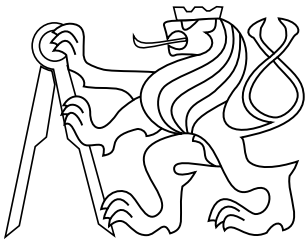
Bakalářská práce

Hybridní úložiště dat pro simulátor přenosové sítě

Ondřej Svoboda



Květen 2015

Vedoucí práce: Ing. Jan Zábojník

České vysoké učení technické v Praze
Fakulta elektrotechnická, Katedra řídicí techniky

České vysoké učení technické v Praze
Fakulta elektrotechnická

katedra řídicí techniky

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Student: **Ondřej Svoboda**

Studijní program: Kybernetika a robotika
Obor: Systémy a řízení

Název tématu: **Hybridní úĺožiště dat pro simulátor přenosové sítě**

Pokyny pro vypracování:

1. Seznamte se s problematikou a prostudujte literaturu dodanou vedoucím práce.
2. Navrhněte strukturu hybridního datového úložiště. Využijte kombinaci relační/objektové databáze a úložiště pro časové řady.
3. K úložišti vytvořte API v Pythonu a C#.
4. Funkčnost aplikace ověřte úspěšným importem a exportem serializovaného scénáře.
5. Vyhodnoťte rychlost importu/exportu scénáře a proveďte případné optimalizace.

Seznam odborné literatury:

[1] The HDF Group. Hierarchical Data Format, version 5, 1997-NNNN, http://www.hdfgroup.org/HDF5/
[2] MariaDB, https://mariadb.com/
[3] Entity framework, https://msdn.microsoft.com/en-us/data/ef.aspx
[4] NHibernate, http://nhibernate.info/
[5] O. Zlevor, "Optimalizace provozu elektrických VN sítí pomocí řízení spotřeby", BP, FEL CVUT, 2014
[6] J. Zabojnik, "Modelování výroby a toků elektrické energie v evropské přenosové soustavě", DP, FEL CVUT, 2012

Vedoucí: Ing. Jan Zábojník

Platnost zadání: do konce letního semestru 2015/2016

L.S.

prof. Ing. Michael Šebek, DrSc.
vedoucí katedry

prof. Ing. Pavel Ripka, CSc.
děkan

V Praze dne 20. 2. 2015

Poděkování

Rád bych poděkoval vedoucímu této bakalářské práce Ing. Janu Zábojníkovi za jeho pedagogickou a odbornou pomoc a další cenné rady při zpracování této bakalářské práce. Dále děkuji své rodině a přátelům za podporu při mém studiu a psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně, a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

V Praze dne ....

Abstrakt

Tato bakalářská práce se zabývá návrhem struktury úložiště dat pro simulátor elektrické přenosové soustavy a aplikace, která toto úložiště bude spravovat. Jejím výsledkem jsou funkční aplikace napsané v jazyce Python a C# využívající jako úložiště dat kombinaci souborů ve formátu HDF5 a SQL databáze. Navržená aplikace ukládá odsimulovaná data a zpracovává a odesílá data do aplikace klienta k jejich vizualizaci.

Klíčová slova

hybridní databáze, přenosová soustava, HDF5 formát, objektově relační mapování, serverová aplikace

Abstract

This bachelor thesis describes design of data storage for electrical power grid simulator and its servicing API. The outcome are a featured API written in Python and C# languages using data storage combination of HDF5 format files and relational database. Application stores simulated data from simulator and prepares data for sending to client application for visualization.

Keywords

hybrid database, power grid, HDF5 format, relation objectional mapping, server application

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Pro budoucí rozvoj elektrické přenosové soustavy je nutné simulovat její chování v závislosti na její struktuře. V současné době pro tento účel existuje aplikace, která simuluje výrobu a přenos elektrické energie v rozvodné síti. Výstupem simulací této aplikace je značné množství dat, které je potřeba pro pozdější analýzu skladovat.

Tyto data jsou v současnosti skladována ve formátu serializovaného scénáře, který neumožňuje jednoduchou správu a editaci scénářů. Z tohoto důvodu je potřeba navrhnout nové hybridní úložiště dat, které kombinuje relační databázi pro ukládání jednotlivých částí rozvodné sítě, s datovým formátem, který je orientovaný na skladování velkého množství dlouhých číselných řad, které mohou obsahovat až desetitisíce hodnot. Pro správu tohoto úložiště je dále potřeba navrhnout a vytvořit API v jazyce Python a C#. API v jazyce Python bude využita pro zpracování a vizualizaci dat skladovaných v úložišti formou interaktivní webové aplikace, zatímco API v jazyce C# bude použita pro import/export dat mezi úložištěm a aplikací simulátoru.

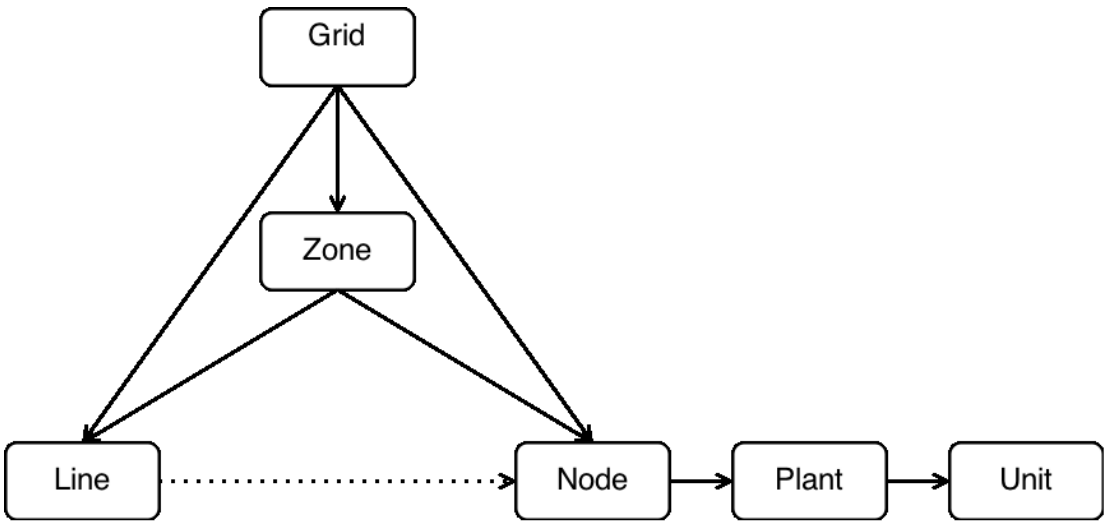
První kapitola uvádí krátký popis přenosové soustavy, návrh a strukturu hybridního úložiště dat a vzorový uchovaný scénář. Ve druhé kapitole je popsána API pro zpracování uložených dat k jejich prezentaci. Třetí kapitola popisuje API pro import serializovaného scénáře ze simulátoru do úložiště a naopak. Čtvrtá kapitola je věnována vyhodnocení rychlosti importu a exportu scénáře prostřednictvím API ze třetí kapitoly. V závěru je provedeno shrnutí dosažených výsledků a předloženy návrhy na budoucí zlepšení.

1. Struktura datového úložiště

Tato kapitola se zabývá navrženou strukturou hybridního datového úložiště a popisem struktury použitého modelu elektrické přenosové soustavy, databáze a souborového formátu.

1.1. Struktura elektrické přenosové soustavy

Každý uložený scénář znázorňuje model rozvodné sítě odsimulovaný s určitými parametry. Model se skládá ze šesti základních prvků. Každý scénář obsahuje právě jednu síť Grid, která představuje oblast simulované rozvodné soustavy. Tato síť se dále dělí na menší oblasti Zone. Každá Zone může obsahovat uzly sítě Node a jejich propojení vedením Line. Také se zde mohou vyskytovat uzly a vedení, které nenáleží do žádné oblasti Zone, ale pouze přímo do sítě Grid. Každý uzel v sobě sdružuje elektrárny Plant, které obsahují bloky Unit. Elektrárny se navíc liší typem (tepelné, přečerpávací, akumulační a s obnovitelnými zdroji), který je uveden u každé z nich. Elektrárny různých typů se liší časovými řadami.



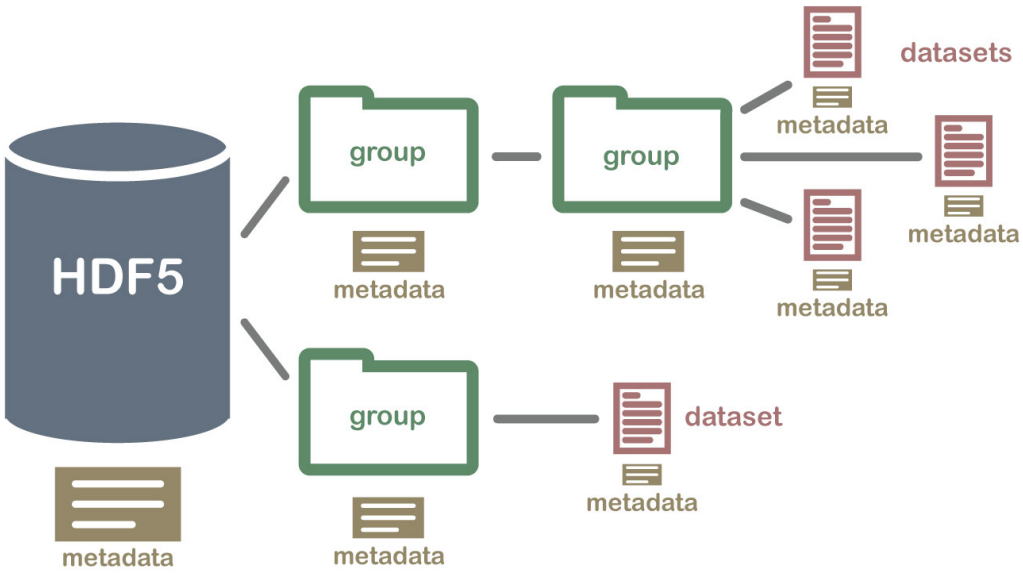
Obr. 1. Diagram struktury modelu přenosové soustavy

Každý z těchto prvků modelu má svá specifická vstupní a výstupní data. Vstupní data jsou parametry daného prvku a výstupní data jsou odsimulované průběhy sledovaných veličin (již zmíněné časové řady). Uložení všech těchto prvků a jejich časových řad je popsány později.

1.2. Soubory formátu HDF5

HDF5 je přenosný open-source hierarchický datový formát, navržený pro uchovávání a organizaci velkého množství komplexních číselných dat. Jedná se o virtuální souborový systém v souboru, využívající symbolické odkazy. Je to otevřený formát a je široce podporovaný mnoha nástroji a programovacími jazyky.

Soubory formátu HDF5 jsou samopopisné a dovolují tak uživateli vytvářet složité vztahy a podmínky mezi daty bez nutnosti dalšího souboru s metadaty. Formát HDF5 také podporuje tzv. data slicing a extrahování částí časových řad, což znamená, že pro získání časové řady nemusí být načten do paměti RAM celý soubor. Typické použití souborů tohoto formátu je ve výzkumu, kdy je potřeba zaznamenat velké množství číselných řad. Vnitřní strukturu těchto souborů, která je vyobrazena na obr. 2, lze zobrazit pomocí programu HDFView, který je také open-source. Tento program umožňuje zobrazit uložená data v poli a základní vykreslení těchto dat do grafu.

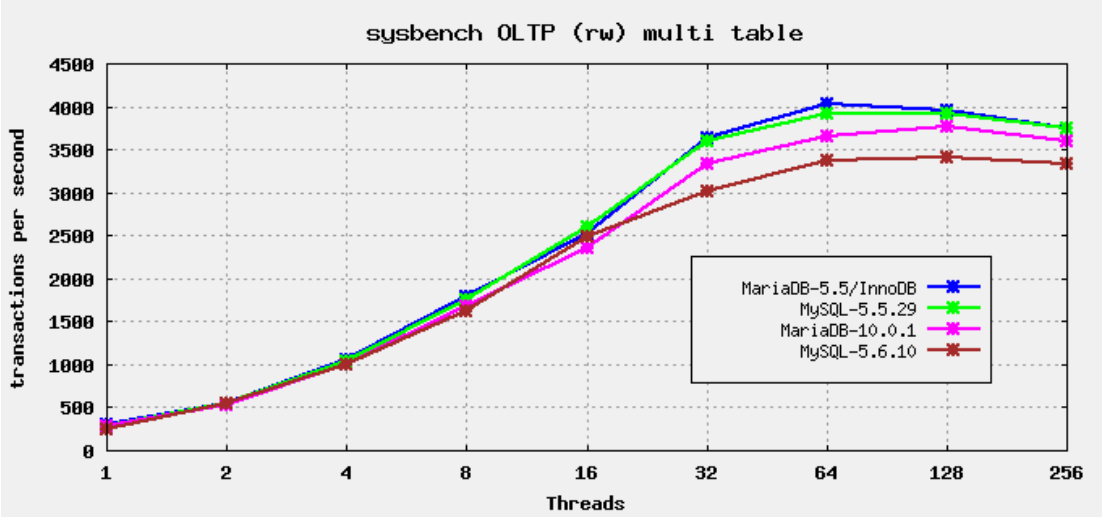


Obr. 2. Vnitřní struktura souboru ve formátu HDF5 [1]

Soubory typu HDF5 se skládají ze dvou základních prvků: ze skupin a časových řad. Skupiny jsou struktury, které obsahují žádnou nebo více instancí dalších skupin nebo časových řad. Skládají se ze dvou částí, z hlavičky, která obsahuje jméno skupiny a seznam jejích vlastností, a ze symbolické tabulky, která obsahuje seznam všech časových řad, náležících do této skupiny. Časové řady jsou struktury, které uchovávají číselná data. Skládají se ze dvou částí: z hlavičky, která obsahuje informace potřebné k interpretaci uložených dat, a z vícedimenzionálního pole, které umožňuje ukládat data homogenního typu. [2]

1.3. Databáze v MariaDB

Pro toto navrhované úložiště byla vybrána databáze MariaDB. Jedná se o open-source relační databáze. MariaDB vznikla jako nástupnická větev databáze MySQL, se kterou je zpětně binárně kompatibilní. To znamená, že všechny rozhraní pro práci s MariaDB jsou stejná jako pro MySQL a všechny klientské API, protokoly a struktury jsou identické. MariaDB obsahuje navíc některé nové funkce a oproti MySQL je v mnoha ohledech více optimalizovaná a tím i rychlejší. [3]



Obr. 3. Porovnání zápis/čtení mezi MariaDB a MySQL [4]

Graf na obr.3 ukazuje porovnání počtu zpracovaných zápisů a čtení za sekundu v závislosti na počtu připojených vláken. Je z něj patrné, že použitá databáze MariaDB verze 10.0.1 převážně převyšuje aktuálně nejnovější verzi databáze MySQL. Pro vizualizaci tabulek z databáze magmadb byla použita open-source aplikace HeidiSQL.

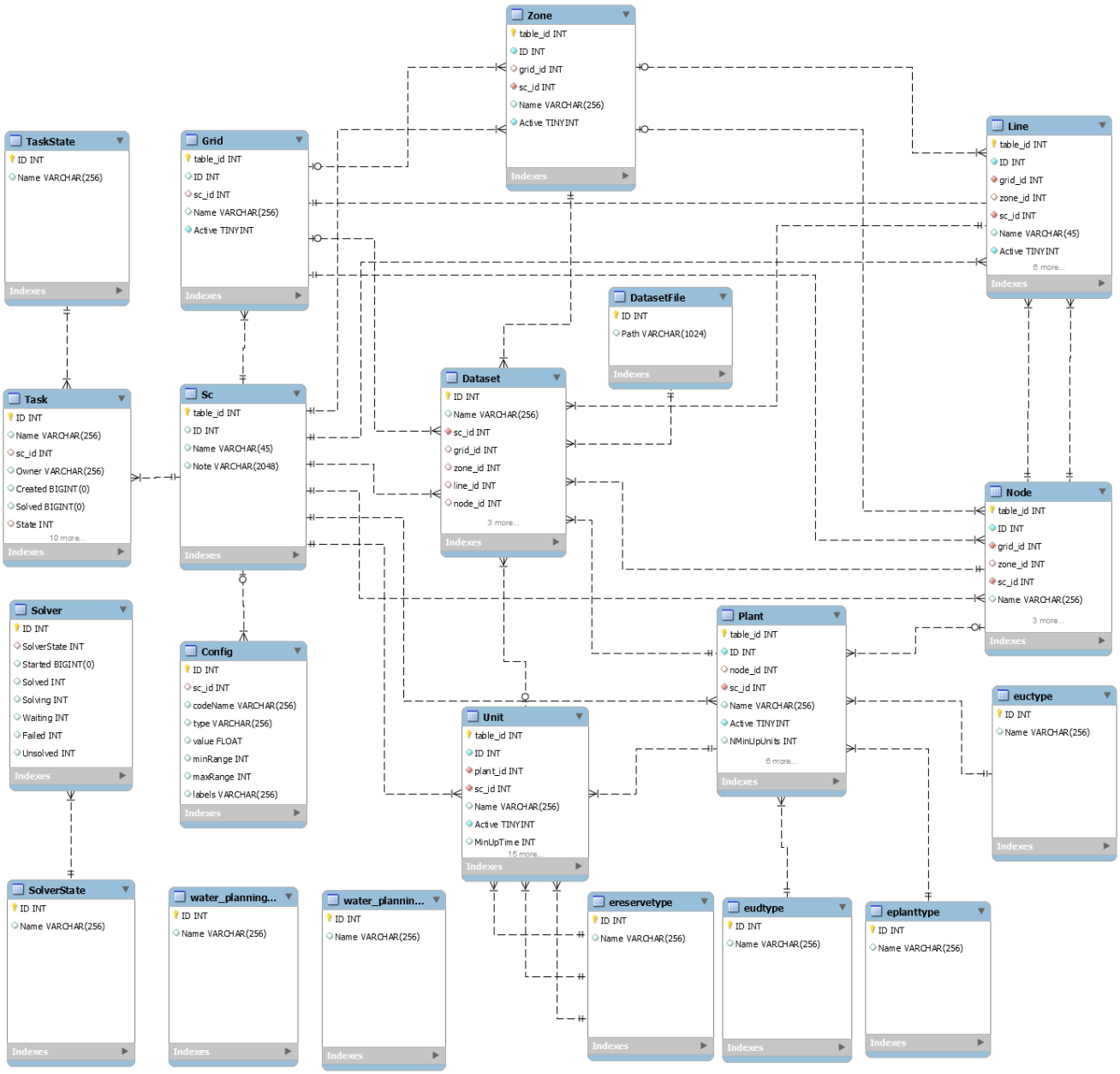
1.4. Navržené úložiště

Tato sekce se zabývá strukturou navrhovaného hybridního úložiště, které se skládá ze dvou částí. Z relační databáze, ve které jsou uloženy informace o prvcích modelu elektrické rozvodné soustavy a dalších pomocných prvcích, a ze souborů formátu HDF5, ve kterých jsou uloženy vstupní a výstupní časové řady. Obě části úložiště jsou propojeny na úrovni API.

1.4.1. Magmadb databáze

Navržená databáze nazvaná magmadb se skládá z celkem dvaceti tabulek. Struktura každé z nich je názorně popsána. Vazby mezi tabulkami jsou realizovány pomocí cizích klíčů. Cizí klíče mezi tabulkami, představující prvky modelu přenosové soustavy, jsou pro jednodušší operace mazání a editace záznamů nastaveny na možnost "Cascade All". To znamená, že při změně hodnoty v odkazované tabulce se změní hodnota i v tabulce odkazující a také to, že při smazání záznamu z odkazované tabulky se smažou všechny záznamy, které na tento záznam odkazují. Tyto vazby jsou důležité také pro zpětnou

rekonstrukci cesty k danému objektu v souboru formátu HDF5, která je generována na úrovni API pro načtení hodnot z příslušných časových řad.



Obr. 4. ER diagram databáze magmadb

Sc Sc je tabulka všech skladovaných scénářů v úložišti. Na tuto tabulku se odkazují cizími klíči všechny tabulky prvků modelu přenosové soustavy, tabulka časových řad, úloh a tabulka s vlastnostmi uložených veličin Config.

Cizí klíče	Unikátní klíče
	table_id
	ID

Tab. 1. Cizí a unikátní klíče tabulky Sc

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	table_id	INT	10	☑	☐	☐	AUTO_INCREMENT
2	ID	INT	10	☑	☑	☐	NULL
3	Name	VARCHAR	45	☐	☑	☐	NULL
4	Note	VARCHAR	2048	☐	☑	☐	NULL

Obr. 5. Tabulka Sc z databáze magmadb

Název sloupce	Popis
table_id	Slouží jako primární klíč tabulky
ID	Uživatelské označení scénáře
Name	Jméno scénáře
Note	Poznámka ke scénáři

Tab. 2. Popis tabulky Sc v databázi magmadb

Config V této tabulce jsou obsaženy konfigurace všech scénářů, uložených v úložišti, a jejich vlastnosti, potřebné pro správnou reprezentaci dat. Ke každému scénáři patří právě jedna sada záznamů v této tabulce.

Cizí klíče	Unikátní klíče
sc_id	Kombinace sc_id a ID

Tab. 3. Cizí a unikátní klíče tabulky Config

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	**ID**	**INT**	**10**	☑	☐	☐	**AUTO_INCREMENT**
2	sc_id	INT	10	☑	☑	☐	NULL
3	codeName	VARCHAR	256	☐	☑	☐	NULL
4	type	VARCHAR	256	☐	☑	☐	NULL
5	value	FLOAT		☐	☑	☐	NULL
6	minRange	INT	11	☐	☑	☐	NULL
7	maxRange	INT	11	☐	☑	☐	NULL
8	labels	VARCHAR	256	☐	☑	☐	NULL

Obr. 6. Tabulka Config z databáze magmadb

Název sloupce	**Popis**
ID	Generované označení, slouží jako primární klíč tabulky
sc_id	Uživatelské ID scénáře, ve kterém se konfigurace nachází
codeName	Označení konfigurace
type	Datový typ konfigurace
value	Hodnota konfigurace
minRange	Nejmenší možná hodnota konfigurace
maxRange	Největší možná hodnota konfigurace
labels	Výpis konstant v případě výčtového typu

Tab. 4. Popis tabulky Config v databázi magmadb

Grid Tabulka prvků Grid obsahuje simulované sítě z každého scénáře, uloženého v úložišti. Každá síť náleží právě jednomu scénáři.

Cizí klíče	**Unikátní klíče**
sc_id	table_id
	Kombinace ID a sc_ID

Tab. 5. Cizí a unikátní klíče tabulky Grid

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	**table_id**	**INT**	**10**	☑	☐	☐	**AUTO_INCREMENT**
2	ID	INT	10	☑	☑	☐	NULL
3	sc_id	INT	10	☑	☑	☐	NULL
4	Name	VARCHAR	256	☐	☑	☐	NULL
5	Active	TINYINT	4	☐	☐	☐	Žádná hodnota

Obr. 7. Tabulka Grid z databáze magmadb

Název sloupce	Popis
table_id	Slouží jako primární klíč tabulky
ID	Uživatelské označení sítě
sc_id	Uživatelské ID scénáře Sc, do kterého daná síť náleží
Name	Jméno sítě
Active	Udává, zda je daná síť aktivní

Tab. 6. Popis tabulky Grid v databázi magmadb

Zone Tabulka prvků Zone obsahuje dílčí oblasti všech sítí Grid. Do jedné sítě může náležet i více podsítí.

Cizí klíče	Unikátní klíče
sc_id	table_id
grid_id	Kombinace ID a grid_id

Tab. 7. Cizí a unikátní klíče tabulky Zone

	#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
	1	table_id	INT	11	☑	☐	☐	AUTO_INCREMENT
	2	ID	INT	10	☑	☐	☐	Žádná hodnota
	3	sc_id	INT	10	☑	☐	☐	Žádná hodnota
	4	grid_id	INT	10	☑	☐	☐	Žádná hodnota
	5	Name	VARCHAR	256	☐	☑	☐	NULL
	6	Active	TINYINT	4	☐	☐	☐	Žádná hodnota

Obr. 8. Tabulka Zone z databáze magmadb

Název sloupce	Popis
table_id	Slouží jako primární klíč tabulky
ID	Uživatelské označení sítě
sc_id	Uživatelské ID scénáře Sc, do kterého daná síť náleží
grid_id	Uživatelské ID sítě, do kterého daná oblast náleží
Name	Jméno sítě
Active	Udává, zda je daná síť aktivní

Tab. 8. Popis tabulky Zone v databázi magmadb

Line Tabulka Line reprezentuje všechny vedení mezi uzly v síti. Hodnota ve sloupci zone_id (cizí klíč na sloupce ID tabulky Zone) může obsahovat hodnotu null, kvůli existenci vedení, které nepatří do žádné oblasti Zone, ale pouze přímo do sítě Grid.

Cizí klíče	Unikátní klíče
sc_id	table_id
grid_id	Kombinace ID a grid_id

Tab. 9. Cizí a unikátní klíče tabulky Line

	#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
🔑 📌	1	**table_id**	**INT**	**11**	☑	☐	☐	**AUTO_INCREMENT**
📌	2	ID	INT	10	☑	☐	☐	Žádná hodnota
📌 🔑	3	grid_id	INT	10	☑	☐	☐	Žádná hodnota
🔑	4	zone_id	INT	10	☑	☑	☐	NULL
🔑	5	sc_id	INT	10	☑	☐	☐	Žádná hodnota
	6	Name	VARCHAR	45	☐	☑	☐	NULL
	7	Active	TINYINT	4	☐	☐	☐	Žádná hodnota
🔑	8	Node1_id	INT	10	☑	☐	☐	Žádná hodnota
🔑	9	Node2_id	INT	10	☑	☐	☐	Žádná hodnota
	10	Max12Pwr	INT	11	☐	☑	☐	NULL
	11	Max21Pwr	INT	11	☐	☑	☐	NULL
	12	R	FLOAT		☐	☑	☐	NULL
	13	X	FLOAT		☐	☑	☐	NULL

Obr. 9. Tabulka Line z databáze magmadb

Název sloupce	Popis
table_id	Slouží jako primární klíč tabulky
ID	Uživatelské označení uzlu
sc_id	Uživatelské ID scénáře, do kterého daný uzel náleží, cizí klíč do tabulky Sc
grid_id	Uživatelské ID sítě, do kterého daný uzel náleží, cizí klíč do tabulky Grid
zone_id	Uživatelské ID oblasti, do které daný uzel náleží, cizí klíč do tabulky Zone
Name	Uživatelské jméno uzlu
Active	Udává, zda je daný uzel aktivní
Node1_id	Uživatelské ID uzlu na jednom konci vedení
Node2_id	Uživatelské ID uzlu na druhém konci vedení
Max12Pwr	Maximální přenosová kapacita v kladném směru
Max21Pwr	Maximální přenosová kapacita v záporném směru
R	Resistance
X	Reaktance

Tab. 10. Popis tabulky Line v databázi magmadb

Node Tabulka prvků Node obsahuje všechny uzly sítě. Také zde může záznam ve sloupci zone_id obsahovat hodnotu null a to kvůli existenci uzlů, které nepatří do žádné oblasti, ale je nutné s nimi počítat v rámci sítě.

Cizí klíče	Unikátní klíče
sc_id	table_id
grid_id	Kombinace ID a grid_id

Tab. 11. Cizí a unikátní klíče tabulky Node

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	table_id	INT	10	☑	☐	☐	AUTO_INCREMENT
2	ID	INT	10	☑	☐	☐	Žádná hodnota
3	grid_id	INT	10	☑	☐	☐	Žádná hodnota
4	zone_id	INT	10	☑	☑	☐	NULL
5	sc_id	INT	10	☑	☐	☐	Žádná hodnota
6	Name	VARCHAR	256	☐	☑	☐	NULL
7	Active	TINYINT	4	☐	☐	☐	Žádná hodnota
8	GpsLat	VARCHAR	256	☐	☑	☐	NULL
9	GpsLong	VARCHAR	256	☐	☑	☐	NULL

Obr. 10. Tabulka Node z databáze magmadb

Název sloupce	Popis
table_id	Slouží jako primární klíč tabulky
ID	Uživatelské označení uzlu
sc_id	Uživatelské ID scénáře, do kterého daný uzel náleží
grid_id	Uživatelské ID sítě, do kterého daný uzel náleží
zone_id	Uživatelské ID oblasti, do které daný uzel náleží
Name	Uživatelské jméno uzlu
Active	Udává, zda je daný uzel aktivní
GpsLat	Zeměpisná šířka polohy uzlu (ve stupních s přesností na 6 desetinných míst)
GpsLong	Zeměpisná délka polohy uzlu (ve stupních s přesností na 6 desetinných míst)

Tab. 12. Popis tabulky Node v databázi magmadb

Plant Tabulka Plant obsahuje všechny elektrárny v uzlech. Plant také využívá jako jedna ze čtyř tabulek tabulky nahrazující enumy. Typ elektrárny je uložen v tabulce PlantType.

Cizí klíče	Unikátní klíče
sc_id	table_id
node_id	Kombinace ID a node_id

Tab. 13. Cizí a unikátní klíče tabulky Plant

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	table_id	INT	10	☑	☐	☐	AUTO_INCREMENT
2	ID	INT	10	☑	☐	☐	Žádná hodnota
3	node_id	INT	10	☑	☐	☐	Žádná hodnota
4	sc_id	INT	10	☑	☐	☐	Žádná hodnota
5	Name	VARCHAR	256	☐	☑	☐	NULL
6	NMinUpUnits	INT	11	☐	☑	☐	NULL
7	MaxEnergySto...	INT	11	☐	☑	☐	NULL
8	PlantType	INT	10	☑	☑	☐	NULL
9	UCType	INT	10	☑	☑	☐	NULL
10	UDType	INT	10	☑	☑	☐	NULL
11	GpsLat	FLOAT		☐	☑	☐	NULL
12	GpsLong	FLOAT		☐	☑	☐	NULL
13	Active	TINYINT	4	☐	☐	☐	Žádná hodnota

Obr. 11. Tabulka Plant z databáze magmadb

Název sloupce	**Popis**
table_id	Slouží jako primární klíč tabulky
ID	Uživatelské označení uzlu
sc_id	Uživatelské ID scénáře, do kterého daný uzel náleží
node_id	Uživatelské ID uzlu, do kterého daná elektrárna náleží
Name	Uživatelské jméno uzlu
Active	Udává, zda je daná elektrárna aktivní
NMinUpUnits	Minimální počet aktivních bloků elektrárny
MaxEnergyStorage	Kapacita nádrže
PlantType	Typ elektrárny
UCType	Způsob nasazování zdroje
UDType	Způsob provozu zdroje
GpsLat	Udává zeměpisnou šířku polohy elektrárny (ve stupních s přesností na 6 desetinných míst)
GpsLong	Udává zeměpisnou délku polohy elektrárny (ve stupních s přesností na 6 desetinných míst)

Tab. 14. Popis tabulky Plant v databázi magmadb

Eplanttype, Ereservetype, Euctype, Eudtype, Water_planning_pricing_strategy, Water_planning_select_strategy Tyto tabulky slouží jako náhrady za enumy pro tabulky prvků modelu sítě. Všechny se skládají ze svou sloupců.

Cizí klíče	**Unikátní klíče**
	ID

Tab. 15. Cizí a unikátní klíče tabulek nahrazujících enumy

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	ID	INT	10	☑	☐	☐	AUTO_INCREMENT
2	Name	VARCHAR	256	☐	☑	☐	NULL

Obr. 12. Tabulka náhrad za enumy

Název sloupce	Popis
ID	Generované označení atributu, slouží jako primární klíč tabulky
Name	Název atributu

Tab. 16. Popis tabulek nahrazujících enumy v databázi magmadb

Unit Tabulka Unit obsahuje všechny bloky v elektrárnách.

Cizí klíče	Unikátní klíče
sc_id	table_id
plant_id	Kombinace ID a plant_id

Tab. 17. Cizí a unikátní klíče tabulky Unit

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	table_id	INT	10	☑	☐	☐	AUTO_INCREMENT
2	ID	INT	10	☑	☐	☐	Žádná hodnota
3	plant_id	INT	10	☑	☐	☐	Žádná hodnota
4	sc_id	INT	10	☑	☐	☐	Žádná hodnota
5	Name	VARCHAR	256	☐	☑	☐	NULL
6	Active	TINYINT	4	☐	☐	☐	Žádná hodnota
7	MinUpTime	INT	11	☐	☑	☐	NULL
8	MinDownTime	INT	11	☐	☑	☐	NULL
9	RampRateStart	FLOAT		☐	☑	☐	NULL
10	RampRateRun	FLOAT		☐	☑	☐	NULL
11	RampRateStop	INT	11	☐	☑	☐	NULL
12	StartFuelId	INT	11	☐	☑	☐	NULL
13	StartFuelAmo...	INT	11	☐	☑	☐	NULL
14	AllowReserve	TINYINT	4	☐	☐	☐	Žádná hodnota
15	VarCost	INT	11	☐	☑	☐	NULL
16	FixCost	INT	11	☐	☑	☐	NULL
17	Effic	INT	11	☐	☑	☐	NULL
18	FuelRatio	INT	11	☐	☑	☐	NULL
19	Ems	INT	11	☐	☑	☐	NULL
20	PriResType	INT	10	☑	☑	☐	NULL
21	SecResType	INT	10	☑	☑	☐	NULL
22	TerResType	INT	10	☑	☑	☐	NULL

Obr. 13. Tabulka Unit z databáze magmadb

Název sloupce	Popis
table_id	Slouží jako primární klíč tabulky
ID	Uživatelské označení bloku
sc_id	Uživatelské ID scénáře, do kterého daná elektrárna náleží
plant_id	Uživatelské ID elektrárny, do které daný blok náleží
Name	Uživatelské jméno bloku
Active	Udává, zda je daný blok aktivní
MinUpTime	Minimální čas, kdy je blok aktivní
MinDownTime	Minimální čas, kdy je blok neaktivní
RampRateStart	Nájezdová rampa
RampRateRun	Operační rampa
RampRateStop	Odstávková rampa
StartFuelId	Id použitého paliva pro start bloku
StartFuelAmount	Množství paliva použitého pro start bloku
AllowReserve	Udává, zda má blok povolené rezervy
VarCost	Variabilní náklady bloku
FixCost	Fixní náklady bloku
Effic	Účinnost bloku
FuelRatio	Palivový mix
Ems	Emise
PriResType	Typ primární podpůrné služby
SecResType	Typ sekundární podpůrné služby
TerResType	Typ terciální podpůrné služby

Tab. 18. Popis tabulky Unit v databázi magmadb

Dataset Tabulka Dataset reprezentuje všechny jména časových řad všech prvků modelu rozvodné sítě v úložišti. V tabulce je uloženo jméno dané řady a podle hodnot cizích klíčů (sc_id, grid_id, zone_id, node_id, line_id, plant_id a unit_id) se na úrovni API automaticky generuje cesta k časovým řadám v souboru formátu HDF5.

Cizí klíče	Unikátní klíče
sc_id	ID
grid_id	
zone_id	
line_id	
node_id	
plant_id	
unit_id	

Tab. 19. Cizí a unikátní klíče tabulky Dataset

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	ID	INT	10	☑	☐	☐	AUTO_INCREMENT
2	Name	VARCHAR	256	☐	☑	☐	NULL
3	sc_id	INT	10	☑	☐	☐	Žádná hodnota
4	grid_id	INT	10	☑	☑	☐	NULL
5	zone_id	INT	10	☑	☑	☐	NULL
6	line_id	INT	10	☑	☑	☐	NULL
7	node_id	INT	10	☑	☑	☐	NULL
8	plant_id	INT	10	☑	☑	☐	NULL
9	unit_id	INT	10	☑	☑	☐	NULL
10	path_id	INT	10	☑	☐	☐	Žádná hodnota

Obr. 14. Tabulka Dataset z databáze magmadb

Název sloupce	Popis
ID	Generované označení časové řady, slouží jako primární klíč tabulky
Name	Uživatelské jméno časové řady
sc_id	Uživatelské ID scénáře
grid_id	Uživatelské ID sítě
zone_id	Uživatelské ID oblasti
node_id	Uživatelské ID uzlu
line_id	Uživatelské ID vedení
plant_id	Uživatelské ID elektrárny
unit_id	Uživatelské ID bloku
path_id	ID v tabulce DatasetFile

Tab. 20. Popis tabulky Dataset v databázi magmadb

DatasetFile V tabulce DatasetFile jsou uloženy systémové cesty ke všem souborům s časovými řadami.

Cizí klíče	Unikátní klíče
	ID

Tab. 21. Cizí a unikátní klíče tabulky DatasetFile

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	ID	INT	10	☑	☐	☐	AUTO_INCREMENT
2	Path	VARCHAR	1024	☐	☑	☐	NULL

Obr. 15. Tabulka DatasetFile z databáze magmadb

Název sloupce	Popis
ID	Generované označení, slouží jako primární klíč tabulky
Path	Systémová cesta k souboru, ve kterém je uložena časová řada

Tab. 22. Popis tabulky DatasetFile v databázi magmadb

Solver V tabulce Solver je uložen aktuální stav solveru scénářů.

Cizí klíče	Unikátní klíče
SolverState	ID

Tab. 23. Přehled cizích a unikátních klíčů tabulky Solver

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	ID	INT	10	☑	☐	☐	AUTO_INCREMENT
2	SolverState	INT	10	☑	☑	☐	NULL
3	Started	BIGINT	20	☐	☑	☐	NULL
4	Solved	INT	10	☑	☑	☐	NULL
5	Solving	INT	10	☑	☑	☐	NULL
6	Waiting	INT	10	☑	☑	☐	NULL
7	Failed	INT	10	☑	☑	☐	NULL
8	Unsolved	INT	10	☑	☑	☐	NULL

Obr. 16. Tabulka Solver z databáze magmadb

Název sloupce	Popis
ID	Generované označení solveru, slouží jako primární klíč tabulky
SolverState	Aktuální stav solveru
Started	Datum, kdy byl solver spuštěn
Solved	Počet vyřešených úloh
Solving	Počet právě řešených úloh
Waiting	Počet úloh, čekájících na vyřešení
Failed	Počet úloh, které při řešení selhaly
Unsolved	Počet nevyřešených úloh

Tab. 24. Popis tabulky Solver v databázi magmadb

Task V tabulce Task jsou uloženy všechny úlohy pro solver.

Cizí klíče	Unikátní klíče
sc_id	ID
State	

Tab. 25. Cizí a unikátní klíče tabulky Task

#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
1	ID	INT	10	☑	☐	☐	AUTO_INCREMENT
2	Name	VARCHAR	256	☐	☑	☐	NULL
3	sc_id	INT	10	☑	☑	☐	NULL
4	Owner	VARCHAR	256	☐	☑	☐	NULL
5	Created	BIGINT	20	☐	☑	☐	NULL
6	Solved	BIGINT	20	☐	☑	☐	NULL
7	State	INT	10	☑	☑	☐	NULL
8	SolvingAllowed	TINYINT	4	☐	☑	☐	NULL
9	Total	INT	11	☐	☑	☐	NULL
10	From	BIGINT	20	☐	☑	☐	NULL
11	To	BIGINT	20	☐	☑	☐	NULL
12	Step	BIGINT	20	☐	☑	☐	NULL
13	Current	INT	11	☐	☑	☐	NULL
14	CurFrom	BIGINT	20	☐	☑	☐	NULL
15	CurTo	BIGINT	20	☐	☑	☐	NULL
16	EstimatedEnd	BIGINT	20	☐	☑	☐	NULL
17	ErrorMessage	VARCHAR	1024	☐	☑	☐	NULL

Obr. 17. Tabulka Task z databáze magmadb

Název sloupce	**Popis**
ID	Generované označení úlohy, slouží jako primární klíč tabulky
Name	Název úlohy
sc_id	Uživatelské ID řešeného scénáře
Owner	Jméno zadavatele úlohy
Created	Datum vytvoření úlohy
Solved	Datum vyřešení úlohy
State	Aktuální stav úlohy
SolvingAllowed	Udává, zda je povoleno řešení této úlohy
Total	Celkový počet intervalů
From	Datum, od kterého se má úloha řešit
To	Datum, do kterého se má úloha řešit
Step	Krok, se kterým se má úloha řešit
Current	Aktuálně řešený interval
CurFrom	Počátek aktuálně řešeného intervalu
CurTo	Konec aktuálně řešeného intervalu
EstimatedEnd	Předpokládaný čas vyřešení úlohy
ErrorMessage	Chybová zpráva

Tab. 26. Popis tabulky Task v databázi magmadb

SolverState, TaskState Tyto tabulky slouží jako náhrady za enumy pro tabulky Solver a Task. Obě se skládají ze svou sloupců.

Cizí klíče	Unikátní klíče
	ID

Tab. 27. Cizí a unikátní klíče tabulek TaskState a SolverState

	#	Název	Datový typ	Délka/Množi...	Unsign...	Nulový	Zerofill	Výchozí
🔑	1	ID	INT	10	☑	☐	☐	AUTO_INCREMENT
	2	Name	VARCHAR	256	☐	☑	☐	NULL

Obr. 18. Tabulka TaskState z databáze magmadb

Název sloupce	Popis
ID	Generované označení atributu, slouží jako primární klíč tabulky
Name	Název atributu

Tab. 28. Popis tabulky Solver v databázi magmadb

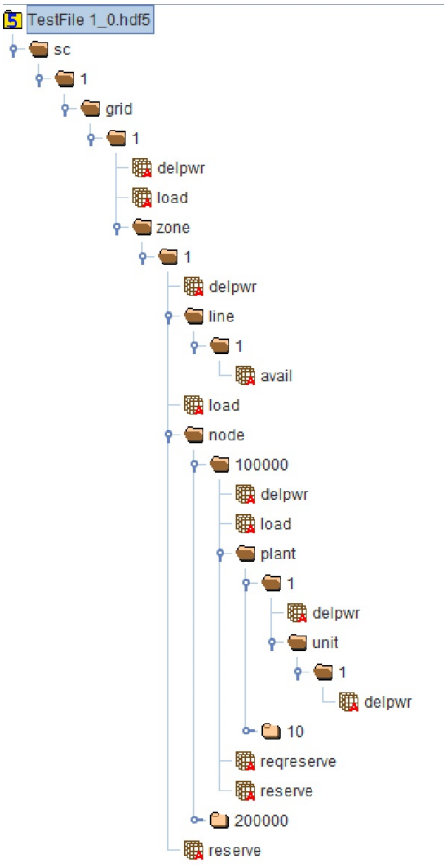
1.4.2. Vzorový scénář v souboru formátu HDF5

Číselné časové řady každého scénáře jsou uloženy v souborech formátu HDF5. Každý takový soubor obsahuje časové řady právě jednoho scénáře. Vnitřní struktura takto uložených dat je popsána na zjednodušeném vzorovém scénáři.

Na obr.19 je vidět zjednodušený dvouuzlový scénář. Jedná se o stromovou strukturu, jejímž kořenem je scénář. Jak již bylo zmíněno, v každém souboru je pouze jeden a každý obsahuje pouze jednu síť. Síť se dále může dělit na oblasti, v tomto případě je zde pouze jedna. V této vzorové oblasti jsou umístěny dva uzly a jedno vedení, které je propojuje. Oba uzly pod sebou sdružují dvě elektrárny a v každé z nich se nachází jeden blok.

Ke každému z těchto prvků modelu patří jeho specifické časové řady. Například uzel 100000 oblasti 1 obsahuje časovou řadu load (zatížení). Tato časová řada jako všechny ostatní obsahuje v hlavičce název časové řady, datový typ uložených dat, počet uložených hodnot, počet atributů v hlavičce, výchozí hodnotu, časový úsek mezi daty, počáteční datum a využité místo. Grafické znázornění hlavičky je na obr. 21.

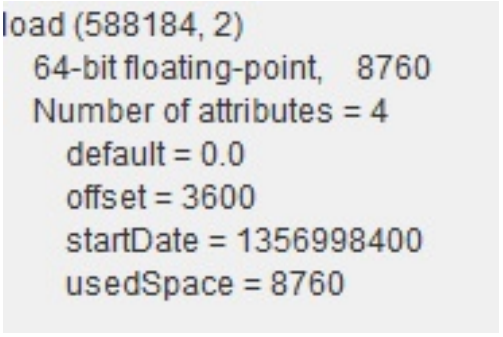
Na obr. 20 je znázorněno pole hodnot časové řady. Je zde vidět, že se ukládají pouze samotné hodnoty, odpovídající data se dopočítávají z počátečního data, časového intervalu mezi hodnotami a pořadím hodnoty.



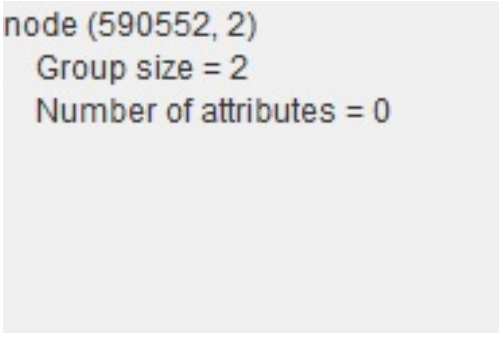
Obr. 19. Stromová struktura vzorového scénáře v souboru HDF5

	0
0	373.0
1	358.0
2	356.0
3	345.0
4	337.0
5	318.0
6	312.0
7	306.0
8	295.0
9	328.0
10	356.0
11	354.0
12	372.0
13	390.0
14	402.0
15	407.0

Obr. 20. Pole hodnot vzorové časové řady v souboru HDF5



Obr. 21. Hlavička časové řady load ze vzorového scénáře



Obr. 22. Hlavička skupiny Node ze vzorového scénáře

Pro získání pole hodnot z časové řady je nutné k ní přistoupit pomocí její absolutní cesty. Ta se skládá ze všech názvů nadřazených skupin oddělených lomítkem a názvu dané řady. Například pro časovou řadu load v uzlu 100000 je její absolutní cesta sc/1/grid/1/zone/1/node/100000/avail.

2. Serverová aplikace pro obsluhu datového úložiště

2.1. Základní popis aplikace

Jedná se o API implementované v jazyce Python, které je navázáno na webserver a slouží ke správě navrženého úložiště. Na této úrovni se spojuje navržená databáze magmadb a soubory formátu HDF5. Jeho úkolem je připravit a odeslat data pro vizualizaci v aplikaci klienta, nebo zpracovat a uložit data vložená uživatelem. API komunikuje s aplikací uživatele pomocí rozhraní REST.

2.2. Využité frameworky

Navržená API využívá při své činnosti čtyři externí frameworky, které umožňují správu navrženého úložiště a usnadňují práci s daty.

2.2.1. H5Py framework

H5Py je open-source knihovna, která funguje jako rozhraní pro práci se soubory formátu HDF5 v jazyce Python. [5] API využívá knihovnu H5Py pro zápis, čtení a mazání časových řad uložených v souborech HDF5.

2.2.2. SQLAlchemy framework

SQLAlchemy je sada nástrojů pro objektově relační mapování (ORM) pro databáze MySQL v jazyce Python. [6] Výhodou ORM je, že tabulky databáze lze jednoduše namapovat na objekty, u kterých lze využívat možnosti dědičnosti a delegace. Rozhraní SQLAlchemy je v API použito právě jako objektově relační mapování.

2.2.3. NumPy framework

NumpPy je rozšíření pro programovací jazyk Python, které umožňuje efektivní práci s velkými vícedimenzionálními poli a přidává široké možnosti matematických nástrojů pro správu velkého množství dat. Toto API využívá knihovnu NumPy pro zpracování přijatých nebo odesílaných časových řad, jako je jejich řazení, hledání minima a maxima a počítání středních hodnot. [7]

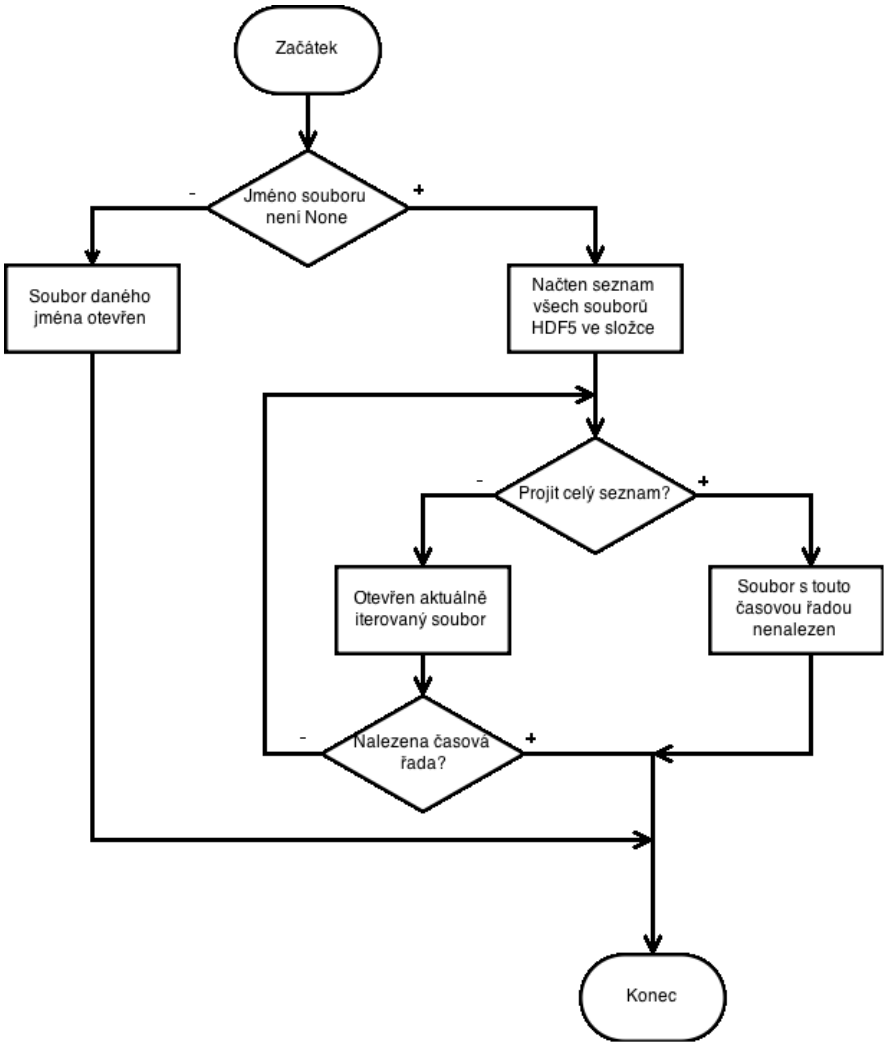
2.2.4. CherryPy

CherryPy je objektově orientovaný webový framework pro programovací jazyk Python, distribuovaný pod licencí BSD, který umožňuje sestavit webovou aplikaci. Navržená API využívá tzv. REST architekturu. [8] To znamená, že všechny požadavky na API mají identifikátor, kterým se specifikuje typ požadavku, a v API jsou předdefinované metody, které jsou volány podle přijatého identifikátoru. Konkrétní popis zpracování je uveden v dalších sekcích.

2.3. Třída pro práci s HDF5

Pro práci se soubory formátu HDF5 byla navržena třída Controller. Každý požadavek na tyto soubory je zpracováván metodami této třídy. Controller zajišťuje čtení, zápis a mazání z těchto souborů. Všechny hlavní metody jsou popsány v další části textu.

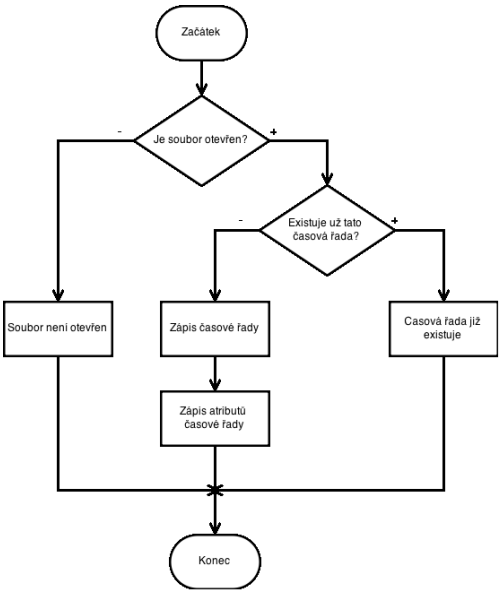
Metoda **___init___** zajišťuje inicializaci objektu třídy Controller otevření správného HDF5 souboru. Jsou volány při zpracování každého dotazu na časovou řadu. Jako jediný povinný parametr přijímá metoda vnitřní cestu k časové řadě, jako volitelné pak název souboru, maximální velikost a mód. Mód určuje, zda má být soubor vytvořen, otevřen pro čtení nebo otevřen pro čtení i zápis. Pokud je metoda zavolána s názvem požadovaného souboru, otevře se. Pokud zůstane název souboru None, pak se načte z předem definované složky celý její obsah, vyfiltrují se soubory formátu HDF5, které se postupně v cyklu prochází a testují, zda se v nich nenachází požadovaná časová řada.



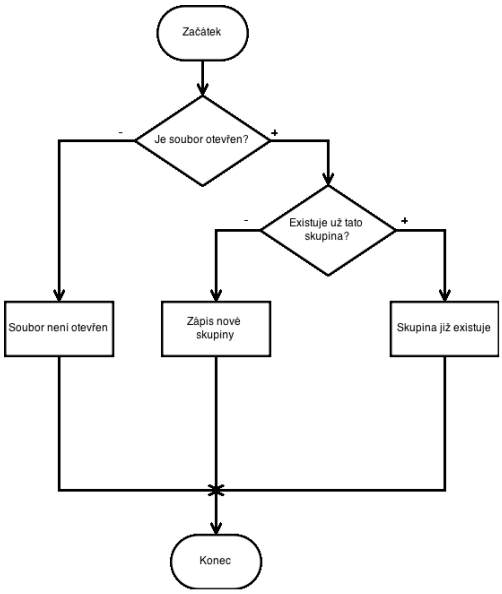
Obr. 23. Vývojový diagram metody ___init___ ve třídě Controller

V případě, že požadovaný soubor nebo soubor s požadovanou časovou řadou není nalezen, uloží se do objektové proměnné file hodnota None. Tuto proměnnou testují při zavolání všechny další metody, které dále s tímto souborem pracují.

Následující tři metody jsou navrženy pro zápis do HDF5 souboru. Pro vytvoření nových časových řad nebo skupin slouží metody **save_new_time_series** a **save_new_subgroup**. Jako vstupní parametry obě metody přijímají vnitřní cestu souboru, kde se má nový prvek vytvořit a metoda save_new_time_series ještě přijímá pole hodnot časové řady a její atributy (počáteční datum, offset a výchozí hodnotu).



Obr. 24. Vývojový diagram metody save_new_dataset třídy Controller

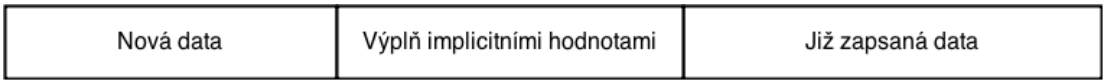


Obr. 25. Vývojový diagram metody save_new_subgroup třídy Controller

Tyto metody, jak bylo zmíněno dříve, na začátku testují, zda je soubor otevřen a pokud ano, pokusí se do souboru zapsat požadovaný prvek (časovou řadu nebo skupinu). Jsou využívány při zpracování požadavku typu POST, které je popsáno v následujících sekcích.

Pro zápis nových dat do již existující časové řady je navržena metoda **save_into_existing**. Metoda přijímá jako vstupní parametry vnitřní cestu k časové řadě, pole nových dat k dopsaní do řady a počáteční datum těchto dat. Na začátku metody se vypočítá datum poslední hodnoty nových dat, načte se počáteční datum již uložených dat a rovněž se dopočítá datum poslední hodnoty. Tyto hodnoty jsou důležité pro správné uložení nových dat do stávající časové řady. Může nastat šest možností:

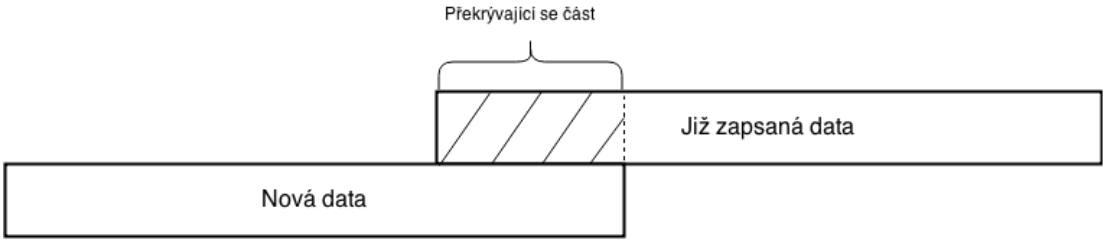
1) Počáteční i koncové datum nových dat je menší než počáteční datum již zapsaných dat. Tato situace je graficky znázorněna na obr. 26.



Obr. 26. Princip zápisu dat do existující časové řady pro první případ

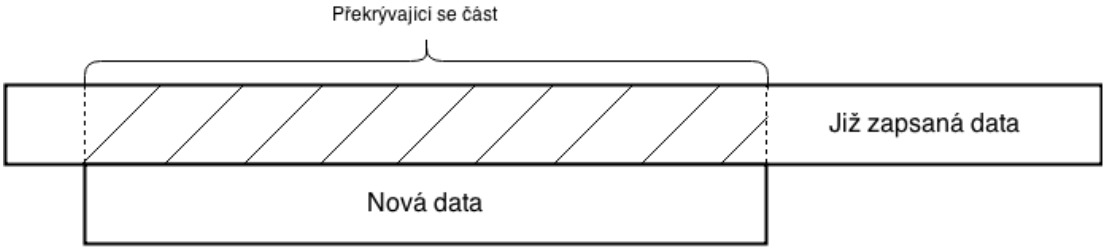
Nová data jsou zapsána před původní a mezera mezi nimi je vyplněna implicitními hodnotami. Počáteční datum časové řady je nastaveno na počáteční datum nových dat a je aktualizována velikost.

2) Tato možnost nastane v případě, že jsou splněny dvě podmínky: počáteční datum nových dat je stále menší než počáteční datum původních dat a koncové datum nových dat je větší než počáteční datum původních dat, ale přitom pořád menší než koncové datum těchto dat. Zde dochází k přepisu části původních hodnot. Počáteční datum časové řady se změní na počáteční datum nových dat a velikost se rovná součtu velikostí původních a nových hodnot bez velikosti jejich průsečíku.



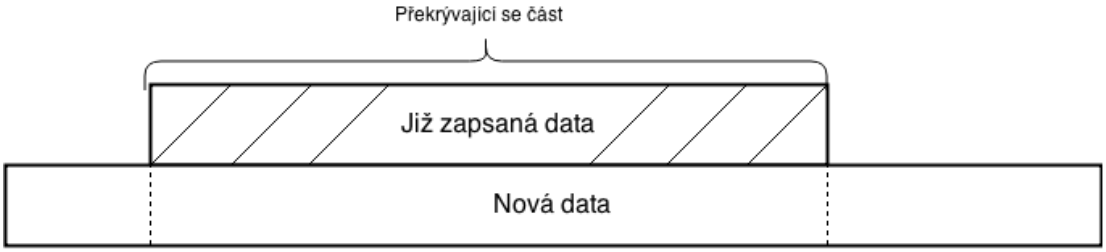
Obr. 27. Princip zápisu dat do existující časové řady pro druhý případ

3) Nejjednodušší možností, kterou zobrazuje obr. 28, je, když se počáteční i koncové datum nových dat vejde mezi počáteční a koncové datum původních dat. Dochází zde k přepsání části původních dat novými hodnotami. Atributy časové řady se nemění.



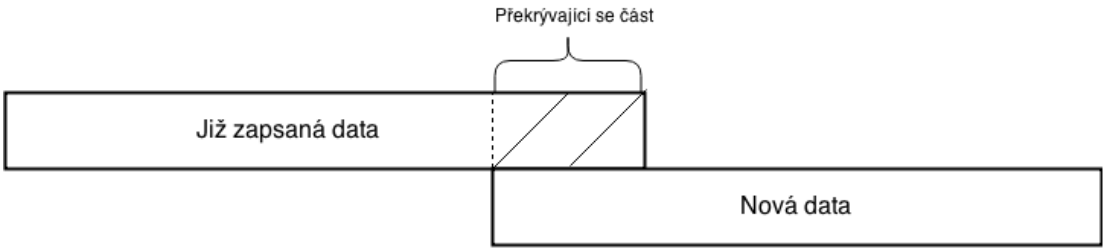
Obr. 28. Princip zápisu dat do existující časové řady pro třetí případ

4) Další situací, která může nastat je, když počáteční datum nových dat menší než původních a koncové datum nových dat větší než původních. Dochází zde k přepsání všech původních dat novými, všechny atributy časové řady jsou nastaveny na vlastnosti nových dat.

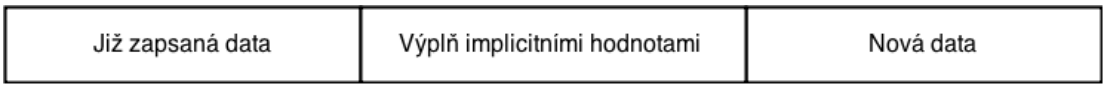


Obr. 29. Princip zápisu dat do existující časové řady pro čtvrtý případ

5,6) Poslední dvě varianty, které jsou znázorněny na obr. 30 a 31, jsou prakticky stejné jako první dvě s rozdílem, že nedochází k přepsání hodnot od začátku směrem ke konci dat, ale od konce směrem k začátku.



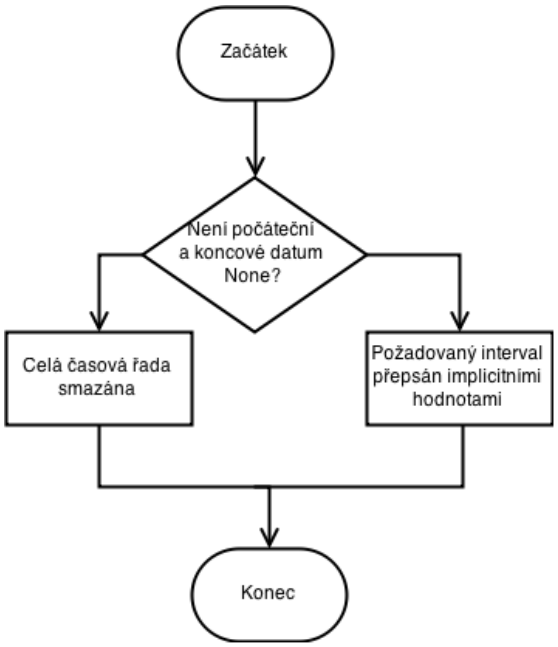
Obr. 30. Princip zápisu dat do existující časové řady pro pátý případ



Obr. 31. Princip zápisu dat do existující časové řady pro šestý případ

Jediný atribut časové řady, který se zde mění, je velikost. V prvním případě je to součet velikostí bez velikosti průsečíku dat a v druhém součet velikostí a velikostí výplně implicitními hodnotami.

Soubory formátu HDF5 v současné době nepodporují žádné jednoduché mechanismy pro smazání časové řady ze souboru a uvolnění použitého místa. Je ale možné smazat všechny odkazy a tím daný objekt znepřístupnit. [9] Pro tuto funkci je navržena metoda **delete_time_series**. Metodu popisuje jednoduchý vývojový diagram na obr. 32.



Obr. 32. Vývojový diagram metody delete_time_series

Jejími vstupními parametry jsou cesta k časové řadě, počáteční a koncové datum. Pokud obě tyto data nejsou nastaveny na None, tak se pouze v tomto zadaném intervalu nahradí původní hodnoty implicitními. Pokud je jedno z těchto dat nebo obě None, smaže se celá časová řada.

Poslední důležitou metodou je metoda **load_time_series**, která slouží k načtení pole dat z načtené časové řady. Vstupními parametry metody jsou cesta k časové řadě a počáteční a koncové datum požadovaných hodnot. V případě, že jsou tyto data nastavena na None, metoda načte hodnoty z celého rozsahu časové řady. Pokud je některá hranice požadovaných dat mimo interval dat časové řady, jsou chybějící hodnoty doplněny implicitními.

2.4. Objektově relační přístup k databázi

Pro práci s navrženou databází magmadb je použito rozhraní SQLAlchemy, které je využito jako objektově relační mapování (dále jen ORM). ORM je mechanismus, který umožňuje namapovat tabulky relační databáze na objekty objektově orientovaného programovacího jazyku. Hlavní výhodou ORM je, že se záznamy z relační databáze lze pracovat jako s objekty a využívat tak možností dědičnosti a polymorfismu. Každý úkon s daty databáze probíhá formou transakce. Protože SQLAlchemy neumožňuje thread-safe přístup, je každé kritické místo zdrojového kódu, kde dochází k přístupu k databázi, ošetřeno jednoduchým zámkem.

Všechny typy dotazů na data, uložená v databázi, jsou zpracovány metodami třídy InitDB, které zavolají další potřebné metody. Při každém spuštění API je nutné provést připojení k databázi. To zajišťuje metoda **___init___**, která je volána při vytváření objektu třídy InitDB. Pro vytvoření spojení s databází je nutné na začátku nadefinovat jeho vlastnosti a typ mapování. V definici připojení je potřeba vyplnit typ, adresu a jméno databáze, přihlašovací údaje a kódování. Dále je nutné nastavit jádro rozhraní, ve kterém je použit nadefinovaný řetězec, a také je zde možné nastavit velikost thread poolu. Zbývá pouze nastavení samotného mapování a tzv. Base, což je třída, která udržuje katalog tříd a tabulek, které mají s Base vztah. V tomto případě byla použita možnost automatického mapování, které automaticky vytváří mapované třídy a vztahy mezi nimi. V tuto chvíli je již možné zavolat metodu **prepare** objektu Base, která provede spuštění mapování. Poslední krokem je vytvořit tzv. Session, která představuje samotné spojení mezi API a databází. [10]

```
self.databasePath = "mysql://root:123456789@localhost/
magmadbcs?charset=utf8&use_unicode=0"
self.engine = create_engine(self.databasePath,
pool_size=1000, pool_recycle=5)
self.Base = automap_base()
self.metadata = MetaData()
self.Base.prepare(self.engine, reflect=True)
self.Session = scoped_session(sessionmaker(bind=self.engine))
```

Pro každé vlákno API, které pracuje s databází, jsou vytvořeny metodou **createTables** objekty tříd, představující tabulky databáze, a metodou **mapTables** se provede jejich namapování.

Všechny tyto třídy představující tabulky databáze mají podobnou strukturu a liší se pouze seznamem názvů sloupců databázových tabulek a jejich zpracováním. Vzorovou třídou je **Scenario**. Obsahuje čtyři základní metody. Úkolwm metody **___init___**, která je volána při vytvoření instance této třídy, je inicializovat základní proměnné a seznam s názvy sloupců databázové tabulky.

CreateTable je metoda, navržená pro vytvoření tabulky v databázi podle vzoru této třídy. Tabulka a její vlastnosti jsou zde předem pevně nadefinované.

```
new = Table(self.tableName, self.metadata,
Column(self.labels[0], INTEGER(unsigned=True),
primary_key=True, autoincrement=True, unique=True,
nullable = False),
Column(self.labels[1], INTEGER(unsigned=True),
nullable = False, unique = True),
Column(self.labels[2], String(256)),
Column(self.labels[2], String(2048)))
```

table_id	ID	Name	Note
1	1	ScenarTest1	
2	2	ScenarTest2	
3	3	ScenarTestCsharp	

Obr. 33. Vizualizace dat uložených v tabulce Sc v databázi magmadb

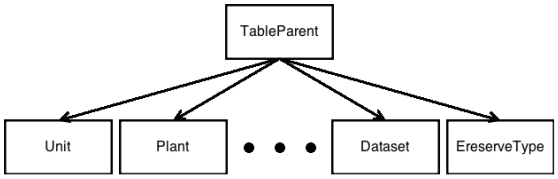
Úkolem metody **addRow** je přidat nový záznam do tabulky v databázi. Jejím jediným vstupním parametrem je slovník, obsahující jako klíče názvy sloupců tabulky a jako hodnoty příslušné údaje k těmto klíčům.

```
session.add(self.rel(**kwargs))
session.commit()
```

Poslední metodou je metoda **updateRow**. Jejími vstupními parametry jsou ID záznamu, uloženého v dané tabulce v databázi, které se má měnit, a slovník. Slovník obsahuje jako klíče názvy sloupců, ve kterých má být provedena změna, a jako hodnoty požadované nové údaje, příslušné daným sloupcům. Tento slovník je v těle metody procházen for cyklem a zpracován konstrukcí elif.

```
row = session.query(self.rel).filter_by(ID = Id).first()
for column in changes:
if column == "ID":
row.ID = changes[column]
elif column == "Name":
row.Name = changes[column]
session.commit()
```

Jediným rodičem všech těchto tříd je třída TableParent, která byla vytvořena pro využití dědičnosti a tím i pro odstranění duplicitních metod. Tyto metody jsou čtyři.



Obr. 34. Ilustrace dědičnosti tříd reprezentujících tabulky databáze

Metoda **setRelation** uloží mapování tabulky, přijaté jako vstupní parametr, do proměnné objektu. Metoda **dropTable** slouží k zahození tabulky z databáze.

Metoda **deleteRow** je navržena pro mazání záznamů z tabulek databáze. Jako vstupní parametr přijímá slovník obsahující parametry záznamu určeného ke smazání.

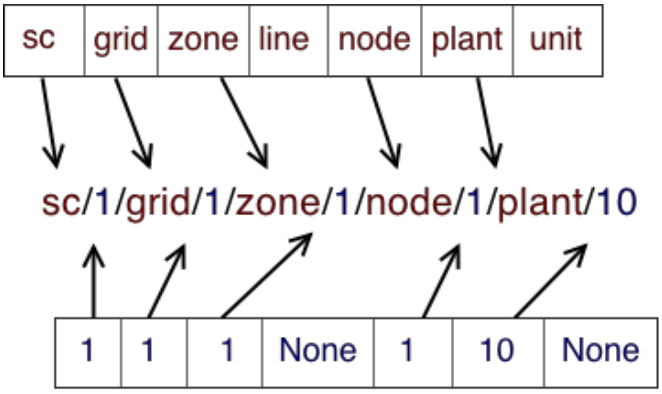
```
session.query(self.rel).filter_by(**delSpecifDict).delete()
session.commit()
```

Poslední a nejsložitější metodou třídy je metoda **toJson**. Vstupními parametry této metody jsou Id požadovaného záznamu, Id scénáře, do kterého záznam patří, název tabulky, ve které se záznam nachází, slovník obsahující jméno a čísla Id všech hierarchických předků modelu přenosové sítě tohoto záznamu a slovník, obsahující další upřesňující údaje, potřebné k získání požadovaných záznamů z databáze.

Metoda se skládá ze dvou částí: stažení dat z databáze a jejich zpracování. Na začátku první části se testuje, zda tabulka, ve které se požadované záznamy nachází, není Node nebo Line. V případě, že se jedná o tabulku uzlů nebo vedení, otestuje se, zda požadované záznamy patří do některé oblasti, nebo patří přímo do sítě. Výsledek testu se uloží do pomocné proměnné a tato informace je následně z původního vstupního parametru metody smazána. Pokud je dotaz na záznam tabulky, která představuje prvek modelu přenosové soustavy, provede se stažení všech časových řad patřících požadovaným záznamům z databáze. Získaná data se uloží do pomocného seznamu.

V dalším kroku dojde ke stažení požadovaných záznamů z tabulky. Kombinace parametrů Id a scId určí, jestli bude z databáze stažen pouze jeden konkrétní záznam s požadovaným Id (Id != None a scId != None), všechny záznamy, které odpovídají scénáři s scId (Id == None a scId != None), všechny záznamy odpovídající Id (Id != None a scId == None) nebo úplně všechny záznamy z tabulky (Id == None a scId == None). Pokud v tabulce nebyl žádný záznam s požadovanými parametry a seznam stažených dat je prázdný, metoda skončí. V opačném případě probíhá zpracování získaných dat.

Zpracování dat probíhá procházením v cyklu for, kdy v každém průchodu dojde ke zpracování právě jednoho záznamu ze seznamu. Uvnitř vnějšího cyklu je další, ve kterém se prochází seznam všech hodnot, které mají být vráceny ve výstupním řetězci, a které jsou pevně definovány v každé třídě, představující tabulku. V tomto vnitřním cyklu dojde nejdříve k testu, zda aktuálně iterovaný název klíče není názvem prvku modelu sítě. Pokud je test kladný, vygeneruje se podle hodnot přijatých jako parametry metody a podle hodnot cizích klíčů, získaných z databáze, cesta k tomuto prvku v souboru formátu HDF5. Princip sestavování cesty je názorně zobrazen na obr. 35. Pokud se nejedná o název prvku modelu sítě, načte se hodnota k tomuto klíči a uloží se i s názvem do výstupního řetězce.



Obr. 35. Princip generování cesty formátu HDF5

```
1: function toJson(Id, scId, datasetRel, table_name, tableParrentsList, kwargs)
   inicializace proměnných
2:     if datasetRel != None then
3:         Načtení všech časových řad do pomocného seznamu z databáze odpovídající
   požadovaným parametrům
4:     end if
5:     if Id != None and scId != None then
6:         Načtení záznamů odpovídající ID a scId z tabulky do pomocného seznamu
7:     else if Id == None and scId != None then
8:         Načtení záznamů odpovídající ID a scId z tabulky do pomocného seznamu
9:     else if Id != None and scId == None then
10:        Načtení záznamů odpovídající ID a scId z tabulky do pomocného seznamu
11:    else
12:        Načtení všech záznamů z tabulky do pomocného seznamu
13:    end if
14:    if query.___contains___(None) then
15:        return None
16:    end if
17:    for record in query do
18:        for column in labels do
19:            for name in tableNames do
20:                if column == name then
21:                    sestavení cesty formátu HDF5 k záznamu
22:                    uložení názvu sloupce a vytvořené cesty do výstupního řetězce
23:                    nenalezeno = false
24:                end if
25:            end for
26:            if nenalezeno == true then
27:                uložení názvu sloupce a jeho hodnoty do výstupního řetězce
28:            end if
29:        end for
30:        if tableName is in tableNames then
31:            for dataset in datasetList do
32:                if dataset.Id = record.Id then
33:                    generování cesty formátu HDF5 k časové řadě
34:                    uložení názvu řady a její cesty do výstupního řetězce
35:                end if
36:            end for
37:        end if
38:    end for
39:    vrácení výstupního řetězce
40: end function
```

Obr. 36. Pseudokód popisující funkci metody toJson

Po skončení vnitřního cyklu je prohledán seznam obsahující časové řady a pokud je nalezena časová řada, náležící k aktuálně iterovanému záznamu, uloží se do pole, které je po skončení prohledávání zapsáno do výstupního řetězce. Názorný popis celé metody je uveden pomocí pseudokódu na obr. 36.

Požadavky na čtení, zápis nebo úpravu dat, uložených v databázi, jsou nejdříve zpracovány třemi metodami třídy InitDB, které dále volají příslušné metody tříd tabulek, ve kterých jsou data uložena, nebo do kterých mají být nová data uložena.

Pro zpracování čtení dat z databáze je navržena metoda **getJson**. Jejími vstupními parametry jsou ID všech tabulek prvků modelu přenosové sítě, řetězec s cestou k požadovanému záznamu v souboru formátu HDF5, která je složená z názvů prvků modelu sítě a jejich ID oddělených lomítkem, a řetězec obsahující název tabulky, ve které se požadovaný záznam nachází. Podle přijatého názvu tabulky je pomocí konstrukce elif zavolána metoda toJson třídy představující požadovanou tabulku.

Pro mazání záznamů z databáze je používána metoda **deleteRecord**. Metoda přijímá jako vstupní parametry ID všech tabulek představujících prvky modelu přenosové sítě, řetězec obsahující název tabulky, ze které se má daný záznam smazat, a proměnnou typu bool, která udává, zda se jedná o časovou řadu nebo ne. Metodu tvoří jednoduchá konstrukce elif, ve které se podle podmínek rozhodne o zavolání metody **deleteRow** instance třídy požadované tabulky. Jako parametr je předáno ID záznamu, který je určen ke smazání.

Metoda **updateRecord** je navržena pro zpracování požadavku na úpravu záznamu v databázi. Přijímá všechny vstupní parametry jako metoda deleteRecord a ještě navíc slovník, obsahující všechny požadované změny daného záznamu. Tělo metody obsahuje jednoduchou elif konstrukci, která zajišťuje volání metody updateRow správného objektu, reprezentujícího cílenou tabulku.

Poslední metodou této třídy je metoda **writeRecord** slouží k ošetření dat, která jsou určena k uložení do databáze, a k zavolání metody addRow instance třídy požadované tabulky. Jejími vstupními parametry jsou ID všech tabulek představujících prvky modelu rozvodné sítě, řetězec s cestou k danému prvku, název tabulky, proměnná typu bool, která udává, zda se jedná o časovou řadu nebo ne, a slovník, obsahující nová data k zapsání. Její tělo je tvořeno konstrukcí elif, které testuje vstupní hodnoty ID a podle nich rozhoduje, o kterou tabulku se jedná.

2.5. Zpracování požadavků na server

Každý požadavek na API je zpracován přes rozhraní REST metodami třídy RunServer. Tyto metody (GET, PUT, POST, DELETE) zachycují jednotlivé typy stejnojmenných požadavků a volají třídy a metody pro jejich zpracování. RunServer zároveň obsahuje metodu ___main___, která slouží jako spouštěč celé API, vytváří objekt třídy TaskDispatcher a volá metody pro vytvoření spojení s databází. Všechna data přijatá v tělech požadavků nebo odesílaná jako výstupní z API jsou ve formátu Json (JavaScript Object Notation).

2.5.1. Požadavek typu GET

Všechny požadavky na získání dat z úložiště jsou typu GET a jsou zpracovány stejnojmennou metodou. Vzorový dotaz na časovou řadu je uveden na obr. 37. Hlavním úkolem této metody je zavolat metodu ProcessRequest třídy GetDispatcher a po jejím skončení rozhodnout o platnosti vracených dat. V metodě ProcessRequest třídy GetDispatcher jsou zpracovány parametry přijatého požadavku ze slovníku params, které jsou uloženy

do pomocných proměnných. V dalším kroku je volána statická metoda pathCreator třídy Path, která z dat přijatého slovníku args sestaví cestu k požadovanému prvku v souboru HDF5.

V případě, že se jedná o dotaz na data v databázi, je volána metoda Read v třídě TaskDispatcher. Tato metoda zajistí volání příslušných metod třídy DatabaseController, jejichž výsledkem je výstupní řetězec ve formátu Json, který obsahuje požadovaná data. Tento řetězec je vrácen do metody GET. Pokud se jedná o dotaz přímo na data časové řady, je volána metoda loadData třídy Aggregation.

```
http://127.0.0.1:1234/data/sc/1/grid/1/zone/1/node/
100000/plant/1/unit/1/delpwr?nSample=100&aggreg=mean
```

Obr. 37. Ukázka požadavku typu GET na data z časové řady

Třída Aggregation je navržena pro zpracování číselných dat načtených z časové řady podle požadavků uživatele. Obsahuje dvě hlavní a tři pomocné metody. Jako první je vždy volána metoda loadData. Na začátku metody se kontroluje, zda přijatá cesta k časové řadě není prázdná a pokud ano, metoda vrátí prázdné pole a skončí. Dále se nastaví volitelné proměnné (počáteční a koncové datum, typ agregace dat, počet vzorků) na předem nastavené hodnoty, pokud nejsou zadány uživatelem. Následně dojde k načtení časové řady ze souboru volání metody getDataSet třídy Controller a k dopočítání zbylých parametrů. Nakonec je volána metoda resampleData, která jako parametry přijímá pole dat, získané z časové řady, a všechny zkontrolované a doplněné parametry.

V metodě resampleData je pole hodnot, získané z časové řady, převzorkováno podle uživatelem požadovaných parametrů. Z každého vzorku a na konci z celé řady jsou vypočítána minima a maxima metodou minmaxFunct a uživatelem požadované agregace (aritmetický průměr, medián) metodou aggregFunction. Metoda vrací pole dat, hodnot, minim, maxim, hodnot po agregaci a počáteční i koncové datum.

Takto zpracované hodnoty jsou přeposlány do metody writeBack třídy Wrap, která je uloží do výstupního řetězce formátu Json, a ten je odeslán z API klientovi. Příklad výstupního řetězce je zobrazen na obr. 38.

```
[{"labs":["Date", "delpwr"],"data":
[[1356998400,[199.87500,199.87500,199.87500]]]},
{"min": [69.62500],"mean": [198.19217],"max": [314.00000],
"dateWindow":[1356998400,1388534400]}]
```

Obr. 38. Ukázka zpracovaných dat zabalených do formátu Json

2.5.2. Požadavek typu DELETE

Metoda DELETE je využívána pro účely mazání záznamů z úložiště. DELETE volá metodu ProcessRequest třídy DeleteDispatcher, ve které je opět sestavena cesta formátu HDF5. Záznam je nejdříve smazán z databáze a následně i ze souboru formátu HDF5. Při smazání záznamu z databáze jsou smazány i všechny jeho hierarchicky podřazené záznamy a stejně tak v souboru.

2.5.3. Požadavek typu POST

Pro vytvoření nového záznamu v úložišti je navržena metoda POST. Stejně jako u předchozích je úkolem této metody zavolat metodu ProcessRequest příslušného dispatcheru, kterým je třída PostDispatcher, a vyhodnotit, zda zápis proběhl úspěšně. V případě úspěšného uložení nového záznamu je volána metoda GET, která vrátí nově zapsaný záznam. V metodě ProcessRequest je přijaté pole args opět zpracováno na cestu formátu hdf5 voláním metody pathCreator a jsou načtena data z těla požadavku. Nejdříve je nový záznam zapsán do databáze a pokud byl zápis úspěšný, provede se i v příslušném souboru formátu HDF5. Vzorové požadavky na uložení nového záznamu jsou uvedeny na následujících obrázcích.

```
Header:
/data/sc/1/Test1
Body:
{
"startTime": 21558185558850000,
"offset": 5451850000,
"data": [548.155, 548.155, 548.155, 548.155, 548.155,
548.155, 548.155, 548.155, 548.155, 548.155, 548.155]
}
```

Obr. 39. Ukázka požadavku na uložení nové časové řady v úložišti

```
Header:
/data/sc
Body:
{
"ID": 15,
"Name": "ScenarioTest",
}
```

Obr. 40. Ukázka požadavku na uložení nového prvku modelu do úložiště

2.5.4. Požadavek typu PUT

Každý požadavek na úpravu dat v úložišti označený jako PUT je zpracován stejnojmennou metodou, jejímž úkolem je zavolat metodu ProcessRequest třídy PutDispatcher a vyhodnotit, zda změna proběhla úspěšně. Cílový záznam je určen cestou sestavenou z přijatého pole args pomocí metody pathCreator stejně jako u zpracování požadavku GET. Nová data, kterými mají být přepsána původní, jsou přijata jako řetězec ve formátu Json v těle požadavku.

Pokud je přijatý požadavek na zápis hodnot do časové řady nebo na přepsání již zapsaných hodnot, je volána metoda save_into_existing třídy Controller a metoda následně skončí. Pokud je ale přijatý požadavek na změnu záznamu v databázi, je volána metoda databaseUpdate třídy TaskDispatcher, která tento požadavek předá metodě v třídě DatabaseController určené pro jeho zpracování. V případě, že dojde ke změně ID prvku modelu sítě nebo se změní její pozice v modelu (změnou cizích

klíčů), je tato změna provedena i v příslušném souboru formátu HDF5 voláním metody changePath třídy Controller. Pokud je požadovaná změna provedena úspěšně, je na daný prvek zavolána metoda GET, jejíž výstup je vrácen API.

```
Header:
    /data/sc/1/grid/1
Body:
    {
     "Name": "RenameTest"
    }
```

Obr. 41. Ukázka požadavku na úpravu záznamu v úložišti

3. Aplikace pro import a export serializovaného scénáře

3.1. Základní popis aplikace

Tato API implementovaná v jazyce c# byla vytvořena pro uložení dat ve formě serializovaných scénářů z výstupu simulátoru elektrických přenosových sítí do navrženého hybridního úložiště a nebo naopak pro přípravu a serializaci dat z úložiště pro vstup simulátoru. V případě nutnosti lze tuto API také využít jako alternativu k webovému klientu, který je taktéž implementován v programovacím jazyce C#.

3.2. Rozhraní pro práci se soubory

Pro práci se soubory formátu HDF5 byla navržena třída HdfController. Jejím účelem je umožnit zápis, čtení a mazání časových řad ze souborů využitím rozhraní HDF5DotNet.

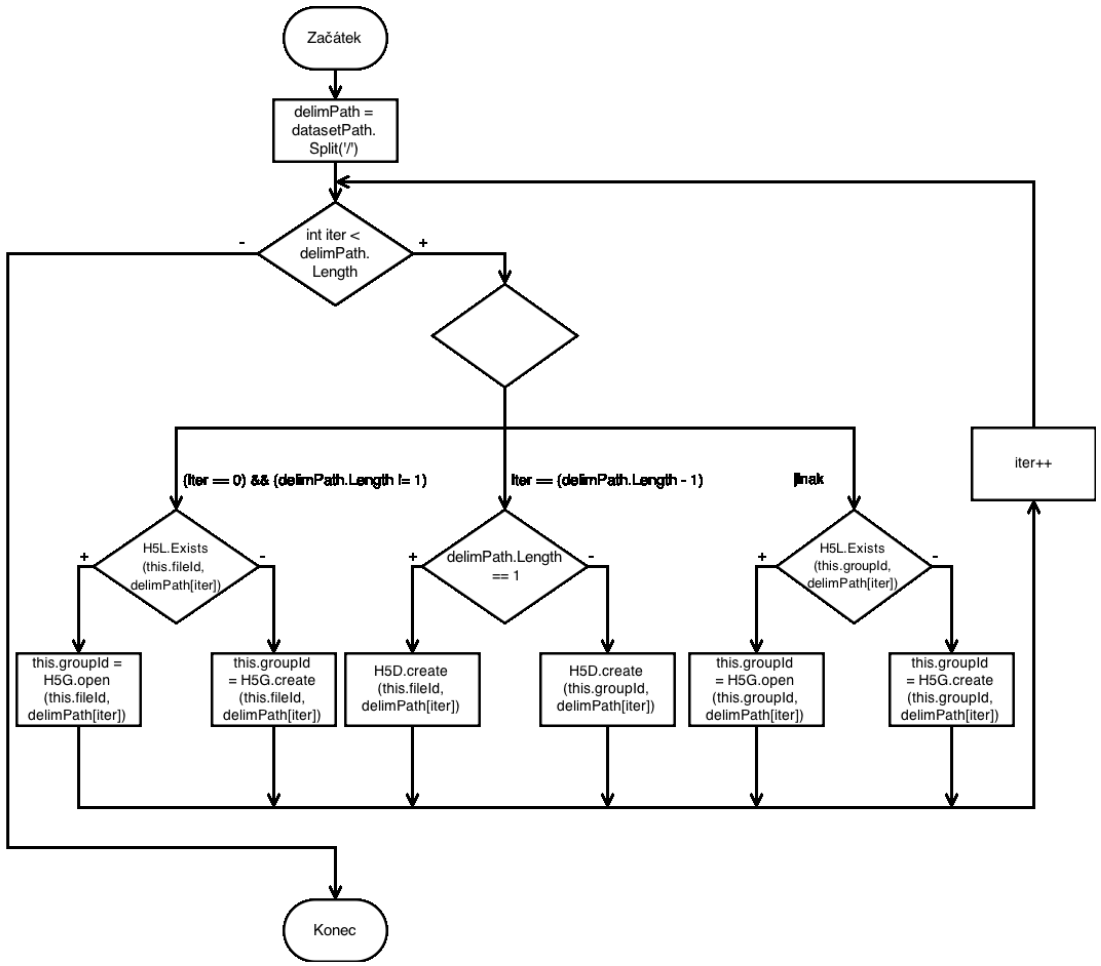
3.2.1. HDF5.NET framework

Rozhraní HDF5DotNet zaobaluje mechanismy knihovny HDF5 pro využití .NET aplikací. Je napsáno v jazyce C++/CLI a využívá P/Invoke mechanismy knihovny .NET. Oproti frameworku H5Py není cílem HDF5DotNetu poskytovat objektově orientované rozhraní, ale pouze zpřístupnit funkce knihovny HDF5.

3.2.2. Zápis dat do souboru

Pro zápis nových časových řad byla navržena metoda Write. Vstupními parametry metody jsou pole hodnot k uložení do nové časové řady, vnitřní cesta v souboru HDF5, kam má být nová řada zapsána, a atributy řady (počáteční datum, offset dat hodnot a implicitní hodnota). Oproti rozhraní H5Py nepodporuje HDF5DotNet možnost zapsání časové řady při zadání řetězce s celou cestou, proto je nutné tento vstupní parametr rozdělit na pole jednotlivých částí a v pomocí cyklu jej projít.

Uvnitř cyklu dochází k větvení do tří možných situací. První z nich může nastat při prvním průchodu cyklu, pokud je velikost pole s rozdělenou cestu vetší než jedna. Zde se testuje, jestli název skupiny v této cestě v souboru už neexistuje. Pokud ano, otevře se, pokud ne, vytvoří se nová skupina. Jako id nadřazené skupiny se použije id souboru. Další možností je, když se iteruje poslední prvek pole s cestou. V tomto případě se testuje, jestli není aktuálně iterovaný prvek zároveň jediný v poli. Pokud ano, zapíše se do souboru a jako id nadřazené skupiny se použije id souboru, pokud není, také se zapíše, ale jako id nadřazené skupiny se použije id skupiny z předchozího průchodu cyklu. Pokud není splněna ani jedna z těchto podmínek, nastává poslední možnost, při které se vytvoří (nebo pokud již existuje otevře) skupina s názvem, který udává aktuální prvek iterovaného pole.



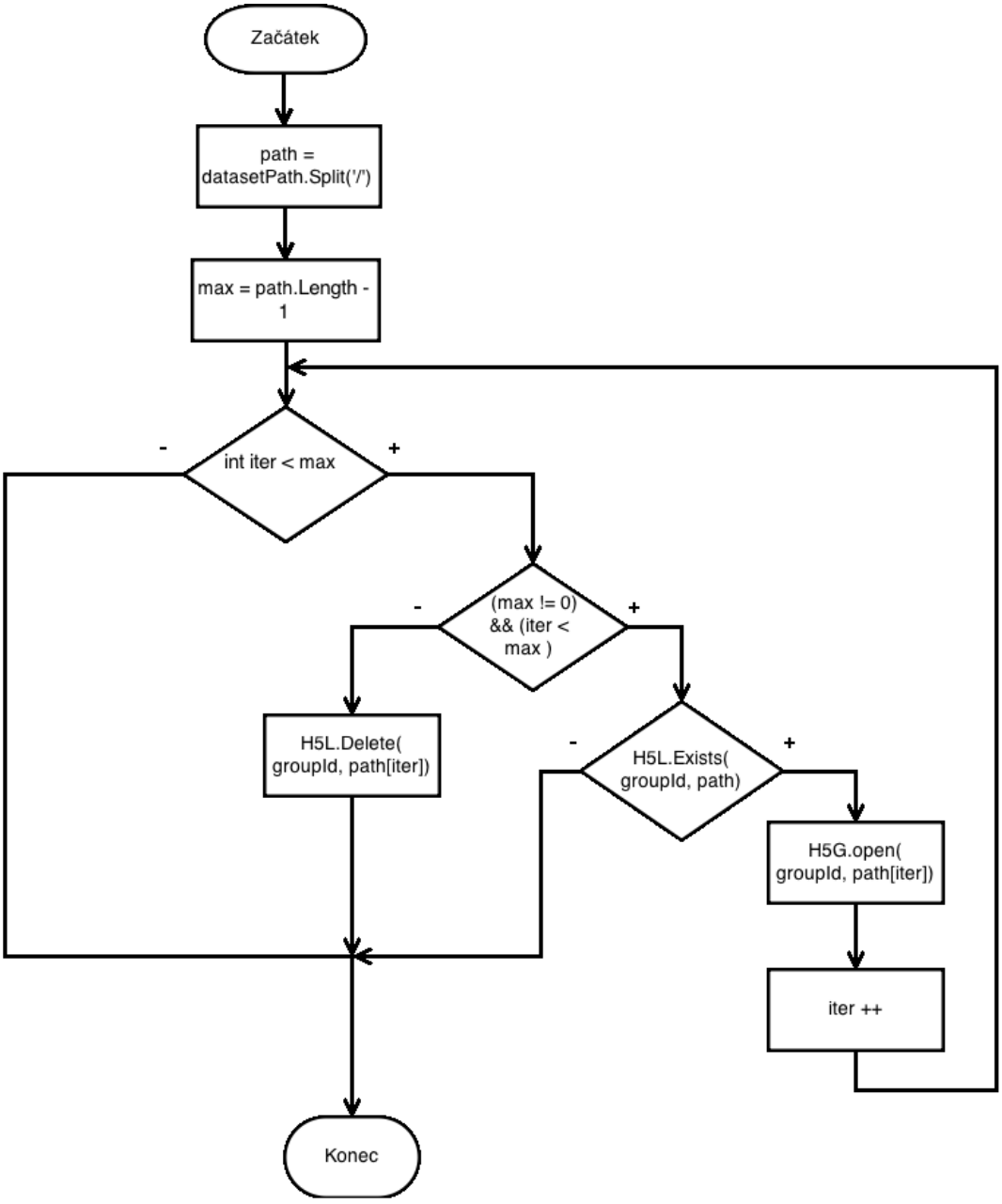
Obr. 42. Zjednodušený vývojový diagram metody Write třídy HdfController

3.2.3. Čtení ze souboru

Čtení časových řad zajišťuje metoda Read. Jediným vstupním parametrem metody je vnitřní souborová cesta k požadované časové řadě. Na začátku metody se tato cesta rozdělí a uloží do pole, které se v cyklu postupně projde. Ve všech průchodech cyklu, kromě posledního, dochází k postupnému otevírání skupin, uložených v poli cesty. Pokud otevíraná skupina neexistuje, metoda skončí a vrátí hodnotu null. V posledním průchodu cyklu dojde k načtení časové řady, získání jejích atributů, podle kterých je alokováno pole pro uložení hodnot z časové řady, a která je vrácena metodou.

3.2.4. Mazání ze souboru

Pro potřeby mazání časových řad ze souboru je navržena metoda Delete. Stejně jako Read přijímá jako vstupní parametr cestu k časové řadě, která má být smazána. Tělo metody je taktéž stejné jako u Read, jediný rozdíl je v posledním průchodu cyklu, kdy místo otevírání časové řady dochází k jejímu smazání, respektive stejně jako v API pro obluhu úložiště dojde k odstranění všech odkazů na časovou řadu a ta se stává nepřístupnou.



Obr. 43. Zjednodušený vývojový diagram metody Delete třídy HdfController

3.3. Objektově relační přístup k databázi

Následující sekce popisuje použité rozhraní pro práci s databází magmadb, navržené třídy tabulek a třídy s jejich mapováním.

3.3.1. Fluent NHibernate framework

Pro přístup k databázi magmadb je použito rozhraní Fluent NHibernate. Je to open-source objektově relační mapování pro aplikace v jazyce C# využívající knihovny .NET. Oproti klasickému NHibernate umožňuje vytvářet mapování pomocí lambda výrazů v jazyce C# bez nutnosti vytvářet mapování v souborech typu XML. [11]

3.3.2. Třídy databázových tabulek a jejich mapování

Stejně jako v API pro obsluhu datového úložiště jsou zde navrženy třídy, odpovídající tabulkám databáze. Každá taková třída obsahuje deklaraci proměnných, představujících sloupce tabulky, třídy pro přístup k těmto proměnným (gettery a settery) a metodu Get, jejíž funkcí je získání záznamu s požadovanými parametry z této tabulky v databázi.

Ke každé třídě představující tabulku databáze existuje třída, obsahující mapování. V těchto třídách jsou nastaveny všechny vlastnosti tabulky a jejích sloupců jako jména, unikátní klíče a vlastnosti ukládaných hodnot.

3.4. Import serializovaného scénáře

Pro účely uložení odsimulovaného serializovaného scénáře do úložiště je navržena metoda ExportSerialized třídy ImportScenario.

```
1: function ImportSerialized(scenarioFile)
2:     Vytvoření a otevření souboru formátu HDF5 pro nový scénář
3:     Načtení serializovaného scénáře
4:     Uložení scénáře do databáze
5:     for Procházení přes všechny prvky Config do
6:         Uložení záznamu do databáze
7:     end for
8:     Uložení sítě do databáze
9:     Ukládání časových řad sítě do souboru formátu HDF5
10:    for Procházení přes všechny oblasti do
11:        Uložení oblasti do databáze
12:        Ukládání časových řad oblasti do souboru formátu HDF5
13:    end for
14:    for Procházení přes všechny uzly do
15:        Uložení uzlu do databáze
16:        Ukládání časových řad uzlu do souboru formátu HDF5
17:        for Procházení přes všechny elektrárny do
18:            Uložení elektrárny do databáze
19:            Ukládání časových řad elektrárny do souboru formátu HDF5
20:            for Procházení přes všechny bloky do
21:                Uložení bloku do databáze
22:                Ukládání časových řad bloku do souboru formátu HDF5
23:            end for
24:        end for
25:    end for
26:    for Procházení přes všechny vedení do
27:        Uložení vedení do databáze
28:        Uložení časových řad vedení do souboru formátu HDF5
29:    end for
30: end function
```

Obr. 44. Pseudokód popisující funkci metody ImportSerialized

Na začátku metody je vytvořen soubor formátu HDF5, do kterého budou uloženy všechny časové řady zpracovávaného scénáře. V dalším kroku je načten serializovaný scénář ze souboru, který je postupně ukládán do úložiště. Nejdříve jsou uloženy všechny veličiny Configu, dále síť a všechny oblasti. Následně jsou postupně v cyklu ukládány uzly, k nim příslušné elektrárny a jejich bloky. Nakonec jsou uloženy všechny vedení. Časové řady jsou do souboru ukládány průběžně při zpracování jednotlivých prvků.

Při návrhu metody bylo zavedeno několik zjednodušení a změn oproti prvotnímu návrhu pro zrychlení metod importu a exportu. Soubory formátu HDF5 se již nezavírají po každém přístupu, ale zavírají se až na konci metody. Při zápisu do těchto souborů již neprobíhá test existence skupiny pomocí try-catch bloků, ale pomocí podmínky H5L.Exists a dále byly odstraněny informační výpisy.

3.5. Export serializovaného scénáře

Pro načtení scénáře z úložiště v serializovaném formátu pro simulátor přenosové sítě slouží metoda ExportSerialized třídy ExportScenario. Metoda je přehledně popsána pomocí pseudokódu na obr. 45. Na začátku jsou načtena všechna data odpovídajícího scénáře z tabulek, které představují prvky modelu přenosové soustavy, a data z tabulky časových řad. Získané záznamy jsou uloženy do pomocných seznamů.

V další části metody jsou tyto seznamy procházeny a zpracované záznamy jsou ukládány do objektů serializovaných tříd. Jako první jsou serializována data z tabulky Config, dále Sc, Grid a Zone. V dalším kroku jsou v cyklu serializovány uzly z tabulky Node. Při každém průchodu cyklu jsou opět v cyklu serializovány elektrárny z tabulky Plant, které náleží do daného uzlu, a stejně tak při každém průchodu cyklu elektráren jsou procházeny a serializovány bloky z tabulky Unit. Na každém konci cyklu je příslušný serializovaný prvek vložen do svého nadřazeného serializovaného prvku. Jako poslední je procházen seznam vedení tabulky Line.

Po projití a serializaci všech záznamů z pomocných seznamů je vzniklý serializovaný scénář, do kterého jsou vloženy všechny ostatní serializované objekty, uložen do souboru a připraven pro simulace. Pro zrychlení exportu a zmenšení velikosti vstupních dat pro simulátor jsou serializovány a exportovány pouze záznamy, které jsou označeny jako aktivní (Active == True).

```
1: function ExportSerialized(scenarioId, fileName)
2:     Stažení dat z databáze a jejich uložení do pomocných seznamů
3:     Serializace záznamu z tabulky Config
4:     Serializace záznamu z tabulky Grid
5:     Uložení časových řad daného prvku do serializovaného záznamu
6:     for Procházení přes všechny oblasti v seznamu do
7:         Serializace záznamu z tabulky Zone
8:         Uložení časových řad daného prvku do serializovaného záznamu
9:         Uložení serializované oblasti do serializované sítě
10:    end for
11:    for Procházení přes všechny uzly v seznamu do
12:        Serializace záznamu z tabulky Node
13:        Uložení časových řad daného prvku do serializovaného záznamu
14:        for Procházení přes všechny elektrárny v seznamu do
15:            Serializace záznamu z tabulky Plant
16:            Uložení časových řad daného prvku do serializovaného
    záznamu
17:            for Procházení přes všechny bloky v seznamu do
18:                Serializace záznamu z tabulky Unit
19:                Uložení časových řad daného prvku do serializovaného
    záznamu
20:                Uložení serializovaného bloku do serializované elektrárny
21:            end for
22:            Uložení serializované elektrárny do serializovaného uzlu
23:        end for
24:        Uložení serializovaného uzlu do serializované oblasti a sítě
25:    end for
26:    for Procházení přes všechny uzly v seznamu do
27:        Serializace záznamu z tabulky Line
28:        Uložení časových řad daného prvku do serializovaného záznamu
29:        Uložení serializovaného vedení do serializované oblasti a sítě
30:    end for
31:    Uložení serializované elektrárny do serializovaného uzlu
32:    Serializace scénáře a uložení serializovaných oblastí a sítě do serializovaného
    scénáře
33:    Uložení serializovaného scénáře do souboru
34: end function
```

Obr. 45. Pseudokód popisující funkci metody ExportSerialized

4. Vyhodnocení rychlosti zpracování serializovaného scénáře

Při vyhodnocování rychlosti zpracování importu a exportu byly použity tři serializované scénáře s různou velikostí a složitostí. Nejmenší z nich je jednoduchý tří uzlový, druhý reprezentuje rozvodnou síť na území České republiky a třetí a největší reprezentuje rozvodnou síť na území části Evropy. Počty prvků jednotlivých serializovaných scénářů jsou uvedeny v tab. 29. Měření bylo provedeno na notebooku s procesorem Intel Core i5-2430M 2,4 GHz, pamětí RAM 8 GB DDR3 a s klasickým pevným diskem.

Importovaný typ prvků / Scénář	**Tří uzlový scénář [ks]**	**Scénář ČR [ms]**	**Scénář EU [ms]**
Sítě	1	1	1
Oblasti	1	5	27
Uzly	3	31	549
Vedení	3	54	1527
Elektrárny	10	61	334
Bloky	10	110	2543
Konfigurace	48	44	48
Časové řady	103	1080	24208

Tab. 29. Počty prvků v jednotlivých serializovaných scénářích

4.1. Import scénáře do úložiště

V této sekci jsou uvedeny naměřené a vypočítané časy importu serializovaného scénáře do úložiště. Tab. 30 obsahuje průměrné časy s jejich průměrnými odchylkami importu jednotlivých záznamů všech prvků modelu přenosové sítě a dalších částí metody, potřebných pro úspěšný import dat. V tab. 31 jsou zaznamenány průměrné časy a jejich průměrné odchylky importu všech záznamů jednotlivých prvků.

Importovaný prvek / Scénář	**Tří uzlový scénář [ms]**	**Scénář ČR [ms]**	**Scénář EU [ms]**
Vytvoření a otevření souboru formátu HDF5	45,7±15,1	97,0±0,0	78,0±0,0
Vytvoření tabulek v databázi a uložení pomocných záznamů	9848±288	9728±0.0	10147±0.0
Vytváření cizích klíčů	35968±2202	34116±0,0	34049±0,0
Import cesty k souboru do databáze	48,7±16,1	39,0±0,0	42,0±0,0
Načtení serializovaného scénáře ze souboru	158 ±14,6	907±0,0	3083±0,0
Import scénáře do databáze	41,8±5,3	49,0±0,0	52,0±0,0
Import sítí do databáze	116±1,7	116±0,0	117±0,0
Import oblastí do databáze	99,5±28,8	92,0±25,2	108±29
Import uzlů do databáze	246±55	220±46	233±50,2
Import vedení do databáze	140±21	129±32,5	130±30,3
Import elektráren do databáze	121±86	63,1±34,5	76,8±51,1
Import bloků do databáze	433±78	352±61	386±80
Import konfigurace do databáze	41,7±9,4	42,2 ±7,8	40,0±11,5
Import časové řady do databáze a uložení číselné řady do souboru HDF5	52,3±14,4	47,6±16,5	48,8±17,4
Celkový čas metody ImportScenario	55036 ±2196	103796 ±0,0	1404288 ±0,0

Tab. 30. Průměrné časy importu jednotlivých prvků do úložiště

Importovaný typ prvků / Scénář	**Tří uzlový scénář [ms]**	**Scénář ČR [ms]**	**Scénář EU [ms]**
Import scénářů do databáze	41,8±5,3	49,0±0,0	52,0±0,0
Import sítí do databáze	116±1,7	116±0,0	117±0,0
Import oblastí do databáze	99,5±28,8	194±44	2922±0,0
Import uzlů do databáze	738±65	3306±53	127780±0,0
Import vedení do databáze	420±40	3482±217	199325±0,0
Import elektráren do databáze	1209±135	1908±363	25655±0.0
Import bloků do databáze	4329±314	19348±37	998540±0.0
Import konfigurací do databáze	2000±164	475±62	40,0±11,5
Import časových řad do databáze a jejich uložení do souboru formátu HDF5	626±74	13090±371	631438±0,0

Tab. 31. Průměrné časy importu všech prvků daného typu do úložiště

4.2. Export scénáře z úložiště

Tato sekce uvádí naměřené a vypočítané časy serializace a exportu scénáře z úložiště do souboru. V následující tabulce jsou zobrazeny průměrné časy s průměrnými odchylkami exportu jednotlivých záznamů všech prvků modu přenosové sítě a jejich časových řad. Čas byl měřen v řádech tisícin milisekundy a serializace záznamů tabulky Config trvala kratší dobu než jedna tisícina milisekundy, z tohoto důvodu jsou hodnoty časů exportu těchto záznamů nulové. V případě exportu scénáře EU bylo provedeno měření pouze jednou a proto jsou uvedeny pouze naměřené hodnoty bez odchylek.

Exportovaný prvek / Scénář	**Tří uzlový scénář [ms]**	**Scénář ČR [ms]**	**Scénář EU [ms]**
Otevření souboru formátu HDF5	42,6±23,5	33,0±10,0	63,3±41,8
Stažení dat z databáze	3527±39	3562±47	4429±378
Serializace scénáře a uložení do souboru	84±2.0	640±13	800±16
Serializace sítí	38,3 ±0,4	39,3±13,1	41±2,6
Serializace oblastí	4,00±0,00	3,40±1,92	0,30±0,48
Serializace uzlů	5,89±5,63	2,01±0,82	0,52±0,59
Serializace vedení	5,56±2,30	3,56±0,86	1,98±0,24
Serializace elektráren	9,47±7,12	10,5±7,2	5,82±5,49
Serializace bloků	4,17±2,07	2,34±1,10	1,02±0,16
Serializace konfigurací	0,0±0,0	0,0±0,0	0,0±0,0
Serializace a export časové	0,56±0,56	0,44±0,34	0,14±0,13
Celkový čas metody ExportScenario	3857±62	5408±62	13348±546

Tab. 32. Tabulka průměrných časů exportu jednotlivých prvků do úložiště

Průměrné časy s průměrnými odchylkami všech záznamů jednotlivých prvků modelu přenosové sítě a všech časových řad uvádí tab. 33.

Importovaný typ prvků / Scénář	**Tří uzlový scénář [ms]**	**Scénář ČR [ms]**	**Scénář EU [ms]**
Serializace scénářů	41,8±5,3	49,0±0,0	52,0
Serializace sítí	38,3±0,4	39,3±13,1	45
Serializace oblastí	4,0±0,0	3,40±1,92	24
Serializace uzlů	17,7±1,1	16±0,8	855
Serializace vedení	14,7±0,4	143±15	9073
Serializace elektráren	94,7±2,9	475±47	5834
Serializace bloků	33,3±0,4	148±6	7580
Serializace konfigurací	0.00	0.00	0,00
Serializace časových řad	1,88±0,56	37,00±0,00	146

Tab. 33. Průměrné časy exportu všech prvků daného typu do úložiště

4.3. Vyhodnocení

Naměřená doba serializace a exportu scénáře z úložiště splnila předběžná očekávání. Největší část představuje stažení všech dat odpovídajícího scénáře z databáze, avšak tento čas již nelze nijak urychlit při daných možnostech. Při importu serializovaného scénáře do úložiště trvá poměrně dlouhou dobu vytvoření a ověření existence tabulek a cizích klíčů v databázi, které zajišťuje framework FluentNhibernate. Použití jiného frameworku by pravděpodobně urychlilo tento krok. Pro další snížení času importu a exportu by měl být zdrojový kód analyzován profilerem. Množství drobných optimalizací bylo provedeno již při návrhu těchto dvou metod.

5. Závěr

Cílem této práce bylo navrhnout nové úložiště pro vstupní a výstupní data simulátoru přenosové sítě, které splní současné požadavky na uložení velkého množství dat. Dalším cílem bylo navrhnout API, které bude toto úložiště spravovat a umožňovat přípravu dat pro jejich vizualizaci. A posledním cílem bylo navrhnout API, které umožní import dat v podobě serializovaného scénáře ze simulátoru do úložiště a také naopak exportovat data z úložiště do souboru pro simulátor. Všechny tyto cíle se podařilo v průběhu práce splnit. Výsledkem je provozuschopné úložiště a plně funkční naimplementované API, jejichž funkčnost byla ověřena úspěšným importem tří dodaných serializovaných scénářů do úložiště, jejichž zpracováním pro vizualizaci a následným exportem z úložiště do souborů.

Hlavní předností navrženého úložiště je oddělenost prvků modelu přenosové soustavy uchovávaných v databázi od číselných řad uložených v souborech typu HDF5. To umožňuje přehlednější orientaci ve struktuře uložených scénářů a především efektivní práci s časovými řadami. Velkou výhodou využití souborů typu HDF5 jsou jednoduché možnosti editace uložených časových řad.

Zdrojový kód programu byl již částečně optimalizován, čímž byly splněny požadavky na rychlý import a export scénářů. Dalšího zrychlení činnosti obou API je možné dosáhnout jejich analýzou programem určeným k profilování zdrojového kódu a optimalizací časově náročných míst kódu. Vývoj tohoto hybridního úložiště bude probíhat i nadále a je plánováno jeho začlenění do nově vyvíjené aplikace určené pro rozvoj přenosové sítě.

Literatura

[1] *Hierarchical Structure*. 2015. URL: http://neondataskills.org/HDF5/About/ (cit. 22. 04. 2015).

[2] *Introduction to HDF5*. 2014. URL: https://www.hdfgroup.org/HDF5/doc/H5.intro.html#Intro-WhatIs (cit. 22. 04. 2015).

[3] *About MariaDB*. 2015. URL: https://mariadb.org/en/about/ (cit. 22. 04. 2015).

[4] *Sysbench OLTP: MySQL-5.6 vs. MariaDB-10.0*. 2012. URL: https://blog.mariadb.org/category/performance/page/3/ (cit. 22. 04. 2015).

[5] *H5Py documentation*. 2014. URL: http://docs.h5py.org/en/latest/ (cit. 27. 04. 2015).

[6] *SQLAlchemy introduction*. 2014. URL: http://www.sqlalchemy.org/ (cit. 27. 04. 2015).

[7] *Numpy main page*. 2014. URL: http://www.numpy.org/ (cit. 27. 04. 2015).

[8] *What is CherryPy*. 2014. URL: http://www.cherrypy.org/ (cit. 27. 04. 2015).

[9] *Deleting a Dataset from a File and Reclaiming Space*. 2014. URL: https://hdfgroup.org/HDF5/doc/UG/10_Datasets.html (cit. 02. 05. 2015).

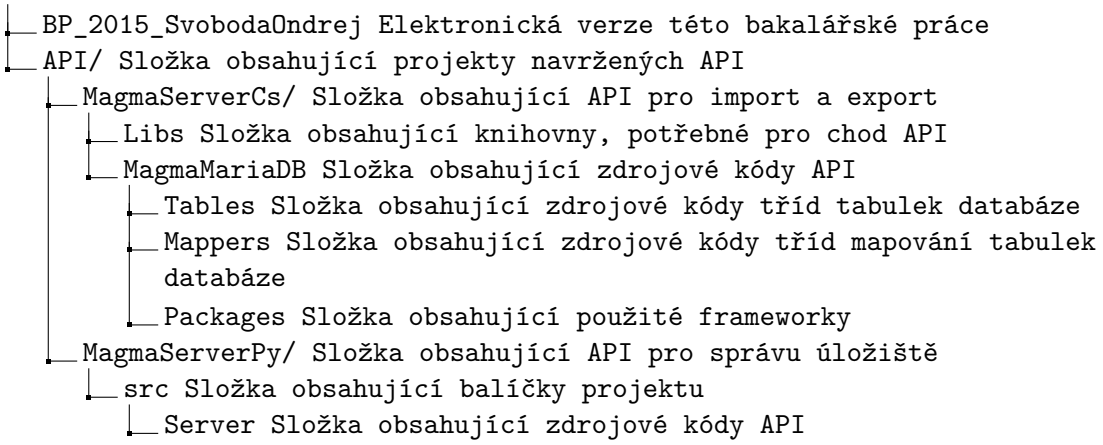
[10] *Automap*. 2015. URL: http://docs.sqlalchemy.org/en/latest/orm/extensions/automap.html (cit. 02. 05. 2015).

[11] *Fluent NHibernate*. 2012. URL: http://www.fluentnhibernate.org/ (cit. 07. 05. 2015).

Příloha A.

Obsah přiloženého CD

Na obr. 46 je uvedena adresářová struktura přiloženého CD.



Obr. 46. Adresářová struktura přiloženého CD