virtuální" tabulky. Tabulky jsou rozděleny podle jednoho sloupce (*partitioning key*) a pokud se tento sloupec vyskytuje v podmínce dotazu, může dojít k vyhledání dat jen v tabulkách, které jeho podmínce vyhovují, a ostatní tabulky mohou být rovnou přeskočeny (*partition pruning*) [49, 265].

Použití partitioningu má několik výhod. Po rozdělení jedné velké tabulky na více menších dojde ke zrychlení dotazů díky menším indexům. Pokud máte aplikaci, která mnohem častěji přistupuje k novým datům než ke starým, můžete zvolit menší velikost této poslední partition a tím dosáhnout zrychlení pro nejnovější data (*hot data*). Další výhodou je možnost lépe pracovat s jednotlivými částmi tabulky, např. mazat staré partitions nebo provádět zálohy jen vybraných partitions a také používat k uložení

jednotlivých partitions různé pevné disky. [49, 266]

Rozdělení tabulek je dáno výrazem definovaným při vytváření tabulky. Tento výraz nejčastěji definuje rozsah (*range*) nebo hash, podle kterého jsou záznamy rozřazovány do jednotlivých partitions. Samotný partitioning je podporován v MySQL od verze 5.1. Do verze 5.5 musel být výraz použitý pro partitioning typu *integer*, ale od této verze už je možné používat další datové typy. Omezením tabulek využívajících partitioning je nemožnost použít cizí klíče. [49, 266]

Partitioning je považován za náhradu starší techniky v MySQL nazývané *merge tables*. Jejich společným rysem je využití více separátních tabulek se stejným schématem, *merge tables* ale vyžadují explicitní práci s dílčími tabulkami - je nutné je ručně spravovat a každou z nich potom přiřadit hlavní tabulce.

3.3 Další optimalizace

Při budování datového skladu je kvůli velkým objemům dat nezbyné používat v MySQL co nejúspornější datové typy sloupců.

Je výhodné použít MyISAM engine, protože umožňuje zmenšení velikosti indexů zhruba na desetinu díky komprimaci klíčů (*pack keys*) [49, 184] [36, 18]. Dále pro něj existuje utilita *myisampack*, která data uloží po sloupcích a zkomprimuje je na 40-70 % původní velikosti. Bohužel kompresi není možné kombinovat s partitioningem [30].

Pro zmenšení zátěže při zápisu dat do velkých tabulek je možné používat techniku zapisování do dočasné tabulky (*staging*) a data pak dávkově zapsat do hlavní tabulky pomocí *INSERT ... SELECT* příkazu [45].

Dobrým přístupem pro zrychlení datového skladu v MySQL je použití *materializovaných pohledů* (*materialized views*). Pohledy umožňují definovat tabulky odvozené od jiných tabulek a rozdělují se na nematerializované (ty neuchovávají data a dotazy jen přeposílají na originální tabulky) a materializované (ty fyzicky uchovávají kopii dat ze zdrojové tabulky), přičemž pro datové sklady je prakticky využitelná jen druhá možnost. Bohužel samotná MySQL ani v nejnovější verzi 5.7 materializované pohledy nepodporuje, takže je nutné je buď nasimulovat pomocnou tabulkou a triggery, anebo využít např. Flexviews rozšíření [51] [49, 138]

Díky materializovaným pohledům je možné mít v databázi předdefinované a případně i předpočítané tabulky optimalizované pro specifické dotazy. Touto cestou se vydávají také specializované nadstavby nad MySQL, které budou popsány v následující podkapitole.

3.4 Specializované nadstavby nad MySQL

Častým řešením pro datové sklady v MySQL je data ze zdrojových tabulek nějakým způsobem předzpracovat. Podle typu úložiště takto extrahovaných dat je možné nadstavby rozdělit na *ROLAP* [41], *MOLAP* [40] a *HOLAP* [39] systémy. Tyto systémy včetně jejich zástupců byly podrobněji popsány v kapitole 1.3.

4 Apache Impala

Apache Impala je engine pro analýzu dat pomocí standardního dotazovacího jazyka SQL. Je vhodná pro datové sklady s větším objemem dat. Jde o poměrně novou technologii, která zpracovává data v distribuovaném prostředí Apache Hadoop.

Impala byla navržena tak, aby odstranila nedostatky soudobých nástrojů pro *ad-hoc* analýzu velkých dat - dlouhou odezvu a nutnost používat nestandardní metody dotazování. Jejím cílem je přiblížit se uživatelskému komfortu, jaký poskytují relační databáze, a zároveň nabídnout dobré škálování systému. Dalším z jejích cílů bylo umožnit souběžnou práci více uživatelů bez výrazných dopadů na výkon.

Pro datový sklad Skliku je tedy Impala díky podpoře SQL jazyka a dobré škálovatelnosti perspektivním kandidátem.

Tato kapitola podrobněji popisuje Apache Impalu ve verzi 2.1 vydanou v prosinci roku 2014.

4.1 Impala a Hadoop

Před podrobnějším popisem Apache Impaly je nutné přiblížit nejprve samotný distribuovaný systém Apache Hadoop, jehož je Impala součástí.

Apache Hadoop je softwarový framework určený pro distribuované ukládání a zpracování dat. Jeho filozofií je využít pro náročné výpočty běžně dostupný hardware - Hadoop tedy počítá s jeho výkonnostními problémy i výpadky a dokáže si s nimi poradit. [53, 6]

Díky open-source licenci, široké škále provázaných komponent (tzv. *Hadoop ekosystém*) a díky zmíněnému využívání běžného hardware je v současné době Hadoop de facto standardem pro zpracování velkých dat.

Počátky Hadoopu sahají až do roku 2004, kdy byl vytvářen v rámci projektu Apache Nutch nový distribuovaný souborový systém inspirovaný GFS (Google File System). Později se k němu přidala implementace MapReduce paradigmatu (taktéž inspirovaná Googlem) a v roce 2006 se z Hadoopu stal samostatný projekt. O dva roky později se Hadoop zařadil mezi top-level projekty *Apache Software Foundation*. [53, 14]

Na podzim roku 2013 byla vydána verze 2 (resp. 2.2), která přinesla kompletně přepracovanou správu výpočetních prostředků i samotného MapReduce.

Za základ současného Hadoopu jsou považovány tyto komponenty:

- *HDFS* (*Hadoop Distributed Filesystem*) - distribuovaný souborový systém, který poskytuje vysokou dostupnost dat a transparentnost přístupu k nim. Je navržen pro uchovávání velkých souborů a provádění sekvenčního čtení a zápisů.

- *YARN* (*Yet Another Resource Negotiator*) - správce výpočetních prostředků v počítačovém clusteru, který umožňuje přidělovat a monitorovat procesor a paměť na jeho jednotlivých uzlech
- *MapReduce algoritmus* - výpočetní paradigma, které umožňuje výpočet efektivně rozdělit na nezávislé části a ty pak provést paralelně na uzlech clusteru

V posledních letech dochází k prudkému rozvoji Hadoopu a dalších projektů, které jsou na něj navázány - souhrnně jsou označovány jako *Hadoop ekosystém*. Mezi nejznámější patří:

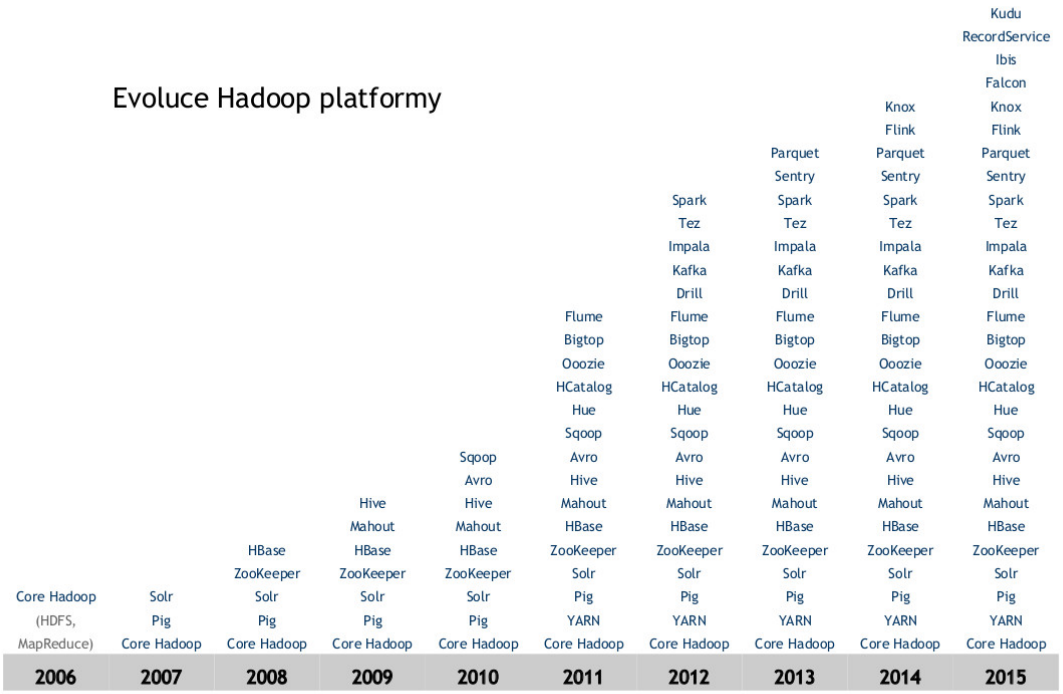
- obecné zpracování dat: Spark
- datová analytika: Hive, Impala, Shark, Spark SQL, Hue
- databáze: HBase, Cassandra, Accumulo, Storm
- real-time zpracování dat: Samza, Kafka, Flume
- souborová úložiště a formáty: Avro, Parquet, Kudu
- vysokoúrovňová manipulace s daty: Pig, Sqoop, Hive, Kite
- nástroje pro umělou inteligenci: Mahout

Mnoho společností nabízí Hadoop a další software z jeho ekosystému jako takzvané *distribuce*. Jde o vhodně poskládané verze jednotlivých komponent, k nimž může být připojen nějaký nástroj pro jejich administraci, případně mohou být propojeny s dalším proprietárním softwarem firmy. Mezi nejvýznamnější distribuce Hadoopu patří ty od firem Cloudera, Hortonworks, MapR, IBM a Amazon.

4.2 Historie Impaly

Impala byla vytvořena v roce 2012 v kalifornské společnosti Cloudera, inc., která patří mezi nejvýznamnější firmy poskytující komplexní integrovaná řešení pro práci s velkými daty (*big data*) nad platformou Hadoop. Inspirací pro tento projekt byl návrh analytické databáze Dremel od Googlu z roku 2010 [46].

Cloudera Impala byla vytvořena se záměrem nahradit Apache Hive, který svým konceptem nesplňoval požadavky na on-line business analýzu dat. Impala byla proto navržena tak, aby co nejvíce výpočetních operací prováděla lokálně, aby nevyužívala MapReduce paradigma a aby pracovala ideálně nad soubory optimalizovanými pro čtení po sloupcích [46]. Záměr, se kterým Impala vznikla, byl tedy podobný tomu, se kterým později vznikla iniciativa Stinger v Hivu (více v 5.1).



Obr. 4.1 Evoluce Hadoop platformy - přidávání nových projektů (komponent) [8]

Impala verze 1.0 byla vydána na jaře 2013 a během několika následujících měsíců přibývalo ve verzích 1.2, 1.3 a 1.4 mnoho vylepšení, díky kterým se stal z Impaly dobře použitelný nástroj. Přibyly nové možnosti pro správu uložených dat, vylepšeny byly všechny dílčí komponenty systému a dotazovací jazyk byl rozšířen o další funkcionalitu. [14]

V říjnu 2014 byla vydána verze 2.0, která znamenala další krok v postupném vývoji. Bylo rozšířeno ukládání mezivýsledků na disk (*disk spilling*), vylepšeny vnořené dotazy (*subqueries*), rozšířeny možnosti řízení přístupu uživatelů, přidány statistické funkce a mnoho dalších vylepšení. [14]

V prosinci 2015 byla Impala (v tu dobu aktuálně ve verzi 2.3) zařazena do inkubátoru Apache [17], což znamenalo prestižní zařazení mezi další open-source projekty pod hlavičkou *Apache Software Foundation* (*ASF*). Impala má tedy v současné době status *incubating*, což je dočasné označení nových projektů v rámci *ASF*. Část z nich je po určité době zařazena na seznam *Top-level* projektů a stanou se z nich plnohodnotné Apache projekty, část je vyřazena. Aktuální vývoj Impaly je nově řízen nikoliv firmou Cloudera, ale open-source komunitou, respektive schvalovací hierarchií definovanou *ASF*.

Tab. 4.1 Přehled verzí Impaly a distrubucí Hadoopu od Cloudery (*CDH*), ve kterých se Impala vyskytuje [22]

Verze Impaly	Datum vydání	Verze CDH
1.0	2013/05	-
1.1	2013/07	-
1.2	2013/11	-
1.3	2014/05	5.0
1.4	2014/07	5.1
2.0	2014/10	5.2
2.1	2014/12	5.3
2.2	2015/04	5.4
2.3	2015/11	5.5
2.4	2016/03	5.6
2.5	2016/04	5.7
2.6	2016/07	5.8
2.7	2016/10	5.9
2.7	2017/01	5.10

V současnosti nabízí Impalu ve svých distribucích všechny tři největší společnosti poskytující Hadoop platformu.

4.3 Architektura

Jedním ze základních rysů Impaly je škálovatelnost, která je umožněna distribuovanou architekturu celého systému. Impala byla vytvořena v prostředí Apache Hadoop, z nějž některé komponenty přímo využívá a s mnoha dalšími poskytuje možnost integrace.

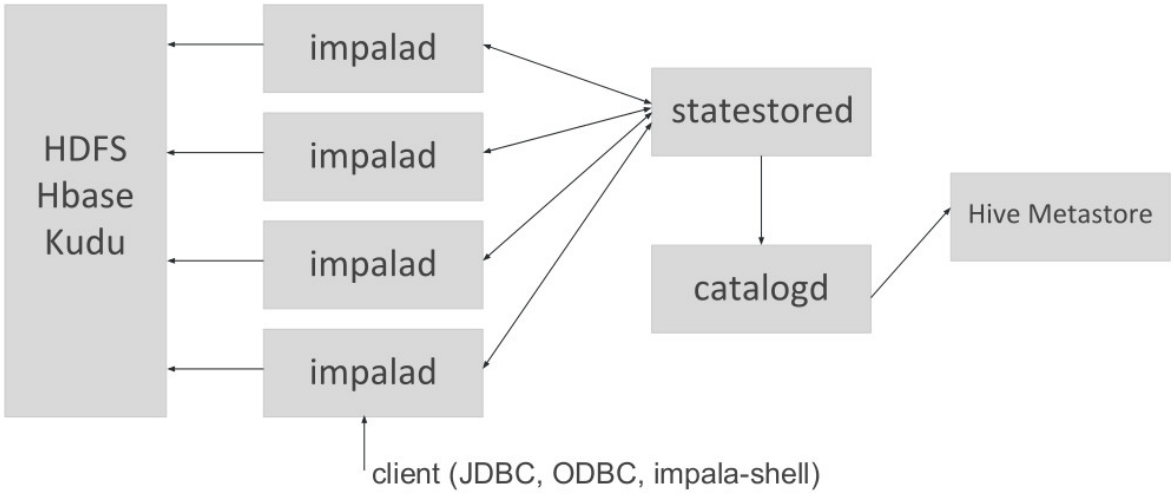
Samotná Impala je složena ze tří typů komponent - výpočetních démonů (*Impala daemon*), z koordinátora přístupu k datům (*Statestore daemon*) a ze správce metadat (*Catalog daemon*). Pro svůj chod vyžaduje další dvě externí Hadoop komponenty: distribuované úložiště dat (HDFS, HBase nebo Kudu) a pro uložení metadat pak Metastore (Hive nebo nově HBase).

4.3.1 Impala démon

Nejvýznamnější komponentou v Impale je takzvaný Impala démon (*Impala daemon*), který provádí operace se samotnými daty. Zároveň také slouží jako přístupový bod pro klientské aplikace.

Impala démon by měl běžet na každém uzlu výpočetního clusteru. Přidávání uzlů a tedy i Impala démonů umožňuje horizontální škálování výkonu.

Klientské aplikace přistupující k Impale, jako jsou například *impala-shell* nebo roz-



Obr. 4.2 Komponenty Impaly

hraní *JDBC* a *ODBC*, se mohou připojit k libovolnému Impala démonu a ten se pro daný dotaz stává koordinátorem. Provede analýzu dotazu, na základě dostupných metadat a statistik vytvoří plán vykonání dotazu (*execution plan*) a deleguje pak výpočty na Impala démony na dalších uzlech. Ti se pak postarají o paralelní zpracování dotazu - načtení dat z úložiště a zpracování jejich určité části (filtrace, agregace). Výsledky následně pošlou zpět koordinátorovi, který z nich vypočítá konečný výsledek a odešle jej zpět ke klientovi. [10]

Impala démon si pro vytvoření optimálního exekučního plánu lokálně uchovává metadata, která mu při každé jejich změně posílá Catalog démon.

4.3.2 Statestore démon

Tato komponenta slouží k monitorování Impala démonů. Periodicky kontroluje jejich stav a udržuje si přehled o aktivních uzlech.

Statestore démon se vyskytuje v clusteru jen v jediné instanci. V případě jeho výpadku není nijak omezeno vykonávání dotazů v clusteru, protože sám pro plánování a následné vykonávání není potřeba. Pokud ale neběží a dojde k výpadku některého z Impala démonů, pak přestává cluster fungovat. Koordinátor dotazu v takovém případě nemá možnost zjistit, že je daný démon nedostupný, a požaduje po něm výpočet části dotazu. Když se pak nedočká odpovědi, rovnou klientovi vrací chybu, protože Impala neposkytuje žádnou možnost zotavení se z chyby na paralelním subprocesu. [10]

Prostřednictví Statestoru probíhá také komunikace mezi Impala démony a Catalog démonem. Při výpadku Statestoru tedy navíc platí totéž co o výpadku Catalog démona, což bude popsáno dále.

V případě, že je požadována vysoká dostupnost celého clusteru, je možné mít States-

tore démony na více strojích a přistupovat na ně přes proxy službu. V případě výpadku jedné z instancí je pak proxy nasměrována na záložní instanci a ta přebírá její roli. [15]

4.3.3 Catalog démon

Pro správné vytváření exekučních plánů dotazů na jednotlivých uzlech clusteru je potřeba mít informace o datech v tabulkách databáze. Tato metadata spravuje komponenta Catalog démon.

Při každé změně struktury databáze a také při změně množství dat v ní je třeba aktualizovat metadata. Impala démon v takovém případě notifikuje Catalog démona a ten dále přes Statestore démona rozešle informaci o změně metadat všem ostatním Impala démonům. [10]

Toto řešení umožňuje udržovat lokální kopii metadat na straně Impala démonů a díky tomu koordinátor dotazu může komunikovat pouze s ostatními Impala démony, kterým naplánoval část paralelního výpočtu dotazu.

Catalog démon se stejně jako Statestore vyskytuje v clusteru pouze v jedné instanci. V případě jeho výpadku není možné v clusteru provádět DDL dotazy. Stejně jako u Statestore démona je možné pro zvýšení dostupnosti mít v clusteru více instancí Catalog démonů a přistupovat na ně pomocí proxy služby. [15]

4.3.4 Hive Metastore

Catalog démon, popsaný v předchozí podkapitole, je pouze službou spravující metadata, ale sám o sobě tato metadata neukládá. Pro jejich uložení slouží Hive Metastore, což je komponenta z projektu Apache Hive. Ta sama ještě k uložení využívá relační databázi - buď interní databázi Apache Derby, anebo externí - MySQL, Oracle nebo Postgre.

4.3.5 Externí úložiště dat

Impala nebývá označována přímo jako databáze, protože sama neobstarává uložení dat. K tomu využívá další služby Hadoopu - distribuované úložiště dat HDFS, HBase nebo Kudu.

HDFS (*Hadoop Distributed File System*; podrobněji byl popsán v podkapitole 4.1 o Hadoopu) je obecné distribuované úložiště. Jde o výchozí úložiště pro Impalu, ve kterém může být uložen soubor v některém z Impalou podporovaných formátů. HBase úložiště interně HDFS také využívá.

Jako vhodné úložiště pro data v Impale se jeví Apache Kudu. Jde o distribuované úložiště s definovaným relačním schématem, které také nabízí replikace dat a jejich

rebalancing. Vzhledem k tomu, že tento projekt je zatím jen v beta verzi, o něm ale není možné uvažovat pro produkční využití. Podrobněji bude dále popsán v samostatné podkapitole 4.7.

Datová úložiště mají na starost perzistentní ukládání dat a měla by být distribuována na všech uzlech clusteru společně s Impala démony, kteří díky tomu mohou číst data lokálně a nemusí si je posílat po síti. Současný výskyt instance distribuovaného datového úložiště a Impala démona na jednom fyzickém uzlu se označuje jako *collocation* a je jednou z podmínek pro rychlé vykonávání dotazů v Impale.

Distribuovaná úložiště poskytují pokročilé funkce, jako je replikace dat a s ní související rebalancing. Pokud dojde k výpadku některého uzlu clusteru, Statestore démon detekuje výpadek Impala démona běžícího na tomto uzlu a oznámí jeho nedostupnost ostatním Impala démonům. Pokud jsou při některém následujícím dotazu potřeba data uložená na nedostupném uzlu, musí je zpracovat náhradní uzel a pro zaručení funkčnosti Impaly je tedy nutné mít data uložena v systému ve více kopiích. Nastavení replikačního faktoru většího než jedna tedy zajišťuje vyšší odolnost systému vůči výpadkům hardwaru.

Rebalancingem dat se rozumí schopnost vrstvy datového úložiště udržovat dynamicky požadovaný počet replik všech částí dat - v případě delšího výpadku jednoho stroje úložiště zajistí vytvoření nových kopií (replik) dat na dostupných strojích tak, aby byl neustále dodržen počet požadovaných replik pro každý datový blok v HDFS.

4.3.6 Klientská rozhraní

Pro volání SQL dotazů v Impale existuje konzolová aplikace distribuovaná přímo s Impalou - *impala-shell*. Dále Impala nabízí rozhraní pro přímý přístup: JDBC, ODBC a Thrift. Tato rozhraní jsou implementována v různých jazycích (přímo nebo pomocí dalších knihoven) - např. v Javě (JDBC, Thrift), C/C++ (ODBC), Python (Impyla) nebo R (RImpala).

4.4 Formáty pro uložení dat

Impala umí pracovat se zdrojovými daty v různých formátech - od prostých textových souborů až po vysoce optimalizované formáty pro sekvenční čtení. Volba datového formátu má vliv na velikost dat a především na celkový výkon Impaly.

Výhodou Impaly je, že datové formáty může kombinovat - a to i na úrovni jedné tabulky. Ve své aplikaci tak například můžete použít formát, který je výstupem z jiné komponenty ve vašem systému, a z něj pak můžete už přímo pomocí Impaly vytvořit optimalizovaný soubor ve formátu Parquet jen s daty, která vás dále zajímají. Jedinou podmínkou je dodržení stejného datového formátu v rámci jedné partition.

4.4.1 Plaintextové soubory

Nejjednodušším souborovým formátem jsou prosté textové soubory, ve kterých jsou datové záznamy uloženy přímo za sebou. Nejčastěji se setkáte s formáty CSV (*comma separated values*) a TSV (*tablespace separated values*), ale Impala umožňuje definovat libovolné oddělovače řádků i sloupců.

Výhodou textových souborů je snadná manipulace s nimi - můžete je lehce číst a přímo do nich zapisovat. Je to tedy ideální formát pro testování, ale pro produkční nasazení není vhodný kvůli své pomalosti. Data jsou v něm zapsána přímo a není tak možné využít žádné optimalizační strategie pro rychlé čtení dat, jako je zpracování konkrétních sloupců a komprese.

4.4.2 Avro a Sequence file

Tyto dva podporované formáty se využívají v Hadoopu pro serializaci dat. Nejsou optimalizovány pro agregace a zpracování po sloupcích, ale nabízí kompresi. Mohou být tedy vhodným formátem pro prozkoumávání dat (*data exploration* ve spojení s jiným softwarem.

4.4.3 HBase

Alternativou k datům uloženým přímo v HDFS je využití HBase databáze. Ta poskytuje vlastní distribuované datové úložiště (sama je ale také postavena nad HDFS) a data v ní jsou členěna do relačních tabulek podobně jako v Impale. Je tedy možné definovat tabulku v Impale a nastavit jí mapování na existující HBase tabulku.

Apache HBase je key-value databáze v Hadoopu, která je navržena pro jiný účel než Impala - je určena pro rychlé vyhledávání jednotlivých položek na základě klíče. Pokud je tedy využívána Impalou, pak by měly dotazy na takovou tabulku obsahovat v podmínce *WHERE* právě jednotlivé hodnoty klíčů. Bez podmínky by bylo procházení celé tabulky (*full scan*) pomalé.

Výhodou HBase je možnost zapisovat data do tabulek pomocí příkazu *INSERT ... VALUES* - ten je pro HBase narozdíl od samotného HDFS vhodný a umožňuje data zároveň i měnit, pokud v příkazu zadáme již existující klíč.

Z možnosti snadno zapisovat data plyne typické využití HBase úložiště v Impale - je vhodné pro data, která je potřeba často měnit. Může být tedy využito například pro menší tabulku dimenzí, která je pomocí klauzule *JOIN* spojována s tabulkou ve formátu Parquet, jež obsahuje velké množství dat. Takováto kombinace využívá výhody HBase tabulky (přímý zápis dat), zatímco nenaráží na její nevýhody (tabulka je malá a dotaz obsahuje podmínku na konkrétní klíč). [16]

4.4.4 RCFile

RCFile (*Record Columnar File*) je sloupcově orientovaný formát pro uložení dat běžně využívaný v Hadoopu. Podporuje adaptivní kompresi v závislosti na vstupních datech. Ačkoliv by tedy mohl být vhodným kandidátem pro využití v Impale, nedosahuje takové efektivity pro ukládání a zpracování dat jako jeho následovník ORC File [50], který však není v Impale podporován.

4.4.5 Apache Parquet

Parquet je sloupcový formát, což znamená, že datové záznamy (řádky) neukládá za sebou, ale ukládá je po sloupcích. Soubor ve formátu Parquet tedy - zjednodušeně řečeno - obsahuje za sebou seřazené hodnoty prvního sloupce všech řádků, za nimi hodnoty druhého sloupce všech řádků a tak dále.

Výhody sloupcových formátů jsou dvě: zaprvé umožňují efektivní kompresi dat a zadruhé urychlují čtení při dotazování se na data. Možnost dobře komprimovat data je dána tím, že za sebou uložené hodnoty jednoho sloupce mají často podobné hodnoty. Vliv na rychlost čtení vyplývá z faktu, že agregační funkce většinou zahrnují jen jeden sloupec a často nejsou některé sloupce z tabulky v dotazu vůbec přítomny. Při zpracování dotazu je tak možné číst naráz pouze požadované sloupce a ostatní vůbec nenačítat z disku.

Vnitřní struktura Parquet souboru Každý Parquet soubor obsahuje 4B dlouhou hlavičku, která vždy nese hodnotu *PAR1*, což je identifikátor formátu Parquet. Za ním už následují vlastní data a až za nimi, v patičce, jsou uložena metadata. Na konci souboru je pak uložena ve 4B velikost patičky a za ní pak znovu identifikátor *PAR1*. [53, 370]

Důvodem uložení metadat až v patičce a ne v hlavičce (jak je tomu u většiny ostatních formátů, např. Avro nebo sekvenčních souborů) je možnost zapisovat do metadat přímo pozici jednotlivých bloků dat. To je umožněno tím, že se patička zapisuje až po samotných datech a v tu chvíli je už velikost a pozice jednotlivých bloků dat známa. Efektivní čtení jednoho sloupce pak tedy vypadá tak, že se na začátku provede posun v souboru na pozici 8B před koncem, přečte se velikost patičky, provede se další posun na začátek patičky a z metadat, která jsou v ní uložena, je pak možné se přímo přesunout na začátek datového bloku, který obsahuje data požadovaného sloupce.

Díky uložení začátků datových bloků v patičce odpadá potřeba oddělovat datové bloky speciálními značkami (které by se musely jinak vyhledávat průchodem přes celý soubor) a Parquet soubor je tak možné jednoduše rozdělit pro paralelní čtení.

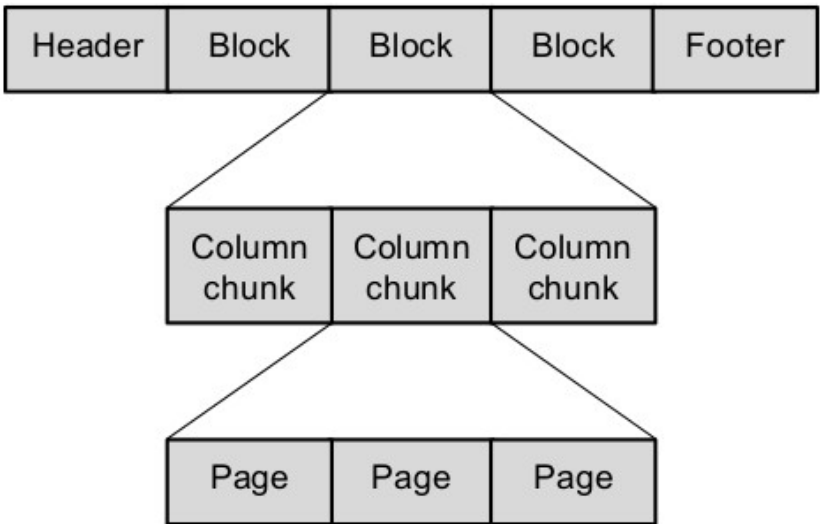
Obr. 4.3 Sloupcové uložení dat v Parquet souboru

Každý datový blok uvnitř Parquet souboru odpovídá jedné skupině řádků (*row group*). Ta se dále dělí na jednotlivé bloky sloupců (*column chunk*). Každý takovýto blok obsahuje data jednoho sloupce a dále se dělí na menší jednotky, nazývané stránky (*page*). Každá stránka tedy nese část dat jednoho sloupce určité podmnožiny řádků datového souboru.

Velikost souboru Parquet není ničím omezena, velké soubory budou rozděleny do více HDFS bloků. Důležitá je však velikost bloku skupiny řádků (*row group*) - ta by pro co největší efektivitu neměla být větší, než velikost datového bloku HDFS, aby mohl být každý takovýto blok načten právě z jednoho HDFS bloku. Je třeba najít optimální velikosti bloku *row group* - čím větší, tím více dat se může naráz zpracovat, ale zároveň je nutné držet celý tento blok v paměti při zápisu [53, 372]. Od verze Impaly 2.0 jsou výchozími hodnotami pro velikost bloku HDFS i *row group* Parquetu 256 MB [31].

Komprese Formát Parquet při zápisu vybírá na základě vstupních dat některé z těchto kódování:

- RLE (*Run Length Encoding*) - sekvence stejných hodnot je kódována jako dvojice hodnota a počet opakování



Obr. 4.4 Vnitřní struktura Parquet souboru

- bitmapové kódování
- delta kódování - je ukládán jen rozdíl mezi po sobě jdoucími hodnotami
- slovníkové kódování - pro omezenou množinu dat je vytvořen překladový slovník, jehož klíče tvoří čísla identifikující hodnoty a ukládány jsou pak jen tyto klíče

Na takto zakódovaná data se následně může aplikovat ještě další úroveň komprese - Snappy, gzip nebo LZO.

Podpora v Hadoopu Kromě Impaly je Parquet formát podporován v nejdůležitějších nástrojích Hadoopu pro manipulaci s daty, jako jsou Pig a Hive. Existuje samozřejmě také rozhraní pro zápis a čtení v Javě. Definici dat je pak možné zadat buď explicitně v textové formě, nebo s využitím formátu Avro, Protocol Buffers nebo Thrift. Zápis do Parquetu nabízí také MapReduce knihovna [53, 377].

Apache Parquet bývá srovnáván s Apache ORC (Optimized Row Columnar) formátem, který má podobné vlastnosti a je preferovaným formátem v Hivu. Oba nabízí velice podobný výkon, Impala však ORC nepodporuje. Parquet je lepší pro vnořené (*nested*) záznamy, ORC zase nabízí možnost budovat nad daty indexy.

4.5 Partitioning

Impala nepodporuje indexy nad daty, ale místo toho využívá pro zrychlení dotazů techniku označovanou jako partitiong. Ta funguje velice podobně jako v MySQL (viz)3.2) - tabulka je fyzicky rozdělena na několik menších částí, které mají stejné schéma, ale jejich data jsou nezávislá. Při zadání dotazu s podmínkou obsahující *partition key* je pak možné zpracovat jen dotčené partitions a ostatní ignorovat (*partition pruning*).

Důležitá je volba velikosti partitions. Každá z nich by ideálně měla mít velikost srovnatelnou s HDFS datovým blokem [48, 30]. Příliš malé partitions by znamenaly neefektivní načítání mnoha malých datových souborů z HDFS, zatímco příliš velké by snižovaly možnost odfiltrovávat partitions už na základě podmínek dotazu (*partition pruning*). Pro partitioning je tedy vhodné vybrat sloupce s rozumně vysokou selektivitou a případně jejich kardinalitu zmenšit dodatečným výrazem, jako je modulo dělení anebo v předchozím příkladu volání funkce *YEAR* nebo *MONTH*. Vhodnými kandidáty na partitioning jsou obecně sloupce vyskytující se často v podmínkách dotazů, typicky to jsou právě časové dimenze.

Impala pro partitions fyzicky vytváří složky v HDFS a umožňuje je hierarchicky zanořovat. Definice partition při vytváření tabulky v takovém případě musí zahrnovat všechny úrovně zanoření. Například pokud chcete mít partitions po měsících, můžete definovat *PARTITIONED BY (year int, month shortint)* a datové soubory pak budou uloženy ve složce *year=2016/month=5*.

4.6 Podpora SQL

Apache Impala je kompatibilní se standardem SQL-92 a podporuje některé konstrukce z pozdějších SQL standardů. Vzhledem ke své podobnosti s Hivem je také vysoce kompatibilní s HiveQL.

DML příkazy Impale jsou velice podobné těm z MySQL. Největším rozdílem mezi nimi je absence příkazů *UPDATE* a *DELETE* v Impale a vzhledem k jinému principu uložení dat se částečně také liší DDL příkazy.

4.6.1 Datové typy

Impala podporuje následující datové typy:

- celočíselné: TINYINT, SMALLINT, INT, BIGINT
- desetinné: FLOAT, DOUBLE
- s definovatelnou přesností: DECIMAL
- booleovské: BOOLEAN
- znakové a řetězcové: CHAR, VARCHAR, STRING
- pro datum a čas: TIMESTAMP
- komplexní: ARRAY, STRUCT, MAP

Kompozitní datové typy byly přidány ve verzi 2.3 a umožňují hierarchické vnořování dat v tabulkách. V případě jejich využití je výhodné ukládat data v Parquet formátu, který vnořené struktury přímo podporuje. Impala však v současnosti neumí zápis tohoto datového typu, takže je nutné jej zapisovat externě - přes Hive, Spark a podobně.

4.6.2 DDL SQL příkazy

Příkazy pro manipulaci s tabulkami a dalšími strukturami v Impale jsou syntakticky podobné příkazům MySQL a lze je rozdělit do těchto souvisejících skupin:

- vytvoření a zrušení databáze: CREATE DATABASE, DROP DATABASE
- manipulace s tabulkou: CREATE TABLE, ALTER TABLE, DROP TABLE
- manipulace s pohledem na tabulku: CREATE VIEW, ALTER VIEW, DROP VIEW
- definice a odstranění uživatelské funkce: CREATE FUNCTION, DROP FUNCTION
- správa oprávnění uživatelů: CREATE ROLE, DROP ROLE, GRANT, REVOKE
- správa metadat a statistik tabulek: INVALIDATE METADATA, REFRESH, COMPUTE STATS

Oproti MySQL nepodporuje Impala v příkazech CREATE TABLE a ALTER TABLE následující konstrukce: *PRIMARY KEY*, *FOREIGN KEY*, *KEY*, *UNIQUE*, *INDEX*, *CONSTRAINT*, *REFERENCE*, *AUTO_INCREMENT* a *NOT NULL*, což plyne ze její samotné architektury. Ekvivalentem nastavením enginu v MySQL (*ENGINE*) je definice formátu uložených dat v Impale (*STORED AS*). Další odlišností je definice partitions - v MySQL k tomu slouží konstrukce *PARTITION BY*, zatímco v Impale *PARTITIONED BY*.

Při vytváření tabulky není vždy nutné specifikovat její strukturu - podobně jako v MySQL je možné vytvořit tabulku podle schématu tabulky již existující (*CREATE TABLE LIKE*) a navíc Impala umí schéma odvodit z datového Parquet souboru (*CREATE TABLE LIKE PARQUET*).

Specialitou Impaly jsou příkazy pro správu metadat a interních statistik tabulek. Metadata mají zásadní vliv na výkon, protože jsou využívána při plánování dotazů. *REFRESH* a *INVALIDATE METADATA* slouží k přepočítání, resp. zneplatnění metadat (tzn. struktury tabulek, nastavení partitions a lokality HDFS bloků datových souborů interně tvořících tabulku) a je nutné je volat jen při externích změnách tabulek a dat - např. při změně datových souborů anebo při změnách zadaných přes Hive. Při práci pouze přes rozhraní Impaly není nutné je volat, zatímco příkaz *COMPUTE STATS* je nutné volat při každé větší změně dat.

INVALIDATE METADATA Tento příkaz slouží ke zneplatnění metadat zapsaných v metastore. Jeho volitelným parametrem je název tabulky, u které se mají zneplatnit; bez parametrů příkaz zneplatní metadata všech tabulek. Příkaz provede pouze vymazání metadat v Hive Metastore, ale metadata samotná se znovu načítají až v okamžiku zavolání dalšího příkazu, který je vyžaduje - typicky jde o *SELECT* nad danou tabulkou. Invalidace metadat tedy může u velkých tabulek s mnoha partitions výrazně zpomalit první následující *SELECT*, který musí projít celé úložiště pro danou tabulku a zjistit lokalitu HDFS bloků datových souborů.

REFRESH Tento příkaz je podobný *INVALIDATE METADATA* - je také nutné jej volat po externí změně tabulky, která měla vliv na její metadata a také prochází datové soubory tabulky. Rozdíly mezi nimi jsou dva - zaprvé *REFRESH* načtení nových metadat provede okamžitě a zadruhé funguje inkrementálně, tedy zjišťuje lokalitu HDFS bloků pouze u nových datových souborů. Ve výsledku tedy kvůli okamžitému provádění přepočtu trvá *REFRESH* typicky déle než *INVALIDATE METADATA* (který metadata jen smaže), ale díky inkrementálnímu přístupu méně dlouho, než *SELECT* následující za *INVALIDATE METADATA*.

COMPUTE STATS K přepočítání interních statistik o datech v tabulce slouží příkaz *COMPUTE STATS*. Projde všechny datové soubory tabulky a spočítá nad nimi statistiky, které pak uloží do Hive Metastore. Existuje varianta příkazu s klíčovým slovem *INCREMENTAL*, která umožňuje specifikovat konkrétní partition a počítání statistik se pak provede jen nad ní. Aktuální statistiky dat v tabulce mají zásadní vliv na vytváření exekučních plánů dotazů a proto se doporučuje volat příkaz *COMPUTE STATS* při každé změně alespoň 30% dat v tabulce [48, 52].

4.6.3 DML SQL příkazy

Impala má omezenou množinu příkazů pro manipulaci s daty - existují pouze příkazy pro vkládání dat, *INSERT* a *LOAD*. Neexistenci modifikačních příkazů *UPDATE* a *DELETE* je nutné obcházet kopírováním dat mezi tabulkami příkazem *INSERT ... SELECT* (je možné přitom podmínkami určitou podmnožinu dat odfiltrovat) anebo modifikací dat přímo v souborech v datovém úložišti externími nástroji, jako je Apache Hive, MapReduce, Spark,

Nejjednodušší variantou příkazu *INSERT* je *INSERT ... VALUES*. Ten vkládá do zadané tabulky hodnoty přímo specifikované v části *VALUES*. Každé zavolání tohoto příkazu ale vytvoří nový datový soubor (ve formátu definovaném ve schématu tabulky) v úložišti, což je z hlediska operací nad daty velice neefektivní a tento způsob je vhodný

spíše pro testovací účely. Jedinou výjimkou z tohoto pravidla jsou tabulky uložené v HBase - viz podkapitola 4.4.3.

Pro produkční používání je určena varianta *INSERT ... SELECT*, která vkládá data načtená z jiné existující tabulky v Impale a jejím výsledkem je jeden datový soubor v úložišti. Používá se k přeuložení dat v jiném formátu, ke sloučení více datových souborů do jednoho (kompaktace) anebo k filtraci dat tabulky.

Pro vložení nových dat do Impaly je určen příkazd *LOAD*, který vkládá data na úrovni vrstvy datového úložiště. Tento příkaz přesune (pozor, nekopíruje - opravdu přesune!) zadaný zdrojový soubor s daty uložený v HDFS do interní složky Impaly. Alternativně je možné data vůbec nepřesouvat do složek Impaly, ale využít místo toho příkaz *CREATE EXTERNAL TABLE*, která pracuje s datovými soubory přímo v definované složce v HDFS. Při každé změně souborů v ní je ale nutné příkazem *REFRESH* uvědomit Impalu o existenci nových datových souborů.

Pro vkládání dat pomocí příkazů *INSERT* i *LOAD* platí, že je vhodné při každém větším vložení zavolat příkaz *COMPUTE STATS*.

U DML příkazů je ještě vhodné zmínit transakční zpracování dat. Impala transakce nepodporuje, takže pokud dojde při vkládání k chybě, může zůstat zapsána jen část dat.

4.6.4 Příkaz SELECT

SELECT je nejdůležitějším příkazem v Impale, protože umožňuje to, k čemu byla Impala navržena: provádět analýzu dat. Možnosti příkazu *SELECT* jsou veliké a podrobně jsou popsány v manuálu [19], zde je uveden jen výčet možností, které příkaz *SELECT* nabízí:

- WHERE podmínky
- agregace dat, GROUP BY
- DISTINCT pro unikátní hodnoty
- JOIN více tabulek
- ORDER BY, LIMIT
- UNION pro spojování výsledků
- pohledy (VIEW)
- CAST a další integrované funkce
- nekorelované poddotazy

4.6.5 Další SQL příkazy v Impale

Přehled dalších SQL příkazů nespadajících do výše popsaného rozdělení:

- SET - umožňuje nastavit systémové proměnné, např. interní velikost bloku Parquet souboru
- SHOW - zobrazí informace o tabulce, partitions tabulky, pohledu nebo dalších entitách
- DESCRIBE - zobrazí stručné informace o tabulce
- EXPLAIN - vypíše exekuční plán zadaného dotazu

4.7 Budoucnost projektu: Apache Kudu a Arrow

Kromě zvyšování stability a výkonu jde vývoj Impaly směrem ke zjednodušování práce na všech úrovních. Aby byla Impala jednoduše použitelná, je nutné umožnit zápis dat přímo přes Impalu a ne pomocí externích nástrojů. Kromě toho je současným trendem ve zpracování velkých dat přesouvat data na úložiště s co nejrychlejším čtením a zápisem - tedy z plotnových disků na SSD disky a nejlépe rovnou do operační paměti. Z těchto dvou důvodů vzniklo Apache Kudu. [44]

Kudu je kompromisem mezi čistě souborově orientovaným, pro sekvenční čtení dat vhodným, distribuovaným souborovým systémem (HDFS) a pro přímé přístupy vhodným HBase úložištěm. Jde o samostatné úložiště, které uchovává sloupcově uložená data v relačních tabulkách a ty dělí na menší části nazvané tablety (*tablets*), které umožňují paralelní uložení i zpracování dat. Záznamy jsou pak uloženy v tabletu definovaném na základě rozsahu nebo hashe primárního klíče. [25]

Hlavním přínosem Kudu pro Impalu bude umožnění modifikace dat přímo pomocí příkazů *UPDATE* a *DELETE*. Kudu je ale na Impale nezávislý projekt a vzhledem ke své univerzální architektuře je vhodný pro širokou škálu aplikací od *OLAP* a v určitých případech i pro *OLTP*. [25]

Kudu vzniklo v roce 2015 ve firmě Cloudera a podobně jako samotná Impala bylo zařazeno do Apache inkubátoru, ze kterého se v červenci 2016 úspěšně dostalo mezi hlavní Apache aplikace. [27] V září 2016 vyšla první stabilní verze. Podpora Kudu byla přidána do Impaly 2.7 a pro jeho používání stačí mít nainstalované Kudu – pak je možné jednoduše definovat úložiště konkrétní tabulky pomocí definice *STORED AS KUDU*. [26]

Další technologií, která se nabízí pro využití v Impale, je Apache Arrow. Jde o standard pro uložení sloupcových dat v paměti tak, aby mohla být efektivně kešována

a zpracovávána *SIMD* instrukcemi na moderních procesorech [1]. Díky tomuto formátu bude v budoucnosti možné načítat data přímo z Parquet souborů nebo z Kudu do paměti bez dodatečných operací, což přinese další zrchlení zpracování dat. Ačkoliv je tento projekt poměrně nový (vznikl začátkem roku 2016 [4]), deklaruje podporu z dalších projektů, mimo jiné od Cassandry, Drillu, HBase, Phoenixu, Sparku, Stormu a dalších. [1]

5 Další technologie pro datové sklady

5.1 Apache Hive

Apache Hive je engine pro manipulaci s daty v Hadoopu pomocí jazyka HiveQL, který je kompatibilní s SQL-92. Využívá se pro dávkové zpracování dat a především pro jejich dotazování.

Apache Hive bývá někdy považován za předchůdce Impaly, což není úplně přesné označení, nicméně Impala je s Hivem v mnoha ohledech kompatibilní a využívá jej k uložení svých metadat. Nové verze Hivu (počínaje 0.13) přinesly zásadní změny a srovnaly jeho výkonnost s Impalou, takže nyní lze Hive chápat spíše jako přímého konkurenta Impaly.

5.1.1 Dotazovací jazyk

HiveQL jazyk je kompatibilní s MySQL-92 a také s SQL jazykem Impaly. Výhodou toho přístupu je jednoduchost - zadávat SQL (resp. HiveQL) dotazy je mnohem jednodušší než psát specializované MapReduce programy a umožňuje to specifikovat vlastní dotazy širšímu okruhu lidí, například datovým analytikům.

Oproti Impale umí Hive data nejen číst, ale také je modifikovat pomocí SQL příkazů *INSERT*, *UPDATE* a *DELETE*. Někdy se této schopnosti využívá i v Impale - data se modifikují Hivem a opětovně čtou v Impale.

5.1.2 Architektura

Dotazy z Hivu jsou překládány do aplikací některého z běhových prostředí v Hadoopu - do MapReduce, Tez nebo Sparku a ty pak zpracovávají datové soubory uložené v Hadoopu. Do verze 0.13 podporoval Hive jen MapReduce, což znamenalo pro každý dotaz čekat na přidělení výpočetních prostředků v Hadoop clusteru a způsobovalo to pomalejší odezvu na dotazy. Novější verze Hivu podporují Tez a Spark, které umožňují držet alokované prostředky a kromě toho výrazně zrychlily dotazy také díky lepší paralelizací výpočtů [47].

Data v Hivu jsou podobně jako v Impale členěna do databází a relačních tabulek. Schéma tabulek musí být definováno příslušným HiveQL dotazem a je uchováno v tzv. Hive Metastore (stejným jako v Impale - viz 4.3.4). Vlastní data jsou pak fyzicky uložena jako soubory v HDFS. Hive pro ně nevyužívá žádný vlastní datový formát, ale naopak umožňuje pracovat s formáty existujícími: prostým textem (CSV nebo s libovolným oddělovačem; soubory můžou být komprimovány [9]), JSON, XML, sekvenčními soubory (*sequence files*), Parquet, RC File, ORC File a také s daty uloženými v HBase (podrobněji v [42, s. 199]). Pokud už tedy v Hadoopu data v některém z těchto for-

mátů máte, pak může zcela odpadnout proces ETL a Hive je možné začít používat bez dalších změn v systému.

5.1.3 Iniciativa Stinger

V roce 2013 vznikla ve firmě Hortonworks iniciativa Stinger [43], která měla za cíl výrazně zrychlit Hive. Ten tehdy podporoval pouze provádění dotazů pomocí MapReduce a při porovnání s Impalou (v tu dobu nově vzniklou) byl výrazně pomalejší. Stinger postupně přinesl do nových verzí Hivu podporu Tez a Sparku, LLAP (*Live Long and Process*, démon pro znovupoužívání alokovaných prostředků v Hadoopu) a podporu optimalizovaného sloupcového formátu ORC (*Optimized Row Columnar*; jeho princip je velice podopný formátu Parquet). Díky tomu nabízí v současné době Hive rychlost zpracování dotazů srovnatelnou s Impalou. Vítězem vzájemného porovnání bývá produkt, který vyvíjí firma provádějící daný test - u Hortonworks tedy vítězí Hive [2] a u Cloudery Impala [3].

5.2 Druid

Druid je sloupcově orientovaná open-source databáze, která je stejně jako Impala a další technologie založena na Dremelu. Je orientována na časově založená data a umožňuje snadné a efektivní přidávání nových dat.

Druid sestává z několika samostatných komponent (Realtime node načítají nová data, Coordinator node monitorují stav Druid clusteru, Historical node uchovávají archivované segmenty dat, Broker node obsluhují dotazy klientů a Indexer node zpracovávají nová data na Realtime node) a podporuje různá úložiště dat (HDFS, S3 nebo lokální) a různá úložiště metadat (MySQL, Derby).

Jednou z hlavních výhod Druida je schopnost zpracovávat v reálném čase streamovaná data, které mohou přicházet ze streamovacích platforem jako jsou Apache Kafka, Apache Samza anebo Apache Storm. Nově příchozí data jsou uchovávána v jiné formě (vhodné pro rychlé zápisy), než později v archivační části.

Druid nativně nabízí HTTP REST API a JSON rozhraní a za použití externích knihoven také omezenou podporu SQL.

5.3 Infobright

Infobright je komerční databáze integrovaná s MySQL. Disponuje vlastní vrstvou pro sloupcové uložení dat a vrstvou pro optimalizaci dotazů.

Infobright nabízí SQL rozhraní, efektivní kompresi dat uložených ve sloupcovém formátu a výkonné nástroje pro import dat. Místo vytváření indexů nad daty používá Infobright automatickou extrakci metadat pro optimalizaci dotazů. Infobright existuje

ve verzi *ICE* (*Infobright Community Edition*), jejíž vývoj ale v posledních letech prakticky neprobíhá [24] a pak komerční *IEE* (*Infobright Enterprise Edition*). [23]

5.4 Apache Kylin

Apache Kylin je open-source nástroj pro analýzu extrémně velkých dat – řádově petabytů. Vznikl původně ve firmě eBay, ale od prosince 2015 patří mezi top-level Apache projekty. [28].

Kylin využívá databázi Hive pro uložení dat a z ní automaticky generuje předagregované tabulky (OLAP kostky) do databáze HBase. Díky tomu i nad velkými daty dokáže vracet výsledky za dobu kratší než jedna sekunda. Kylin také nabízí podporu SQL jazyka. [28]

II. ANALYTICKÁ ČÁST

6 Požadavky na nový datový sklad Sklik.cz

Současné řešení datového skladu v MySQL popsané v kapitole 2 je nevyhovující ze dvou hlavních důvodů. Prvním je velmi dlouhá doba odezvy na některé dotazy - pokud je dotaz prováděn nad delším časovým obdobím nebo nad více dimenzemi, dochází k vypršení časového limitu webového rozhraní a není vrácen žádný výsledek. Celý systém je tak pro náročnější dotazy nepoužitelný.

Druhým důvodem je velikost databáze, která už se přibližuje k hardwarovým limitům stroje, na kterém běží. Část dat byla zkomprimována pomocí *myisampack* (viz 3.3), ale do budoucna je tento stav neudržitelný.

Požadavky na nové datové úložiště by se daly shrnout do těchto základních bodů:

- nutnost zvládat dotazování nad velkým objemem dat - stovky GB až TB
- dobrá škálovatelnost pro nová data - kromě nových dat přibývajících do systému bude v budoucnosti třeba přidat nové dimenze, které velikost dat znásobí
- vysoká dostupnost dat - odolnost vůči výpadku hardwaru, na kterém databáze běží
- možnost asynchronní replikace mezi serverovnami - v budoucnu bude třeba mít možnost provozovat dvě nezávislé instance datového skladu a zajistit mezi nimi synchronizaci dat
- odezva na dotazy v řádu sekund
- dotazovací jazyk pokud možno kompatibilní s SQL - díky němu bude možné bez velkých úprav napojit na datový sklad existující statistické servery
- open-source bezplatné řešení - Seznam využívá téměř výhradně bezplatný open-source software a proto by měl i nový datový sklad být pod takovou licencí

6.1 Porovnání vhodností uvažovaných řešení

Následující tabulky přináší porovnání vhodnosti jednotlivých technologií vzhledem k požadavkům Sklik.cz, které byly popsány v minulé podkapitole. Vlastnosti systémů jsou s ohledem na jejich popis v úvodní části práce pouze vypsány bez hlubšího vysvětlování.

6.1.1 Datový sklad v MySQL

Argumentem pro zachování datového skladu v MySQL by bylo z hlediska Sklik.cz zachování plné kompatibility se stávajícím SQL rozhraním. Důležitým faktorem by

bylo i to, že jde o většině vývojářům známou technologii. Oproti mnohým dalším technologiím pro datové sklady také nabízí replikaci mezi serverovnami.

Jejím hlavním problémem jsou ale výkonnostní problémy vyplývající a špatná škálovatelnost.

U MySQL by se sice dalo provést několik optimalizací probraných v kapitole 3.2, jako je partitioning, ale MySQL nebude nikdy plnohodnotným datovým skladem vzhledem k tomu, že data vždy ukládá a zpracovává po řádcích a také proto, že nepodporuje nezávislé paralelní zpracování dat.

6.1.2 Apache Impala

Apache Impala byla vybrána jako nejvhodnější kandidát na nový datový sklad pro Sklik.cz. Její hlavní výhodou oproti ostatním technologiím je rychlost dotazů (aktuální srovnání z dubna 2017 je možné nalézt v [12]), lineární škálovatelnost a integrace s Hadoop ekosystémem. Dobrá podpora SQL jazyka zajišťuje kompatibilitu se starým řešením i snadné psaní ad-hoc dotazů přes webové rozhraní Hue (viz 7.4.5). Další výhodou Impaly je umožnění efektivního spouštění více souběžných dotazů.

Požadovaná replikace dat mezi serverovnami může být u Impaly vyřešena na úrovni vrstvy úložiště (HDFS).

Nevýhodou Impaly je nutnost zapisovat data ve větších dávkách a nemožnost je poté editovat. Tento nedostatek není z hlediska Sklik.cz velký a v nové verzi 2.7 navíc přichází Impala s propojením s úložištěm Apache Kudu, které jej řeší.

6.1.3 Apache Hive

Apache Hive poskytuje lineární škálovatelnost, výbornou integraci s Hadoopem a také dobrou podporu SQL jazyka. Nabízí vysokou rychlost dotazů srovnatelnou s Apache Impalou, někdy dokonce i lepší [11]. Navzdory podobným vlastnostem je ale přece jen více určen pro datové vědce pro analyzování větších dat než Impala, která je spíše „tradiční“ analytickou databází.

Výhodou Hivu oproti Impale je možnost zapisovat data přímo do tabulek. Více důležitý je však pro Sklik.cz jeho nedostatek - není určen pro souběžnout práci více uživatelů, což by mohl být v reálném nasazení problém.

6.1.4 Apache Druid

Apache Druid byl jedním z vážných kandidátů na nový datový sklad vzhledem k tomu, že je robustní, data zpracovává paralelně a jde o rozšířenou a zralou technologii. Jeho výhodou je podpora pro vkládání nových dat, jejíž využití by zjednodušilo proces zpracování dat na Sklik.cz.

Druid ale není dostatečně flexibilní - je orientován na časovou dimenzi a nenabízí obecné možnosti pro dotazování dat – například nepodporuje příkaz JOIN nad více tabulkami. Další nevýhodou je omezená podpora SQL jazyka, která navíc není integrována přímo v systému.

6.1.5 Presto

Presto je architekturou i vlastnostmi podobná Apache Impale, ale z měření nad testovacími datasety vychází hůře – např. [32].

III. PROJEKTOVÁ ČÁST

7 Migrace datového skladu z MySQL do Apache Impala

Tato část popisuje vlastní provedení přechodu datového skladu Sklik.cz z MySQL na technologii vybranou v předchozí kapitole, tedy na Apache Impalu.

Výměna technologií zahrnovala jak přípravu nové databáze (instalace, návrh schématu pro data, vyřešení přidávání nových dat), tak samotnou migraci dat a také úpravu komponent, které s datovým skladem pracují (rozhraní pro komunikaci a úprava SQL dotazů).

7.1 Instalace Apache Impaly

Tato podkapitola má za cíl popsat rámcově instalaci Apache Impaly na provozní stroje určené pro provoz datového skladu Sklik.cz.

7.1.1 Příprava strojů pro provoz Impaly

Provozní instalaci Impaly byly vyhrazeny čtyři stroje. Konfigurace každého z nich obsahuje:

- 2×procesor Intel Xeon E5-2630L v3 (1,8 GHz, 8 jader)
- 128 GB RAM (8×16 GB, 2133 MHz)
- 4×SSD pevný disk (1,2 TB)
- 1 Gbit síťová karta

Na strojích byl nainstalován operační systém Debian Wheezy a Hadoop z distribuce od Cloudery, konkrétně ve verzi CDH 5.3. Pro instalaci balíků od Cloudery byl použit veřejný debian repozitář této firmy.

Výše uvedené stroje byly určeny pro běh Impala serveru (démonů), které budou vykonávat veškeré výpočty nad daty a vyžadují tedy výkonný hardware. Kromě nich byly ještě vytvořeny pomocí virtualizace OpenVZ na jiných fyzických strojích další virtuální servery:

- jeden pro instalaci a spouštění importu dat a dalších případných MapReduce programů
- dva pro HDFS Namenody a pro YARN Resource Managery
- jeden historyserver pro YARN
- tři pro Zookeeper quorum (služba potřebná pro YARN a HDFS High Availability)

7.1.2 Instalace Hive Metastore

Impala vyžaduje pro práci s metadaty Hive Metastore. Ten byl nainstalován z debian repozitáře Cloudery na jeden ze čtyř strojů určených pro Impala servery a jako úložiště používá lokální MySQL databázi. V ní bylo třeba vytvořit uživatele a nainicializovat tabulky pro Hive, dále pak nainstalovat java ovladač pro MySQL a nakonfigurovat Hive, aby jeho Metastore mohl přistupovat do připravené MySQL databáze. Podrobný popis zprovoznění Hive Metastore je možné najít na stránkách Cloudery. [5].

7.1.3 Instalace a konfigurace Impaly

Impala server (démon) byl nainstalován na všechny čtyři připravené servery z debian balíku impala-server získaného z repozitáře Cloudery. Kromě něj bylo na tyto uzly nainstalováno klientské rozhraní *impala-shell*. Na jeden ze čtyř Impala strojů byl nainstalován Impala Statestore démon a také Catalog démon (z debian balíků *impala-state-store* a *impala-catalog*).

Každý Impala server je třeba dále nastavit upravením konfiguračních souborů. V souboru */etc/impala/conf/hive-site.xml* nastavit adresu Impala Metastoru (*hive.metastore.uris*), povolit práci s metadaty HDFS bloků na datanodech (proměnná *dfs.datanode.hdfs-blocks-metadata.enabled* v */etc/hadoop/conf/hdfs-site.xml*), zkopírovat konfigurační soubory Hadoopu *core-site.xml* a *hdfs-site.xml* ze složky */etc/hadoop/conf* do */etc/impala/conf* a dále pak povolit čtení z lokálně uložench HDFS bloků na datanodech (proměnná *dfs.client.read.shortcircuit* a související *dfs.domain.socket.path* v souboru */etc/impala/conf/hdfs-site.xml*). Podrobnosti k instalaci lze nalézt na [18].

Po spuštění všech služeb Impaly a Hive metastoru na příslušných serverech (příkazy *service impala-<název služby> start*, resp. *service hive-metastore start*) by měla být instalace Apache Impaly plně funkční a připravená pro zadávání SQL dotazů přes *impala-shell*.

Ke správě Impala clusteru slouží webové rozhraní běžící defaultně na portu 25000 na každém stroji s Impala serverem.

7.2 Schéma databáze

Cílem migrace datového skladu bylo zachovat co nejvíce strukturu databáze. Výsledná podoba tabulek v Impale tedy odpovídá tabulkám v MySQL a je popsána v příloze 1. Oproti MySQL bylo nutné upravit v databázovém schématu datové typy, odstranit některé konstrukty nevyskytující se v Impale a naopak přidat některé nové.

7.2.1 Úprava datových typů sloupců

Číselné datové typy má Impala stejné jako MySQL. Pro uložení textových řetězců je v Impale preferován datový typ s variabilní délkou *STRING*, takže sloupce s MySQL typem *VARCHAR* bylo třeba změnit.

Sloupec pro uložení data (*day*) byl v původním schématu MySQL typu *DATE*, avšak Impala podporuje pouze jediný datový typ pro datum – *TIMESTAMP*.[21] Ten má velikost 16 bytů a je tedy výhodnější použít pro uložení data typ *STRING* a ISO formát data (např. "2017-04-20"), který zabere jen 10 bytů a lépe se s ním pracuje při ručním zadávání SQL dotazů. Varianta uložení pouze dne v měsíci ve sloupci *day* a využití dalších sloupců pro práci s datem (*year* a *month*) nebyla vybrána, protože by vyžadovala u většiny tabulek přidání sloupce *month*, zesložitila by psaní SQL dotazů a ani úspora při uložení sloupce by nebyla velká díky slovníkovému kódování dat v Parquet formátu. Při testování navíc tato varianta nepřinesla žádné měřitelné zrychlení dotazu.

7.2.2 Odstranění konstrukcí MySQL

Z DDL příkazů pro definici databáze bylo třeba odstranit Impalou nepodporované konstrukce pro definici klíčů *PRIMARY KEY* a *KEY*, s ním související deklarace *AUTO_INCREMENT* a také definice typu úložiště – *ENGINE*.

V Impale může být každý sloupec nedefinovaný, takže bylo navíc nutné odstranit nepodporované deklarace *NOT NULL* u sloupců.

7.2.3 Přidání konstrukcí specifických pro Impalu

Do příkazů pro vytváření tabulek bylo třeba přidat informaci o formátu uložení dat – *STORED AS PARQUET*.

Pro využití partitioningu v Impale, popsaného podrobně v kapitole 4.5, bylo potřeba definovat sloupce pro partitioning. Nejvhodnějším kandidátem byla časová dimenze dat, která je součástí prakticky každého dotazu na globální statistiky Sklik.cz. Sloupec *day* má však granularitu jeden den a pro partitioning se tak přímo nehodí, protože data z jednoho dne by byla příliš malá (v řádku KB nebo maximálně jednotek MB) pro samostatnou partition. Vhodnější bylo tedy definovat pro partitioning v tabulkách celý rok a pouze v tabulce *contextstat.user_daily*, která má největší velikost, navíc ještě měsíc. Tabulky *web_user* a *zone_web* jsou vazební a žádný partitioning nemají.

7.3 Migrace dat

V této podkapitole je popsán způsob, jakým byly do Impaly přemigrována všechna data z databáze MySQL s využitím čistě textového formátu. Tento formát byl pro migraci

nejvhodnější, protože do něj umí exportovat data jak starý datový sklad v MySQL, tak jej umí číst Apache Impala.

Pro migraci by kromě textového formátu mohl být použit také program Apache Sqoop, o kterém podrobněji pojednává následující podkapitola 7.4. Ten ale do operační paměti načítá všechna data ze zadaného *SELECT* dotazu, takže není vhodný pro přímou jednorázovou migraci velkých tabulek. Sice je možné jej postupně volat vícekrát a data migrovat v dávkách (např. po dnech nebo týdnech), avšak migrace s využitím textového formátu je přímočařejší a z hlediska vykonání i rychlejší.

Při migraci dat je využita možnost přímo z MySQL vyexportovat textový soubor TSV (Tab-Separated Values – tabulátorem oddělené hodnoty), který je pak nahrán do HDFS do adresáře pomocné tabulky Impaly. Z ní jsou následně data přes *impala-shell* překopírována SQL příkazem *INSERT ... SELECT* do cílové tabulky.

Přesný postup exportu dat s využitím textového souboru a pomocné tabulky v Impale je popsán v následujících odstavcích.

7.3.1 Provedení exportu dat z MySQL do textového souboru

Export byl proveden pomocí SQL příkazu zadaného v MySQL konzoli na libovolném stroji, odkud je přístup jak do databáze MySQL, tak do HDFS v Hadoop clusteru, kde běží Impala. Jako separátor byl vybrán tabulátor, výstupem tedy bude textový soubor TSV (Tab-Separated Values). Pro escapování znaků bylo použito zpětné lomítko a stejné musí být nastaveno později v tabulce Impaly. Dále je třeba nenastavovat uzavírání řetězců do uvozovek (*OPTIONALLY ENCLOSED BY '"'*), protože by to způsobilo problémy při načítání dat v Impale (ta by totiž uvozovky považovala za součást textového řetězce). V dotazu dále není třeba explicitně specifikovat formát uložení sloupců (FIELD) ani řádků (LINE).

Ukázka MySQL dotazu pro contextstat.user_daily, který uloží data celé tabulky (přibližně 70 GB) do adresáře */tmp*:

```
mysql > SELECT user_id, master_id, walletAgency, zone_id, `day`, clicks,
    impressions, avg_position, money, web_id, discarded_clicks,
    impression_money
FROM user_daily
INTO OUTFILE "/tmp/contextstat.user_daily.tsv"
FIELDS TERMINATED BY "\t" ESCAPED BY "\\";
```

7.3.2 Vytvoření dočasné tabulky v Impale

Dalším krokem v migraci je vytvoření dočasné tabulky v Impale, která má stejnou strukturu jako cílová tabulka, jen nemá nastavený partitioning a také má definovaný formát dat TSV.

```
impala-shell> CREATE TABLE tmp_user_daily (
   user_id BIGINT,
   master_id BIGINT,
   walletAgency BOOLEAN,
   zone_id BIGINT,
   `day` STRING,
   clicks BIGINT,
   impressions BIGINT,
   avg_position BIGINT,
   money BIGINT,
   web_id BIGINT,
   discarded_clicks BIGINT,
   impression_money BIGINT
)
ROW FORMAT DELIMITED FIELDS TERMINATED BY "\t" ESCAPED BY '\\'
STORED AS TEXTFILE;
```

7.3.3 Vložení dat do HDFS adresáře Impaly

Data vyexportovaná z MySQL nyní můžeme nahrát do HDFS přímo do složky určené pro data dočasné tabulky *tmp_ user_ daily*. Volání využívá standardního HDFS příkazu *hdfs dfs*.

```
hdfs dfs -put /tmp/contextstat.user_daily.tsv
    /user/hive/warehouse/contextstat.db/tmp_user_daily
```

7.3.4 Zkonvertování dat do cílové tabulky v Impale

Data z dočasné tabulky vložíme do cílové tabulky do formátu Parquet pomocí SQL dotazu v *impala-shell*. Je v něm nutné specifikovat oba sloupce pro partitioning:

```
impala-shell> INSERT INTO user_daily
  PARTITION (`year`, `month`)
  SELECT *, YEAR(`day`), MONTH(`day`)
    FROM tmp_user_daily;
```

7.3.5 Kontrola dat v Impale a úklid dočasné tabulky

Po provedení předchozích kroků můžeme zkontrolovat data dotazem na Impala tabulku:

```
impala-shell> SELECT YEAR(`day`), MONTH(`day`), `day`, SUM(impressions),
    SUM(clicks)
  FROM tmp_user_daily
  WHERE day="2017-03-15";
```

Nakonec je vhodné odstranit dočasnou tabulku:

```
impala-shell> DROP TABLE tmp_user_daily;
```

7.4 Přidávání nových dat do Impaly

Kromě počáteční migrace dat, popsané v přechozí kapitole, bylo třeba vyřešit průběžné vkládání nových dat do datového skladu.

Apache Impala sice podporuje vkládání dat pomocí příkazu *INSERT INTO*, ten ale při každém zavolání vytváří nový datový soubor a není tedy vhodný pro reálné využití. Nejlepším způsobem přidávání dat je vytváření Parquet souborů externími nástroji.

7.4.1 Import pomocí Apache Sqoop

Během přechodného období, kdy je v provozu jak nový, tak starý datový sklad a ve starém se data stále aktualizují, bylo nejjednodušší možností využití nástroje Apache Sqoop (SQL-to-Hadoop). Je to nástroj pro import dat mezi relačními databázemi a Hadoopem, takže je pomocí něj možné načítat nová data ze starého skladu v MySQL a zapisovat je do formátu Parquet přímo do složek Impaly v HDFS. Sqoop je třeba spouštět v Hadoop clusteru kde běží Impala právě kvůli zápisu do HDFS. Sqoop dále ke svému běhu vyžaduje YARN, protože vytváří MapReduce joby.

Sqoop příkaz pro import jednoho dne z tabulky *contextstat.user_ daily* vypadá takto:

```
sqoop import
  -Dmapreduce.task.timeout=0
  -Dmapreduce.map.java.opts=-Xmx10500m
  -Dmapreduce.map.memory.mb=14336
  --connect "jdbc:mysql://source-db-server.cz:3306/db_name?autoReconnect=true"
  --driver com.mysql.jdbc.Driver
  --username db-username
  --password db-pass
  --query "SELECT *, YEAR(`day`) AS `year`, MONTH(`day`) AS `month` FROM
     user_daily WHERE `day` = '2017-03-15' "
  --map-column-java day=String
  --target-dir
     "/user/impala/warehouse/contextstat.db/user_daily/year=2017/month=3"
  --as-parquetfile -m 1
```

V příkazu je třeba kromě nastavení parametrů pro běh Javy (*-D* přepínače) specifikovat adresu zdrojového MySQL serveru a přístupové údaje k němu, dále pak výstupní formát Parquet (*–as-parquet*) a cestu do HDFS, kde Sqoop vytvoří Parquet soubory (*–target-dir*). Přepínač *-m* definuje počet paralelních map tasků a tedy i počet výstpních Parquet souborů. Kvůli poměrně malým souborům z jednoho dne je vhodné mít Parquet souborů co nejméně a proto byla nastavena hodnota na 1.

V parametru *query* je definován SQL dotaz pro získání dat z MySQL. Všechny sloupce z tohoto *SELECT* dotazu budou použity jako sloupce pro výstupní Parquet soubor, který bude mít datové typy odpovídající typům v MySQL. Je možné explicitně přemapovat datový typ pomocí přepínače *–map-column-java*, jako je to provedeno u sloupce *day* (u něj měníme typ z *DATE* na *STRING*). Pokud mapování typů nespecifikujeme, tak se použije defaultní mapování definované Sqoopem.

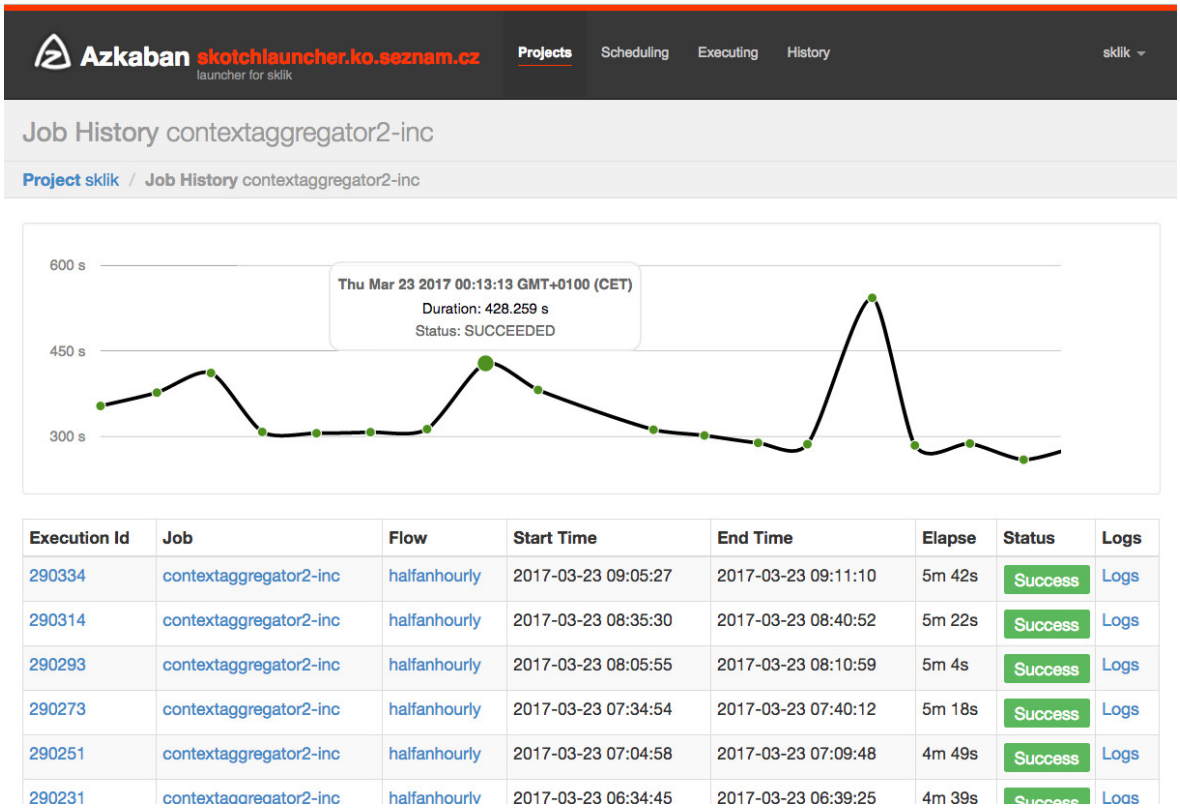
7.4.2 Import speciálním agregátorem

Po zrušení starého datového skladu bylo nutné napsat vlastní program (aggregátor), který vytváří přímo Parquet soubory. Tento program byl napsán v rámci firmy mimo

tuto diplomovou práci. Byl napsán v programovacím jazyku Scala a využívá Hadoop framework Spark (alternativa MapReduce). Zpracovává na vstupu statlogy (popsané v teoretické části v 2.2.2) a pomocí Spark SQL s nimi pracuje a výstup ukládá do Parquetu.

7.4.3 Automatické spouštění importu

Import nových dat do Impaly probíhá podobně jako u starého datového skladu každou půlhodinu. Kromě tohoto inkrementálního importu se každý den po půlnoci pouští „opravná“ přeagregace, která bere statlogy za celý předchozí den a přepíše tak výstupy jednotlivých inkrementálních běhů. Díky tomu se do datového skladu dostanou kompletně všechna data i v případě, že některý inkrementální běh v předchozím dni neproběhl úspěšně.



Obr. 7.1 Ukázka spouštění importů do Impaly pomocí Azkabanu

K pravidelnému spouštění inkrementálních i opravných denních importů v Seznamu používáme nástroj Azkaban. Ten umožňuje přes konfigurační soubory nebo přes grafické uživatelské rozhraní definovat programy (v terminologii Azkabanu *jobs*) a časy, ve které se mají automaticky spouštět (*schedules*). Azkaban ve svém webovém rozhraní umožňuje tyto joby pohodlně procházet a zobrazovat dodatečné informace o jejich běhu včetně výstupních logů spouštěných programů.

7.4.4 Inkrementální a denní importy

Sqoop i nové Spark agregátory vytváří nové Parquet soubory. Kromě toho je ale třeba zajistit spojování těchto souborů na straně Impaly, protože bez toho by datový sklad obsahoval velké množství malých (řádově stovky KB až jednotky MB) souborů, což je nevhodné z hlediska HDFS i zpracování v Impale.

Tento problém řeší Python skript *warehouse-feeder* vytvořený v rámci této diplomové práce. Ten vytváří na jednom (výpočetním) Hadoop clusteru Parquet soubory a nahrává je na Impala cluster. Jeho činnost se liší v závislosti na tom, jestli běží v inkrementálním nebo denním módu. Inkrementální mód lze zjednodušeně popsat následovně:

1. **zamkni zámek** na externím serveru (zabrání souběžnému spuštění více inkrementálních importů)
2. **Nastav** `last_run_time` = **čas posledního úspěšného běhu** – načte se z externího serveru
3. **Nastav** `now_time` = **aktuální čas**
4. **vytvoř dočasnou složku** pro importní nástroj
5. **zavolej importní agregátor nebo Sqoop** na data z intervalu `<last_run_time, now_time>`)
6. **přidej výsledný Parquet soubor do HDFS** na Impala cluster do složky aktuální partition
7. **ulož** `now_time` na externí server jako nový čas posledního úspěšného běhu
8. **odemkni zámek** na externím serveru

Inkrementální import umožňuje zpracovat data v rozmezí od svého posledního úspěšného běhu až do aktuálního času. Případné výpadky importu se tedy srovnají při dalším úspěšném importu. Při každém svém běhu vytváří nový datový soubor v příslušném adresáři partition Impaly v HDFS.

K inkrementálnímu běhu se váží dva drobné implementační detaily nezachycené v pseudokódu: při importu přes Sqoop je třeba importovat data vždy za celý příslušný den, protože data ve zdrojové MySQL mají nejnižší časovou granularitu jeden den. Druhým detailem je nutnost zaokrouhlit aktuální čas (který se použije jako koncový čas pro import) na nejbližší čas v minulosti, ke kterému byly kompletní všechny potřebné statlogy (v praxi jde o půlhodinu nebo hodinu).

Denní běh *warehouse-feederu* pro tabulku, která má partitions po rocích, lze popsat takto:

1. **zamkni zámek** na externím serveru (zabrání souběžnému spuštění více importů)
2. **vytvoř dočasnou složku** pro importní nástroj
3. **vytvoř dočasnou tabulku** `tmp_table` v Impale - bez partitioningu
4. **zavolej importní agregátor/Sqoop** na data z celého dne `imported_day`
5. **vlož výsledný Parquet soubor do HDFS** na Impala cluster do složky tabulky `tmp_table`
6. **zkopíruj data do dočasné tabulky** `tmp_table`: `INSERT INTO tmp_table * SELECT FROM orig_table WHERE` `` `year` `` `= YEAR(imported_day)AND` `` `day` `` `!= imported_day` (do dočasné tabulky se tak k importovanému dni přidají všechny ostatní dny z dotčené partition)
7. **odstraň dotčenou partition z původní tabulky**: `ALTER TABLE orig_table DROP PARTITION WHERE` `` `year` `` `= YEAR(imported_day)`
8. **vlož data zpět do původí tabulky**: `INSERT INTO orig_table * SELECT * FROM tmp_table` (tímto se fyzicky provede kompaktace – z dočasné tabulky se ze dvou soborů – pro importovaný den a zbytek dnů – stane jeden Parquet soubor v původní tabulce)
9. **odemkni zámek** na externím serveru

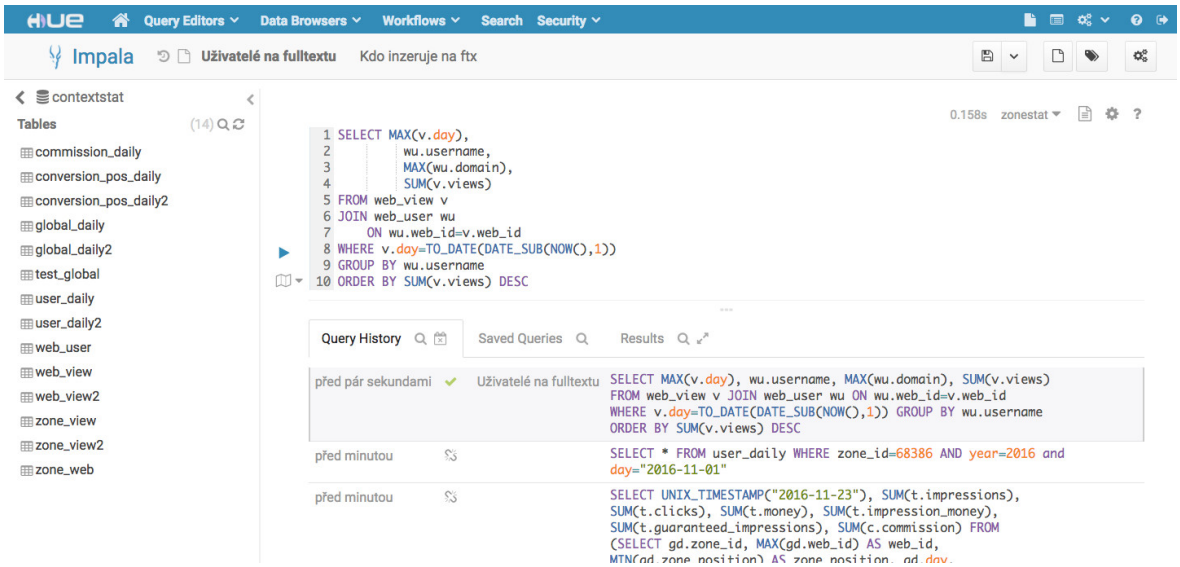
Klíčovou operací je při denním importu začlenění dat z naimportovaného dne do zbytku partition. K tomu se využívá dočasné tabulky, ze které se přelévají data celé partition zpět do původní tabulky příkazem `INSERT ... SELECT` a tím dochází ke kompaktaci dat do jednoho výsledného Parquet souboru.

7.4.5 Webový nástroj Hue pro práci s Impalou

Pro pohodlnější práci s Impalou byl na Hadoop clusteru nainstalován webový nástroj Hue (Hadoop User Experience). Ten kromě Impaly umí pracovat i s Hivem, MySQL databází, umožňuje přistpovat do HDFS a spravovat běžící MapReduce úlohy.

Při práci s Impalou umožňuje pomocí grafického rozhraní snadné zadávání SQL dotazů - umí napovídat názvy existujících sloupců, tabulek a funkcí a také pracovat s historií dotazů a uloženými dotazy. Výsledky dotazů umí zobrazovat přímo formou grafů.

Dále umožňuje v grafickém rozhraní procházet schémata tabulek, zobrazovat metadata, statistické informace a náhledy na data tabulek.



Obr. 7.2 Ukázka práce s webovým rozhraním Hue

7.5 Úprava SQL dotazů u komponent navázaných na datový sklad

Jedním z kritérií pro nový datový sklad Sklik.cz bylo zachování SQL kompatibility, protože do datového skladu přistupují další komponenty a získávají z něj data poměrně složitou logikou. Vybraná Apache Impala podporuje vlastní SQL jazyk, který je až na pár drobností kompatibilní s SQL-92 standardem.

Ačkoliv dotazy konstruované nad datovým skladem obsahují mnoho různých konstrukcí a poddotazů, v praxi bylo nutné provést jen tři typy úprav: přidat agregační funkce u některých sloupců, nahradit volání funkcí pro práci s časem a přidat podmínky pro partitioning.

7.5.1 Přidání agregačních funkcí ke sloupcům v klauzuli SELECT

Apache Impala narozdíl od MySQL databáze vyžaduje, aby každý z výrazů, které vrací dotaz *SELECT*, byl buď specifikován v *GROUP BY* klauzuli, anebo aby nad ním byla použita některá agregační funkce. [48, s. 115] Pokud tomu tak není, tak dotaz skončí s chybou *AnalysisException: select list expression not produced by aggregation output.*

V případě dotazů nad datovým skladem Sklik.cz bylo třeba z tohoto důvodu nejčastěji přepsat dotazy, které obsahovaly *web_id* při agregaci podle *zone_id*. Vzhledem k jejich vazbě 1:N (zóna tedy patří vždy k jednomu webu) a k tomu, že schéma da-

tabáze je denormalizované a záznamy obvykle obsahují oba tyto sloupce, byly dotazy pro MySQL psány zjednodušeně například takto:

```
SELECT web_id, zone_id, SUM(clicks)
FROM global_daily
GROUP BY zone_id;
```

Tento dotaz vrací celkový počet kliků pro každou zónu a zároveň k ní vrací i web, do kterého zóna patří. Sloupec *web_id* ale není součástí žádné agregační funkce (narozdíl od *clicks*), ani není uveden v klauzuli *GROUP BY* (narozdíl od *zone_id*).

Vzhledem k tomu, že *web_id* je pro zagregovanou skupinu řádků se stejným *zone_id* vždy stejné, je pro tento účel třeba v Impale použít např. agregační funkce *MAX()* nebo *MIN()* a přepsat dotaz do následující formy:

```
SELECT MAX(web_id), zone_id, SUM(clicks)
FROM global_daily
GROUP BY zone_id;
```

7.5.2 Nahrazení funkcí pro práci s časem

Další úpravou SQL dotazů bylo nahrazení některých funkcí pro práci s časem. Datum ve sloupci *day* bylo totiž v MySQL uloženo v datovém typu *DATE*, zatímco v Impale jako *STRING* (viz kapitola 7.2.1).

Impala narozdíl od MySQL nepracuje s interně nastavenými časovými zónami [13], což v kombinaci s tím, že aplikace generující SQL dotazy nad datovým skladem pracovaly s datem jako s číslem (Unix timestamp), způsobovalo nekompatibilitu. Například dotaz pro získání data z timestampu v MySQL při nastavené časové zóně pro Českou republiku:

```
SELECT * FROM global_daily
WHERE `day` >= DATE(FROM_UNIXTIME(1420066800));
```

je tedy nutné pro Impalu přepsat do tvaru:

```
SELECT * FROM global_daily
WHERE `day` >= TO_DATE(FROM_UTC_TIMESTAMP(CAST(1420066800 AS timestamp),
    'CET'));
```

Další nekompatibilitou při práci s datem byly funkce pro určení týdne v roce a čtvrtletí. MySQL nabízí v tomto ohledu širší paletu možností [29] a kvůli neexistujícím funkcím *YEARWEEK* a *QUARTER* v Impale [13] tak bylo nutné některé dotazy upravit. Například určení týdne v roce a čtvrtletí pro datum 1.3.2017 v MySQL:

```
SELECT
   YEARWEEK("2017-03-01", 3) AS yearweek,
   QUARTER("2017-03-01") AS quarter;
```

bylo nutné pro Impalu přepsat do tvaru:

```
SELECT
   CONCAT(CAST(IF(WEEKOFYEAR("2017-03-01") > DAYOFYEAR("2017-03-01"),
       YEAR("2017-03-01")-1, YEAR("2017-03-01")) AS string),
       LPAD(CAST(WEEKOFYEAR("2017-03-01") AS string), 2, '0')) AS yearweek,
   CEIL(CAST(EXTRACT("2017-03-01", "month") AS int)/3) AS quarter;
```

7.5.3 Přidání podmínek pro využití partitioningu

Po úpravách uvedených výše v této kapitole byly původní SQL dotazy plně funkční i v Impale. Pro optimální načítání dat v Impale je ale vhodné zahrnout do podmínek dotazů také sloupce s datem, které jsou využity k partitioningu (podrobněji popsaném v kapitole 4.5). Díky tomu bude Impala schopna načítat jen data z uvedených partitions a ostatní bude ignorovat.

Například dotaz na zjištění celkového počtu kliků za březen 2017 v tabulce *global_daily*, která má jeden sloupec pro partitioning *year*, je vhodné přepsat ze:

```
SELECT SUM(clicks) FROM global_daily
WHERE `day` >= "2017-03-01" AND `day` <= "2017-03-31";
```

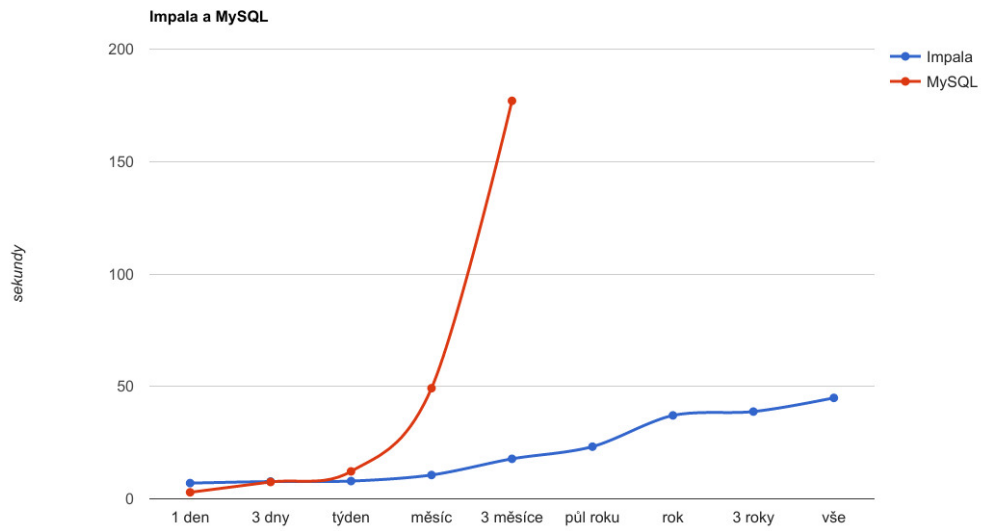
do tvaru:

```
SELECT SUM(clicks) FROM global_daily
WHERE `day` >= "2017-03-01" AND `day` <= "2017-03-31" AND
    `year`=YEAR("2017-03-01");
```

8 Porovnání starého a nového řešení

Díky současnému provozu starého a nového datového skladu bylo možné přímo porovnat vlastnosti obou řešení.

Nejmarkantnějším rozdílem je rychlost databáze. Měření nad SQL dotazy ukázala, že s rostoucím množstvím zpracovaných dat roste doba vracení odpovědi u Impaly lieárně, zatímco u MySQL exponenciálně. MySQL je v případě datového skladu Sklik.cz rychlejší pouze při načítání velmi malého objemu dat, například jednoho celého dne. Impala v takvém případ malého množství dat neumí využít, protože vždy zpracovává data z celé partition. Graf a tabulka 8.1 ukazují naměřené hodnoty z obou databází.



Obr. 8.1 Porovnání délky běhu SQL dotazů na globální statistiky ve starém a novém datovém skladu. Podrobnější popis měření je uveden u tabulky 8.1.

Dalším kritériem, které by mohlo být porovnáváno, je škálovatelnost. Impala by měla za předpokladu dostatečného hardwaru nabízet lineární škálovatelnost. [20] Prakticky byla tato vlastnost vyzkoušena na vývojovém clusteru provedením složitějšího agregačního dotazu nad tabulkou *contextstat.user_daily* při různém počtu aktivních strojů s běžícím Impala démonem. Ačkoliv nešlo o komplexní test, naměřený výsledek v tabulce 8.2 ukazuje lineární škálovatelnost Impaly.

Nový datový sklad je připraven na přidávání nových dimenzí. V takovém případě

Tab. 8.1 Srovnání délky běhu SQL dotazů nad starým a novým datovým skladem. Hodnoty jsou v sekundách.[1]

	MySQL	Impala
1 den	7,0	2,9
3 dny	7,7	7,5
týden	7,9	12,2
měsíc	10,6	49,2
3 měsíce	17,8	177
půl roku	23,2	-
rok	38,8	-
vše	44,9	-

Tab. 8.2 Délka běhu agregačního dotazu nad tabulkou *contextstat.user_daily* při různém počtu aktivních Impala serverů.[2]

počet uzlů	1 uzel	2 uzly	3 uzly	4 uzly	5 uzlů
délka běhu (s)	232	133	108	93	62

je třeba vytvořit novou tabulku a příkazem `INSERT ... SELECT` do ní zkopírovat data z původní tabulky. Už při počáteční migraci dat do Impaly bylo v plánu přidat jednu novou dimenzi (*device_type* - zařízení, kde se návštěvníkovi zobrazila reklama) s kardinalitou 4 a pro účely testování rozšiřitelnosti z ní pak byly odvozeny další dvě tabulky s novou dimenzí. Z výsledků testování nad tabulkou *contextstat.global_daily*, uvedených v tabulce 8.3, plyne, že se vzrůstajícím počtem záznamů v tabulce (např. přidání dimenze s kardinalitou 10 zdesetinásobí počet řádků tabulky) sice roste čas pro vrácení odpovědi na dotaz, ale nikoliv násobně. Přidání dimenze zařízení a testovací dimenze s kardinalitou 10 sice přibližně zvacetinásobilo počet dat v tabulce, ale testovaný dotaz měl průměrnou délku trvání zhruba jen dvojnásobnou.

Podobný test nemohl být bohužel nad starým datovým skladem vykonán vzhledem k tomu, že ten už bez přidání testovací dimenze narážel na hardwarové limity.

1 Hodnoty jsou průměrem z deseti měření nad provozním starým a novým datových skladem. Měření zahrnuje délku vykonání všech devíti dotazů, ze kterých se skládá výsledek zobrazovaný na interní webové stránce globálních statistik Sklik.cz.

2 Testováno na vývojovém Hadoop clusteru (5 strojů, každý s Intel Xeon CPU 3,3 GHz, 6 jader, 24 GB RAM, 150 GB SAS disk 15000 otáček), hodnoty jsou průměrem z deseti měření. Testovací dotaz: `SELECT MAX(user_id), MAX(zone_id), SUM(clicks), SUM(impressions)FROM user_daily WHERE YEAR = 2016 GROUP BY user_id, web_id ORDER BY user_id LIMIT 10;`

Tab. 8.3 Délka běhu agregačního dotazu nad tabulkami odvozenými z *contextstat.global_ daily* s vyšším počtem dimenzí.[1]

	velikost dat (MB)	počet nodů	řádků (mil.)	čas (s)
global_ daily	255	2	2	5,4
přidán *device_ type*	450	4	5	4,9
testovací dimenze (kard. 4)	1560	4	19	7,4
testovací dimenze (kard. 10)	3520	4	47	8,9

Kromě výše uvedené rychlosti a škálovatelnosti splnil nový datový sklad realizovaný pomocí Apache Impala všechny další požadavky zadané v kapitole 6 a předpoklady uvedené v 6.1.

8.1 Další možná rozšíření

V souvislosti s novým datovým skladem by v budoucnosti bylo možné zavést další úpravy a technologie, které by zrychlily dotazování anebo přinesly zjednodušení systému.

Pro zrychlení analytických dotazů nad datovým skladem by bylo možné v Impale vytvořit několik předagregovaných tabulek, které by se využívaly pro specifické dotazy (jde o přístup podobný Apache Kylin, popsanému v 5.4). Většina dotazů totiž požaduje ve výsledku agregovaná data a kdyby existovala tabulka s daty již předem agregovanými nad danou dimenzí, tak by mohl dotaz provést přímo nad ní a tedy nad menším objemem dat. Nevýhodou tohoto přístupu je složitost systému - bylo by nutné zajistit spolehlivé přeagregovávání dat do všech tabulek při každém zápisu dat. Dalším problémem by byl velký počet dimenzí, protože pokud bychom požadovali kombinace všech dimenzí, tak by počet tabulek rostl exponenciálně.

1 Testováno na provozním datovém skladu v Impale. Hodnoty jsou průměrem z deseti měření. Testovací dotaz pro jednu z tabulek: `SELECT SUM(clicks), SUM(impressions), SUM(impression_money), SUM(guaranteed_impressions)FROM global_daily WHERE year=2016 AND day >= "2016-12-01"AND day <= "2016-12-31"AND web_id > 1000 GROUP BY zone_id, feed_tag ORDER BY feed_tag DESC LIMIT 1;` Tabulka s novou dimenzí *device_ type* obsahovala přibližně dvojnásobek dat oproti původní (přidaná dimenze měla sice kardinalitu 4, ale u mnoha nově vzniklých záznamů se nevyskytly všechny čtyři možnosti). Testovací tabulky s novou dimenzí byly odvozeny už z tabulky obsahující *device_ type* a měly nastaveny všechny možné hodnoty nové dimenze pro každý záznam, takže obsahovaly 8×, resp. 20× více dat než původní tabulka. Zajímavým zjištěním v tomto testu byla skutečnost, že po přidání dimenze *device_ type* se dotazy mírně zrychlily. Je to způsobeno tím, že původní tabulka (resp. partition, protože Impala pracovala jen s daty partition pro daný rok) měla příliš málo dat a nemohl se tak využít paralelismus. Z tabulky je vidět, že byly využity jen dva ze čtyř Impala serverů, protože zpracovávaná data partition se vešla jen do dvou Parquet souborů o velikosti 128 MB a vyšší paralelismus tedy nebyl možný.

Ke zjednodušení logiky importu dat do datového skladu by přispělo ukládání dat do Apache Kudu (popsaného v kapitole 4.7) místo na HDFS. Při vkládání dat by pak nebylo třeba zajišťovat žádné operace pro kompaktaci dat, které nyní provádí *warehouse-feeder* popsaný v 7.4). Toto řešení by ale přineslo mírné zpomalení čtecích dotazů [44]. Při současné povaze importu dat, který v drtivé většině modifikuje pouze data jednoho posledního dne, tedy není využití Kudu výhodné. Přínosem by ale mohlo být v případě použití předagregovaných tabulek popsaných v předchozím odstavci, protože by umožnilo přímý zápis dat do všech tabulek.

Současným trendem, který se dotýká i datových skladů, je zpracovávání dat v reálném čase. Pokud by takový požadavek vznikl i pro datový sklad Sklik.cz a nebyla by k tomu použita specializovaná databáze, tak by vhodnou volbou opět mohlo být Apache Kudu.

ZÁVĚR

V této diplomové práci byla provedena analýza existujícího datového skladu pro globální statistiky reklamního systému Sklik.cz firmy Seznam.cz a bylo navrženo optimální řešení pro nový datový sklad.

V teoretické části byl představen současný datový sklad Sklik.cz a dále pak obecně datové sklady – jak z hlediska teoretického, tak konkrétně vybrané technologie. Podrobněji byly popsány databáze MySQL, Apache Impala a stručně pak i Apache Hive.

Nedostatky starého datového skladu Sklik.cz a požadavky na nové řešení byly rozebrány v analytické části. Dále v ní byly diskutovány jednotlivé technologie z hlediska vhodnosti pro Sklik.cz. Na základě této analýzy byla pro nový datový sklad zvolena Apache Impala.

Apache Impala nabízí paralelní zpracování dat a díky tomu dosahuje lineární škálovatelnosti, což je pro přidávání dimenzí v datovém skladu Sklik.cz důležité. Díky využití sloupcového formátu Apache Parquet a paralelního zpracování snižuje délku analytických dotazů, především těch spuštěných nad větším objemem dat. Dále poskytuje SQL rozhraní, které bylo kvůli zachování kompatibility se starým datovým skladem požadováno. Další výhodou Impaly je integrace do Hadoop ekosystému, ve kterém jsou i další součásti systému Sklik.cz. Výhodou Impaly oproti Apache Hive, který nabízí podobné vlastnosti, byla lepší výkonnost při současném běhu více dotazů.

Projektová část práce obsahuje popis procesu migrace dat do Apache Impala včetně struktury databáze, migračního nástroje a změn nutných na aplikační úrovni systému.

Během vytváření této diplomové práce došlo k reálnému nasazení nového datového skladu a projektová část díky tomu obsahuje přímé srovnání starého a nového řešení a zhodnocení přínosu nového řešení. Při testech nad datovými sklady byla zjištěna výrazně vyšší rychlost Impaly pro dotazy nad většími datovými objemy, což při praktickém využívání znamená odpovědi na dotazy nad delšími časovými obdobími v řádu jednotek a desítek sekund. Impala dále v souladu s předpoklady poskytuje dobrou škálovatelnost a možnost rozšíření databázového schématu o nové dimenze.

SEZNAM POUŽITÉ LITERATURY

[1] *Apache Arrow* [online]. [cit. 30. 4. 2017]. Dostupné z: https://arrow.apache.org/.

[2] *Impala Vs. Hive Performance Benchmark* [online]. [cit. 30. 4. 2017]. Dostupné z: http://hortonworks.com/blog/impala-vs-hive-performance-benchmark/.

[3] *New SQL Benchmarks: Apache Impala (incubating) Uniquely Delivers Analytic Database Performance* [online]. [cit. 30. 4. 2017]. Dostupné z: https://blog.cloudera.com/blog/2016/02/new-sql-benchmarks-apache-impala-incubating-2-3-uniquely-delivers-\analytic-database-performance/.

[4] *Introducing Apache Arrow: A Fast, Interoperable In-Memory Columnar Data Structure Standard* [online]. [cit. 30. 4. 2017]. Dostupné z: https://blog.cloudera.com/blog/2016/02/introducing-apache-arrow-a-fast-interoperable-in-memory-columnar-\data-structure-standard/.

[5] *Configuring the Hive Metastore* [online]. [cit. 30. 4. 2017]. Dostupné z: https://www.cloudera.com/documentation/enterprise/5-6-x/topics/cdh_ig_hive_metastore_configure.html.

[6] *Data Lake vs Data Warehouse: Key Differences* [online]. [cit. 30. 4. 2017]. Dostupné z: http://www.kdnuggets.com/2015/09/data-lake-vs-data-warehouse-key-differences.html.

[7] *Fact Table - Wikipedia* [online]. [cit. 30. 4. 2017]. Dostupné z: https://en.wikipedia.org/wiki/Fact_table.

[8] *Apache Hadoop at 10*, 2015. Dostupné z: http://www.slideshare.net/cloudera/apache-hadoop-at-10-59397028.

[9] *CompressedStorage - Apache Hive* [online]. [cit. 30. 4. 2017]. Dostupné z: https://cwiki.apache.org/confluence/display/Hive/CompressedStorage.

[10] *Impala Concepts and Architecture* [online]. [cit. 30. 4. 2017]. Dostupné z: http://www.cloudera.com/documentation/cdh/5-1-x/Impala/Installing-and-Using-Impala/ciiu_concepts.html.

[11] *Apache Hive Vs Apache Impala Query Performance Comparsion* [online]. [cit. 30. 4. 2017]. Dostupné z: https://hortonworks.com/blog/apache-hive-vs-apache-impala-query-performance-comparison/.

[12] *Apache Impala Leads Traditional Analytic Database* [online]. [cit. 30. 4. 2017]. Dostupné z: https://blog.cloudera.com/blog/2017/04/apache-impala-leads-traditional-analytic-database/.

[13] *Impala Date and Time Functions* [online]. [cit. 30. 4. 2017]. Dostupné z: https://www.cloudera.com/documentation/enterprise/5-8-x/topics/impala_datetime_functions.html.

[14] *What's New in Apache Impala (incubating)* [online]. [cit. 30. 4. 2017]. Dostupné z: http://www.cloudera.com/documentation/enterprise/5-5-x/topics/impala_new_features.html\#new_features_200_unique_1.

[15] *Do statestore and catalog have HA feature?* [online]. [cit. 30. 4. 2017]. Dostupné z: https://issues.cloudera.org/browse/IMPALA-2702.

[16] *Using Impala to Query HBase Tables* [online]. [cit. 30. 4. 2017]. Dostupné z: http://www.cloudera.com/documentation/archive/impala/2-x/2-0-x/topics/impala_hbase.html.

[17] *What's New in Apache Impala (incubating)* [online]. [cit. 30. 4. 2017]. Dostupné z: http://blog.impala.io/whats-new-12-013-2015.html.

[18] *Impala Installation* [online]. [cit. 30. 4. 2017]. Dostupné z: https://www.cloudera.com/documentation/enterprise/latest/topics/impala_install.html.

[19] *SELECT Statement* [online]. [cit. 30. 4. 2017]. Dostupné z: http://www.cloudera.com/documentation/archive/impala/2-x/2-1-x/topics/impala_select.html.

[20] *Cluster Sizing Guidelines for Impala* [online]. [cit. 30. 4. 2017]. Dostupné z: https://www.cloudera.com/documentation/enterprise/5-8-x/topics/impala_cluster_sizing.html.

[21] *Impala TIMESTAMP Data Type* [online]. [cit. 30. 4. 2017]. Dostupné z: https://www.cloudera.com/documentation/enterprise/5-8-x/topics/impala_timestamp.html.

[22] *What's New in Apache Impala (incubating)* [online]. [cit. 30. 4. 2017]. Dostupné z: http://www.cloudera.com/documentation/enterprise/release-notes/topics/impala_new_features.html.

[23] *Upgrading Your Row-based Database to a Columnar Database in Infobright* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://www.infobright.org/resources/Upgrading-to-Infobright-from-Row-based-Databases.pdf`.

[24] *Latest ICE Release Notes* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://www.infobright.org/index.php/Resources/release\_notes/`.

[25] *Introducing Apache Kudu* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://kudu.apache.org/docs/`.

[26] *Using Apache Kudu with Apache Impala (incubating)* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://kudu.apache.org/docs/kudu\_impala\_integration.html`.

[27] *Kudu Project Incubation Status* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://incubator.apache.org/projects/kudu.html`.

[28] *Apache Kylin* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://kylin.apache.org/`.

[29] *MySQL Date and Time Functions* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://dev.mysql.com/doc/refman/5.5/en/date-and-time-functions.html`.

[30] *myisampack - MySQL Reference Manual* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://dev.mysql.com/doc/refman/5.7/en/myisampack.html`.

[31] *PARQUET_FILE_SIZE* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://www.cloudera.com/documentation/archive/impala/2-x/2-1-x/topics/impala\_parquet\_file\_size.html`.

[32] *Presto* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://prestodb.io/`.

[33] *Seznam.cz* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://cs.wikipedia.org/wiki/Seznam.cz`.

[34] *Měření úspěšnosti - Sklik nápověda* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://napoveda.sklik.cz/mereni-uspesnosti/`.

[35] *Nápověda reklamního systému Sklik* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://napoveda.sklik.cz/`.

[36] *Designing Scalable Data Warehouse Using MySQL* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://www.slideshare.net/vanuganti/designing-scalable-data-warehouse-using-mysql`.

[37] *Bussiness Intelligence* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://en.wikipedia.org/wiki/Business\_intelligence`.

[38] *Data warehouse* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://en.wikipedia.org/wiki/Data\_warehouse`.

[39] *HOLAP* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://en.wikipedia.org/wiki/HOLAP`.

[40] *MOLAP* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://en.wikipedia.org/wiki/MOLAP`.

[41] *ROLAP* [online]. [cit. 30. 4. 2017]. Dostupné z: `https://en.wikipedia.org/wiki/ROLAP`.

[42] Capriolo, E. – Wampler, D. – Rutherglen, J. *Programming Hive.* O'Reilly Media, Inc., 1. edition, 2012. 328 s. ISBN 978-1-449-31933-5.

[43] Gates, A. *The Stinger Initiative: Making Apache Hive 100 Times Faster* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://hortonworks.com/innovation/stinger/`.

[44] Gutow, A. *Introducing Kudu: The New Hadoop Storage Engine for Fast Analytics on Fast Data* [online]. 2015. [cit. 30. 4. 2017]. Dostupné z: `https://vision.cloudera.com/introducing-kudu-the-new-hadoop-storage-engine-for-fast-analytics-\on-fast-data/`.

[45] James, R. *Data Warehouse in MySQL* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://mysql.rjweb.org/doc.php/datawarehouse`.

[46] Kornacker, M. – Erickson, J. *Cloudera Impala: Real-Time Queries in Apache Hadoop, For Real* [online]. 2012. [cit. 30. 4. 2017]. Dostupné z: `http://blog.cloudera.com/blog/2012/10/cloudera-impala-real-time-queries-in-apache-hadoop-for-real/`.

[47] Posthuma, J. P. *Hadoop from Hive with Stinger to Tez* [online]. [cit. 30. 4. 2017]. Dostupné z: `http://www.slideshare.net/janpieterposthuma/hadoop-from-hive-with-stinger-to-tez`.

[48] Russel, J. *Getting Started with Impala.* O'Reilly Media, Inc., 1. edition, 2014. 107 s.

[49] Schwartz, B. – Zaitsev, P. – Tkachenko, V. *High Performance MySQL: Optimization, Backups, and Replication.* O'Reilly Media, Inc., 3. edition, 2012. 826 s. ISBN 1449314287, 9781449314286.

[50] Shanklin, C. *ORC File In HDP 2: Better Compression, Better Performance* [online]. [cit. 30. 4. 2017]. Dostupné z: http://hortonworks.com/blog/orcfile-in-hdp-2-better-compression-better-performance/.

[51] Swanhart, J. *Using Flexviews – part one, introduction to materialized views* [online]. 2011. [cit. 30. 4. 2017]. Dostupné z: https://www.percona.com/blog/2011/03/23/using-flexviews-part-one-introduction-to-materialized-views/.

[52] Swanhart, J. *Data mart or data warehouse? (Percona blog)* [online]. 2010. [cit. 30. 4. 2017]. Dostupné z: https://www.percona.com/blog/2010/07/15/data-mart-or-data-warehouse/.

[53] White, T. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 4. edition, 2015. 756 s. ISBN 978-1-491-90163-2.

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. STRUKTURA DATABÁZE V IMPALE

```
-- SQL DDL for Impala DB contextstat
CREATE DATABASE contextstat;
USE contextstat;
CREATE TABLE commission_daily (
   zone_id BIGINT COMMENT 'zone ID',
   `day` STRING COMMENT 'The stats are aggregated for single days',
   ad_type TINYINT COMMENT 'Ad type',
   feed_tag TINYINT COMMENT 'Feed tag',
   commission BIGINT COMMENT 'commission for the zone'
)
PARTITIONED BY (
   `year` BIGINT
)
STORED AS PARQUET;
CREATE TABLE conversion_pos_daily (
   conversion_id BIGINT COMMENT 'Conversion ID',
   keyword_id BIGINT COMMENT 'Keyword ID',
   ad_id BIGINT COMMENT 'Ad ID',
   `day` STRING COMMENT 'The conversions are aggregated for single days',
   user_id BIGINT COMMENT 'User ID',
   campaign_id BIGINT COMMENT 'Campaign ID',
   group_id BIGINT COMMENT 'Group ID',
   web_id BIGINT COMMENT 'Web ID',
   zone_id BIGINT COMMENT 'Zone ID',
   zone_position BIGINT COMMENT 'Zone position',
   ad_type TINYINT COMMENT 'Ad type',
   feed_tag TINYINT,
   `value` BIGINT COMMENT 'Aggregated value of the conversions (money or
      whatever)',
   conversions BIGINT COMMENT 'The number of conversions',
   transactions BIGINT COMMENT 'The number of transactions'
)
PARTITIONED BY (
   `year` BIGINT
)
STORED AS PARQUET;
CREATE TABLE global_daily (
   web_id BIGINT COMMENT 'Web ID',
   zone_id BIGINT COMMENT 'Zone ID',
   zone_position BIGINT COMMENT 'Zone position',
   `day` STRING COMMENT 'The stats are aggregated for single days',
   ad_type TINYINT COMMENT 'Ad type',
   feed_tag TINYINT COMMENT 'Feed tag',
   clicks BIGINT COMMENT 'The number of clicks',
   impressions BIGINT COMMENT 'The number of impressions',
   money BIGINT COMMENT 'The cost of the clicks, in hellers',
   discarded_clicks BIGINT COMMENT 'Number of discarded clicks from column
      "clicks"',
   impression_money BIGINT COMMENT 'The cost of the impressions, in hellers',
   guaranteed_impressions BIGINT COMMENT 'Guaranteed_impressions'
)
PARTITIONED BY (
   `year` BIGINT
)
STORED AS PARQUET;
CREATE TABLE user_daily (
   user_id BIGINT COMMENT 'User ID',
   master_id BIGINT,
   walletAgency BOOLEAN,
   zone_id BIGINT COMMENT 'Zone ID',
   `day` STRING COMMENT 'The stats are aggregated for single days',
   clicks BIGINT COMMENT 'The number of clicks',
   impressions BIGINT COMMENT 'The number of impressions',
   avg_position BIGINT COMMENT 'The average position * 100',
   money BIGINT COMMENT 'The cost of the clicks, in hellers',
   web_id BIGINT COMMENT 'Web ID',
   discarded_clicks BIGINT COMMENT 'Number of discarded clicks from column
      "clicks"',
   impression_money BIGINT COMMENT 'The cost of the impressions, in hellers'
)
PARTITIONED BY (
   `year` BIGINT,
   `month` BIGINT
)
STORED AS PARQUET;
```

```
CREATE TABLE web_user (
   web_id BIGINT COMMENT 'Web ID',
   user_id BIGINT COMMENT 'User ID',
   username STRING COMMENT 'Username',
   domain STRING COMMENT 'Domain',
   destination_id TINYINT COMMENT 'Ad destination type id',
   is_szn_web TINYINT COMMENT 'is seznam web',
   web_domain STRING COMMENT 'web domain'
)
STORED AS PARQUET;
CREATE TABLE web_view (
   web_id BIGINT COMMENT 'web ID',
   `day` STRING COMMENT 'The stats are aggregated for single days',
   views BIGINT COMMENT 'web views',
   views_with_result BIGINT COMMENT "Pageviews with displayed ad."
)
PARTITIONED BY (
   `year` BIGINT
)
STORED AS PARQUET;
CREATE TABLE zone_view (
   zone_id BIGINT COMMENT 'Zone ID',
   `day` STRING COMMENT 'The stats are aggregated for single days',
   views BIGINT COMMENT 'Zone views',
   ad_type TINYINT COMMENT 'Ad type',
   feed_tag TINYINT COMMENT 'Feed tag'
)
PARTITIONED BY (
   `year` BIGINT
)
STORED AS PARQUET;
CREATE TABLE zone_web (
   zone_id BIGINT COMMENT 'Zone ID',
   web_id BIGINT COMMENT 'Web ID'
)
STORED AS PARQUET;
```

Databáze *zonestat* je až na chybějící sloupce *ad_type*, *feed_tag* a *impression_money* stejná.