Univerzita Hradec Králové

Fakulta informatiky a managementu

Katedra informatiky a kvantitativních metod

Optimalizace ukládání časových řad v energetice

Diplomová práce

Autor: Lukáš Chvátal
Studijní obor: Aplikovaná informatika

Vedoucí práce: Ing. Barbora Tesařová, Ph.D.

Hradec Králové duben 2020

Prohlášení:

Prohlašuji, že jsem diplomovou práci zpracoval samostatně a s použitím uvedené literatury.

V Hradci Králové dne 29.4.2021 Lukáš Chvátal

Poděkování:

Děkuji vedoucí práce paní Ing. Barboře Tesařové, Ph.D. za přínosné konzultace a připomínky během zpracování. Dále bych chtěl poděkovat panu Ing. Lubomíru Andrle, za jeho čas a přínosné konzultace ve společnosti Unicorn.

Anotace

Diplomová práce je věnována optimalizaci datového modelu časových řad v relační databázi Oracle, jež se nachází v informačním systému vyvíjeném společností Unicorn. Cílem práce je analýza aktuálního stavu a navržení nového modelu, který je více efektivní z pohledu datové náročnosti a výkonu dotazování. Motivaci pro nutné zlepšení představuje blížící se změna severského energetického trhu, která s sebou přinese čtyřnásobný objem vstupních dat do systému.

Teoretická část práce se zabývá rozborem severského energetického trhu, na jehož základě jsou stanovené požadavky na časové řady. Poté následuje průzkum optimalizačních technik a správných praktik návrhu databáze Oracle. Podle získaných poznatků z teoretické části a analýzy nedostatků stávajícího modelu je následně proveden návrh nového modelu. Závěrečná část práce je dedikována srovnání obou datových modelů v několika testovacích scénářích.

Annotation

Title: Time series storage optimization for energetics

The master's thesis is devoted to the optimization of the time series data model in the Oracle relational database, which is utilized in the information system developed by Unicorn. The aim of the thesis is to analyze the existing solution and design a new one, which is more efficient in terms of data storage and query performance. The motivation for the necessary improvement is the impending change in the Nordic energy market, which will quadruple the volume of input data to the system.

The theoretical part deals with the analysis of the Nordic energy market, on the basis of which the time series requirements are described. This is followed by a survey of optimization techniques and good Oracle database design practices. According to the knowledge gained from the theoretical part and the analysis of the shortcomings of the existing model, a proposal for a new model is made. The final part of the thesis is dedicated to the comparison of both data models in several test scenarios.

Obsah

Seznam obrázků

Seznam tabulek

Seznam kódů

Seznam grafů

Úvod

Jedním ze specifik energetiky je neskladovatelnost elektrické energie ve velkých kvantitách, ideálně by tak vyprodukovaná elektřina měla být spotřebována ihned. Tento stav může nastat pouze za předpokladu, že je energetický trh v rovnováze, tedy když se nabídka a poptávka energie rovná. Udržení rovnováhy vyžaduje kooperaci účastníků trhu, skrze takzvané balancování, v rámci kterého dochází mezi účastníky k obchodům s energií. I při sebelepší snaze o vybalancování trhu však mohou vzniknout nežádané odchylky.

O výpočet a následnou fakturaci odchylek, se na severském energetickém trhu stará systém dodávaný společností Unicorn Systems a.s. (dále jen Unicorn). Data v tomto systému jsou uchovávaná v časových řadách. Vstupní data systému aktuálně představují hodinová obchodní data, která jsou reportovaná účastníky celého severského trhu, což představuje značný přírůstek dat v rámci dne. Jako úložiště je zde použita relační databáze Oracle, jejíž datový model od spuštění systému prošel již drobnými úpravami, ale pro dosavadní účely byl bez větších výhrad dostačující.

V nadcházejících letech však na severském energetickém trhu dojde k drastické změně. Místo reportování hodinových obchodních dat, je v plánu přechod na data čtvrthodinová, což pro systém představuje čtyřikrát větší objem vstupních dat. Tato změna s sebou přinese negativní dopady na datový model v podobě čtyřnásobného nárůstu datové náročnosti ukládaných dat a tím i potencionální úbytek výkonu samotného úložiště. Migrace úložiště do jiné technologie by představovala příliš velký zásah do existující infrastruktury, a tudíž není vhodným východiskem.

Cílem této práce je tedy analýza a následná optimalizace stávajícího datového modelu časových řad, za účelem mitigace negativních dopadů několikanásobně většího objemu vstupních dat po přechodu na reportování čtvrthodin. Zejména je zde kladen důraz na snížení datové náročnosti ukládaných dat, jelikož velikost databáze produkčního prostředí se již nyní pohybuje v řádech TB.

První kapitola práce je věnována obecnému pohledu na časové řady a přiblížení jejich využití skrze analýzu. Druhá kapitola se zabývá rozborem severského

energetického trhu a jeho specifik, pro jednoznačné stanovení požadavků na časové řady v systému. Třetí a poslední teoretická kapitola obsahuje průzkum různých optimalizačních technik a soupis správných praktik návrhu relační databáze Oracle. Na začátku kapitoly je představena i struktura samotné databáze, jelikož některé optimalizační techniky jsou na ní přímo závislé.

Čtvrtá a zároveň první praktická kapitola práce je věnována analýze stávajícího datového modelu, na základě které jsou vytyčené jeho nedostatky a uvedené možné vylepšení. V páté kapitole je proveden návrh nového datového modelu, jež napravuje stanovované nedostatky původního modelu. Šestá a poslední praktická kapitola obsahuje srovnání starého a nového modelu napříč několika disciplínami, za účelem ověření úspěšnosti návrhu.

1 Časová řada

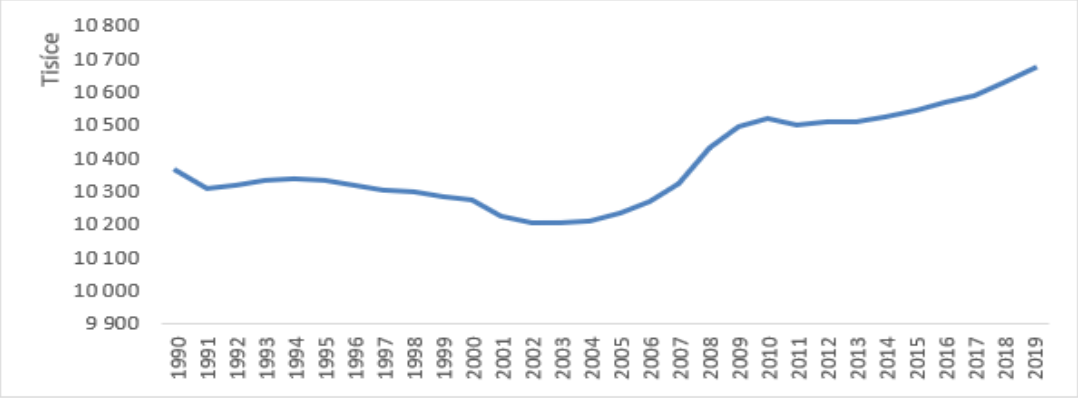
Časovou řadou rozumíme chronologickou posloupnost měření hodnoty vybraného parametru, či více parametrů v časových intervalech. Měření se většinou provádí ve stejně dlouhých časových intervalech, ale není to pravidlem. Například tedy každou minutu, hodinu, nebo také nepravidelně, a to třeba v okamžiku výskytu sledované události. Hodnotu měřeného parametru velmi často představuje číslo, může být ale libovolného typu. Časové řady obvykle obsahují data, která po prvotním uložení nejsou již dále modifikovaná. (1)

1.1 Druhy časových řad

Časové řady se člení dle charakteru sledovaného parametru na: (1)

- **okamžikové** – hodnota parametru v určitému okamžiku, například aktuální rychlost automobilu
- **intervalové** – hodnota parametru závisející na délce zvoleného intervalu, například objem vyrobené elektřiny v rámci dne

Příkladem časové řady může být počet obyvatel České republiky napříč roky. K měření středního stavu počtu obyvatel dochází každý rok na přelomu měsíce června a července. Jde tedy o okamžikovou časovou řadu s pravidelnou roční periodou, u které nedochází k modifikaci již získaných hodnot (2). Naměřené hodnoty této časové řady reprezentuje následující graf.



Graf 1: Vývoj počtu obyvatel ČR v letech 1990 až 2019
Zdroj: vlastní zpracování, data (3)

1.2 Využití časových řad

Ukládání dat do časové řady se hodí v situacích, kdy chceme sledovat změnu parametru v čase a nestačí nám pouze aktuální stav (4). Tento princip sběru dat však není žádnou novinkou digitální doby a je využíván již stovky let. Například v podobě lodních deníků, do kterých v libovolných intervalech kapitáni lodí zaznamenávali různé parametry, jako jsou kupříkladu povětrnostní podmínky během plavby. Takto vedené deníky byly následně použity v 19. století, jako základ pro nalezení optimálních námořních tras, které v některých případech zkrátily čas plavby i o několik dní (1).

Novodobým ekvivalentem lodních deníků je černá skříňka v leteckém průmyslu, jejímž úkolem je několikrát za vteřinu zaznamenávat parametry letu, které poskytují senzory z různých částí letadla. Tyto data nabízí velice detailní přehled vývoje parametrů v čase, což hraje klíčovou roli při vyšetřování příčin případné havárie nebo poruchy letadla. Data však nejsou využívaná pouze pro rekonstrukci událostí předcházejících poruchu. Podobně jako u lodních deníků, jsou data vhodná i pro zlepšení a monitorování různých aspektů letadla, například pro snížení spotřeby paliva nebo sledování opotřebení dílů, za účelem nalezení optimálního intervalu výměny. (1)

Počet případů, kdy je vhodné ukládat data jako časovou řadu je nespočet, využití lze najít napříč různými oblastmi, od medicíny, až po počasí (5). Aktuálním příkladem může být rozmach Internet of things zařízení, které produkují enormní množství dat. Sběr těchto dat do časové řady se na první pohled nemusí zdát přínosný, jejich analýza však může odhalit skrytý potenciál. Časové řady totiž nabízí krom otisku minulosti daleko více využití, jelikož mohou být užitečné například pro odhalení opakujících se úseků nebo trendů (1).

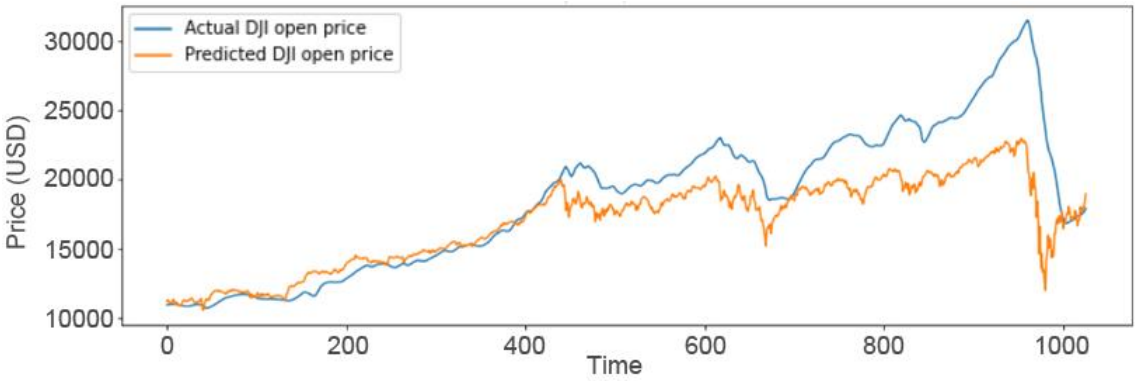
1.3 Analýza časových řad

Analýza časových řad představuje extrakci smysluplných souhrnů a statistických informací z dat časové řady. Účelem je snaha o nalezení modelu, který co nejblíže

popisuje analyzovaná data. Na základě vhodně zvoleného modelu poté můžeme například predikovat budoucí vývoj nebo doplnit chybějící úseky v časové řadě. (5)

Analýza časových řad jakožto disciplína je velice rozsáhlá a svými počátky sahá až do počátku 20. století. K obrovským pokrokům však došlo teprve v posledních desetiletích a to díky rapidnímu nárůstu výpočetního výkonu, který umožňuje mnohonásobně rozsáhlejší a kvalitnější sběr dat. Neboli, to, co dříve musel člověk měřit manuálně, nyní zvládnou senzory automaticky a s lepší přesností. Objem sbíraných dat tedy každým rokem raketově roste, což s sebou přináší vyšší nároky na analýzu, ale i samotné ukládání časových řad. V tomto směru představuje velký posun kupředu rozvoj technik strojového a hlubokého učení, jež jsou dnes pro analýzu hojně využívané. Osvědčily se zde především neuronové sítě a z nich vycházející frameworky, které celý proces značně zjednodušují. (5)

Typický příklad analýzy časových řad představuje následující graf, kde je vyobrazena predikce vývoje ceny akcií společnosti Dow Jones po otevření burzy v jednotlivých dnech. K predikci byla použita neuronová síť LSTM a jak je z grafu zřejmé, tak vzniklý model popisuje realitu velmi přesně. (6)



Graf 2: Predikce ceny akcií společnosti DJIA

Zdroj: (6)

Jednotlivé techniky analýzy v rámci této práce nebudou dále rozvedeny, jelikož zaměřením práce není analýza časových řad jako takových, ale optimalizování způsobu jejich ukládání a čtení. Tato kapitola sloužila pro přiblížení aktuálních možností v analýze časových řad a za účelem vybudování představy o jejich využití.

2 Časová řada v energetice

Většina si pod spojením pojmů časová řad a energetika představí nejspíše real-time sběr dat o produkci, nebo naopak spotřebě elektrické energie. Tedy posloupnost naměřených hodnot v čase, získaných přímo z měřících čidel, což představuje obrovský objem dat za krátký časový úsek. Takovéto real-time data jsou využívána například pro predikci produkce a spotřeby elektrické energie, což je kritický proces pro takzvané balancování trhu (7). V systému dodávaného společností Unicorn se ale real-time data nevyskytují. Systém totiž sběr dat přímo z čidel neprovádí a místo toho přijímá již předzpracovaná obchodní data od třetích stran, které následně ukládá do časových řad pro další využití (8).

Co představují obchodní data, kdo je poskytuje a jak jsou dále využita, přiblíží následující sekce, která je věnována severskému energetickému trhu, kde vystupuje společnost eSett Oy (dále jen eSett), pro kterou společnost Unicorn dodává systém zaměřený na výpočet a zúčtování vzniklých odchylek v dodávkách elektřiny.

2.1 Energetický trh

Elektrická energie je komodita, kterou využívají téměř všichni a život bez ní je dnes jen těžko představitelný, přináší s sebou ale jeden doposud nevyřešený zásadní problém. Tento problém představuje nalezení efektivního způsobu skladování elektřiny ve velkých kvantitách. Z toho důvodu je ideální, aby vyprodukovaná elektřina byla spotřebována ihned, což vyžaduje neustálou rovnováhu nabídky a poptávky na trhu (9). V případě nerovnováhy nastává nežádoucí situace, kdy je elektřina produkována nadbytečně, nebo naopak nedostatečně. Subjekty vystupující na trhu tedy musí plánovat své úkony v zájmu zachování rovnováhy, to ale není vždy realizovatelné. Neočekávané poruchy přenosové soustavy, náhlé nárůsty poptávky, nepříznivé počasí či povětrnostní podmínky nebo chybná predikce, všechny tyto situace mohou mít negativní dopad na rovnováhu trhu. Dlouhodobé přehlížení nerovnováhy trhu má potencionálně fatální následky a může vést až k plošnému výpadku dodávky elektřiny, takzvanému blackoutu. Za účelem udržení rovnováhy tedy účastnici energetického trhu musí provádět různé obchody, při kterých prodávají, nebo naopak nakupují energii od ostatních účastníků (8).

2.1.1 Rozdělení trhu

Energetický trh se dělí podle časového hlediska, tedy horizontu, v jakém je elektřina obchodována. Grafické rozdělení trhu je k dispozici na **obrázku 1**.

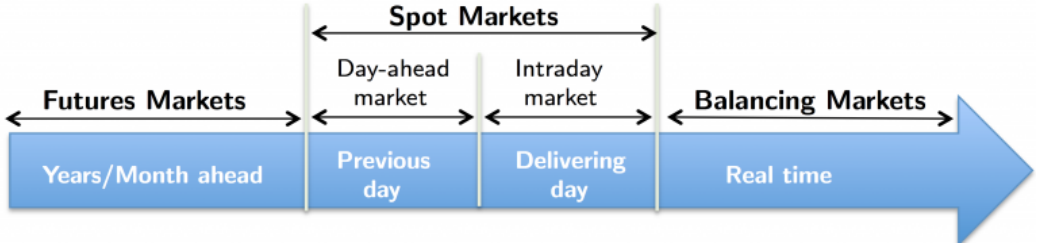
Krátkodobý trh

Na krátkodobém trhu se obchoduje v řádu dní až hodin před realizací dodávky. Rozhoduje zde den dodávky (delivery day), nebo hodina dodávky (delivery hour), což představuje čas, ve kterém skutečně dojde k přenosu elektrické energie. (9)

- **Denní trh (Day-ahead)** – Trh na kterém se obchoduje na den dopředu, tedy den před dodávkou. Účastník trhu zde obchoduje s energii pro celý den najedou, tudíž všech 24 hodin (9). Po zveřejnění dostupných kapacit operátorem trhu mají účastníci dvě hodiny na předložení svých nabídek. Cenu za jednotku určuje operátor trhu podle nabídky a poptávky z předchozího dne (10).
- **Vnitrodenní trh (Intraday)** – Trh na kterém se obchoduje v daný den, většinou minimálně hodinu předem dodávkou, ale v některých případech to může být i těsně před. Obchoduje se zde s energií v jednotlivých hodinách. Cenu určují účastníci trhu, jelikož se jedná o aukci a nejlepší nabídka vyhrává. (11)
- **Vyrovnávací trh (Balancing)** – Trh na kterém dochází k uzavírce 30 minut před dodávkou. Provozovatelé přenosové soustavy zde nakupují regulační energii, která je důležitá pro nastolení rovnováhy trhu na poslední chvíli. (9)

Dlouhodobý trh

Na dlouhodobém trhu se obchoduje týdny, měsíce i roky před skutečnou dodávkou energie. Obchody zde uzavírají dvě strany skrze předem vyjednaný kontrakt. (9)



Obrázek 1: Grafické rozdělení trhů dle časového hlediska
Zdroj: (12)

2.1.2 Balancování trhu

Balancování trhu neboli udržení rovnováhy mezi nabídkou a poptávkou je velmi důležitý úkon, bez kterého by se energetický trh neobešel. Jednotlivé tržní segmenty mohou mít odlišné postupy balancování, jelikož podléhají regulacím příslušných států, pod které spadají. Obecně lze však říci, že balancování funguje na základě obchodování energie mezi účastníky a operátory trhu v podobě aukce. Pokud účastník vyprodukuje nadbytek elektřiny, tak ho skrze operátora může prodat jinému účastníkovi, který má nedostatek, čímž se zachová rovnováha. Operátoři trhu tedy zastávají roli energetické burzy, která v obchodech vystupuje jako třetí strana. Obchody realizované skrze operátora spadají pod krátkodobý trh (13). Jedním z operátorů vystupujících na severském trhu je společnost Nord Pool, která umožnuje obchodování s elektřinou mezi severskými státy (14).

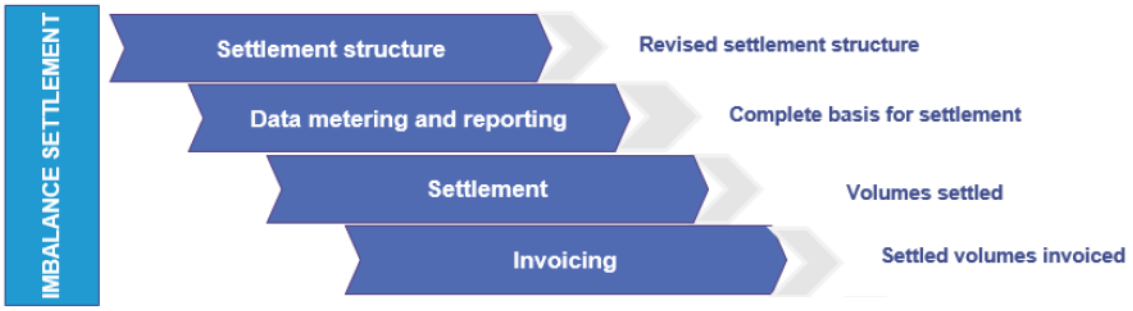
Další možnost představují bilaterální obchody, které probíhají přímo mezi dvěma účastníky trhu, bez nutnosti prostředníka. Tyto obchody musí být ale nahlášeny operátorovi trhu se značným předstihem. Bilaterálním obchodem je možné vyjednat velmi specifické podmínky, které by skrze burzu nebyly realizovatelné. Bilaterální obchod spadá pod dlouhodobý trh. (9)

Vyrovnání odchylek

I přes veškerou snahu o udržení rovnováhy mohou vzniknout odchylky, které samozřejmě nejsou žádané a proto jsou pokutované skrze poplatky. Za odchylku odpovídá ten účastník trhu, který jí způsobil. Odpovědnost může být ale i přenesená, běžným příkladem je domácnost, jakožto spotřebitel, jež využívá elektřinu podle vlastní potřeby a bez ohledu na ostatní okolnosti. Pokud tato domácnost způsobí odchylku nadprůměrnou spotřebou, tak za ní nenese odpovědnost samotná domácnost, ale obchodník se kterým má uzavřenou smlouvu. (9)

Výpočet a následné finanční vypořádání odchylek se na severském trhu řídí podle takzvaného Imbalance Settlementu modelu (dále jen settlement), který se skládá ze čtyř úrovní uvedených na **obrázku 2**. Úkolem settlementu je dále sjednocení způsobu vypořádání odchylek ve státech severského trhu. Aktuálně mezi tyto státy patří Finsko, Švédsko, Norsko a Dánsko. V settlementu hraje důležitou roli spojení

účastníků trhu a tržních entit, na základě kterého je postaven celý datový model a tudíž i časové řady. Následující sekce jsou tedy věnované popisu těchto propojení a poté jednotlivým úrovním settlementu pro přiblížení v něm figurujících dat. (8)



Obrázek 2: Úrovně modelu vyrovnání odchylek
Zdroj: (8)

2.1.3 Market Participants (MP)

Neboli účastníci trhu, jsou hlavní stakeholdeři vystupující v settlementu severského energetického trhu. Každý účastník má definované povinnosti a odpovědnosti, jež musí dodržovat (8). Aktuálně na trhu vystupuje více než 1,6 tisíc účastníků (15).

Transmission System Operator (TSO)

Provozovatel přenosové soustavy, který je zodpovědný za zabezpečení a stabilitu dodávky elektřiny a tudíž i provoz vedení velmi vysokého napětí, které ji přenáší na delší vzdálenosti v rámci celého státu nebo regionu. Tato přenosová soustava, občas také nazývaná páteřní síť, propojuje jednotlivé distribuční soustavy a slouží i jako přípojka velkých elektráren. Pro všechny severské státy platí, že v každém z nich vystupuje pouze jedno TSO. (8)

Distribution System Operator (DSO)

Vlastník distribuční soustavy, jež má odpovědnost distribuovat elektřinu ke spotřebitelům, kteří jsou k ní připojeni (8). DSO na rozdíl od TSO přenáší elektřinu na kratší vzdálenosti, skrze vedení nízkého napětí a na jeho soustavu jsou připojeny pouze menší elektrárny (16). Povinností DSO je měření reálně vyprodukované, spotřebované a vyobchodované elektřiny s jinou soustavou a následné reportování těchto dat na hodinové bázi (8). V settlementu momentálně vystupuje více než 500 účastníků v roli DSO (15).

Balance Responsible Party (BRP)

Společnost odpovídající za průběžné balancování za účelem vyrovnání nabídky a poptávky na hodinové bázi. BRP odpovídá za nahlášení očekávané produkce všech svých RO. Na základě této predikce poté TSO tvoří produkční plány pro výrobu elektřiny. BRP má přenesenou odpovědnost za jednoho nebo více RE, které zastupuje. BRP může uzavírat bilaterální obchody s ostatními BRP. V případě způsobení odchylky při balancování, je za ní BRP plně finančně zodpovědný (8). S rolí BRP v settlementu aktuálně vystupuje přibližně 200 účastníků (15).

Retailer (RE)

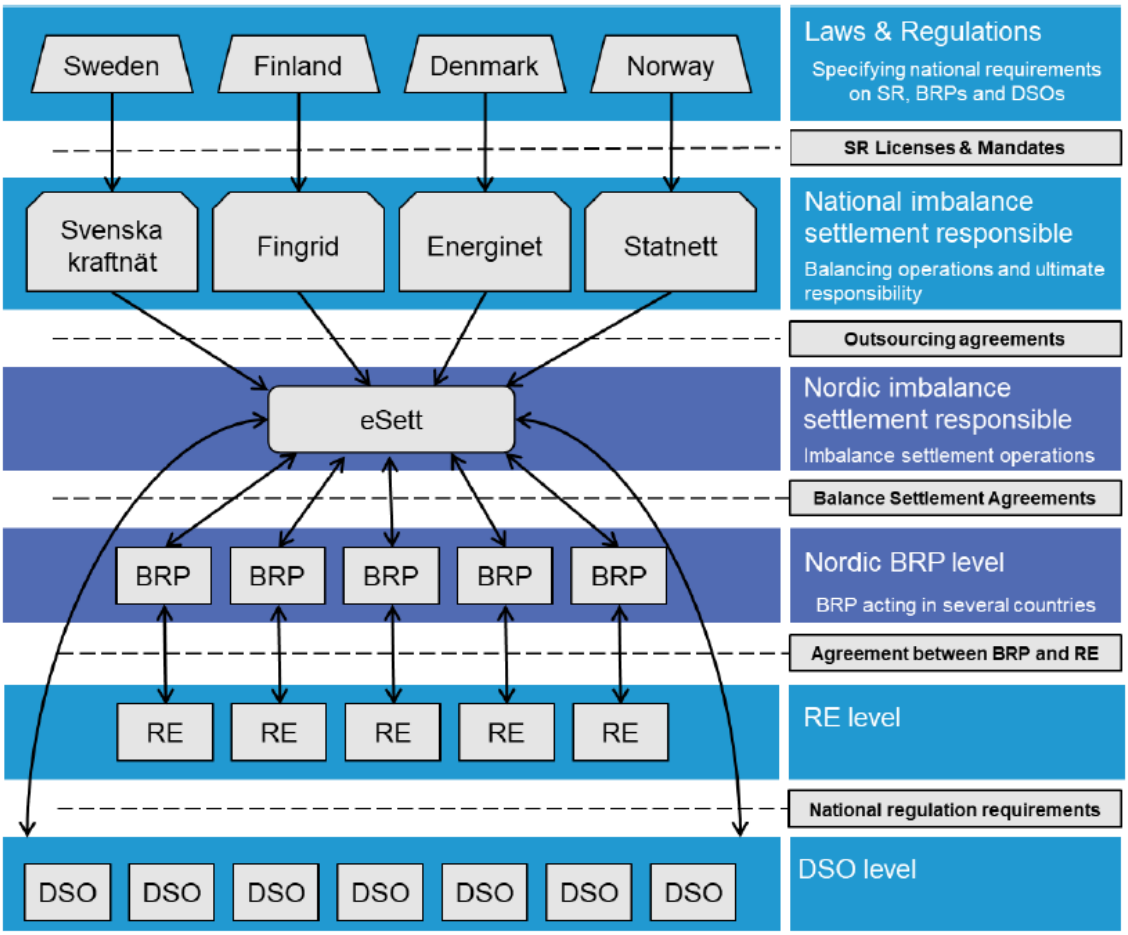
Obchodník prodávající elektřinu koncovým uživatelům, například tedy již zmíněným domácnostem. Skrze BRP nakupuje a prodává elektřinu přímo od výrobce, jiného RE nebo NEMO, tyto obchody musí nahlásit u TSO. Vyrovnání účtů mezi RE a BRP probíhá mimo systém společnosti Unicorn (8). RE představuje nejobsáhlejší roli v settlementu, aktuálně je jich evidováno téměř 900 (15).

Nominated Electricity Market Operator (NEMO)

NEMO představuje energetickou burzu, kde ostatní účastníci mohou prodávat nebo kupovat elektřinu na denních a vnitrodenních trzích. NEMO navíc umožňuje provádět i přeshraniční obchody s burzami jiných států. NEMO je odpovědný za reportování všech realizovaných obchodů (8). V settlementu vystupují celkem tři energetické burzy (15).

Imbalance Settlement Responsible (ISR)

Společnost odpovědná za výpočet a následné vyúčtování odchylek, které mohou vzniknout jednotlivým BRP. Důležitým úkolem ISR je poskytnout ostatním účastníkům komplexní IT řešení pro reportování všech potřebných obchodních dat, za účelem vyrovnání trhu. Součástí řešení také musí být umožněn přístup k těmto datům a jejich případná modifikace, dále publikace odvozených výpočtů a výsledků settlementu a v poslední řadě fakturace příslušných účastníků. Na severském energetickém trhu tuto roli zastupuje již zmíněná společnost eSett, pro kterou společnost Unicorn dodává systém splňující všechny tyto požadavky. (8)



Obrázek 3: Znázornění vztahů mezi účastníky trhu

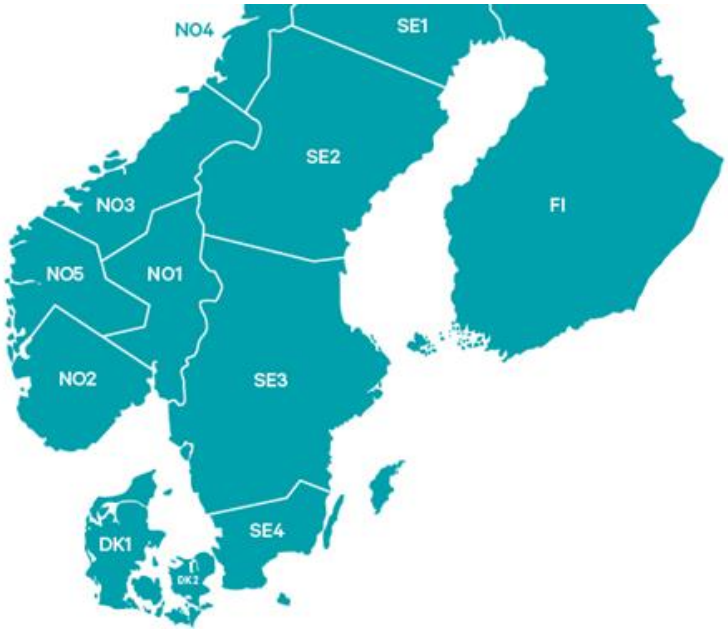
Zdroj: (8)

2.1.4 Market Entities (ME)

Neboli tržní subjekty, které slouží pro strukturované uspořádání dat potřebných pro vyrovnání trhu. Pomáhají specifikovat oblasti, ve kterých došlo k produkci, spotřebě nebo obchodu s elektrickou energií. (8)

Market Balance Area (MBA)

Geografická oblast předepisující pravidla pro obchod s elektřinou v rámci státu. Tyto oblasti definuje TSO daného státu, například Norsko je rozděleno do pěti MBA. V některých případech může jedno MBA pokrýt i celý stát. Pro obchodování na denním trhu v rámci MBA a je použita jednotná cena, kterou stanovuje NEMO na hodinové bázi. (8)



Obrázek 4: Rozdělení severského trhu do MBA

Zdroj: vlastní zpracování, data z (17)

Metering Grid Area (MGA)

Fyzická oblast, kde v rámci distribuční soustavy dochází k měření produkce, spotřeby a objemu obchodované elektřiny. Za měření těchto parametrů v MGA odpovídá DSO. MGA představuje podrobnější rozpad MBA do více sektorů. (8)

Regulation Object (RO)

Regulační objekt představuje skupinu jednoho či více generátorů v rámci jednoho nebo více MBA. Generátorem je zde myšlen zdroj elektrické energie, například tedy solární panel, větrná turbína nebo jaderná elektrárna. O seskupení generátorů do RO rozhoduje TSO dané země, v níž se RO nachází. (8)

Production Unit (PU)

Produkční jednotka představuje jeden či více generátorů nacházejících se v jedné elektrárně v rámci jednoho MGA. Podobně jako v případě vztahu MBA-MGA jde o podrobnější rozpad RO do menších skupin. (8)

2.1.5 Market Entity Connections (MEC)

Mezi účastníky na severském trhu dochází k reportování velkého objemu různých obchodních dat. Pro organizaci těchto dat jsou použité takzvané MECy, což jsou logické struktury, na základě kterých se uchovávají data v časových řadách. MEC

představuje propojení účastníků a entit za účelem identifikace aktérů vystupujících v různých obchodech a jiných aktivitách v rámci settlementu. O vytváření a případné úpravy MECů se starají pověření účastníci trhu. (8)

MECy představují první úroveň settlementu na **obrázku 2**.

Obchodní dimenze

Jsou dimenze představující jednotlivé účastníky a entity vystupující v rámci MECu. Příkladem může být MEC bilaterálního obchodu, který má následující dimenze: (8)

- BRP_1 – kupující strana, nakupuje energii od BRP_2
- BRP_2 – prodávající strana, prodává energii BRP_1
- RE_1 – obchodník zastupovaný od BRP_1
- RE_2 – obchodník zastupovaný od BRP_2
- MBA – oblast výskytu obchodu

Časová dimenze

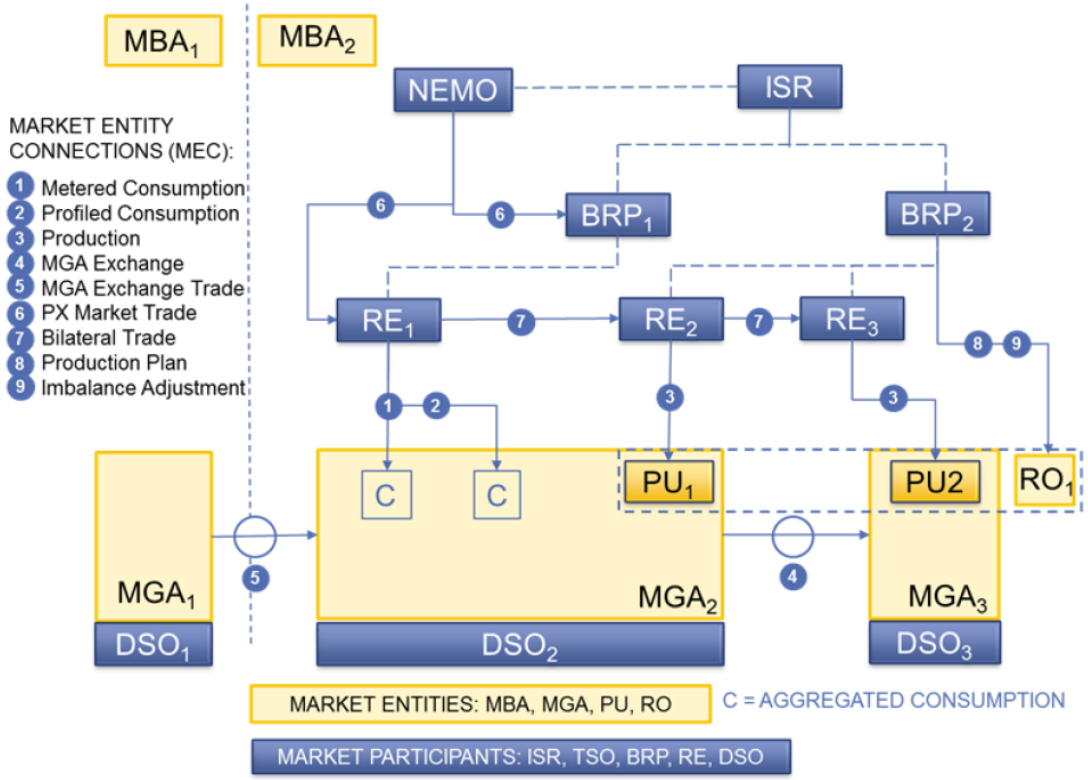
Každý MEC má kromě obchodních dimenzí také časovou dimenzi, neboli validitu, která určuje kdy MEC začíná a končí. Tato validita představuje libovolný interval o délce několika dní, roků, ale také i neomezeného časového úseku. (8)

Typy MECů

V settlementu se vyskytují tyto hlavní typy MECů: (8)

- **Consumption (CONS)** – naměřená spotřeba energie, reportuje DSO
- **Production (PROD)** – naměřená produkce energie, reportuje DSO
- **MGA Exchange (MGX)** – suma obchodované energie mezi MGA, reportuje DSO
- **MGA Exchange Trade (MGT)** – objem energie potřebný pro obchody mezi MGA v různých MBA, reportuje DSO
- **PX Market Trade (PXT)** – obchod energie na burze v rámci jednoho MBA (denní i vnitrodenní trh), reportuje NEMO
- **PX Market Flow (PXF)** – obchod energie na burze napříč dvěma MBA (denní i vnitrodenní trh), reportuje NEMO
- **Bilateral Trade (BIT)** – obchod mezi dvěma BRP, reportuje BRP
- **Production Plan (PP)** – plánovaná produkce energie, reportuje TSO

- **Imbalance Adjustment (IA)** – regulační energie použitá pro balancování (vyrovnávací trh), reportuje TSO



Obrázek 5: Znázornění vztahů mezi účastníky, entitami a MECy
Zdroj: (8)

2.1.6 Reportování obchodních dat

Reportování obchodních dat za účelem settlementu provádí téměř všichni účastníci trhu. ISR jim dává k dispozici různé komunikační kanály, skrze které zasílají takzvané datové toky, což jsou zjednodušeně řečeno XML soubory obsahující obchodní data a další potřebné identifikační informace o odesílateli a MECu. Příklad datového toku pro MEC bilaterálního obchodu i s popisky je dostupný v **příloze 1**. Povinností ISR je validování přijatého datového toku a obeznámení odesílatele o výsledku. Po validaci je provedeno uložení obchodních dat do časových řad. (8)

Reportování představuje druhou úroveň settlementu na **obrázku 2**.

Granularita dat

Aktuální časovou jednotkou použitou pro reportování obchodních dat je hodina. Ve většině případů tedy datové toky obsahují obchodní data na hodinové bázi. Jedna

hodnota tak představuje například objem vyobchodované energie v rámci specifické hodiny a MECu. (8)

V druhé polovině roku 2023 je však plánovaný přechod na reportování čtvrthodinových dat (18). Tato změna bude mít samozřejmě dopady i na denní a vnitrodenní trhy, místo hodin se bude zde obchodovat s čtvrthodinami. Kvůli takové zásadní změně byl přechod již několikrát odložen, prvotní návrh počítal s uskutečněním přechodu koncem roku 2020 (19). Tato změna je hlavním důvodem pro nutnou optimalizaci ukládání časových řad v úložišti, jelikož se jedná o čtyřikrát větší objem dat reportovaných v rámci jednoho dne.

Jednotky dat

Jakýkoliv objem elektrické energie je vždy reportován v megawatt hodinách (MWh) a hodnoty jsou vždy zaokrouhlené na šest desetinných míst. Některé datové toky obsahují krom objemu energie také hodnotu udávající cenu za tento objem, která je reprezentována jako číslo zaokrouhlené na dvě desetinná místa. Jelikož reportovaná data pochází od účastníků z různých států, tak je u cen nutné specifikovat měnu. (8)

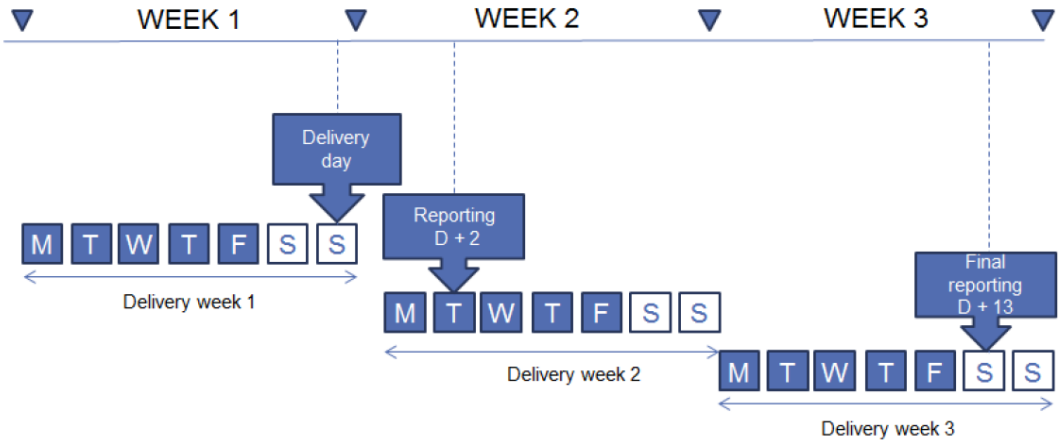
Gate Closure (GC)

Neboli uzávěrka, která určuje nejzazší termín pro reportování obchodních dat za účelem výpočtu settlementu. Obecně platí, že data týkající se obchodování mezi účastníky trhu a plánování produkce musí být reportována nejpozději v den dodávky. Naopak data získaná až při měření reálné produkce a spotřeby mohou být reportována i několik dní po dodávce.

2.1.7 Výpočet settlementu

Kvůli prodlevě při reportování reálně naměřených obchodních dat, dochází nejdříve k výpočtu takzvaného předběžného settlementu, který se počítá každý den, počínaje druhým dnem po dodávce a dvanáctým dnem konče. V tomto intervalu mají účastníci trhu čas na reportování zbývajících dat a jejich případnou korekci. Výsledky předběžného settlementu mají pouze informativní charakter. Třináctý den po dodávce dochází k výpočtu finálního settlementu, na základě kterého probíhá již finanční vyrovnání skrze fakturaci. Účastníci kteří nedodali potřebná obchodní data před GC jsou z výpočtu vynechání. (8)

Výpočet settlementu představuje třetí úroveň settlementu na **obrázku 2**.



Obrázek 6: Harmonogram výpočtu settlementu
Zdroj: (8)

Výstupní data

Detailní rozbor postupu výpočtů settlementu a odpovídající business logiky na pozadí není pro účely této práce podstatný. Velmi důležité jsou však výstupní data, které jsou stejně jako vstupní data ukládané do časových řad. Výstupní data zahrnují výsledky settlementu na základě plánované produkce/spotřeby, reálně naměřené produkce/spotřeby a použité regulační energie. Jelikož výpočty probíhají nad hodinovými daty, tak hlavní výsledky představují opět data na hodinové bázi. Především se jedná o objem přebytečné, nebo nedostatečné elektrické energie ve specifickou hodinu. Krom objemu energie se v datech mohou vyskytovat i finanční částky za tento objem, cena je určená podle MBA a různých poplatků. (8)

Povinností ISR je umožnit zobrazení výstupních dat účastníků trhu, ale ne každý z nich má vidět data ve stejné podobě. Například DSO zajímají data na úrovni MGA, naopak TSO je zase interesován v přehledech za celé MBA. Z tohoto důvodu dochází v rámci settlementu i ke generování odvozených časových řad, které mohou mít odlišné obchodní dimenze. (8)

Dále musí být jednotlivým účastníkům umožněno zobrazení souhrnů za delší časový úsek, což může představovat několik týdnů, měsíců, ale i roků. Z tohoto důvodu jsou některé časové řady agregované do vyšších časových jednotek. Agregované časové řady si většinou zachovávají původní obchodní dimenze, ale výsledná hodnota zde

představuje sumu hodinových hodnot, která rozsahem odpovídá zvolené granularitě. Výpočet této sumy se v některých případech může lišit i na základě požadované business logiky. Agregované časové řady jsou počítané v rámci settlementu a jejich výpočet se provádí i pro vstupní data. (8)

2.1.8 Fakturace

Fakturace odchylek jednotlivých BRP probíhá na základě výsledků settlementu. ISR zde vystupuje na straně dodavatele a BRP na straně odběratele. Samotný proces fakturace se pak řídí podle zákonů státu, v němž se BRP nachází. Na základě fakturace nevznikají žádná další data časových řad a podrobná analýza postupu tedy není pro účely této práce potřebná. (8)

Fakturace je poslední úrovní settlementu na **obrázku 2**.

2.2 Specifikace časových řad

Po rozboru settlementu severského energetického trhu je možné definovat požadavky na časové řady, jež hrají hlavní roli v systému dodávaného společností Unicorn. Datové úložiště časových řad v tomto systému představuje relační databáze Oracle, tato sekce je ale věnována obecným požadavkům a rozbor stávajícího datového modelu bude uskutečněn v praktické části práce. Uvedené objemy dat a některé doplňující požadavky jsou převzaté z interních zdrojů společnosti Unicorn.

2.2.1 Typy časových řad

Časové řady vystupující v settlementu jsou vždy intervalové, jelikož ve všech případech obsahují parametry, které představují hodnotu sledované veličiny v závislosti na délce zvolené časové jednotky. Časové řady lze ale dále klasifikovat na dva typy dle původu ukládané hodnoty.

Primární

Jsou to časové řady obsahující především reportovaná obchodní data v nejnižší časové granularitě, která jsou do časové řady uložena v originální získané podobě. Aktuálně jde tedy o časové řady obsahující zejména hodinová data, v budoucnu se ale bude jednat o data čtvrthodinová. Hodnoty primární řady jsou po prvotním

uložení jen zřídka modifikované a ve většině případech se dají považovat za read-only. Modifikace účastníky trhu je navíc možná pouze do uplynutí GC, obecně tedy platí, že data starší více než měsíc po dodávce už nebudou nikdy modifikována. (8)

Odvozené

Hodnoty těchto časových řad jsou vypočítávané na základě jiných časových řad, obvykle primárních. Typicky se jedná o agregované časové řady, například denní agregace, kde jedna hodnota představuje sumu všech hodin v daném dni. Možností je ale také výpočet hodnoty na základě libovolné business logiky, například rozdíl dvou parametrů z jiných časových řad. Na rozdíl od primárních řad, může u odvozených dojít k modifikaci hodnoty i po delším časovém úseku. Obecně ale platí, že data starší více než půl roku již nikdy nejsou modifikována.

2.2.2 Granularita dat

Granularita dat časové řady je určená především druhem ukládaných dat. Reportovaná data mají aktuálně vždy hodinovou granularitu. U odvozených hodnot může být časová jednotka libovolná (8). V systému se aktuálně vyskytují tyto pevné jednotky: hodina, den, týden a měsíc.

2.2.3 Dimenze časových řad

Časová řada má vždy časovou dimenzi a jednu, nebo více obchodních dimenzí.

Časová dimenze

Řídí se granularitou uložených dat a říká, jaký časový interval představuje jedna hodnota časové řady. V případě denní časové řady tedy udává začátek a konec specifického dne, například 1.1.2021 00:00 až 1.2.2021 00:00. Časová dimenze by měla být vždy ukládána v UTC[1], jelikož systém vystupuje na severském trhu, kde figuruje více časových zón. (8)

Obchodní dimenze

V běžných časových řadách hraje hlavní roli čas, respektive časová dimenze (1). V energetice jsou ale důležité i obchodní dimenze, které identifikují účastníky a

[1] Koordinovaný světový čas

entity, jimž náleží jednotlivé hodnoty časové řady. Počet a složení obchodních dimenzí záleží na požadovaných vlastnostech. Například časová řada, ve které se nachází sumy vyprodukované energie v rámci jednotlivých MBA, bude mít jedinou obchodní dimenzi – MBA. Většina řad má však dvě až pět obchodních dimenzí (8).

Krom účastníků a entit mohou obchodní dimenze obsahovat i další doplňující informace, například identifikátor měny v případě ukládání peněžní částky.

2.2.4 Typy hodnot

V řadách se uchovávají především číselné hodnoty s požadovanou maximální délkou šesti desetinných míst. V některých případech může dojít i k ukládání textového řetězce a logické hodnoty. Textové řetězce jsou většinou krátké a nevyžadují žádný speciální datový typ. Nutností je ovšem možnost uložení řetězců obsahující speciální znaky, které se vyskytují ve znakových sadách severských států.

2.2.5 Objem dat

Vstupní data do úložiště představují především reportovaná obchodní data, která jsou hromadně zasílána skrze datové toky (8). Datová propustnost při ukládání do časových řad tedy nehraje tak kritickou roli, jako například při sběru real-time dat z více zdrojů (7). To ovšem neznamená, že časové řady obsahují málo dat, přibližný denní přírůstek představuje téměř 2,5 miliónů nových hodnot do primárních řad. Denní přírůstek do odvozených časových řad je poté necelých 8 milionů hodnot. Ačkoliv se tyto objemy na první pohled zdají velké, tak je nutné brát v potaz, že se jedná o sumu napříč všemi časovými řadami daného typu. Kvůli značnému objemu a různorodosti dimenzí obchodních dat, je totiž vhodné ukládat data do oddělených časových řad, které jsou specifické pro daný MEC. Data přijatá datovým tokem bilaterálního obchodu jsou tedy uložena do časové řady, která je pro ně určena.

Primární časové řady

V settlementu je evidováno přibližně 40 primárních časových řad, což v průměru odpovídá zhruba 60 tisícům nových hodnot za den pro každou z nich. Rozdělení nových hodnot mezi jednotlivé řady ovšem není rovnoměrné, u některých řad se denní přírůstek pohybuje v řádu jednotek tisíců a u jiných zase v řádu 100 tisíců

hodnot. Časovou řadu s největším přírůstkem představuje naměřená spotřeba energie pro jednotlivé RE, kde za den může přibýt až 500 tisíc nových hodnot.

Odvozené časové řady

Odvozených časových řad je v settlementu znatelně více, aktuálně přibližně 700. Průměrný denní přírůstek na řadu je tedy zhruba 11 tisíc hodnot. Stejně jako u primárních řad, je zde přírůstek také nerovnoměrný. Nejvíce frekventovanou řadu opět představuje naměřená spotřeba energie, tentokrát ale přepočítaná na úroveň RE a BRP, přírůstek se zde pohybuje okolo 400 tisíc nových hodnot v rámci dne.

Přechod na čtvrthodinová data

Změna reportování znamená pro úložiště čtyřikrát větší objem vstupních hodnot, z nynějších 2,5 milionů za den, bude rázem 10 milionů. U výstupních dat se tato změna dotkne pouze neagregovaných odvozených řad, jelikož agregace čtvrthodin do hodin není plánovaná. Celkový nárůst odvozených řad tedy nebude čtyřnásobný, ale přibližně dvojnásobný.

U primární časové řady naměřené spotřeby, která má největší přírůstek ze všech řad, je po přechodu nutné počítat s 2 miliony nových hodnot za den. Očekávaný stav této řady po roce reportování čtvrthodinových dat je poté přibližně 800 milionů hodnot, jelikož je zde započítán i možný nárůst počtu účastníků trhu.

2.2.6 Doba uchování dat

Doba, po kterou je nutné uchovávat data v úložišti, je diktována zákony severských států a aktuálně se jedná o interval 5 let. Data starší více než 5 let je tedy možné odebrat z časových řad a eventuálně zálohovat. V případě zmiňované primární časové řady naměřené spotřeby, představuje uložení čtvrthodinových dat za 5 let minimálně 4 miliardy hodnot. Takový objem dat už pro úložiště může představovat určité problémy, jak z pohledu výkonosti čtení, tak ze strany potřebné kapacity pro uchování. Z tohoto důvod je nutné, aby časové řady byly uloženy co nejefektivněji. Výhodu zde představuje skutečnost, že ne všechna data musí být přístupná za stejnou dobu. Na základě provedených analýz bylo zjištěno, že ke čtení historických dat dochází jen velmi zřídka. Uživatele systému tedy zajímají především „čerstvá data“ v rámci posledního roku a čtení starších dat má tudíž menší prioritu.

3 Relační databáze Oracle

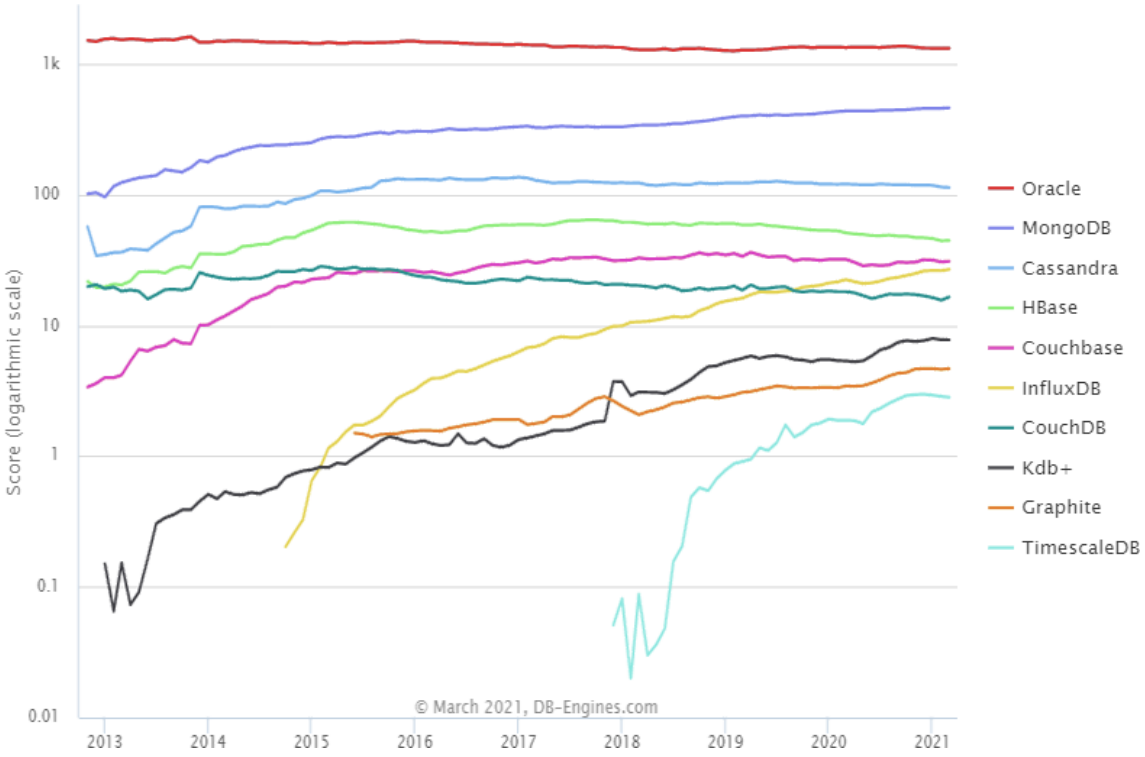
Společnost Oracle nabízí hned několik databázových systémů určených pro využití v různých oblastech, od databáze typu klíč-hodnota, až po datové sklady. Tato práce je ale konkrétně zaměřena na relační databázi Oracle Enteprise Edition ve verzi 19c, jež je použita jako úložiště časových řad v systému společnosti Unicorn.

3.1 Proč Oracle

Data časových řad v settlementu obsahují velké množství různých závislostí (8). V některých situacích tak pro přístup k hodnotám nestačí pouze časová a obchodní dimenze časové řady, ale musí dojít i k připojení dodatečných informací pro filtraci na základě požadované business logiky. Relační databáze je v tomto ohledu vhodnou volbou, díky možnosti definování logických vztahů.

Argumentem proti použití relační databáze by mohla být například menší škálovatelnost, oproti jiným dostupným technologiím, především NoSQL a time series databázím (1). Ačkoliv by dnes tyto technologie teoreticky představovaly lepší alternativu, tak je nutné brát v úvahu, že prvotní návrh úložiště časových řad proběhl před více než 10 lety. V té době, některé dnes populární nerelační databáze, ještě neexistovaly, nebo nebyly tak rozšířené a prověřené pro použití v podnikové sféře, viz **graf 3**. Teoretický přechod na jinou technologii by v nynějším stádiu představoval příliš velký zásah do již existujícího systému.

Databáze Oracle však nabízí mnoho optimalizačních technik, pro dosažení lepší škálovatelnosti a úspory diskového místa. Oracle je navíc multiplatformní a umožňuje podrobnou customizaci pro naplnění i velmi specifických požadavků (20). Tyto vlastnosti patří mezi důvody, proč databáze Oracle je, a v dohledné minulosti i byla jedničkou na databázovém trhu (21).



Graf 3: Popularita Oracle vs nerelační databáze napříč roky
Zdroj: (21)

3.2 Struktura databáze

Struktura relační databáze Oracle je tvořena z několika částí. Některé části představují logické dělení pro lepší správu a udržitelnost databáze, jiné zase fyzické struktury pro uložení dat na disk. Tyto části je nutné představit, jelikož budou zmiňované v následujících sekcích.

3.2.1 Datový slovník

Reprezentuje jednu z nejdůležitějších částí databáze. Jedná se o kolekci tabulek, které uchovávají podrobné informace o databázi. Mezi tyto informace se řadí: (22)

- definice všech databázových objektů
- přehled alokovaného místa pro objekty
- uživatelé a jich práva pro jednotlivá schémata
- auditní informace o správě databázových objektů

Tabulky datového slovníku se nachází ve schématu SYS a jejich modifikaci může provádět pouze samotná databáze. Pro uživatele jsou ale k dispozici speciální pohledy, které poskytují různé užitečné informace. Příkladem může být pohled DBA_TABLES, který poskytuje informace o všech tabulkách v databázi. Prefix DBA zde značí, že pohled obsahuje opravdu všechny tabulky a tudíž k němu má přístup pouze uživatel s DBA právy (obvykle jen administrátor). Pro ostatní uživatele existují pohledy s prefixy USER a ALL. USER_TABLES tedy poskytuje pouze informace o tabulkách ve vlastnictví uživatele. ALL_TABLES zase poskytuje informace o tabulkách, na které má daný uživatel udělaná přístupová práva. (23)

3.2.2 Schéma

Je logická organizační jednotka databáze. Schéma vždy patří určitému uživateli a definuje kolekci jemu dostupných databázových objektů a tabulkových prostorů. Každé schéma má definované uživatelské jméno a musí být zabezpečeno heslem. Pomocí schématu tedy lze vytvářet přístupová a kapacitní práva. Mezi hlavní výchozí schémata databáze Oracle patří SYS a SYSTEM. (23)

3.2.3 Tabulkový prostor

Představuje kontejner obsahující jeden nebo více fyzických datových souborů. Tabulkový prostor umožňuje logické rozdělení databáze do více částí, což pomáhá proti takzvanému boji o I/O[2] prostředky mezi uživateli databáze. Mezi hlavní výhody definování více tabulkových prostorů patří: (24)

- rozprostření databáze na více disků
- oddělení dat pro různé účely (často aktualizovaná, pouze pro čtení, …)
- možnost odpojení starých dat z databáze (offline režim)

Slovníková správa

Tabulkové prostory se slovníkovou správou uchovávají informace o volném místě svých objektů v datovém slovníku, který je globální. Například seznam volných extentů je uložen ve slovníkové tabulce FET$. Pokud se v databázi nachází více

[2] Input/Output neboli vstup/výstup, zde tedy zápis a čtení dat.

tabulkových prostorů s vysokou frekvencí DML[3] operací, tak může docházet k poklesu výkonu, kvůli boji o přístup ke sdíleným slovníkovým tabulkám. (25)

Lokální správa

Lokálně spravované tabulkové prostory uchovávají informace o volném místě v bitmapové struktuře, která je uložena v jednom z datových souborů, jež se nachází v tabulkového prostoru. Tím je docíleno nezávislosti tabulkového prostoru a zabráněno souboji o sdílené tabulky slovníku. (24)

3.2.4 Datový soubor

Je fyzický soubor uložený na disku, který obsahuje různé databázové objekty. Jeden objekt může být uložen v rámci více datových souborů, které spadají pod tentýž tabulkový prostor. Objekty uvnitř datového souboru jsou složeny z datových bloků (26). Maximální velikost datového souboru je možno konfigurovat manuálně, stejně tak i jeho přírůstky. Běžným postupem je ale využití automatického módu, který zajistí, že databáze sama rozšíří datový soubor dle potřeb a nedojde tak k odstávkám kvůli nedostatečnému místu (27).

Datový blok

Je nejmenší jednotkou pro ukládání dat. Jeden blok odpovídá specifikovanému počtu bajtů zabírajících fyzický prostor na disku. Velikost datového bloku databáze a operačního systému se může lišit. Od verze Oracle 9i je možné nastavit odlišnou velikost datového bloku pro různé tabulkové prostory, obecně tento postup ale není doporučovaný. Specifikace velikosti bloku probíhá při instalaci databáze a defaultní hodnota je 8192 bajtů (8 KB). Velikost bloku je vhodné volit dle potřeb úložiště, OLTP[4] systém může benefitovat z menších bloků, OLAP[5] naopak z větších. (23)

---

[3] Data Manipulation Language – INSERT, UPDATE, DELETE klauzule

[4] Online Transaction Processing - zpracovávání velkého počtu transakcí (relační databáze)

[5] Online Analytical Processing – provádění analýz z velkého objemu dat (datový sklad)

Extent

Představuje logickou kolekci souvislých datových bloků pro uložení určitého typu objektu v rámci datového souboru. Velikost extentu je nastavitelná pro jednotlivé tabulkové prostory, defaultní hodnota je 64 KB. (28)

Segment

Reprezentuje logickou kolekci extentů z jednoho nebo více datových souborů, které jsou alokované pro uložení specifického typu objektu v rámci jednoho tabulkového prostoru. (28)

Správa místa v datovém bloku

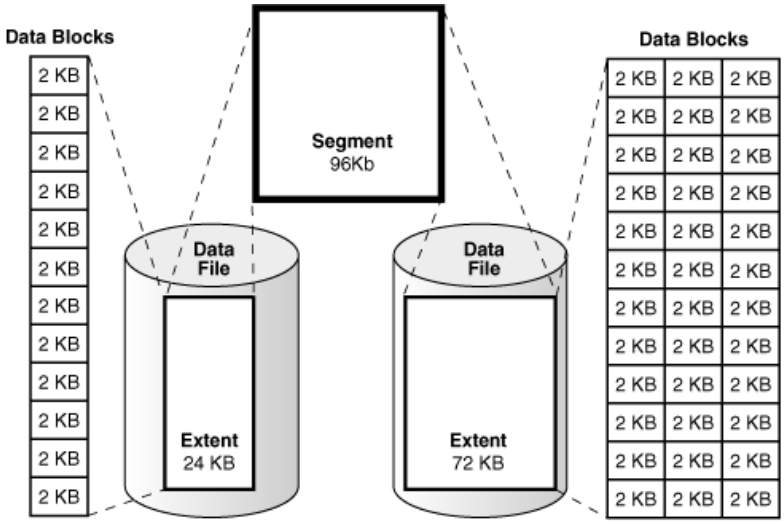
Vytváření nových extentů je závislé na volném místě existujících datových bloků. Databáze pro každý segment uchovává seznam volných datových bloků podle zvoleného způsobu správy. Hranice pro specifikaci volného a zaplněného bloku je možné konfigurovat skrze parametry PCTFREE a PTCUSED. (28)

- **PCTFREE** – Určuje procentuální velikost rezervy pro modifikaci uložených řádků v bloku. Defaultní hodnota 10 značí, že do bloku je možné ukládat nové řádky do té doby, dokud není vyčerpána kapacita bloku z 90%. Takto zaplněný blok již nepřijímá nové řádky a je považován za plný. Zbylých 10% je rezervováno jako místo pro možné modifikace již uložených řádků. (28)
- **PCTUSED** – Určuje procentuální zaplnění, kterého je nutné dosáhnout pro znovuotevření bloku novým řádkům. Defaultní hodnota 40 značí, že pro otevření bloku je potřeba, aby byl zaplněn z méně než 40%. Pokud tedy z uzamčeného bloku dojde k odstranění řádků a volná kapacita překročí 60%, tak je blok označen jako volný. (28)

Při špatně zvolených hraničních hodnotách může docházet k plýtvání místa a zřetězení řádků. Pro většinu případů je tak doporučeno používat automatický mód, který se časem přizpůsobuje požadavkům dat a dosahuje tak obvykle lepších výsledků. (28)

Zřetězené řádky

Představují řádky, které se nevejdou do jednoho datového bloku, a tudíž je nutné jejich rozdělení do více bloků. Čtení těchto řádku vyžaduje více I/O operací, což může mít negativní vliv na výkon databáze. Pro prevenci vzniku zřetězených řádků je vhodné nevytvářet příliš široké tabulky a vyhnout se objemným datovým typům. Existují však případy, kdy je požadované mít v tabulce mnoho sloupců nebo použít objemné datové typy, možné řešení zde představuje navýšení velikosti datového bloku. (28)

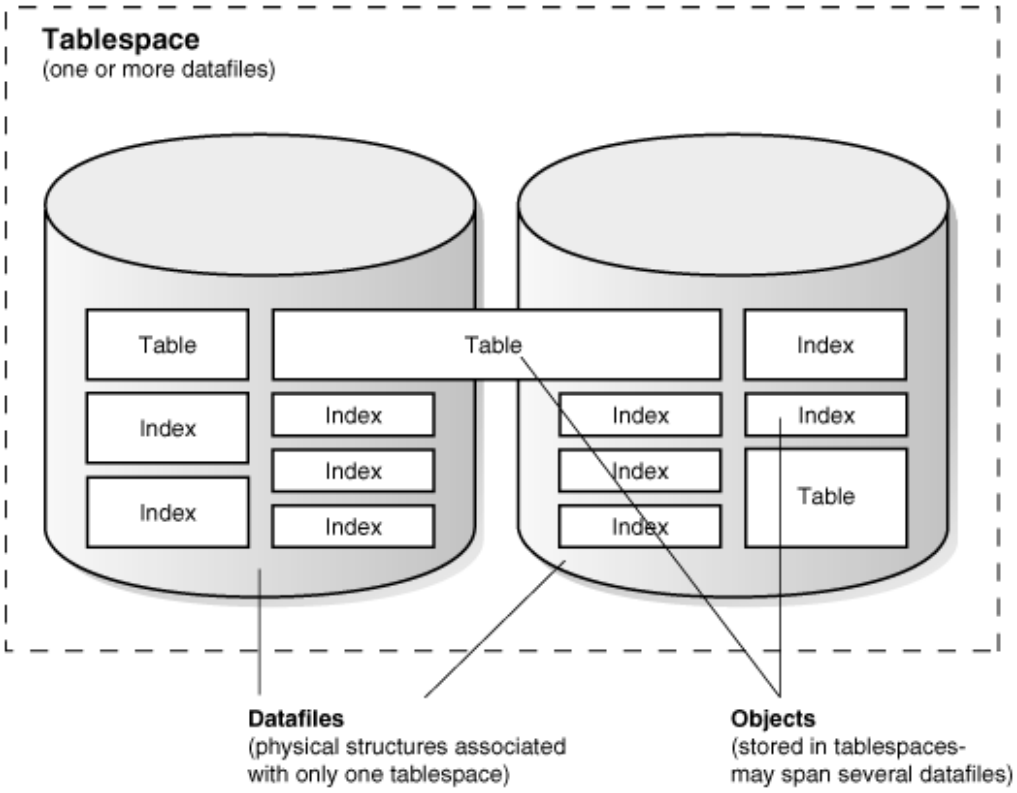


Obrázek 7: Vztah mezi datovými soubory, bloky, extenty a segmenty
Zdroj: (29)

3.2.5 Databázový objekt

Představuje definovaný objekt databáze, který je určený pro uložení, referencování nebo manipulování dat. Mezi hlavní databázové objekty v databázi Oracle patří: (20)

- **Tabulka** – kolekce řádků s jedním nebo více sloupci
- **Pohled** – logická tabulka složena z jedné nebo více fyzických tabulek
- **Integritní omezení** – vynucení požadovaných vlastností hodnot ve sloupcích
- **Index** – urychlení přístupu k datům
- **Sekvence** – generování unikátních celých čísel, především pro primární klíče
- **Synonym** – definice alternativního jména pro databázový objekt
- **Uložené procedury** – různé pomocné PL/SQL skripty
- **Trigger** – PL/SQL skript, který je proveden automaticky pro určitou událost



Obrázek 8: Struktura databáze Oracle

Zdroj: (26)

3.3 Tabulky

Tabulka představuje hlavní datový objekt relační databáze. Základní tabulka v databázi Oracle je ovšem téměř totožná tabulkám z ostatních relačních databází. Jde tedy o kolekci řádků, které mají jeden nebo více sloupců. (30)

Každá tabulka má v Oracle navíc pseudo-sloupec ROWID, který je unikátním identifikátorem řádku. Tento sloupec vzniklá složením několika informací, které zahrnují například číslo databázového objektu (řádku) a relativní číslo datového souboru v tabulkovém prostoru, ve kterém se řádek nachází. Ačkoliv je sloupec ROWID unikátním identifikátorem, tak jeho použití pro primární klíč není vhodné. Při odstranění a opětovném přidání řádku totiž může dojít ke změně hodnoty ROWID. Sloupec ROWID není fyzicky uložen v databázi a v případě potřeby je dopočítán. I přes tuto vlastnost ROWID představuje nejrychlejší způsob pro získání jediného řádku v tabulce. (31)

3.3.1 Haldově uspořádaná tabulka

Je defaultní a tedy i nejpoužívanější typ tabulky v databázi Oracle. Řádky tabulky jsou uchovávány v haldové struktuře (heap) bez určeného pořadí, které se postupem času může měnit. Se změnou pozice řádku se změní i jeho ROWID. Při dotazování se na data tabulky bez explicitního řazení tedy není zaručené, že stejný dotaz vrátí řádky vždy ve stejném pořadí (32). Nově vkládaný řádek je uložen do prvního volného místa, které je pro něj dostačující (33).

```
CREATE TABLE KNIHA
(
    NAZEV VARCHAR2(200) PRIMARY KEY,
    DATUM_VYDANI DATE NOT NULL
) ORGANIZATION HEAP; -- Není nutné explicitně uvádět
```

Kód 1: Příklad vytvoření haldové tabulky

Zdroj: vlastní zpracování

3.3.2 Indexově uspořádaná tabulka

Je typ tabulky, ve které jsou data uchovávána ve formě indexů. Na rozdíl od haldové tabulky, kde jsou indexy volitelné, je v indexové tabulce každý sloupec indexem. Indexově uspořádaná tabulka musí povinně obsahovat primární klíč, skrze který by mělo následně docházet k dotazování. Dotazování skrze ostatní sloupce je taky možné, ale není tak efektivní. Primární klíč by měl zahrnovat co nejvíce sloupců tabulky, za účelem dosažení maximální efektivity. Tento typ tabulky je vhodný pro statická data, která se často nemění (20). Nevýhodou indexových tabulek je fakt, že nepodporují rozdělení, které bude popsáno v následujících sekcích (23).

```
CREATE TABLE KNIHA
(
    NAZEV VARCHAR2(200) NOT NULL,
    DATUM_VYDANI DATE NOT NULL,
    CONSTRAINT PRIMARNI_KLIC_KNIHA PRIMARY KEY (NAZEV, DATUM_VYDANI)
) ORGANIZATION INDEX;
```

Kód 2: Příklad vytvoření indexové tabulky

Zdroj: vlastní zpracování

3.3.3 Dočasná tabulka

Představuje tabulku uchovávající data pouze po dobu transakce, nebo relace[6]. Tabulka existuje globálně pro všechny uživatele, ale každý v ní může pracovat jen se svými řádky, o které po ukončení transakce/relace nenávratně přijde (20).

Od verze Oracle 18c je možné definovat i privátní dočasnou tabulku, která je dostupná pouze uživateli, jež jí vytvořil. Po ukončení transakce/relace je tabulka nenávratně odstraněna. (34)

```
-- Globální dočasná tabulka
CREATE GLOBAL TEMPORARY TABLE KNIHA
(
    NAZEV VARCHAR2(200) PRIMARY KEY,
    DATUM_VYDANI DATE NOT NULL
) ON COMMIT DELETE ROWS; -- Odstranění dat po konci transakce

-- Privátní dočasná tabulka
CREATE PRIVATE TEMPORARY TABLE KNIHA
(
    NAZEV VARCHAR2(200) PRIMARY KEY,
    DATUM_VYDANI DATE NOT NULL
) ON COMMIT PRESERVE DEFINITION; -- Odstranění tabulky po konci relace
```

Kód 3: Příklad vytvoření dočasné tabulky
Zdroj: vlastní zpracování

3.3.4 Seskupená tabulka

Neboli cluster, umožňuje uložení často spojovaných tabulek do jednoho fyzického místa na disku. Při seskupení tabulek tedy databáze nemusí přistupovat do více bloků a tím se sníží počet potřebných I/O operací pro získání dat. Seskupení je definováno na základě klíče clusteru, což je sloupec nebo spojení více sloupců, které spojované tabulky sdílí. Každá unikátní hodnota klíče je v clusteru a jeho indexu uložena pouze jedenkrát, což představuje možnou úsporu místa na disku (20). Seskupení není vhodné pro tabulky, které jsou často aktualizovány nebo je potřeba

[6] Aktivní síťové spojení mezi uživatelem a databází (session).

jejich procházení skrze přístupovou metodu full table scan. Značnou nevýhodou je také nemožnost definování rozdělení (32).

```
-- Vytvoření clusteru a jeho indexu
CREATE CLUSTER KNIHA_AUTOR(AUTOR_ID NUMBER(8, 0));
CREATE INDEX INDEX_KNIHA_AUTOR ON CLUSTER KNIHA_AUTOR;

-- Vytvoření první tabulky clusteru
CREATE TABLE AUTOR
(
    ID NUMBER(8, 0) PRIMARY KEY,
    JMENO VARCHAR(300) NOT NULL
) CLUSTER KNIHA_AUTOR(ID);

-- Vytvoření druhé tabulky clusteru
CREATE TABLE KNIHA
(
    NAZEV VARCHAR2(200) PRIMARY KEY,
    AUTOR_ID NUMBER(8, 0) NOT NULL
) CLUSTER KNIHA_AUTOR(AUTOR_ID);
```

Kód 4: Příklad vytvoření seskupených tabulek
Zdroj: vlastní zpracování

3.4 Pohledy

Pohled v databázi Oracle představuje klasickou logickou reprezentaci jedné nebo více zdrojových (master) tabulek. Pohled sám o sobě neuchovává data, ale pouze definici dotazu, kterým data v případě potřeby načítá. Pohledy jsou užitečné pro zjednodušení přístupu k datům, ale také pro definovaní restrikcí. Příkladem může být situace, kdy uživatel nemá přidělená práva na celou tabulku, ale pouze na pohled, který neobsahuje určité sloupce nebo řádky z master tabulky. (23)

```
CREATE VIEW KNIHY_OD_A AS
(
    SELECT NAZEV, DATUM_VYDANI
    FROM KNIHA
    WHERE UPPER(NAZEV) LIKE 'A%'
) WITH READ ONLY; -- Specifikace režimu pouze pro čtení
```

Kód 5: Příklad vytvoření pohledu
Zdroj: vlastní zpracování

3.4.1 Materializovaný pohled

Oproti klasickému pohledu ukládá kromě definice dotazu i jeho výsledek. Ve své podstatě tedy jde o fyzickou tabulku, jejíž řádky jsou tvořeny na základě uložené definice. Definice materializovaného pohledu může být složena z více tabulek a jiných materializovaných pohledů (23). Využití materializovaných pohledů lze najít například v OLAP pro tabulky faktů nebo k replikaci dat v distribuované databázi (30). Další použití představují i různé souhrnné výpočty a agregace dat, které by při často opakovaném přístupu skrze klasický pohled zabíraly mnoho času (20). Materializované pohledy je nutné aktualizovat, což je možné provádět více způsoby.

Kompletní aktualizace

Při kompletní aktualizaci dojde nejdříve k odstranění všech dosavadních řádků, následným krokem je poté naplnění pohledu novými řádky podle definovaného dotazu. Tento způsob bývá oproti přírůstkové aktualizaci značně pomalejší, jelikož se jedná o úplné přegenerování pohledu. (35)

Přírůstková aktualizace

U přírůstkové aktualizace dochází k vyhodnocení pouze modifikovaných řádků z master tabulek. Získání informace o modifikaci řádku tabulky je možné skrze takzvané logy materializovaných pohledu, nebo sledováním změn oddílu v případě rozdělené tabulky. Log je nutné vytvořit manuálně, ale k jeho plnění již dochází automaticky. (35)

Provedení aktualizace

Samotné provedení aktualizace pohledu je možné provést na základě: (36)

- **ON COMMIT** – Automatická aktualizace před ukončení transakce týkající se master tabulky. Tento způsob může mít negativní dopad na původní transakci, jelikož aktualizace pohledu je její součástí.
- **ON DEMAND** – Manuální aktualizace uživatelem nebo uloženou procedurou.
- **START WITH a NEXT** – Naplánování automatické aktualizace v pravidelných intervalech se specifikovaným začátkem.

Indexy v materializovaném pohledu

Na rozdíl od klasického pohledu, který využívá indexy master tabulky, je u materializovaného pohledu nutné vytvořit indexy nové, což může představovat větší nároky na diskovou kapacitu. (37)

Přepsání dotazu

Materializované pohledy umožňují takzvané přepisování dotazu za účelem jeho optimalizace. Pokud v databázi existuje materializovaný pohled, který by výměnou za master tabulku v dotazu předešel zdlouhavému přepočtu za běhu, tak se databáze Oracle může rozhodnout provést jejich záměnu. Toto chování je nutno povolit při tvorbě materializovaného pohledu. (37)

```
CREATE MATERIALIZED VIEW KNIHY_OD_A
REFRESH COMPLETE -- Vždy kompletní aktualizace
ON DEMAND -- Pouze manuální aktualizace
ENABLE QUERY REWRITE -- Povolení přepsání dotazu
AS
SELECT NAZEV, DATUM_VYDANI
FROM KNIHA
WHERE UPPER(NAZEV) LIKE 'A%';

DBMS_MVIEW.REFRESH('KNIHY_OD_A'); -- Provedení manuální aktualizace
```

Kód 6: Příklad vytvoření materializovaného pohledu

Zdroj: vlastní zpracování

3.5 Možnosti optimalizace

Způsobů optimalizace výkonu a datové náročnosti databáze Oracle je mnoho. Největším limitním faktorem však bývají I/O operace, které jsou potřebné pro veškerou činnost databáze. Možným řešením je provoz databáze na nejlepších dostupných databázových serverech, což je ale velmi nákladné a daleka nevyřeší všechny problémy. Trendem poslední doby je řešení optimalizace na softwarové úrovni, tedy především v konfiguraci a návrhu databáze (38). Tato sekce je věnována různým metodám optimalizace, jež jsou převážně softwarového rázu.

3.5.1 Konfigurace databáze

Konfigurace databáze je kromě výběru hardwaru pro úložiště prvním místem, kde lze provádět optimalizaci. Při konfiguraci je vhodné brát v potaz očekávaný rozsah databáze a výkonnostní nároky. (38)

Fyzická správa dat

Pro snadnou fyzickou správu dat nabízí databáze Oracle automatickou správu úložiště (ASM). Při použití ASM jsou data ukládána do diskových skupin, kde každá skupina vystupuje jako jednotka, což umožnuje rovnoměrné rozprostření dat mezi disky v rámci skupiny. Disky mohou být do skupiny přidávány a odebírány za chodu databáze a tedy bez nutnosti odstávky. Po přidání nového disku do skupiny je automaticky provedena reorganizace dat. Rozdělení do diskových skupin také umožňuje vytváření skupin s různými diskovými technologiemi. Často používané databázové objekty tak mohou být uloženy ve skupině s ultrarychlými NVMe disky s menší kapacitou. Pro méně frekventované objekty pak postačí uložení ve skupině s klasickými SSD, nebo dokonce HDD. ASM podporuje i automatické zrcadlení skupin, jako ochranu proti případnému selhání disku. Díky redundantním skupinám je možné dočasné přesměrování požadavků na záložní disk bez odstávky. (39)

Logická správa dat

Představuje rozdělení databáze na tabulkové prostory a další podřazené logické struktury. Každý tabulkový prostor v databázi by měl být lokálně spravovaný, pro zamezení bojům o tabulky datového slovníku. Pro uživatele by měl být definován defaultní tabulkový prostor, který není systémový. Důvodem je, aby uživatel nemohl vytvářet databázové objekty v systémovém prostoru, a tím ho zbytečně zanášet. Správu volného a přírůstkového místa v prostorech je silně doporučeno nastavovat na automatickou, pro lepší využití diskové kapacity a efektivnější provádění transakcí. Ve vzácných případech lze dosáhnout lepších výsledků manuální správou, ale rozhodnutí pro tuto volbu by mělo být podložené důkladným testováním. (40)

Vyrovnávací paměť

V databázi Oracle nazývaná jako buffer cache, je způsob ukládání datových bloků s častým přístupem do operační paměti. Uložené bloky tak při dalším požadavku

nemusí být znovu čtené z disku, což má pozitivní vliv na výkon. Velikost této paměti je nastavitelná skrze inicializační parametr DB_CACHE_SIZE. Obdobně jako u většiny funkcionality databáze Oracle, ale existuje i automatický mód, který si velikost vyrovnávací paměti upravuje dle aktuálních potřeb a dosahuje tak optimálních výsledků. Obsah vyrovnávací paměti je k vidění v pohledu V$BH. (38)

Velikost bloku

Nastavení velikost bloku je závislé na typu úložiště a ukládaných datech. Obecně ale platí, že menší bloky jsou vhodné pro OLTP úložiště a větší pro OLAP. O ideální velikost bloku v OLTP úložištích, za účelem dosažení co nejlepšího výkonu, se ale v Oracle komunitě vedou spory. Větší bloky teoreticky sníží počet potřebných I/O operací pro skenování tabulek a indexů, na druhou stranu se tak ale do mezipaměti databáze můžou dostat i nepotřebná data, která akorát zbytečně zabírají místo. Menší velikost zase pomáhá v boji o přístup k blokům velmi frekventovaných tabulek. Argumentů pro menší či větší velikost bloku je spousta. (38)

Změna velikosti bloku by neměla být brána jako zázračné optimalizační řešení, které vyřeší veškeré problémy s výkonem databáze. Obě varianty mají své výhody i nevýhody. Defaultní velikost 8 KB je vhodná pro většinu OLTP úložišť a zvolení jiné velikost by mělo být podložené důkladným testováním. Pro vytvoření představy může posloužit následující tabulka, která popisuje dopad velikosti bloku na jednotlivé DML operace v podobě časové náročnosti.

Tabulka 1: Délka zpracování operací pro milión datových souborů

Zdroj: (41)

Velikost bloku	Vytvoření	Čtení	Mazání
4 KB	8m	17m 14s	4m 31s
8 KB	7m 39s	15m 20s	4m 28s
16 KB	7m 50s	14m 47s	5m 2s
32 KB	8m 41s	14m 39s	7m 7s
64 KB	11m 3s	14m 31s	11m 22s

3.5.2 Databázové schéma

Schéma relační databáze by správně mělo dodržovat alespoň třetí normální formu. Normalizované tabulky by tak neměly obsahovat redundantní data a všechny možné vztahy modelu by měly být jasně definované skrze primární a cizí klíče. Ačkoliv je normalizace považována za ideál efektivního ukládání dat v relační databází, tak ne vždy musí vést k nejlepší výkonu při dotazování. (38)

Denormalizace

Neboli proces porušení normální formy, skrze zavedení redundantních sloupců do tabulek, za účelem zlepšení výkonu dotazování. Denormalizací lze například předejít spojování tabulek, což může mít pozitivní dopad na dotazování. Nevýhodu ovšem představuje větší náročnost na diskovou kapacitu a možné zpomalení DML operací. Redundantní data také mohou způsobit snadnou nekonzistenci, jelikož při modifikacích je nutné aktualizovat data na více místech současně. Správu redundantních dat tak mají na starosti většinou triggery nebo dedikované externí aplikace. Před použitím denormalizace by mělo být zváženo, zda možný nárůst výkonu převáží nevýhody. (38)

3.5.3 Výběr tabulek

Pro většinu potřeb postačí klasické haldové tabulky, které mají velmi flexibilní využití. Občas se ale může jevit jako lepší řešení použití speciálního typu tabulky. V takovýchto případech je ale nutné brát v potaz, že speciální typy tabulek mohou mít různá omezení z pohledu dalších optimalizačních technik, které jsou popsané v následujících sekcích. (38)

3.5.4 Datové typy

Databáze Oracle nabízí mnoho datových typů pro různé účely, od běžných čísel s plovoucí desetinnou čárkou, po velké binární bloky, které mohou uchovávat například obrázky. Tato sekce je zaměřena především datovým typům, jež se vyskytují v časových řadách systému. U některých datových typů lze specifikovat maximální délku a přesnost uložené hodnoty. Nevhodně zvolené nebo omezené datové typy mohou mít vyšší nároky na diskovou kapacitu a případný negativní vliv na dotazovaní v podobě nutného přetypování. (38)

Číselné typy

Příkladem může být ukládání peněžních částek, které bývají často zaokrouhlené na dvě desetinná místa. Vhodný datový typ pro tuto situaci je NUMBER s délkou podle maximální očekávané částky a omezením přesnosti. Například pro částky do 100 korun je ideální použít sloupec definovaný jako NUMBER(4, 2), jež přijme maximální hodnotu 99,99. Definici sloupce by samozřejmě bylo možné ponechat jako NUMBER, bez jakékoliv další specifikace, v tomto případě byl Oracle použil defaultní maximální délku. Takovýto sloupec ale umožní i ukládání hodnot s nadbytečnou přesností, která by v případě částek nikdy nebyla využita a zabírala by tak akorát místo na disku. Počet potřebných bajtů pro uložení datového typu NUMBER udává reálná podoba uložené hodnoty. (38)

Databáze Oracle krom NUMBER nabízí například i datový typ INTEGER, který by se v nějakých situacích mohl zdát vhodnější pro celočíselné hodnoty. Jedná se však ale o pouhý alias a INTEGER je na pozadí převeden na NUMBER(38, 0). (40)

Textové typy

Pro textové řetězce je obecně doporučeno využít datový typ VARCHAR2, který má variabilní délku a na rozdíl od datového typu CHAR, není vyplňován prázdnými znaky při nevyužití stanovené maximální délky. Datová náročnost VARCHA2 se tedy odvíjí od délky reálně uložené hodnoty a zvoleného kódování databáze. (38)

Datumové typy

Hlavní datové typy pro uchování datumu a času jsou DATE a TIMESTAMP. Oba tyto typy mají v základu stejnou datovou náročnost, což je 7 bajtů. Hlavním rozdílem je maximální možná přesnost uložené hodnoty. DATE podporuje pouze sekundy, kdežto TIMESTAMP se specifikovanou přesností 9, dokáže uložit i nanosekundy. Datová náročnost TIMESTAMP(9) je poté 11 bajtů (42). Díky stejné základní datové náročnosti je obecně vhodnější využít více flexibilní TIMESTAMP i za předpokladu, že větší přesnost uložené hodnoty není potřebná (38).

3.5.5 Indexy

Jsou datové struktury určené pro urychlení přístupu k požadovaným řádkům tabulky. Index patří vždy specifické tabulce a je složen z jednoho nebo více sloupců.

Vícesloupcový index je nazýván kompozitní (23). Funkce indexu je podobná rejstříku v knize, kde si čtenář může dohledat číslo stránky požadovaného pojmu a nemusí tak zdlouhavě procházet celou knihu stránku po stránce. Při použití indexu databáze ví, kde se požadovaný řádek nachází a nemusí provádět procházení celé tabulky (40).

Index je uložen ve speciální struktuře existující mimo tabulku a obsahuje méně dat, než celý řádek tabulky. Čtení indexu místo řádků tabulky tak vyžaduje méně I/O operací a tudíž je ve většině případů rychlejší. Většina typů indexů je automaticky řazena, což má pozitivní dopad na dotazování podle rozsahu nebo řazení výsledku na základě indexovaného sloupce. (23)

Použití indexů s sebou přináší určitou režii v podobě zpomalení DML operací. Při libovolné změně dat v tabulce je totiž nutné synchronizovat i odpovídající indexy tak, aby byly vždy aktuální. Další nevýhodu představuje dodatečná disková kapacita, která je nutná pro uchování indexů. Z těchto důvodu je vhodné použít indexy pouze tam, kde to opravdu dává smysl. Vhodné kandidáty pro aplikaci indexů představují například: (30)

- sloupce cizích klíčů
- sloupce často používané ve filtrování, řazení a grupování

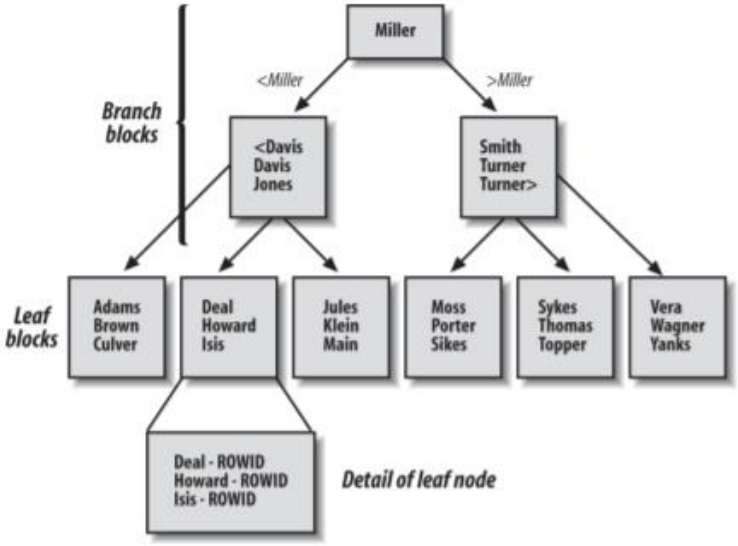
Oracle nabízí několik typů indexů, které mají různé případy užití. Mezi hlavní a nejvíce používané se však řadí B-tree a bitmap index.

B-tree

Je defaultní index databáze Oracle a jak již název napovídá, tak jde o index založený na stromové struktuře. Index je složen z jedné nebo více úrovní takzvaných branch (větev) bloků a jedné úrovně leaf (lístek) bloků (23).

První úroveň stromu představuje kořen, ze kterého vedou větve, na jejichž konci jsou jednotlivé lístky obsahující indexované hodnoty a jejich ROWID. Počet úrovní mezi kořenem a lístky je nazvaný hloubka indexu. Všechny lístky se nachází ve stejné hloubce, čímž je strom vyvážený a každá koncová větev tak odkazuje na přibližně stejný počet lístků. Vyhledání jakéhokoliv lístku ve stromu tedy stojí přibližně stejný

počet I/O operací (23). B-tree index je vhodný pro dotazování na základě přímé rovnosti a rozsahu hodnot (20).



Obrázek 9: Struktura B-tree indexu
Zdroj: (23)

B-tree má i další různé varianty, mezi nejvíce používané patří:

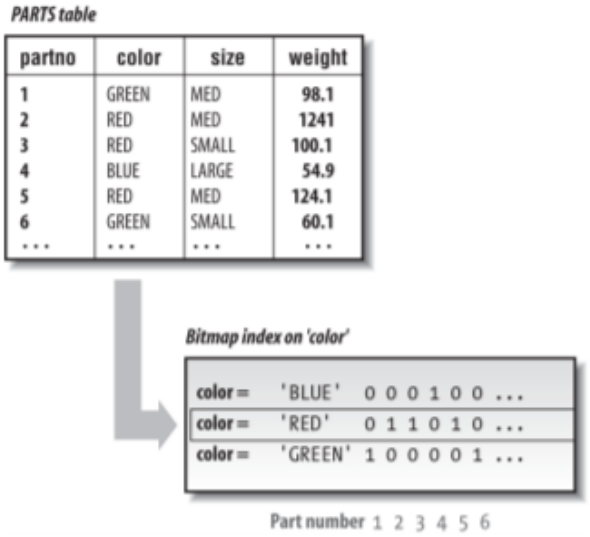
- **Unikátní index** - Jak již název napovídá, tak se jedná o index vynucující unikátnost hodnot. Je používaný především pro primární klíče tabulky, jako prevence proti duplicitě hodnot v sloupci. (20)
- **Reverzní index** - Je index, který má indexované hodnoty v listech v opačném pořadí. Hodnoty 119 a 120 tak jsou převedeny na 911 a 021 a tím uloženy do odlišných lístků. Účelem reverzního indexu je rovnoměrnější rozložení hodnot mezi lístky při ukládání sekvenčních hodnot. Nevýhodou je možné zpomalení čtení, jelikož hodnoty už nejsou seřazené. (43)

```
CREATE INDEX INDEX_NAZEV_KNIHY ON KNIHA(NAZEV); -- Klasický index
CREATE UNIQUE INDEX INDEX_NAZEV_KNIHY ON KNIHA(NAZEV); -- Unikátní index
CREATE INDEX INDEX_NAZEV_KNIHY ON KNIHA(NAZEV) REVERSE; -- Reverzní index
```

Kód 7: Příklad vytvoření B-tree indexu
Zdroj: vlastní zpracování

Bitmap

Je index založený na bitové mapě, kde každá indexovaná hodnota má vlastní bitovou posloupnost, jejíž délka je rovna počtu řádků v tabulce. Pozice bitu v posloupnosti je mapovaná na ROWID řádku. Pokud řádek obsahuje indexovanou hodnotu, tak je příslušný bit nastaven na 1 (true), v opačném případě na 0 (false). (23)



Obrázek 10: Struktura bitmap indexu
Zdroj: (23)

Použití bitmapy je vhodné pro sloupce s malou kardinalitou, tedy sloupce, které obsahují malý počet unikátních hodnot. V takovémto případě bitmapa oproti B-tree představuje výraznou úsporu místa. Pokud se ve sloupci nachází například pět unikátních hodnot, tak v indexu bude uloženo pouze pět bitových posloupností. Bitmapa na rozdíl o ostatních indexů zahrnuje i NULL hodnoty (23). Typické použití představují indexy v OLAP pro tabulky faktů a dimenzí (20).

```
CREATE BITMAP INDEX INDEX_NAZEV_KNIHY ON KNIHA(NAZEV);
```

Kód 8: Příklad vytvoření bitmap indexu
Zdroj: vlastní zpracování

3.5.6 Partitioning

Představuje fyzické rozdělení tabulek a jejich indexů do menších oddílů. Jednotlivé oddíly mohou být uložené i na různých místech, například tak může docházet k přesunu starších dat na pomalejší disky. Oddíly vznikají na základě hodnot

vybraného sloupce nebo kombinace sloupců, nejčastěji však bývá rozdělení prováděno na základě datumu. Rozdělení obvykle přináší obrovský nárůst výkonů při dotazování nad obsáhlými tabulkami, jelikož databáze vůbec nemusí přistupovat k oddílům obsahující data, která nevyhovují podmínkám dotazu (23). Přínos rozdělení je znatelný i při mazaní a zálohování dat (20).

Rozdělení je možné provádět manuálně, nebo automaticky. Možnou nevýhodu automatického rozdělování může představovat nekonzistentní pojmenování oddílů, jelikož Oracle si může zvolit libovolný název. Oddíly je možné přidat i do již existujících tabulek (20). Databáze Oracle nabízí hned několik způsobu rozdělení.

Rozsahové rozdělení

Rozdělení dat je prováděno na základě rozsahu hodnot, nejčastěji dle datumu. Velice obvyklé je rozdělení na kvartály, kde tak každý oddíl obsahuje data za tři specifické měsíce kalendářního roku. U rozsahového rozdělení může docházet k výrazným rozdílům ve velikostech oddílů. (20)

```
CREATE TABLE KNIHA
(
    NAZEV VARCHAR2(200) PRIMARY KEY,
    DATUM_VYDANI DATE NOT NULL
)
PARTITION BY RANGE(DATUM_VYDANI)
(
    PARTITION KNIHY_2021_Q1 VALUES LESS THAN('01/04/2021'),
    PARTITION KNIHY_2021_Q2 VALUES LESS THAN('01/07/2021')
);

-- Přidání dalšího oddílu do již existující tabulky
ALTER TABLE KNIHA ADD PARTITION KNIHY_2021_Q3
VALUES LESS THAN('01/10/2021');
```

Kód 9: Příklad rozsahového rozdělení tabulky

Zdroj: vlastní zpracování

Hashové rozdělení

Data jsou rozdělena na základě výsledné hash hodnoty takzvaného klíče rozdělení, který se skládá z jednoho nebo více sloupců tabulky. Umožnuje definovat rozdělení pouze na základě uvedení požadovaného počtu oddílů. Hashové rozdělení je

užitečné v případech, kdy je potřeba rovnoměrně rozdělit data napříč více oddíly, nebo když nelze jednoznačně specifikovat jiný způsob rozdělení. (20)

```
CREATE TABLE KNIHA
(
    NAZEV VARCHAR2(200) PRIMARY KEY,
    KATEGORIE VARCHAR2(30) NOT NULL
)
PARTITION BY HASH(KATEGORIE)
PARTITIONS 2;
```

Kód 10: Příklad hashového rozdělení tabulky

Zdroj: vlastní zpracování

Seznamové rozdělení

K rozdělení dochází podle specifických hodnot ve sloupci. Pro všechny ostatní neodpovídající hodnoty je vhodné definovat samostatný oddíl pomocí klíčového slova DEFAULT. Pokud tento oddíl není definovaný, tak Oracle nedovolí vložení řádku s neodpovídající hodnotou do tabulky. (20)

Od verze Oracle 12c je možné definovat seznamové rozdělení podle více sloupců (44). Přibyla také možnost automatického rozdělení, které samo vytvoří oddíl pro každou novou unikátní hodnotu (45).

```
CREATE TABLE KNIHA
(
    NAZEV VARCHAR2(200) PRIMARY KEY,
    KATEGORIE VARCHAR2(30) NOT NULL
)
PARTITION BY LIST(KATEGORIE)
(
    PARTITION UCEBNICE VALUES('ZEMEPIS', 'CHEMIE', 'MATEMATIKA'),
    PARTITION KUCHARKY VALUES('ITALSKA_KUCHYNE', 'CESKA_KUCHYNE')
);
```

Kód 11: Příklad seznamového rozdělení tabulky

Zdroj: vlastní zpracování

Kompozitní rozdělení

Jde o sub-rozdělení, tedy rozdělení jiného rozdělení, skrze libovolnou kombinaci předchozích způsobů. Tento způsob je vhodný především pro velmi velké tabulky, ve kterých by jedna úroveň rozdělení nemusela být dostačující. (20)

```
CREATE TABLE KNIHA
(
    NAZEV VARCHAR2(200) PRIMARY KEY,
    DATUM_VYDANI DATE NOT NULL,
    KATEGORIE VARCHAR2(30) NOT NULL
)
PARTITION BY RANGE(DATUM_VYDANI)
SUBPARTITION BY HASH(KATEGORIE)
SUBPARTITIONS 6
(
    PARTITION KNIHY_2021_Q1 VALUES LESS THAN('01/04/2021'),
    PARTITION KNIHY_2021_Q2 VALUES LESS THAN('01/07/2021')
);
```

Kód 12: Příklad kompozitního rozdělení tabulky

Zdroj: vlastní zpracování

Rozdělení indexů

Pro rozdělené tabulky je vhodné vytvářet i rozdělené indexy, ty můžou být lokální, nebo globální. Lokální index je rozdělen na části pro jednotlivé oddíly tabulky, jedna část indexu tak obsahuje pouze hodnoty jednoho oddílu. Globální index naopak může obsahovat hodnoty z více oddílů, jelikož je zde možné definovat odlišné rozdělení, než je použité v tabulce. (20)

Při vytváření lokálního rozděleného indexu je možné specifikovat již existující oddíly tabulky, pro které mají být vytvořené části indexu. Pokud žádné oddíly nejsou specifikované, tak Oracle automaticky vytvoří části indexu pro každý existující i nově přidaný oddíl. (20)

V případě odlišného rozsahového rozdělení u globálního indexu je nutné specifikovat část pro všechny ostatní hodnoty, které nevyhovují uvedenému rozdělení. Pro tento účel se používá klíčové slovo MAXVALUE. (46)

```
-- Lokální index se specifikací oddílů
CREATE INDEX INDEX_DATUM_VYDANI ON KNIHA(DATUM_VYDANI) LOCAL
(
    PARTITION KNIHY_2021_Q1,
    PARTITION KNIHY_2021_Q2
);

-- Globální index s odlišným rozdělením oproti tabulce
CREATE INDEX INDEX_DATUM_VYDANI ON KNIHA(DATUM_VYDANI) GLOBAL
PARTITION BY RANGE(DATUM_VYDANI)
(
    PARTITION KNIHY_2021_LEDEN VALUES LESS THAN('01/02/2021'),
    PARTITION KNIHY_2021_UNOR VALUES LESS THAN('01/03/2021'),
    PARTITION KNIHY_MAX VALUES LESS THAN(MAXVALUE)
);
```

Kód 13: Příklad vytvoření rozdělených indexu

Zdroj: vlastní zpracování

3.5.7 Komprese tabulek

Databáze Oracle nabízí hned několik možností komprese dat pro ušetření místa na disku. Komprese je založena na vyhledání duplicitních hodnot a jejich nahrazení za referenci, která ukazuje do tabulky symbolů. Odstraněním duplicity je tak možné dosáhnout vysokých kompresních poměrů, a to především u dat s nízkou kardinalitou. Komprimované řádky jsou navíc uložené ve speciálním formátu pro sjednocení všech sloupců tabulky do jediného bloku, což může mít pozitivní dopad na dotazování. Kompresi je možné deklarovat pro celý tabulkový prostor a jednotlivý oddíl i sub-oddíl tabulky (32). Databáze Oracle nabízí i různé nástroje pro odhad a monitorování kompresních poměrů, které jsou dostupné v balíčku DBMS_COMPRESSION (47).

Základní komprese

Je starší funkcionalita, která byla navržena převážně pro statická data. Základní varianta nepodporuje komprimaci přírůstkových řádků, které jsou vložené nebo modifikované skrze klasické DML operace, tedy INSERT a UPDATE. Za účelem jejích komprimace musí být použité takzvané direct path operace, které zapisují data přímo do datového souboru. Využití základní komprese tedy nachází uplatnění především v OLAP úložištích nebo u neaktivních oddílů tabulek. (32)

Pokročilá komprese

Řeší nedostatky základní komprese pro OLTP úložiště, kde dochází k časté modifikaci tabulek. Tato varianta již podporuje komprimaci dat vložených skrze klasické DML operace (32). Další výhodou je, že čtená data mohou zůstat nadále komprimovaná i v operační paměti databázového severu (48).

```
CREATE TABLE KNIHA
(
    NAZEV VARCHAR2(200) PRIMARY KEY,
    DATUM_VYDANI DATE NOT NULL
)
PARTITION BY RANGE(DATUM_VYDANI)
(
    PARTITION KNIHY_2021_LEDEN VALUES LESS THAN('01/02/2021')
    ROW STORE COMPRESS BASIC, -- Základní komprese
    PARTITION KNIHY_2021_UNOR VALUES LESS THAN('01/03/2021')
    ROW STORE COMPRESS ADVANCED, -- Pokročilá komprese
    PARTITION KNIHY_OSTATNI VALUES LESS THAN(MAXVALUE)
    NOCOMPRESS -- Bez komprese
);
```

Kód 14: Příklad komprese oddílů tabulky
Zdroj: vlastní zpracování

3.5.8 Komprese indexů

Obdobně jako u komprese tabulek je hlavním účelem ušetření místa na disku. Komprese je k dispozici pouze pro index B-tree nebo v indexově uspořádané tabulce[7]. Princip je podobný jako u komprese tabulky, v indexu jsou identifikovány duplicitní hodnoty a následně nahrazeny za referenci. Kompresi je možné definovat i na úrovni jednotlivých částí rozděleného indexu. (50)

Základní komprese

Umožňuje kompresi unikátních i neunikátních indexů. Při vytváření indexu je možné specifikovat kolik z sloupců indexu je potřeba komprimovat, za předpokladu, že se jedná o složený index. Pokud počet sloupců ke komprimaci není uvedený, tak

[7] Pouze základní komprese je podporována u indexově orientované tabulky. (49)

Oracle použije defaultní hodnoty. Pro neunikátní index je defaultní hodnota rovna počtu sloupců indexu, pro unikátní to je poté počet sloupců indexu – 1. (50)

Pokročilá komprese

Je více sofistikovaná varianta komprese, která dosahuje daleko vyšších kompresních poměrů. Pokročilá komprese je ale již příplatková položka, která není zahrnuta v enterprise edici databáze Oracle. Při použití pokročilé komprese jsou k dispozici dva módy, LOW a HIGH. (49)

- **LOW** – Jak již jméno napovídá, tak tento mód dosahuje menších kompresních poměrů, ale na oplátku vyžaduje při DML operacích méně procesorového času navíc. Tento mód je vhodný pro často modifikovaná data OLTP úložišť. (49)
- **HIGH** – Mód aplikuje více komplexní algoritmy, které při DML operacích více vytěžují procesor. Výhodou jsou ale velmi vysoké kompresní poměry, které jsou užitečné pro zřídka modifikovaná data OLAP úložišť. (49)

```
CREATE INDEX INDEX_DATUM_VYDANI ON KNIHA(DATUM_VYDANI)
COMPRESS 1; -- Základní komprese

CREATE INDEX INDEX_DATUM_VYDANI ON KNIHA(DATUM_VYDANI)
COMPRESS ADVANCED LOW; -- Pokročilá komprese
```

Kód 15: Příklad komprese indexu

Zdroj: vlastní zpracování

3.5.9 Dotazování

Za účelem dosažení co možná nejlepšího výkonu databáze je nutné se na optimálně uložená data i správně dotazovat. Rozdíl mezi správným a špatným dotazem může být u některých tabulek i v řádu desítek sekund. V databázi Oracle se o efektivní zpracování dotazu stará modul nazvaný optimalizátor (query optimizer).

Optimalizátor

Úkolem optimalizátoru je zvolení nejlepšího způsobu zpracování libovolného SQL dotazu. Tento proces nazývající se optimalizace dotazu je aplikován na všechny typy SQL klauzulí. Při optimalizaci může dojít i modifikaci dotazu za účelem dosažení vyšší efektivity. Logika původního dotazu však zůstává a modifikovaný dotaz vrací

totožné výsledky. Příkladem modifikace může být již zmiňované nahrazení tabulky materializovaným pohledem. (38)

Oracle v minulosti využíval optimalizátor založený na jednoduchých pravidlech (RBO), který se ale časem ukázal jako méně efektivní pro komplexní dotazování. Ve verzi Oracle 7 tedy došlo k představení optimalizátoru založeném na výpočtu ceny (CBO). Při hledání nejlepšího způsobu podle ceny, optimalizátor prozkoumá několik možných scénářů zpracování dotazu a pro každý krok v nich vypočítá odhad relativní ceny. Optimalizátor následně vybere scénář s nejmenší celkovou cenou, který se nazývá exekuční plán dotazu. Cena představuje míru odhadovaného množství potřebných zdrojů pro vykonání dotazu. Při výpočtu ceny se zvažují například tyto faktory: (38)

- odhadované nároky na I/O, procesorový čas a operační paměť
- odhadovaný počet vracených řádků
- možné přístupové cesty, použití různých typů spojení a indexů

Výpočet ceny je možné ovlivnit nastavením různých parametrů optimalizátoru. Mezi ně se například řadí: (51)

- **CURSOR_SHARING** – Povolení převodu literálů v dotazu na bindované proměnné. Sdílení kurzorů, místo přímo zapsaných hodnot může mít pozitivní vliv na exekuční plán.
- **DB_FILE_MULTIBLOCK_READ_COUNT** – Specifikuje počet bloků čtených jednou I/O operací. Podle zvolené hodnoty dochází k zvýhodnění různých přístupových cest.
- **OPTIMIZER_MODE** – Určuje cíl optimalizátoru. První variantou je prioritizace nejrychlejšího vrácení všech řádků dotazu. Druhou možností je nejrychlejší vrácení prvních n řádků dotazu, což je vhodné zejména při načítání pro koncového uživatele se stránkováním.
- **OPTIMIZER_INDEX_COST_ADJ** – Ovlivňuje výpočet ceny podle indexů, čím menší hodnota, tím více jsou indexy zvýhodněné (levnější).

```sql
-- Nastavení cíle optimalizátoru na prvních 100 řádků (pro relaci)
ALTER SESSION SET OPTIMIZER_MODE = FIRST_ROWS_100;

-- Nastavení převodu všech literálů (globálně, vyžaduje admin práva)
ALTER SYSTEM SET CURSOR_SHARING = FORCE;
```

Kód 16: Příklad nastavení parametrů optimalizátoru

Zdroj: vlastní zpracování

Statistiky

Pro informované rozhodnutí potřebuje optimalizátor dobrý přehled o objektech v dotazu a dostupných prostředcích databáze. Tento přehled mu nabízí kolekce dat nazvaná statistiky, která detailně popisuje databázi a její objekty. Tato kolekce je uložena v datovém slovníku a zahrnuje statistiky pro: (52)

- **Tabulky** – počet řádků a bloků, průměrná délka řádku
- **Sloupce** – počet unikátních a NULL hodnot, distribuce hodnot
- **Indexy** – počet lístků a úrovní indexu
- **Systém** – I/O a procesový výkon a vytíženost

Za účelem získání co nejlepších exekučních plánů je nutné statistiky udržovat aktuální. Zastaralé, nebo dokonce chybějící statistiky mají velice negativní vliv na rozhodovací schopnosti optimalizátoru. Sběr statistik je možné provádět dvěma způsoby: (52)

- **Automaticky** – Je defaultní a doporučený způsob sběru. Oracle tak sbírá statistiky pravidelně bez zásahu administrátora databáze. Statistiky jsou aktualizované pouze v případě nutnosti, tedy pokud neexistují, nebo jsou zastaralé. Sběr je defaultně plánován přes noc a nepracovní dny, pro co nejmenší dopad na výkon databáze v provozních hodinách.
- **Manuálně** – Je vhodný způsob pokud automatický sběr není dostačující, například kvůli velmi častým změnám dat nebo když je nutné aktualizovat statistiky ihned. Manuální sběr nabízí takzvané vzorkování, které určuje kolik procent položek objektu stačí projít, pro získání dostatečně přesných statistik. Díky vzorkování tak databáze nemusí procházet všechny položky objektu.

```
-- Manuální sběr statistik indexu se vzorkováním 10%
EXECUTE DBMS_STATS.GATHER_INDEX_STATS('IDX_DATUM', 10);
```

Kód 17: Příklad manuálního sběru statistik

Zdroj: vlastní zpracování

Přístupové cesty

Jsou techniky používané databází Oracle pro efektivní vyhledávání řádků tabulek v různých situacích. Přístupové cesty jsou závislé na typu použité tabulky a indexu. Mezi hlavní přístupové cesty pro haldové tabulky a B-tree indexy se řadí: (53)

- **Full Table Scan** – Postupné procházení všech řádků tabulky, které většinou bývá nejmíň efektivní možností, především u velkých tabulek. Většinou k němu dochází pokud má tabulka málo řádků nebo nemá použitelný index.
- **Table Access by ROWID** – Vyhledání řádků skrze pseduo-sloupec ROWID je nejrychlejší způsob nalezení jednoho řádku. Využívá se většinou až po skenu indexů nebo pokud je ROWID specifikováno v dotazu.
- **Index Unique Scan** – Vrací maximálně jeden řádek a používá se pokud predikát dotazu obsahuje sloupec s unikátním indexem.
- **Index Range Scan** – Vyhledání řádků podle sloupce s řazeným indexem. Rozsah může být skenován z jedné nebo obou stran. Tato cesta je využita především pokud predikát dotazu obsahuje operátory < nebo >. Vrácené řádky jsou ve vzestupném pořadí, existuje ale i varianta pro sestupné pořadí.
- **Index Full Scan** – Dochází k prohledání všech lístků indexu ve větvích podle pořadí, dokud není nalezen první odpovídající lístek. Cesta je vhodná pokud se v dotazu nachází klauzule ORDER BY nad sloupcem bez NULL hodnot, jelikož v tomto případě je řazení možno přeskočit.
- **Index Fast Full Scan** – Prohledává všechny bloky včetně větví bez pořadí a na rozdíl od předchozí cesty pracuje pouze s indexem. Cesta je vhodná pokud predikát dotazu obsahuje pouze indexované sloupce.
- **Index Skip Scan** – Vyhledání na základě logického rozpadu kompozitního indexu sloupce na sub-indexy. Cesta bývá použita pokud predikát dotazu neobsahuje hlavní sloupec indexu nebo pokud hodnoty v hlavním sloupci mají

malou kardinalitu, ale vedlejší sloupce již ne. Skip scan je více efektivnější než předchozí dva způsoby.

- **Index Join Scan** – Provádí hashové spojení na úrovni více indexů, pokud se v nich nachází všechny požadované sloupce z dotazu.

Spojování

Je kombinace dvou řádků z různých zdrojů do jediného řádku na základě spojovací podmínky. Zdrojem je zde myšlena klasická či dočasná tabulka nebo pohled. V databázi Oracle dochází k těmto možným spojením: (54)

- **Nested Loops Join** – Spojení vnořenými cykly, kde dochází k procházení každého řádku prvního zdroje a v druhém zdroji se hledají odpovídající řádky, které splňují spojovací podmínku. Toto spojení je vhodné pokud oba spojované zdroje obsahují méně dat nebo když je cílem optimalizátoru nejrychlejší vrácení prvních n řádků.
- **Hash Join** – Spojení skrze hash tabulku. Databáze vybere zdroj s menším počtem řádků a pro hodnoty v jejím spojovacím sloupci vytvoří tabulku hashovaných hodnot v operační paměti. Ve větším zdroji jsou poté za pomoci hashové tabulky hledány řádky splňující podmínku spojení. Hashové spojení je vhodné pro velké zdroje a spojení na základě přímé rovnosti.
- **Sort Merge Join** – Je variace nested loops spojení. Oba zdroje jsou nejdříve seřazeny podle spojovacího sloupce a následně spojeny. Tento způsob bývá používán především, pokud některé další operace dotazu vyžadují řazení nebo pro spojení na základě nerovnosti.
- **Cartesian Join** – Představuje kartézský součin obou zdrojů, tedy spojení každého řádku s každým. Kartézské spojení je použito pouze pokud není definovaná spojovací podmínka nebo pokud jsou oba zdroje velmi malé.

Výběr vhodného spojení provádí optimalizátor, za předpokladu, že není ovlivněn hintem. Spojení je obvykle náročná operace, jež stojí hodně prostředků (38). Různé metody pro eliminaci spojování byly popsané v předchozích sekcích. Způsob potlačení dopadů spojování může představovat paralelizace.

Paralelizace

Umožnuje vícevláknové zpracování libovolného SQL dotazu, pro efektivní využití vícejádrových procesorů. Paralelizace tedy představuje rozpad zpracování dotazu na menší části, kde každé vlákno zpracovává svůj přidělený úkol, například jedno vlákno může provádět abecední řazení hodnot od A až K a druhé vlákno zbytek, tedy L až Z. Paralelizace dále dovoluje přístup k více diskům současně, kde každé vlákno může číst či modifikovat data na jiném disku. U sériového jednovláknového zpracování musí být přístup k diskům prováděn postupně (38). Paralelizace je hojně využívaná především v OLAP úložištích, pro OLTP ale také nabízí možný přínos. Paralelizace zlepšuje výkon zpracování například při: (55)

- procházení velkých tabulek a indexů nebo komplexním spojování
- vytváření a aktualizování obsáhlých indexů
- sběru statistik
- hromadnému ukládání, aktualizování a mazání dat

Paralelizace naopak není vhodná pro úložiště, které jsou hardwarově limitované a často velmi vytížené. Důvodem jsou částečně i náklady spojené s koordinací více vláken. Obecně by měla být paralelizace využita pouze při zpracování dlouhodobých dotazů a za předpokladu, že má úložiště dostupné požadované hardwarové prostředky, tedy vícejádrové procesory a data rozprostřená mezi více disky. (38)

Paralelizace je v databázi Oracle defaultně povolená a o jejím využití rozhoduje optimalizátor na základě dostupných prostředků. Paralelizaci je možné konfigurovat zvlášť pro DDL a DML operace, dotazy a funkce. Při konfiguraci je možné specifikovat takzvanou úroveň paralelizace, což je maximální počet rozpadů operace. Například hodnota čtyři určuje, že operace může být rozložena do čtyř menších části. Pokud úroveň není uvedena, tak si jí optimalizátor určí sám. (55)

```
-- Povolení paralelizace DML operací
ALTER SESSION ENABLE PARALLEL DML;
-- Vynucení paralelizace DML operací s úrovní 4
ALTER SESSION FORCE PARALLEL DML PARALLEL 4;
```

Kód 18: Příklad nastavení parametrů paralelizace

Zdroj: vlastní zpracování

Hinty

Představují již několikrát zmiňované ovlivnění optimalizátoru prostřednictvím instrukcí v dotazu. Instrukcemi lze specifikovat například požadovaný mód optimalizátoru, přístupovou cestu, použití indexů, pořadí a způsob spojování tabulek. Optimalizátor použije hint pouze v případě, že se jedná o validní operaci. Pokud tedy v hintu bude uveden například neexistující sloupec tabulky, tak ho optimalizátor bude ignorovat a dotaz provede podle svého uvážení. (51)

Použití hintů je obecně doporučené spíše pro testovací účely. Hint sice může vyřešit aktuální problém, ale díky různým změnám v databázi a konfiguraci prostředí se může stát zastaralým a přinést tak negativní dopady na dotazování (51). Seznam všech hintů je k dispozici v oficiální dokumentaci.

```
-- Hint pro použití přístupové cesty full table scan
SELECT /*+ FULL(KNIHA) */ NAZEV
FROM KNIHA
WHERE DATUM_VYDANI >= TO_DATE('01-01-2021', 'DD-MM-YYYY');
```

Kód 19: Příklad dotazu s hintem

Zdroj: vlastní zpracování

Exekuční plán

Je soupis všech potřebných kroků pro vykonání libovolného SQL dotazu. Jednotlivé kroky představují již zmiňované přístupové cesty, použité spojení a další potřebné operace pro přípravu dat. Čtením exekučního plánu je tedy možné jistit operace a jejich pořadí, tak, jak je optimalizátor zvolil. Pro každou operaci jsou v plánu k dispozici následující parametry:

- cena a kardinalita[8] operace
- počet čtených oddílů, pokud je tabulka rozdělena do oddílů
- přehled paralelního zpracování, pokud bylo využito
- počet čtených bajtů operací
- čas v sekundách potřebný pro zpracování operace

[8] Kardinalita zde představuje očekávaný počet řádků vrácených operací.

Získání exekučního plánu pro ladění dotazů a testování výkonu databáze je možné více způsoby. Mezi ně patří klauzule EXPLAIN PLAN a funkcionalita AUTOTRACE.

- **EXPLAIN PLAN** – Vygeneruje odhadovaný exekuční plán do zvolené tabulky. Při odhadování plánu nedochází k provedení dotazu a výsledek se tedy od reálného plánu může lišit. Pro vygenerování plánu musí být dotaz modifikován přidáním klauzule EXPLAN PLAN. Po provedení modifikovaného dotazu je poté nutné manuálně plán vypsat pomocí dodatečného dotazu. (56)
- **AUTOTRACE** – Na rozdíl od předchozí varianty, zde dochází k provedení dotazu a výsledkem je tak reálně použitý exekuční plán. AUTOTRACE je dostupný pouze ve vybraných vývojářských nástrojích, například Oracle SQL Developer. Výhodu představuje snadnější použití, jelikož pro získání plánu není nutná modifikace dotazu klauzulí a k vypsání plánu dochází automaticky. Součástí vygenerovaného plánu mohou být i detailnější statistické informace. Pro použití AUTOTRACE musí mít uživatel přidělená patřičná práva. (57)

Interpretace exekučního plánu

Pro správnou interpretaci a hledání možných problémů je nutné umět se orientovat ve vypsaném exekučním plánu. Pro formátovaný plán platí dvě základní pravidla. Čím více je operace ve výpisu odsazena, tím dříve byla provedena. Když se na stejné úrovni nachází více operací, tak byly prováděny směrem od shora dolů. (38)

Jako příklad můžeme mít následující situaci. Mějme haldovou tabulku KNIHA, která obsahuje sloupce NAZEV, DATUM_VYDANI a KATEGORIE_ID. Tabulka KNIHA je rozsahově rozdělaná podle sloupce DATUM_VYDANI na jednotlivé měsíce roku 2021. Pro tabulku existuje rozdělený kompozitní B-tree index IDX_KNIHA, který se skládá se ze sloupců DATUM_VYDANI a KATEGORIE_ID. Druhá tabulka je halda s názvem KATEGORIE, která obsahuje sloupce ID a NAZEV. Tabulka KATEGORIE není rozdělená a pro sloupec NAZEV existuje nerozdělený unikátní B-tree index IDX_KATEGORIE. Nad těmito tabulkami provedeme následující dotaz, pro který si necháme vypsat odhadovaný exekuční plán.

```
-- Dotaz na plán získání učebnic vydaných v prvním týdnu února 2021
EXPLAIN PLAN FOR
SELECT t1.NAZEV
FROM KNIHA t1
JOIN KATEGORIE t2 ON t2.ID = t1.KATEGORIE_ID
WHERE t1.DATUM_VYDANI >= '01-02-2021'
  AND t1.DATUM_VYDANI <= '07-02-2021'
  AND t2.NAZEV = 'UCEBNICE';

SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY); -- Výpis plánu z tabulky

-- Získaný plán (za účelem ušetření místa byly odebrány některé sloupce)
-------------------------------------------------------------------------------------------------
| Id | Operation                           | Name          | Rows  | Cost (%CPU) | Pstart | Pstop |
-------------------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT                    |               |   158 |     3   (0) |        |       |
|  1 |  NESTED LOOPS                       |               |   158 |     3   (0) |        |       |
|  2 |   TABLE ACCESS BY INDEX ROWID       | KATEGORIE     |     1 |     1   (0) |        |       |
|* 3 |    INDEX UNIQUE SCAN                | IDX_KATEGORIE |     1 |     0   (0) |        |       |
|  4 |   PARTITION RANGE SINGLE            |               |   158 |     2   (0) |      2 |     2 |
|  5 |    TABLE ACCESS BY LOCAL INDEX ROWID| KNIHA         |   158 |     2   (0) |      2 |     2 |
|* 6 |     INDEX RANGE SCAN                | IDX_KNIHA     |     4 |     2   (0) |      2 |     2 |
-------------------------------------------------------------------------------------------------
3 - access("T2"."NAZEV"='UCEBNICE')
6 - access("T2"."ID"="T1"."KATEGORIE_ID"
          AND "T1"."DATUM_VYDANI">=TO_DATE('2021-02-01 00:00:00', 'syyyy-mm-dd hh24:mi:ss')
          AND "T1"."DATUM_VYDANI"<=TO_DATE('2021-02-07 00:00:00', 'syyyy-mm-dd hh24:mi:ss'))
    filter("T2"."ID"="T1"."KATEGORIE_ID")
```

Kód 20: Příklad exekučního plánu

Zdroj: vlastní zpracování

Získaný exekuční plán se skládá ze 7 operací. Odhadovaný počet vrácených řádků je 158 a celková cena plánu je 3. Podle exekučního plánu dojde k procházení pouze jednoho oddílu tabulky KNIHA, konkrétně oddílu s číslem 2, což je ideální, jelikož se ptáme pouze na knihy vydané v únoru. Prořezání ostatních oddílů tabulky proběhlo na základě predikátů operace #6. Nyní již k jednotlivým operacím, které by podle plánu byly prováděné v následujícím pořadí:

- Operace #6 – rozsahový sken rozděleného indexu IDX_KNIHA
- Operace #3 – unikátní sken indexu IDX_KATEGORIE
- Operace #5 – přístup k řádkům tabulky KNIHA podle IDX_KNIHA
- Operace #2 – přístup k řádkům tabulky KATEGORIE podle IDX_KATEGORIE
- Operace #4 – přístup k oddílu číslo 2 tabulky KNIHA
- Operace #1 – spojení řádků z operací #2 a #4 skrze nested loops

- Operace #0 – vrácení všech nalezených řádků

Ve výsledku tedy lze říci, že odhadovaný exekuční plán je pro uvedenou situaci efektivní. Optimalizátor správně zvolil prořezání oddílů tabulky KNIHA a přístup k řádkům obou tabulek skrze dostupné indexy.

4 Stávající datový model

Nyní přichází na řadu představení datového modelu časových řad, který je v systému společnosti Unicorn aktuálně využíván. Před rozborem jednotlivých tabulek a ostatních databázových objektů je ale nutné se nejdříve podívat na konfiguraci samotné databáze.

Uvedené parametry a informace byly získané z interní dokumentace společnosti Unicorn a přímou analýzou stávajícího modelu na interním testovacím prostředí. Některé použité dotazy pro analýzu jsou dostupné v **příloze 2**.

4.1 Inicializační parametry

Databáze používá defaultní hodnoty téměř pro všechny inicializační parametry. Samozřejmostí je upravený název databáze a parametry týkající se kultury. Jediné upravené parametry, které stojí za zmínku jsou:

- **DB_FILES** – Maximální počet datových souborů s hodnotou 60 tisíc, defaultní je 200. Důvodem je rozdělení databáze do velkého množství tabulkových prostorů, které je popsané v následující sekci.
- **ENABLE_PLUGGABLE_DATABASE** – Povolení pluggable módu (PDB), ve kterém může být databáze součástí kontejneru s více databázemi. Tento mód je defaultně vypnutý, ale zde je zapnutý. Aktuálně je na produkčním prostředí v kontejneru pouze jediná databáze. Důvodem pro volbu módu PDB je snadnější správa a monitorování databáze pro administrátora.
- **OPTIMIZER_ADAPTIVE_PLANS –** Povolení adaptivních exekučních plánů, které se mohou měnit na základě průběhu aktuálně prováděného dotazu. Tato funckionalita je defaultně zapnuta, ale zde je vypnuta. Důvodem pro vypnutí adaptivních plánů byla nekonzistence dotazování v některých extrémních případech. Při zapnutém adaptivním módu se totiž optimalizátor může rozhodnout pro přerušení dlouhodobého dotazu a vyzkoušet jiný exekuční plán. Toto nemusí být požadované chování, jelikož alternativní exekuční plán může být horší a ve výsledku pak dlouhodobý dotaz trvá ještě déle.

4.2 Schéma

Časové řady a ostatní data jsou vedené pod schématem USRBUF. Dočasný tabulkový prostor tohoto schématu je předdefinovaný TEMP a jako defaultní prostor je využit opět předdefinovaný USERS, na kterém je nastavena kvóta 50 MB. Název schématu USRBUF nedodržuje konvenci pojmenování schémat v PDB a správně by mělo být vedeno jako C##USRBUF (58). Důvodem pro porušení jmenné konvence je fakt, že databáze je v módu PDB poměrně krátkou dobu a schéma USRBUF existuje již od samotného vytvoření databáze. Schéma USRBUF má neomezené kvóty na všechny tabulkové prostory, které uchovávají data. Schéma USRBUF kromě názvu vyhovuje všem správným praktikám a žádná změna zde není nutná.

4.3 Tabulkové prostory

Pro časové řady jsou definované vlastní tabulkové prostory, jež se dělí podle typu uchovávaných dat. Pro data tabulek existují prostory s prefixem DAE_DATA a pro indexy se používá prefix DAE_INDX. Tabulkové prostory jsou dále dělené do kvartálů. Například pro první kvartál roku 2021 tak existují tabulkové prostory s názvy DAE_DATA_Q_2021_1 a DAE_INDX_Q_2021_1. Krom těchto kvartálových prostorů existují i „obecné“ prostory DAE_DATA a DAE_INDEX, ve kterých jsou uchovány data tabulek a indexů z méně objemných časových řad, například obchodní dimenze, které mají maximálně desítky až tisíce hodnot.

Rozdělení do jednotlivých tabulkových prostorů je výhodné z pohledu snazší správy a možnosti přesunu datových souborů obsahujících starší data do jiné diskové skupiny, která může být pomalejší a tudíž i levnější. Rozdělení do více diskových skupin se ale aktuálně na produkčním prostředí neděje a všechny datové soubory jsou uloženy v jedné diskové skupině na velkoobjemových SSD discích.

Každý tabulkový prostor je lokálně spravovaný a má nastavenou automatickou správu volného místa, což je ideální pro efektivní využití diskové kapacity a zabránění souboji o tabulky datového slovníku.

Velikost datového bloku je ve všech tabulkových prostorech nastavena na defaultních 8 KB. Žádná tabulka neobsahuje zřetězené řádky a tudíž by nastavení

větší velikosti bloku, krom pokusu o dosažení marginálně většího výkonu dotazování, nedávalo smysl. Menší velikost bloku by sice mohla pozitivně působit na DML operace, ale naopak negativně na čtení hodnot.

4.4 Tabulky časových řad

Všechny časové řady jsou uloženy v klasických haldových tabulkách, které mají defaultní inicializační parametry. Tabulky využívají pouze indexy typu B-tree a žádné jiné typy indexů nejsou v modelu použité. Názvy tabulek začínají prefixem TS, jež následuje unikátní identifikátor v podobě celého 17místného čísla, například TS_14500000002085720.

Pro využití speciálních druhů tabulek není v modelu žádný důvod. Haldové tabulky poskytují všechny požadované vlastnosti a přechod na jiný druh by pravděpodobně přinesl negativní dopady na výkon dotazování a datovou náročnost.

V teoretické části bylo zmíněno rozdělení řad podle typu ukládaných dat na primární a odvozené. Ačkoliv je toto rozdělení správné, tak ho model časových řad přehlíží a tabulky obou typů mají totožnou strukturu. Datový model pro tabulky časových řad je tak velice obecný, což umožňuje možné znovupoužití, pro ukládání různých druhů dat.

4.4.1 Časové řady hodnot

Hodnotové časové řady představují majoritní objem uložených dat. Jedná se o časové řady obsahující reportovaná obchodní a settlementem vypočítaná data s různou časovou granularitou. V rámci jedné hodnotové časové řady je však granularita jednotná. Aktuálně v systému existují časové řady s hodinovou, denní, týdenní a měsíční granularitou. Nově je nutné počítat i s uložením hodnot na čtvrthodinové bázi. Každá hodnotová časová řada je dedikovaná hodnotám specifického typu MECu, například již zmiňovaná řada naměřené spotřeby energie, která má aktuálně dvě verze, primární hodinovou a odvozenou denní. Počet řádků v hodnotových časových řadách se pohybuje v milionech. Po přechodu na čtvrthodinová data můžou mít některé časové řady i miliardy řádků.

4.4.2 Časové řady obchodních dimenzí

Účastníci a entity trhu v datovém modelu vystupují jako obchodní dimenze. Pro uložení informací o nich jsou využité časové řady. Důvodem je to, že vazby mezi účastníky a entitami se mohou v čase měnit, například vazba DSO-MGA, kde DSO může postupem času odpovídat za různé MGA. Pro tyto situace je tedy nutné definování intervalu platnosti, takzvané validity. Validita určuje od kdy a do kdy je dimenze, nebo vztah mezi nimi platný. Pro každou obchodní dimenzi existuje dedikovaná časová řada, kde se nachází pouze dimenze jednoho typu. Časová granularita je zde dynamická, jelikož obchodní dimenze může mít libovolný začátek i konec a tudíž i libovolnou délku intervalu platnosti. Krom validity obsahují tyto časové řady také doplňující informace o dimenzi, například její název a interní kód. Časové řady obchodních dimenzí mají jednotky i tisíce řádků.

4.4.3 Časové řady MECů

V časových řadách jsou ukládané i samotné MECy, což jsou ve své podstatě také jen vazby mezi účastníky a entitami, které se v čase mohou měnit. Stejně jako obchodní dimenze, má i MEC svou validitu, která může být libovolná. Časové řady uchovávající MECy mají tedy dynamickou granularitu. Každý typ MECu má vlastní dedikovanou časovou řadu. Počet řádků se zde pohybuje ve stovkách až tisících.

4.4.4 Sloupce časové řady

Většina časových řad má stejné sloupcové obsazení, bez ohledu na to, zda se jedná o primární, nebo odvozenou řadu. Konkrétně se jedná o následující sloupce.

OID

Unikátní celočíselný identifikátor řádku v rámci tabulky, který začíná jedničkou. Pro generování OID není použita sekvence a hodnota je určována externí aplikací. Datový typ sloupce OID je definován jako NUMBER(22).

T103TIME_UNIT_VALUE

Unikátní identifikátor intervalu hodnoty z kalendářové tabulky, která bude rozvedena v následující sekci. Jelikož se jedná o cizí klíč sloupce OID z kalendářové tabulky, tak je zde datový typ totožný, tedy NUMBER(22).

FROM_TIME

Časová značka udávající začátek intervalu hodnoty, vztahu nebo objektu. Je zde použit datový typ TIMESTAMP(6), jež podporuje přesnost na sekundy.

TO_TIME

Časová značka udávající konec intervalu hodnoty, vztahu nebo objektu. Stejně jako v případě FROM_TIME je použit datový typ TIMESTAMP(6).

BD

Identifikátor libovolné obchodní dimenze. Sloupec BD má datový typ NUMBER(22) a obsahuje identifikátor dimenze, což je ve skutečnosti 17místné celé číslo, například 14660000011841514. Tento identifikátor je unikátní v rámci všech obchodních dimenzí a nemůže se tedy stát, že by například BRP a DSO sdíleli stejný identifikátor.

Časové řady obchodních dimenzí mají vždy jeden sloupec BD, který udává identifikátor dané dimenze. U hodnotových a MECových časových řad může být sloupců BD hned několik, obvykle se ale počet pohybuje okolo dvou až pěti. U těchto řad se jedná o cizí klíč do časové řady dané obchodní dimenze.

V názvu sloupce BD je postfix s identifikátorem, který je unikátní pro časovou řadu, ale již ne pro celý model. Například tedy pokud budeme mít dvě hodnotové časové řady, které obsahují obchodní dimenzi RE, tak v první časové řadě může být pro dimenzi RE definován sloupec BD_14660000000000001 a v druhé řadě může být sloupec pojmenován jako BD_14660000000000002.

TS_VALUE

Hodnota časové řady, kde je datový typ závislý na charakteru uložené hodnoty. U hodnotových časových řad se především jedná o NUMBER(20, 8), pro uložení objemu elektrické energie nebo peněžní částky. V časových řadách obchodních dimenzí může hodnota představovat název dimenze, pro Norské MBA tedy například 'NO4'.

TS_VALUE v časových řadách MECů obsahuje vždy MEC ID, což je unikátní textový identifikátor MECu v systému. Obsahem sloupce TS_VALUE může být ale i hodnota sloupce BD z jiné časové řady, tedy identifikátor cizího klíče.

Časová řada může být i vícehodnotová, tj. v tabulce se nachází více TS_VALUE sloupců. Pro každý sloupec TS_VALUE pak existují vlastní doprovodné sloupce COMP_ERR, CORR_VALUE a T406VALSTATE se specifickým identifikátorem.

COMP_ERR

Příznak udávající, zda při výpočtu hodnoty došlo k chybě. Datový typ je definován jako NUMBER(1). Sloupec může nabývat pouze hodnot 1 (pravda) a 0 (nepravda).

CORR_VALUE

Korigovaná hodnota v případě výskytu chyby při výpočtu. Datový typ je totožný s typem sloupce TS_VALUE. Pokud při výpočtu nedošlo k chybě, tak je hodnota sloupce CORR_VALUE ponechána prázdná (NULL).

T406VALSTATE

Příznak udávající status hodnoty TS_VALUE. Systém totiž umožňuje definování různých stavů hodnoty časové řady. Datový typ je CHAR(1) a sloupec může kromě defaultní hodnoty NULL obsahovat pouze následující znaky, které udávají stavy:

- **P** (precreated) – hodnota TS_VALUE byla předvytvořena, ale je zatím prázdná
- **F** (final) – hodnota TS_VALUE je finální a nebude již upravována
- **D** (deleted) – hodnota TS_VALUE byla měkce smazána

Udržování informace o statusu hodnoty je nutné pro business logiku systému, to platí jak pro primární, tak odvozené časové řady. Při dotazování na data je nutné odfiltrování měkce smazaných hodnot, jelikož nesmí být použité ve výpočtech settlementu a ani by neměly být zobrazované účastníkům trhu.

Všechny sloupce cizích klíčů, tj. T103TIME_UNIT_VALUE, BD[9] a v některých případech i TS_VALUE, sice obsahují ukazatele z jiných tabulek, ale nejsou zde pro

[9] V případě hodnotové a MECové časové řady.

ně definované žádné integritní omezení (constraints). Tento postup je zvolen za účelem dosažení rychlejšího zpracování DML operací. O integritu cizích klíčů v časových řadách se tedy nestará databáze, ale externí aplikace.

4.4.5 Denormalizace

Kromě neexistujících integritních omezení na cizích klíčích časových řad, model využívá i dalších denormalizačních technik. Sloupce FROM_TIME a TO_TIME jsou v časových řadách redundantní, jelikož údaje o validitě hodnoty jsou k dispozici v kalendářní tabulce prostřednictvím sloupce T103TIME_UNIT_VALUE. Původní návrh datového modelu sloupce FROM_TIME a TO_TIME neobsahoval a počítal s nutným spojením kalendářové tabulky pro filtraci hodnot podle validity. V rané fázi se ale tento způsob ukázal jako velmi neefektivní a pro zrychlení dotazovaní byly přidány právě tyto dva redundantní sloupce.

U hodnotových časových řad představují redundantní sloupce jednotlivé BD. Podle ideálního návrhu relační databáze by zde měl být pouze jeden sloupec, který by obsahoval cizí klíč daného MECu, jemuž hodnota náleží. Tato změna by sice představovala velkou úsporu diskového místa, ale v případě filtrace skrze jednotlivé obchodní dimenze by přinesla horší výkon dotazování kvůli nutnému spojení časové řady MECu. Denormalizace skrze sloupce BD je v tomto případě oprávněná. Dotazování prostřednictvím jednotlivých dimenzí je používané především při výpočtu agregovaných hodnot v settlementu.

4.4.6 Partitioning

K rozdělení tabulek časových řad do oddílů dochází pouze u hodnotových řad, ostatní řady rozdělené kvůli relativně malému počtu záznamů nejsou. Hodnotové tabulky všech časových granularit využívají rozsahové rozdělení na základě sloupce TO_TIME na jednotlivé kvartály, pro každý rok tedy existují čtyři oddíly. Každý oddíl je zařazen do odpovídajícího kvartálového tabulkového prostoru.

4.4.7 Indexy

Každá tabulka hodnotové časové řády má minimálně dva indexy. Prvním je kompozitní index primárního klíče, jež je složen ze sloupců OID a TO_TIME. Druhým

indexem je unikátní kombinace sloupce TO_TIME a všech BD řady. Indexy jsou zároveň použity jako integritní omezení za účelem zajištění unikátnosti. Pro každý MEC (kombinaci BD) tedy může v časové řadě existovat pouze jedna hodnota pro daný interval. Zajištění této unikátnosti je kritické pro integritu dat a proto je přenecháno na databázi. Oba základní indexy jsou lokální a využívají oddílů tabulky. Data indexů jsou ale zařazena do indexových tabulkových prostorů.

U časových řad obchodních dimenzí a MECů existují vždy pouze dva indexy. První index je primární klíč obsahující sloupec OID. Druhý index je kombinace TO_TIME, FROM_TIME a všech sloupců BD. Oba tyto indexy jsou globální a nerozdělené.

4.4.8 Komprese

Komprese je stejně jako rozdělení použita pouze u hodnotových časových řad. Jednotlivé oddíly využívají základní kompresi, výjimkou je oddíl nevyhovujících hodnot MAXVALUE, ten má kompresi vypnutou skrze klauzuli NOCOMPRESS. Indexy hodnotových řad využívají také základní kompresní mód.

4.4.9 Pohledy časových řad

Jmenná konvence nazvu časových řad a jejich sloupců je pro běžného uživatele velmi nečitelná, z tohoto důvodu se pro přístup k časovým řadám používají pohledy, které jsou interně nazývané access file (AF). Sloupce obchodních dimenzí a hodnot zde mají uživatelsky přívětivé názvy. V AF jsou zároveň vynechány i některé sloupce tabulek, které jsou pro uživatele nepodstatné. Na úrovni těchto pohledů dochází i k odfiltrování nechtěných hodnot, například měkce smazaných.

Pro každou časovou řadu existuje odpovídající AF, který je používaný při dotazování na hodnoty řady. Tabulky časových řad se v dotazech objevují přímo jen velmi zřídka. Příklady samotných AF budou uvedeny v následujících sekcích. Ukázky řádků AF a kalendářové tabulky jsou k dispozici v **příloze 3**.

4.4.10 Dotazování na časové řady

Při dotazování na hodnotové časové řady je vždy kladen důraz na použití indexovaných sloupců. Validita je tedy v drtivé většině případů filtrována skrze indexovaný sloupec TO_TIME. U obchodních dimenzí zpravidla dochází k filtraci

napříč všemi dimenzemi s přímou rovností. Indexy hodnotových časových řad se tedy jeví jako vhodně zvolené, jelikož obsahují všechny klíčové sloupce používané při dotazování.

4.5 Časová řada hodnot naměřené spotřeby

Tato primární časová řada byla zmiňovaná napříč celou prací a jelikož reprezentuje zdaleka nejvíce zatíženou řadu z pohledu přírůstku hodnot na den, tak je ideálním kandidátem na ukázkové představení a následnou optimalizaci. Aktuální hodinová verze časové řady v modelu vystupuje pod názvem TS_14500000002085720. SQL skript pro vytvoření tabulky a příslušných indexů je k dispozici v **příloze 5**.

4.5.1 Obchodní dimenze

Časová řada má čtyři obchodní dimenze, které jsou řazené sestupně dle kardinality.

MGA (BD_14500000002173483)

Identifikátor MGA, ve kterém došlo ke spotřebě.

RE (BD_14500000002173485)

Identifikátor RE, který spotřebu reportoval.

CONSUMPTION TYPE (BD_14500000002173487)

Identifikátor typu spotřeby, který specifikuje o jakou spotřebu přesně šlo.

MEASUREMENT TYPE (BD_14500000002173489)

Identifikátor typu měření spotřeby udávající způsob naměření hodnoty.

4.5.2 Hodnoty

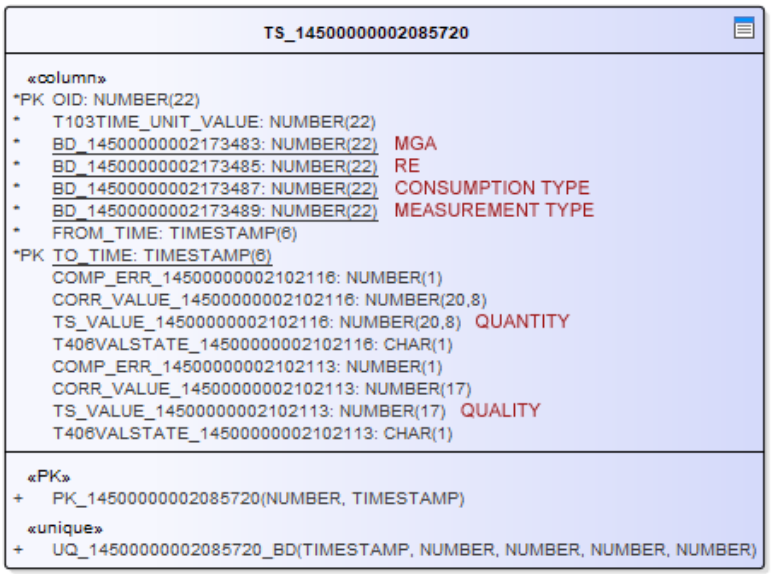
Časová řada má dvě hodnoty, pro každou z nich zde existují i doprovodné sloupce (COMP_ERR, CORR_VALUE a T406VALSTATE) popsané výše.

QUANTITY (TS_VALUE_14500000002102116)

Představuje kvantitu naměřené spotřeby uvedené v MWh za daný časový interval. Podle business logiky jde o číslo s délkou maximálně šesti desetinných míst, sloupec je ale definován jako NUMBER(22, 8). Příkladem uložené hodnoty může být číslo 42,12345678. Výchozí hodnota je NULL.

QUALITY (TS_VALUE_14500000002102113)

Reprezentuje „kvalitu“ uvedené hodnoty spotřeby, která může mít pouze tři stavy: reálně naměřená, odhadnutá, nebo dočasná. Jedná se o cizí klíč a hodnota je tudíž 17místný identifikátor z jiné řady. Datový typ je definovaný jako NUMBER(17).



Obrázek 11: Tabulka časové řady hodnot spotřeby (starý model)

4.5.3 Pohled

Pro přístup k hodinové verzi řady je použit pohled AF_VAL_CONS_RE. Pohledy pro ostatní verze časové řady pro jiné granulairity mají navíc postfix s časovou jednotkou. V případě denní verze má pohled název AF_VAL_CONS_RE_DAY.

```
CREATE OR REPLACE VIEW USRBUF.AF_VAL_CONS_RE AS
SELECT
    OID,
    FROM_TIME,
    TO_TIME,
    BD_14500000002173483 AS MGA,
    BD_14500000002173485 AS RE,
    BD_14500000002173487 AS CONSUMPTION_TYPE,
    BD_14500000002173489 AS MEASUREMENT_TYPE,
    TS_VALUE_14500000002102116 AS QUANTITY,
    T406VALSTATE_14500000002102116 AS QUANTITY_I,
    TS_VALUE_14500000002102113 AS QUALITY,
    T406VALSTATE_14500000002102113 AS QUALITY_I
FROM TS_14500000002085720
WHERE T406VALSTATE_14500000002102116 != 'D'
    OR T406VALSTATE_14500000002102116 IS NULL;
```

Kód 21: Definice pohledu AF_VAL_CONS_RE (starý model)

4.6 Časová řada obchodní dimenze MGA

Pro představení časových řad obchodních dimenzí jsem zvolil řadu MGA, jelikož právě dimenze MGA je na prvním místě v hodnotové časové řadě spotřeby. Časové řady ostatních obchodních dimenzí mají obdobnou podobu.

4.6.1 Obchodní dimenze

Protože se jedná o časovou řadu obchodní dimenze, tak se zde nachází pouze jediný sloupec BD, a to unikátní 17místný identifikátor MGA, který je použit ve všech ostatních časových řadách jako cizí klíč pro specifikaci MGA. Tento sloupec nese název BD_14500000002229243 a má datový typ NUMBER(22).

4.6.2 Hodnoty

Časové řady obchodních dimenzí mají velké množství hodnotových sloupců, které pro účely práce nejsou podstatné, v případě MGA se jich v řadě nachází celkem 12. Pro zachování přehlednosti a vytvoření představy postačí tyto 3.

NAME (TS_VALUE_14500000002116332)

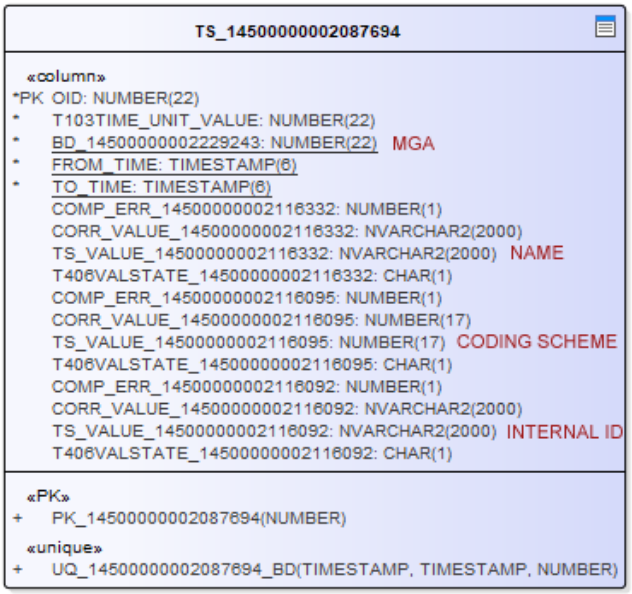
Název MGA, tedy textový řetězec s maximální délkou 2 tisíc znaků. Průměrná délka názvů MGA je však 15 znaků, u ostatních dimenzí může být průměrná délka názvu i okolo 25 znaků. Název se používá především jako uživatelsky přívětivý popisek při zobrazování dat.

CODING SCHEME (TS_VALUE_14500000002116095)

Identifikátor schématu kódování, které je používané pro komunikaci (reportování skrze datové toky) v rámci MGA. Je to tedy cizí klíč z jiné časové řady a datový typ je NUMBER(17).

INTERNAL ID (TS_VALUE_14500000002116092)

Interní identifikátor MGA. Jedná se textový řetězec, jež je složený z interního identifikátoru kódování a názvu MGA, například tedy EIC_MGA1. Datový typ je definován jako textový řetězec s maximální délkou 2 tisíc znaků.



Obrázek 12: Tabulka časové řady obchodní dimenze MGA (starý model)

4.6.3 Pohled

K dimenzím MGA se přistupuje skrze pohled AF_MGA, jež je definován následovně.

```
CREATE OR REPLACE VIEW USRBUF.AF_MGA AS
SELECT
    OID,
    FROM_TIME,
    TO_TIME,
    BD_14500000002229243 AS MGA,
    TS_VALUE_14500000002116332 AS NAME,
    T406VALSTATE_14500000002116332 AS NAME_I,
    TS_VALUE_14500000002116095 AS CODING_SCHEME,
    T406VALSTATE_14500000002116095 AS CODING_SCHEME_I,
    TS_VALUE_14500000002116092 AS INTERNAL_ID,
    T406VALSTATE_14500000002116092 AS INTERNAL_ID_I
FROM TS_14500000002087694
WHERE T406VALSTATE_14500000002116092 != 'D'
    OR T406VALSTATE_14500000002116092 IS NULL;
```

Kód 22: Definice pohledu AF_MGA (starý model)

4.7 Časová řada MECu spotřeby

Všechny možné kombinace obchodních dimenzí z hodnotové časové řady spotřeby jsou uvedené v této MECové řadě. Aktuálně je těchto unikátních kombinací pro spotřebu vedených v systému téměř 20 tisíc.

4.7.1 Obchodní dimenze

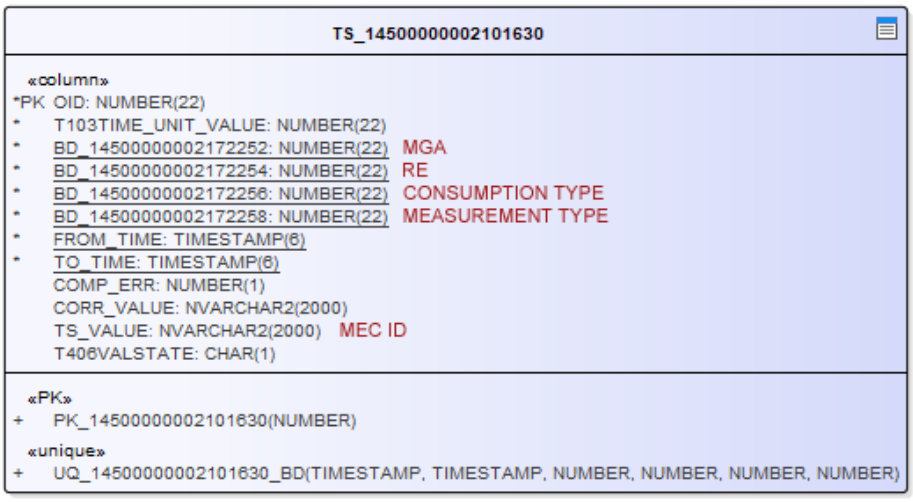
Složení obchodních dimenzí MECové řady je totožné s příslušnou hodnotovou časovou řadou. Seznam všech sloupců dimenzí MECu spotřeby je tedy uvedený v kapitole: **4.5.1 Obchodní dimenze**.

4.7.2 Hodnoty

Časová řada má pouze jedinou hodnotu a tou je MEC ID.

MEC ID (TS_VALUE)

Jak již bylo uvedeno, tak MEC ID je unikátní textový identifikátor. V případě spotřeby se jedná o spojení textového řetězce CONS a celého čísla, například 'CONS42'. MEC ID je automaticky generováno externí aplikací, bez možnosti uživatelského zásahu. Datový typ je definovaný jako NVARCHAR(2000).



Obrázek 13: Tabulka časové řady MECu spotřeby (starý model)

4.7.3 Pohled

Pro přístup k řadě se používá pohled AF_MEC_CONS_RE, který má tuto definici.

```
CREATE OR REPLACE VIEW USRBUF.AF_MEC_CONS_RE AS
SELECT
    OID,
    FROM_TIME,
    TO_TIME,
    BD_14500000002172252 AS MGA,
    BD_14500000002172254 AS RE,
    BD_14500000002172256 AS CONSUMPTION_TYPE,
    BD_14500000002172258 AS MEASUREMENT_TYPE,
    TS_VALUE AS MEC_ID,
    T406VALSTATE AS MEC_ID_I
FROM TS_14500000002101630
WHERE T406VALSTATE != 'D' OR T406VALSTATE IS NULL;
```

Kód 23: Definice pohledu AF_MEC_CONS_RE (starý model)

4.8 Kalendářová tabulka

Již zmíněná kalendářová tabulka s názvem T103TIME_UNIT_VALUE obsahuje všechny možné intervaly pro všechny používané časové granularity a existující obchodní dimenze v systému. Hodnoty kalendáře jsou generovány externí aplikací a aktuálně tabulka obsahuje několik miliónů řádků, jelikož se zde nacházejí intervaly od počátku roku 2015, až do konce roku 2026. Tabulka není přizpůsobena ke spojování s časovými řadami, jelikož nevyužívá ani vertikálního rozdělení na oddíly. Není zde zapnuta ani základní metoda komprese.

4.8.1 Sloupce

Kalendářová tabulka není časovou řadou a používá odlišnou jmennou konvenci.

OID

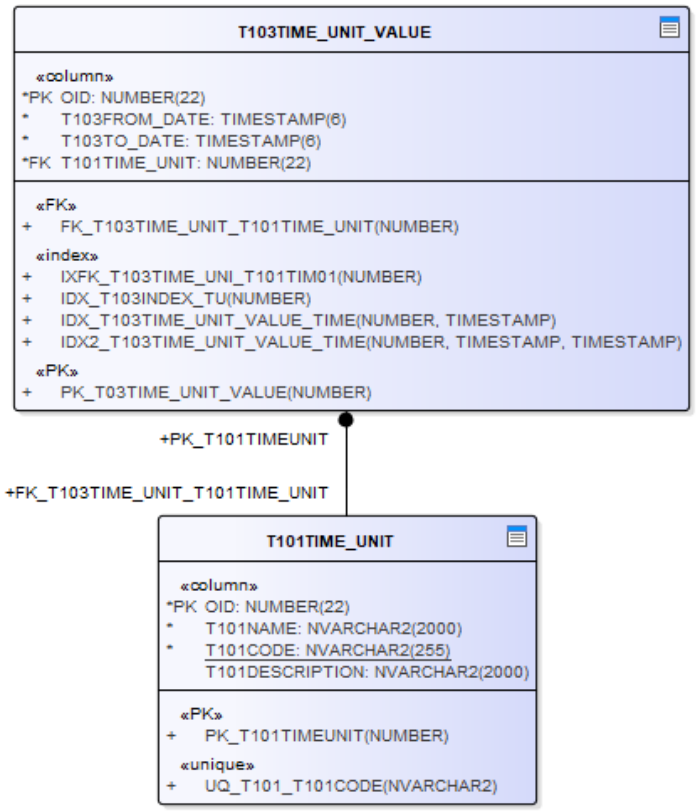
Stejně jako u tabulek časových řad se jedná o unikátní identifikátor řádku, pro který je použitý datový typ NUMBER(22).

T103FROM_DATE a T103TO_DATE

Obdoba sloupců FROM_TIME a TO_TIME z tabulky časové řady.

T101TIME_UNIT

Cizí klíč identifikátoru časové jednotky (granularity) z tabulky T101TIME_UNIT, která obsahuje doplňující informace o jednotce. Na tomto sloupci je definováno integritní omezení cizího klíče FK_T103TIME_UNIT_T101TIME_UNIT.



Obrázek 14: Tabulky kalendáře

4.8.2 Indexy

Kalendářová tabulka má celkem čtyři globální B-tree indexy. Prvním je primární klíč sloupce OID. Ostatní indexy jsou neunikátní a jedná se o postupné kombinace všech ostatních sloupců.

4.8.3 Využití

Jak již bylo řečeno, tak původním záměrem kalendářově tabulky byla filtrace dle validity skrze spojení s hodnotovými časovými řadami. Aktuálně má však tabulka velice minoritní využití a při dotazování na časové řady není používána. V časových řadách tedy spíše zabírá jen zbytečně místo.

4.9 Nedostatky modelu

Na základě rozboru lze určit nedostatky stávajícího datového modelu, které by měly být v novém modelu adresovány a patřičně napraveny. Některé další nedostatky a potencionální zlepšení byly již zmíněné v předchozím textu.

4.9.1 Čitelnost

Ačkoliv čitelnost nemá nic společného s optimalizací modelu z pohledu datové náročnosti ani výkonu dotazování, tak bych chtěl tento problém v novém modelu adresovat. Jak je z příkladů zřejmé, tak časové řady jsou bez AF velmi uživatelsky nepřívětivé. Jmenná konvence skrze unikátní identifikátory je sice obecná, ale pro uživatele ve výsledku nepřehledná. Další problém zde představuje fakt, že názvy tabulek časových řad a jejich sloupců můžou být na různých prostředích odlišné. Identifikátory jsou totiž při vytvoření tabulky generované externí aplikací a bez možnosti uživatelského vstupu.

4.9.2 Obecnost

Časové řady se sice logicky dělí na primární a odvozené, ale fyzicky mezi nimi rozdíl není. Obecný návrh řad je vhodný pro znovupoužitelnost, ale ne pro dosažení optimálních výsledků.

Nadbytečné sloupce

Primární a odvozené řady mají totožné sloupce i přestože to není nutné. U primárních řad sloupce COMP_ERR a CORR_VALUE nejsou nikdy využité, jelikož zde hodnoty nevznikají na základě výpočtů settlementu, ale pouze zápisem nebo případnou modifikací. Ve sloupci COMP_ERR jsou tak vždy uložené 0 (nepravda). Celočíselná hodnota 0, podle **tabulky 2** zabírá na disku 1 bajt, pokud tedy budeme mít primární časovou řadu s miliardou řádků, kde jsou navíc dva hodnotové sloupce, tak pouze sloupce COMP_ERR zaberou zbytečně 2 GB[10] diskové kapacity. To samé platí pro sloupce CORR_VALUE, které sice mají hodnotu NULL, ale jelikož nejsou

[10] 1 000 000 000 řádků * 1 bajt * 2 sloupce / (1000 * 1000 * 1000 převod B na GB)

nikdy poslední v pořadí řádku, tak jejich prázdná hodnota musí být na disku zastoupena 1 bajtem (38).

Nadbytečný je i sloupec T103TIME_UNIT_VALUE kalendářově tabulky, o čemž vypovídá i fakt, že v AF není uvedený a tudíž není ani používaný při dotazování.

Partitioning

Ačkoliv hodnotové časové řady mají různé časové granularity, tak všechny využívají rozdělení na oddíly podle kvartálů. Pro denní a vyšší granularitu toto rozdělení dává smysl, jelikož systém uživateli umožňuje zobrazování stovek dní naráz a drobnější rozdělení by zde mohlo být spíše na škodu. U hodinových a nově i čtvrthodinových dat je však nastaven limit zobrazení maximálně jednoho měsíce. Při návrhu nového modelu je tedy nutné provést průzkum, zda by drobnější rozdělení u menších časových jednotek přineslo lepší výkon při dotazování.

Nepřesné datové typy

Obecnost modelu se dotýká i zvolených datových typů, které neodpovídají požadované business logice. Například pro uložení čísla s plovoucí desetinnou čárkou je vždy použit datový typ NUMBER(20, 8). V případě ukládání peněžní částky se jedná o přebytečnou přesnost, která potencionálně může zabírat zbytečné místo na disku. Business logika udává pro peněžní částky zaokrouhlení na dvě desetinná místa, pro uložení hodnoty je tak dostačující využít přesnost na 3 nebo 4 místa. Ideálně by uložená přesnost měla být na dvě desetinná místa, ale je zde nutná rezerva pro výpočet přesných agregovaných hodnot.

4.9.3 Cizí klíče

Všechny sloupce cizích klíčů v časových řadách používají 17místné unikátní identifikátory obchodních dimenzí. Některé hodnotové časové řady mohou mít takových sloupců i 5 a více.

Když si vezmeme jako příklad opět časovou řadu naměřené spotřeby, která má obchodní dimenzi MEASUREMENT TYPE. Tato dimenze má pouze 3 položky, tj. 3 řádky v tabulce časové řady obchodní dimenze MEASUREMENT TYPE. I přes tento fakt je ve sloupcích cizích klíčů v ostatních řadách použita hodnota ze sloupce BD,

tedy 17místné celé číslo. Pokud budeme brát v potaz, že do hodnotové časové řady naměřené spotřeby po přechodu na čtvrthodinová data přibyde denně 2 miliony nových řádků, tak jen pro uložení dimenze MEASUREMENT TYPE je nutné počítat s 20 MB[11] diskové kapacity. Reálně by však cizí klíč mohl využívat identifikátor, který není unikátní v rámci všech obchodních dimenzí, ale pouze v rámci časové řady obchodní dimenze, například sloupec OID. V případě dimenze MEASUREMENT TYPE sloupec OID nabývá pouze hodnot 1 až 3, tedy 1místné celé číslo. Kdyby byl pro cizí klíč využit sloupec OID, tak pro uložení 2 milionů nových řádků by bylo potřeba pouze 4MB[12] diskové kapacity, což by představovalo 5 násobné zlepšení.

Stejná logika je aplikovatelná na všechny ostatní obchodní dimenze. U objemnějších dimenzí je však nutné počítat již s 2-3 bajty na řádek, jelikož například MGA je v systému více než tisíc. Počet bajtů nutný pro uložení hodnoty lze v databázi Oracle získat pomocí vestavěné funkce VSIZE. Jednotlivé počty bajtů potřebné pro uložení celého čísla jsou viditelné v následující tabulce.

Tabulka 2: Bajty potřebné pro uložení celočíselného NUMBER

(bez komprese řádků)

Počet míst	0	1	2	3	4	5	6	…	17
Počet bajtů	1	2	2	3	3	4	4	…	10

Unikátnost a délka identifikátorů

Může se naskytovat otázka, proč vlastně obchodní dimenze používají identifikátor, který je unikátní v rámci celého modelu. Důvodem je to, že externí aplikace spravující databázi tuto unikátnost vyžaduje za účelem správy dimenzí, jelikož na pozadí existují různé závislosti, které zde nejsou zmíněné z rozsahových důvodů. Rozbor celého systému by sám o sobě dal minimálně na jednu knihu a v rámci této práce realizovatelný není. Důležité však je, že zmíněné omezení se netýká cizích

[11] 2 000 000 řádků * 10 bajtů / (1000 * 1000 převod B na MB)

[12] 2 000 000 řádků * 2 bajty / (1000 * 1000 převod B na MB)

klíčů používaných v časových řadách a použitý identifikátor zde může být prakticky jakýkoliv.

Druhou otázkou může být, proč má vlastně unikátní identifikátor dimenze 17míst, když takový počet obchodních dimenzí v systému ani není. Aktuální počet všech dimenzí je téměř 20 tisíc, tudíž by zde postačil identifikátor 5, nebo rovnou 6místný, jelikož obě varianty vyžadují 4 bajty diskové kapacity pro uložení. Důvodem je datová migrace v raném stádiu systému, kdy se ke všem již existujícím identifikátorům přičetlo 17místné číslo. To, že všechny doposud uvedené identifikátory začínají trojčíslím 145 a 146, tedy není náhoda ani výmysl, ale skutečný stav aktuálního modelu. Jednoznačnou odpověď na otázku, proč přičtené číslo mělo tento formát a zároveň bylo tak velké, se mi bohužel zjistit nepodařilo.

5 Nový datový model

Na základě rozboru stávajícího modelu a všech zjištěných nedostatků byl navržen nový datový model, jež bude představen v této kapitole.

5.1 Inicializační parametry

Pro nový model byl upraven pouze jediný inicializační parametr s názvem COMMON_USER_PREFIX. Tento parametr byl přidaný v novější verzi databáze Oracle a udává povinný prefix názvu schémat v PDB, defaultní hodnota obsahuje textový řetězec 'C##'. Hodnota parametru byla změněna na prázdný textový řetězec, což umožňuje definování schémat bez prefixu.

Všechny ostatní inicializační parametry databáze původního modelu zůstaly nezměněné. Většina parametrů používala výchozí hodnoty a v případě úpravy byl pro změnu pádný důvod, jež platí nadále i pro nový model.

5.2 Schéma

Schéma USRBUF bylo ponecháno v původním stavu a pro umožnění názvu bez prefixu byl upraven inicializační parametr předepisující jmennou konvenci. Vynucené přejmenování schématu by představovalo pro externí aplikaci a celý systém pouze zbytečnou překážku.

5.3 Tabulkové prostory

Původní rozdělení dat do tabulkových prostorů na kvartály je sice správné, ale v novém modelu mohou být data rozdělena i podle měsíců. Pro zohlednění této změny se v modelu nachází navíc měsíční prostory se stejnou jmennou konvencí. Například pro hodnoty měsíce ledna roku 2021 tedy nově existuje prostor pojmenovaný DAE_DATA_M_2021_1 a pro data indexu poté DAE_INDX_M_2021_1. Řady vyšších granularit (den a výše) využívají stávající kvartálové prostory.

Nový model počítá s přesunem datových souborů obsahující stará data na pomalejší/sekundární diskové skupiny. Jako stará data jsou považovaná data starší více než rok. Přesun datového souboru při použití ASM je možné provést například

pomocí nástroje Oracle Recovery Manager (RMAN). Celý postup přesunu datových souborů je velice snadný a dostupný v oficiální Oracle dokumentaci (59).

Velikost datového bloku ve všech prostorech byla ponechána na defaultních 8 KB, jelikož tato hodnota dosahuje dobrého poměru výkonu čtení a DML operací.

5.4 Tabulky časových řad

Nový model využívá nadále haldové tabulky, jelikož nabízí flexibilní využití a podporují všechny potřebné optimalizační techniky. Použití jiných typů tabulek zde ani možné není, protože pro tabulky časových řad je kritické rozdělení do oddílů, což většina ostatních typů tabulek nepodporuje.

5.4.1 Jmenná konvence

Čitelnost názvů tabulek časových řad byla ve starém modelu velmi omezená, k čemuž mohl potencionálně přispět i striktní limit názvů na 30 znaků ve starších verzích databáze Oracle. Od verze 12.2 je ale možné definovat názvy o délce 128 bajtů, tedy maximálně 128 znaků (60). Nové názvy časových řad jsou tedy více informativní a skládají se z těchto částí.

Každá tabulka časové řady začíná prefixem TSP nebo TSD. TSP indikuje primární (primary) časovou řadu a TSD odvozenou (derived). Prefix následuje jeden z těchto možných identifikátorů pro rozlišení druhu časové řady:

- **BD** – časová řada obchodní dimenze
- **MEC** – časová řada MECu
- **VAL** – časová řada hodnot

Další v pořadí je uživatelsky přívětivý název časové řady, například CONS_RE, jež indikuje řadu naměřené spotřeby energie na úrovni RE.

Jako poslední je identifikátor granularity, který se vyskytuje pouze u hodnotových časových řad. Tento identifikátor je inspirován normou ISO 8601 pro intervaly a může mít například tyto hodnoty:

- **PT15M** – čtvrthodinová časová řada

- **PT1H** – hodinová časová řada
- **P1D** – denní časová řada

Podle takto stanovené jmenné konvence je tedy název hodinové časové řady naměřené spotřeby nově jako TSP_VAL_CONS_RE_PT1H. Uživatel databáze tak pouze z názvu časové řady může získat všechny potřebné informace.

5.4.2 Sloupce časových řad

V časových řadách nepřibyl žádný nový sloupec, naopak některé byly odebrány. Sloupcové obsazení se nově řídí především typem časové řady.

OID

Sloupec OID je nezměněný. Nově je ale použit pro cizí klíče v časových řadách.

FROM_TIME

Sloupec FROM_TIME je nově vynechán v hodnotových časových řadách, které mají granularitu čtvrthodiny, hodiny a dne. Důvodem pro vynechání sloupce v těchto případech je značná úspora místa. Hodnotu FROM_TIME lze velice snadno a levně odvodit na úrovni AF ze sloupce TO_TIME. Databáze Oracle totiž nabízí literál INTERVAL, pomocí kterého lze od časové značky odečítat a přičítat libovolnou periodu (61). Například v případě čtvrthodinové granularity, tak pro získání FROM_TIME stačí od TO_TIME odečíst 15 minut. Virtualizace sloupce FROM_TIME nepřináší pro běžného uživatele databáze žádné znatelné změny, jelikož sloupec nebyl ve starém modelu indexovaný a tudíž nebyl ani použit při filtrování.

Tato modifikace by teoreticky mohla být použita také u větších granularit, například týdne a měsíce. V těchto řadách se ale nachází velmi málo hodnot a změna by zde nepřinesla téměř žádný benefit.

```
-- Výpočet FROM_TIME odečtením 15 minut od TO_TIME
SELECT TO_TIME - INTERVAL '15' MINUTE AS FROM_TIME, ...
```

Kód 24: Použití literálu INTERVAL pro získání FROM_TIME

TO_TIME

Sloupec je nezměněný a nachází se ve všech časových řadách. Sloupec stále hraje hlavní roli při filtrování hodnotových časových řad podle validity.

BD

Sloupec identifikátoru obchodní dimenze se nově nachází pouze v časových řadách obchodních dimenzí. Pro cizí klíče již není použit, ale jeho existence je nutná pro externí aplikaci. Hodnotou BD je nadále 17místný unikátní identifikátor v rámci všech obchodních dimenzí.

V hodnotových a MECových časových řadách byl sloupec nahrazen za OID s postfixem dané dimenze. Například tedy sloupec OID_MGA indikuje, že časová řada má obchodní dimenzi MGA, ve které se nachází OID z časové řady této dimenze.

VAL

Obdoba původního sloupce TS_VALUE, tj. sloupec hodnoty časové řady. Datový typ sloupce je sice závislý na uložených hodnotách, nově jsou zde ale zavedené striktnější přesnosti na základě business logiky. Například pro uložení peněžní částky je sloupec definován již jako NUMBER(20, 3). U sloupců s datovým typem NVARCHAR2 byl snížen maximální počet znaků podle potřeby.

VAL_E

Obdoba původního sloupce COMP_ERR. Nově se tento sloupec nachází pouze v odvozených časových řadách, jelikož v primárních řadách nemá žádné využití. Datový typ byl změněn na CHAR(1) a nově sloupec může nabývat pouze hodnot 'Y' (pravda) a 'N' (nepravda). Důvodem pro změnu typu je úspora místa na disku, protože CHAR(1) vyžaduje 1 bajt, zatímco NUMBER(1) potřebuje 2 bajty.

VAL_C

Obdoba původního sloupce CORR_VALUE. Stejně jako sloupec VAL_E se tento sloupec nově nachází pouze v odvozených časových řadách.

VAL_S

Obdoba původního sloupce T406VALSTATE. Datový typ byl ponechán jako CHAR(1), nově je ale tento sloupec povinný a jeho defaultní hodnota je 'P', tedy stav

precreated. Díky nastavení defaultní hodnoty je možné odstranění podmínky o NULL hodnotě stavového sloupce ze všech AF. Sloupec se stále nachází v primárních i odvozených řadách, jelikož stav hodnoty je nutné uchovávat pro oba typy řad.

5.4.3 Cizí klíče

Nový model využívá pro cizí klíče sloupce OID z časových řad obchodních dimenzí, namísto původního sloupce BD. OID je unikátní identifikátor řádku a tudíž i obchodní dimenze v rámci tabulky časové řady dané dimenze. Důvodem pro přechod byla nemalá úspora diskového místa v hodnotových časových řadách.

Na cizích klíčích ale nadále nejsou definované integritní omezení, jelikož by měly za následek drastické zpomalení DML operací nad modelem.

5.4.4 Sloupec kalendářní tabulky

Ze všech časových řad byl odebrán sloupec T103TIME_UNIT_VALUE. Využití kalendářové tabulky bylo pro časové řady minimální a sloupec tak zabíral akorát zbytečné místo na disku. Kalendářní tabulka zůstává zachována na pozadí, ale v rámci nového modelu již nebude zmiňována v souvislosti s časovými řadami.

5.4.5 Partitioning

Rozdělení hodnotových časových řad na oddíly je nově závislé na granularitě řady. Denní a vyšší granularity jsou stále rozdělené na kvartály, ale hodinové a nižší jsou nově rozdělené podle jednotlivých kalendářních měsíců. Pro rozdělení je v obou případech využit sloupec TO_TIME. Měsíční rozdělení je dostačující pro čtvrthodinová data, ale pro nižší granularity, například minuty, by bylo již vhodné definovat kompozitní rozdělení.

5.4.6 Indexy

Indexy hodnotových časových řad ve starém modelu jsou sice datově náročné, ale pro zajištění integrity dat jsou kritické. V novém modely jsou tedy zahrnuty bez jakékoliv změny, jelikož pořadí jejich sloupců je vhodně zvoleno na základě sestupné kardinality hodnot.

U starého modelu docházelo v hodnotových časových řadách k filtrování primárně přes unikátní index kombinace TO_TIME a všech BD. Díky unikátnosti má však tento index stejný počet záznamů jako samotná tabulka řádků. Při enormním počtu řádků tak index ztrácel na efektivitě. Jako řešení byl v hodnotových časových řadách definován bitmap index složený ze všech obchodních dimenzí, který využívá faktu, že počet unikátních kombinací obchodních dimenzí (MECů) je v rámci hodnotové řady poměrně malý. V případě časové řady naměřené spotřeby je to přibližně 20 tisíc možných kombinací. Při filtrování přes všechny obchodní dimenze, tak nově optimalizátor může zvolit efektivnější přístup skrze bitmap index. Výhodou bitmap indexu je i jeho velmi malá datová náročnost. Nevýhodu nového indexu může představovat teoretické zpomalení DML operací. Z tohoto důvodu je vhodné podrobit nový model důkladnému testování výkonu.

5.4.7 Komprese

Původní model již využíval základní kompresi pro tabulky a indexy hodnotových časových řad. Jelikož je pokročilá komprese finančně nákladnou záležitostí, se kterou by klient společnosti Unicorn nemusel souhlasit, tak nový model opět využívá pouze základní kompresní metody. Ve výsledném srovnání modelů ale budou pro zajímavost uvedené i získané metriky při teoretickém použití pokročilé komprese v režimu LOW. Režim HIGH by sice dosáhl lepších kompresních poměrů, ale za cenu zpomalení výkonu dotazování a větší náročnosti na čas procesoru.

5.4.8 Pohledy časových řad

Jmenná konvence AF byla ponechána, jelikož změna názvů by vyžadovala hromadnou úpravu všech dotazů v externích aplikacích. Pro uživatele databáze na úrovni AF nedošlo k žádné změně a všechny původní sloupce jsou nadále dostupné. Sloupec FROM_TIME se sice v některých řadách již nevyskytuje, ale na úrovní AF je virtualizován skrze dopočtení podle TO_TIME. Díky nastavení výchozí hodnoty stavového sloupce (VAL_S) byla z AF odebrána podmínka o jeho NULL hodnotě. Ukázky řádků AF jsou k dispozici v **příloze 4**.

5.5 Časová řada hodnot naměřené spotřeby

Tato primární časová řada byla použita pro představení starého modelu a tudíž je vhodné její použití i v tomto případě. Hodinová verze časové řady vystupuje podle nové jmenné konvence pod názvem TSP_VAL_CONS_RE_PT1H. SQL skript pro vytvoření tabulky a všech indexů je k dispozici v **příloze 6**. Skripty pro vytvoření celého modelu jsou dostupné v externí příloze.

5.5.1 Obchodní dimenze

Jak již bylo řečeno, tak pro cizí klíče dimenzí jsou nově použité hodnoty sloupce OID.

MGA (OID_MGA)

Identifikátor MGA, ve kterém došlo ke spotřebě.

RE (OID_RE)

Identifikátor RE, který spotřebu reportoval.

CONSUMPTION TYPE (OID_CONSUMPTION_TYPE)

Identifikátor typu spotřeby, který specifikuje o jakou spotřebu přesně šlo.

MEASUREMENT TYPE (OID_MEASUREMENT_TYPE)

Identifikátor typu měření spotřeby udávající způsob naměření.

5.5.2 Hodnoty

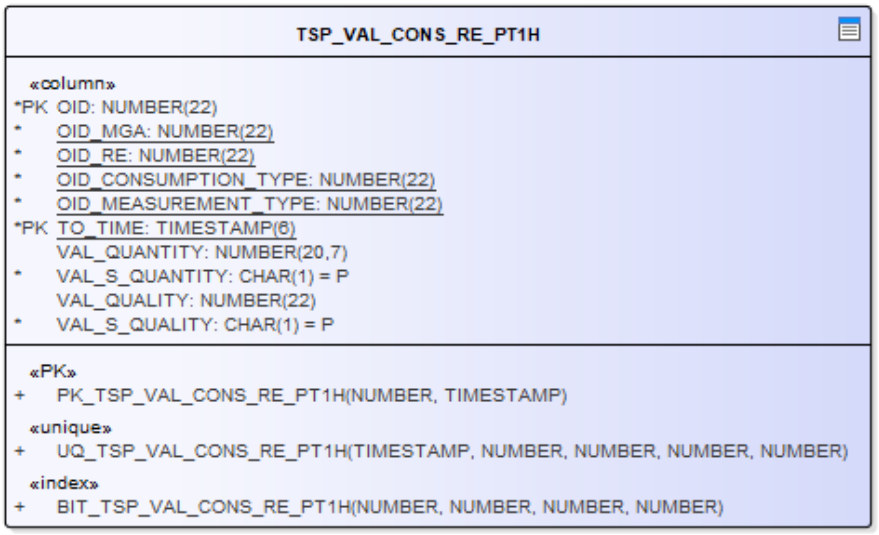
Časová řada má stejné hodnoty. V novém modelu došlo pouze k vynechání sloupců výpočetní chyby a korigované hodnoty.

QUANTITY (VAL_QUANTITY)

Obdoba původního sloupce TS_VALUE_14500000002102116. Nově je datový typ definován jako NUMBER(22, 7). Výchozí hodnota je ponechaná jako NULL.

QUALITY (VAL_QUALITY)

Obdoba původního sloupce TS_VALUE_14500000002102113. Datový typ byl pro konzistenci sloupců cizích klíčů upraven na NUMBER(22). Ačkoliv se nově ve sloupci mohou nacházet pouze hodnoty 1-3, tak definice sloupce s přesností 22 nezabírá v případě nevyužití kapacity žádné diskové místo navíc.



Obrázek 15: Tabulka časové řady hodnot spotřeby (nový model)

5.5.3 Pohled

K hodnotám řady se přistupuje skrze pohled s následující definicí.

```
CREATE OR REPLACE VIEW USRBUF.AF_VAL_CONS_RE AS
SELECT
    OID,
    TO_TIME - INTERVAL '1' HOUR AS FROM_TIME,
    TO_TIME,
    OID_MGA AS MGA,
    OID_RE AS RE,
    OID_CONSUMPTION_TYPE AS CONSUMPTION_TYPE,
    OID_MEASUREMENT_TYPE AS MEASUREMENT_TYPE,
    VAL_QUANTITY AS QUANTITY,
    VAL_S_QUANTITY AS QUANTITY_I,
    VAL_QUALITY AS QUALITY,
    VAL_S_QUALITY AS QUALITY_I
FROM TSP_VAL_CONS_RE_PT1H
WHERE VAL_S_QUANTITY != 'D';
```

Kód 25: Definice pohledu AF_VAL_CONS_RE (nový model)

5.6 Časová řada obchodní dimenze MGA

Hlavní změny v této časové řadě představují nové názvy sloupců dle nové jmenné konvence. Obsah obchodní dimenze a hodnotových sloupců však zůstal totožný.

5.6.1 Obchodní dimenze

V časové řadě byl ponechán sloupec BD pro zpětnou kompatibilitu. Pro cizí klíč v ostatních řadách se ale používá sloupec OID.

5.6.2 Hodnoty

Nová verze časové řady má stále stejné hodnoty.

NAME (VAL_NAME)

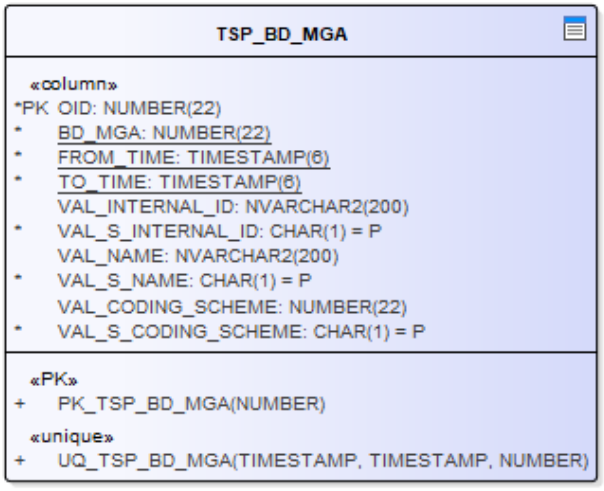
Obdoba sloupce TS_VALUE_14500000002116332. Uložený řetězec může mít nově maximální délku 200 znaků.

CODING SCHEME (VAL_CODING_SCHEME)

Obdoba sloupce TS_VALUE_14500000002116095. Datový typ byl upraven na NUMBER(22), pro konzistenci s ostatními sloupci cizích klíčů.

INTERNAL ID (VAL_INTERNAL_ID)

Obdoba sloupce TS_VALUE_14500000002116092. Jediná změna je v omezení řetězce na maximální délku 200 znaků.



Obrázek 16: Tabulka časové řady obchodní dimenze MGA (nový model)

5.6.3 Pohled

K dimenzím MGA se přistupuje skrze upravený pohled AF_MGA.

```
CREATE OR REPLACE VIEW USRBUF.AF_MGA AS
SELECT
    OID AS MGA,
    BD_MGA,
    FROM_TIME,
    TO_TIME,
    VAL_NAME AS NAME,
    VAL_S_NAME AS NAME_I,
    VAL_CODING_SCHEME AS CODING_SCHEME,
    VAL_S_CODING_SCHEME AS CODING_SCHEME_I
    VAL_INTERNAL_ID AS INTERNAL_ID,
    VAL_S_INTERNAL_ID AS INTERNAL_ID_I
FROM TSP_BD_MGA
WHERE VAL_S_INTERNAL_ID != 'D';
```

Kód 26: Definice pohledu AF_MGA (nový model)

5.7 Časová řada MECu spotřeby

Řada stále obsahuje všechny kombinace obchodních dimenzí tvořící MECy spotřeby a došlo zde pouze k přejmenování tabulky a sloupců.

5.7.1 Obchodní dimenze

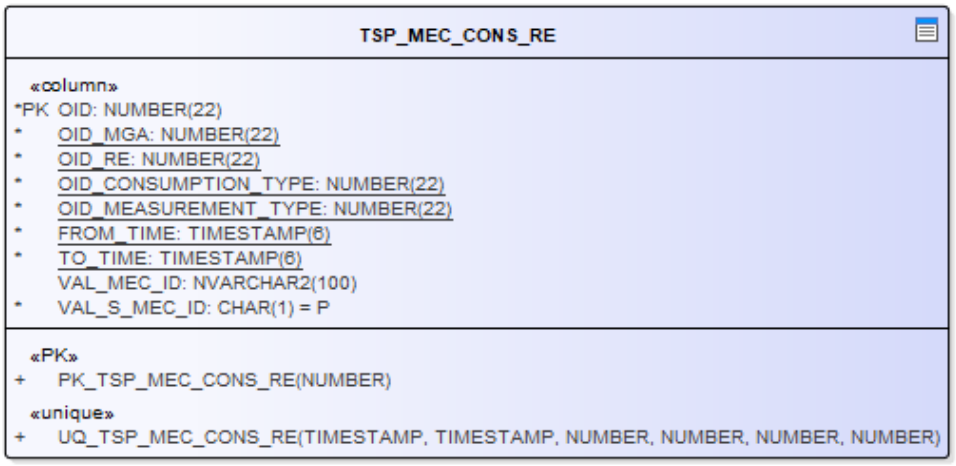
Složení obchodních dimenzí je totožné s hodnotovou časovou řadou spotřeby, jež je uvedené v sekci **5.5.1 Obchodní dimenze**.

5.7.2 Hodnoty

Řada má nadále jediný hodnotový sloupec.

MEC ID (VAL_MEC_ID)

Obdoba sloupce TS_VALUE původní řady. Datový typ je nově definován jako NVARCHAR2(100), jelikož všechny MEC ID v systému jsou poměrně krátké a původní limit 2 tisíc znaků byl zbytečný.



Obrázek 17: Tabulka časové řady MECu spotřeby (nový model)

5.7.3 Pohled

Hodnoty časové řady jsou k dispozici skrze upravený pohled AF_MEC_CONS_RE.

```
CREATE OR REPLACE VIEW USRBUF.AF_MEC_CONS_RE AS
SELECT
    OID,
    FROM_TIME,
    TO_TIME,
    OID_RE AS RE,
    OID_MGA AS MGA,
    OID_CONSUMPTION_TYPE AS CONSUMPTION_TYPE,
    OID_MEASUREMENT_TYPE AS MEASUREMENT_TYPE,
    VAL_MEC_ID AS MEC_ID,
    VAL_S_MEC_ID AS MEC_ID_I
FROM TSP_MEC_CONS_RE
WHERE VAL_S_MEC_ID != 'D';
```

Kód 27: Definice pohledu AF_MEC_CONS_RE (nový model)

6 Srovnání modelů

Pro srovnání modelů byla zvolena stará známá řada naměřené spotřeby na úrovni RE, která má největší denní přírůstek hodnot, tentokrát ale v nové čtvrthodinové verzi. Tato řada reprezentuje worst case ze všech časových řad v settlementu a její optimalizace je tedy nejvíce žádaná. Za účelem testování byla čtvrthodinová verze řady pro nový a starý model naplněna hodnotami za prvních 6 měsíců roku 2021. V každé řadě tak bylo uloženo téměř 350 miliónů řádků[13]. Důvodem pro naplnění „pouze" dvou kvartálů 2021 byla velká časová náročnost plnění a zjištění, že data mimo aktuálně dotazované oddíly nemají vliv na výkon dotazování. Pro docílení co možná nejlepších rozhodovacích schopností optimalizátoru byly před zahájením testování manuálně přepočítané kompletní statistiky datového schématu USRBUF.

Pro generování hodnot a plnění řad vznikl v rámci práce C# program, který velkou část procesu automatizuje. Zdrojový kód programu je k dispozici v externí příloze. Skrze program jsou nejdříve vygenerovány obchodní dimenze, ze kterých následně dojde k definovaní požadovaného počtu unikátních MECů spotřeby. Pro všechny definované MECy jsou poté podle konfiguračních parametrů vygenerované skripty na naplnění obou verzí testovaných řad. Naplněné časové řady nového a starého modelu ve finále obsahuji hodnoty pro stejné MECy a je tedy možné filtrovat na základě stejných obchodních dimenzí.

Testování a návrh modelu probíhal na cloudové platformě Microsoft Azure, konkrétně na VM s operačním systémem Windows Server 2019. Společnost Unicorn využívá pro hosting databáze operační systém Windows a tudíž jsem ho zvolil také. Použitá VM měla následující parametry:

- 2 jádrový procesor (E2s v3)
- 16 GB operační paměti
- 127 GB premium SSD (500 IOPS) pro operační systém a instalaci databáze
- 512 GB premium SSD (2300 IOPS) pro databázová data

[13] 181 dnů (6 měsíců) * 20 000 MECů * 96 čtvrthodin (1 den) = 347 520 000

Extra disk dedikovaný databázovým datům byl zvolen za účelem dosažení konzistentního I/O výkonu. Verze nainstalované databáze byla totožná s verzí, jež je používaná společností Unicorn, tedy 19c Enterprise Edition. Pro ladění a benchmark dotazů byl použit oficiální nástroj Oracle SQL Developer.

Porovnání starého a nového datového modelu je rozděleno do tří disciplín. První disciplínou je datová náročnost modelu, druhou je výkon dotazování a v poslední řadě výkon DML operací nad modelem. Všechny získané exekuční plány a metriky jsou k dispozici v externí příloze, jejíž obsah je popsán v **příloze 7**.

6.1 Datová náročnost

Přechod na reportování čtvrthodinových dat představuje ve starém modelu problém především z pohledu datové náročnosti hodnotových časových řad. Stávající databáze produkčního prostředí má aktuálně několik TB a po přechodu lze počítat s přibližně čtyřnásobnými nároky na diskovou kapacitu. Jelikož aktuální model využívá jako úložiště pouze SSD disky a data je nutné uchovávat minimálně po dobu 5 let, tak by se tento přechod velice prodražil. Při návrhu nového modelu tedy byl kladen důraz na co možná největší úsporu diskové kapacity.

Porovnání datové náročnosti starého a nového modelu proběhlo na úrovni databáze, skrze součet bajtů všech alokovaných segmentů pro jednotlivé tabulkové prostory. Používaný dotaz pro získání velikosti tabulkového prostoru je dostupný v **příloze 2**. Je nutné podotknout, že reálná velikost dat na disku se může nepatrně lišit od hodnoty získané z databáze, jelikož velikost datového bloku databáze a operačního systému se může lišit. Pro porovnání datové náročnosti byla hodnotová časová řada v obou modelech naplněna očekávaným denním přírůstkem v případě čtvrthodinových dat, tj. téměř 2 miliony hodnot[14]. Pro zajímavost byly vytvořené i varianty starého a nového modelu s použitím pokročilých kompresních metod. Získané velikosti rozpadnuté na tabulkové prostory dat a indexů a dále podle kompresních metod uvádí následující tabulka.

[14] 96 čtvrthodin (1 den) * 20 000 MECů = 1 920 000

Tabulka 3: Datová náročnost čtvrthodinové časové řady naměřené spotřeby

Uvedené velikosti jsou v MB na 1 den pro 20 tisíc MECů

	Základní komprese		Pokročilá komprese (LOW)	
	Starý model	Nový model	Starý model	Nový model
Hodnoty	215	96	80	64
B-tree indexy	262	176	174	116
Bitmap index	-	2	-	2
Celkem	477	274	254	182

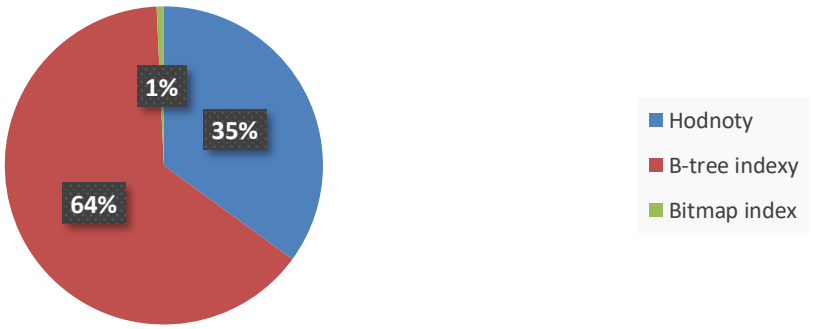
Základní komprese

Jak je z tabulky zřejmé, tak nový model vykazuje **43%** celkové snížení datové náročnosti. Pro očekávaný denní přírůstek čtvrthodinových dat v případě časové řady spotřeby tedy vyžaduje téměř polovinu diskové kapacity původního modelu.

Větší část úspory představují řádky tabulky, kde došlo ke snížení o **55%**. Za majoritní část této úspory může přechod na OID v cizích klíčích a virtualizace sloupce FROM_TIME. Průměrná délka řádku získaná ze statistiky tabulky je nově 47 bajtů, oproti původním 111 bajtům.

U indexů je snížení o **33%** již menší, ale také znatelné. Za velkým objemem B-tree indexů stojí velmi vysoká kardinalita kombinací. Nově zavedený bitmapový index zabírá navíc pouhé 2 MB diskové kapacity na den.

Graf 4: Rozložení datové náročnosti nového modelu



Pokročilá komprese

Použití pokročilé komprese pro komerční účely vyžaduje nákup licence, která je poměrně nákladná. Z tohoto důvodu je brána jako možný bonus a ve výstupním modelu není využita. Licenci je možné zakoupit na základě počtu uživatelů, nebo procesorů databáze. Ceník je k dispozici na oficiálním online obchodu Oracle[15].

Pouhé zavedení pokročilé komprese do stávajícího modelu by snížilo datovou náročnost o **47%**, což je lepší výsledek, než v případě nového modelu se základní kompresí. Implementace pokročilé komprese v novém modelu by poté oproti starému modelu se základní kompresí přineslo **62%** snížení datové náročnosti. Při průběžném testování bylo zjištěno, že zavedením pokročilé komprese by nedošlo k pozorovatelnému snížení výkonu dotazování. Metody pokročilé komprese tedy lze považovat za potencionální vhodné doplnění nového modelu.

Závěr

Nový model dosahuje výrazně menší datové náročnosti a přitom koncovému uživateli databáze nabízí stejná data a funkčnosti starého modelu. Použití pokročilé komprese by posunulo nový model ještě o stupeň výš. Pokud budeme brát v úvahu celkové velikosti uvedené v tabulce, tak pro uložení celého kalendářního roku 2021 by starý model vyžadoval celkem **174,1 GB** diskové kapacity. Nový model s pokročilou kompresí by pro uchování stejného objemu dat potřeboval jen **66,4 GB**.

Z pohledu datové náročnosti lze nový model považovat za úspěšný a záleží jakých výsledků dosáhne v dalších disciplínách.

6.2 Výkon dotazování

Dotazováním je zde myšleno získávání řádků tabulek pomocí SQL klauzule SELECT. Pro ověření výkonu dotazování byly zvoleny dva scénáře typického dotazování na hodnotové časové řady v systému. Dotaz každého scénáře byl spuštěn několikrát (minimálně 5x) a ze všech pokusů byl vybrán ten nejlepší. Důvodem pro několik pokusů bylo získání výsledků, které nejsou ovlivněné externími faktory, například

[15] https://shop.oracle.com/apex/f?p=dstore:2:0::NO:RIR:PROD_HIER_ID:4509953293451805720010

nečekaným vytížením procesoru VM. Před každým dotazem došlo ke kompletnímu promazání mezipaměti databáze skrze příkazy uvedené v **příloze 2**. Pro získání exekučních plánů a statistik byla použita primárně funkcionalita AUTOTRACE. Všechny získané exekuční plány a statistiky jsou k dispozici v externí příloze. K dotazování byly použité příslušné AF a pro rozlišení starého a nového modelu mají pouze prefix O (old) a N (new). Obchodní dimenze pro filtraci byly zvoleny náhodně.

6.2.1 Scénář 1

Jako první scénář byla zvolena běžná situace dotazu zobrazení hodnot specifického MECu v intervalu měsíce. Testováno je zde tedy načtení 31 dní z čtvrthodinových dat s filtrací přes všechny obchodní dimenze. Dotaz by tak měl vrátit přesně 2976 řádků[16]. Filtrace podle validity je schválně nastavena tak, aby databáze musela přistoupit k dvěma oddílům tabulky a jde tedy o worst case, jelikož načítání ze tří a více oddílů naráz není v případě nižších granularit uživateli umožněno skrze business logiku systému.

Pro dotazováni byl optimalizátor nastaven do módu ALL_ROWS, tj. prioritizace vrácení všech řádků dotazu. Protože ale testování probíhalo v nástroji Oracle SQL Developer, tak plán počítal jen s vrácením prvních 50 řádků výsledku.

Dotaz starého modelu

Dotaz na hodnoty starého modelu vypadal následovně:

```
SELECT FROM_TIME, TO_TIME, QUANTITY, QUALITY
FROM USRBUF.O_AF_VAL_CONS_RE
WHERE TO_TIME > TO_TIMESTAMP('2021-03-15', 'YYYY-MM-DD')
  AND TO_TIME <= TO_TIMESTAMP('2021-04-15', 'YYYY-MM-DD')
  AND MGA = 14660000853234864
  AND RE = 14660000103559374
  AND CONSUMPTION_TYPE = 14660000762936617
  AND MEASUREMENT_TYPE = 14660000853927168
ORDER BY TO_TIME;
```

Kód 28: Dotaz scénáře 1 (starý model)

[16] 31 dní * 96 čtvrthodin (1 den)

Exekuční plán a statistiky starého modelu

Pro zpracování dotazu se optimalizátor rozhodl využít následující exekuční plán:

```
-------------------------------------------------------------------------------------------------------------
| Id  | Operation                          | Name                 | Rows  | Cost (%CPU)| Pstart | Pstop |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                   |                      |    50 |  3891  (24)|        |       |
|   1 |  PARTITION RANGE ITERATOR          |                      |    50 |  3891  (24)|      1 |     2 |
|*  2 |   TABLE ACCESS BY LOCAL INDEX ROWID | TS_1450000000208572… |    50 |  3891  (24)|      1 |     2 |
|*  3 |    INDEX SKIP SCAN                 | UQ_1450000000208572… |     2 |  3888  (24)|      1 |     2 |
-------------------------------------------------------------------------------------------------------------
2 - filter("T406VALSTATE_14500000002102116"<>'D' OR "T406VALSTATE_14500000002102116" IS NULL)
3 - access("TO_TIME">TIMESTAMP'2021-03-15 00:00:00' AND "TO_TIME"<=TIMESTAMP'2021-04-15 00:00:00'
          AND "BD_14500000002173483"=14660000853234864 AND "BD_14500000002173485"=14660000103559374
          AND "BD_14500000002173487"=14660000762936617 AND "BD_14500000002173489"=14660000853927168)
    filter("BD_14500000002173483"=14660000853234864 AND "BD_14500000002173485"=14660000103559374
          AND "BD_14500000002173487"=14660000762936617 AND "BD_14500000002173489"=14660000853927168)

-- Statistiky nejlepšího pokusu
consistent gets: 2741
DB time: 87
physical reads: 618
physical reads bytes: 5 062 656
recursive calls: 1874
```

Kód 29: Exekuční plán scénáře 1 (starý model)

Optimalizátor v exekučním plánu využil jediný vhodný index. Konkrétně unikátní index složený ze sloupce TO_TIME a všech BD. Skip scan indexu zde představuje nejvíce efektivní přístupovou cestu pro danou situaci. Výhodou použití unikátního indexu je přeskočení řazení podle sloupe TO_TIME, jelikož získané hodnoty podle indexu jsou již seřazené. Optimalizátor také správně rozhodl o přístupu pouze k dvěma kvartálním oddílům tabulky.

Dotaz nového modelu

Dotaz na hodnoty nového modelu vypadal následovně:

```sql
SELECT FROM_TIME, TO_TIME, QUANTITY, QUALITY
FROM USRBUF.N_AF_VAL_CONS_RE
WHERE TO_TIME > TO_TIMESTAMP('2021-03-15', 'YYYY-MM-DD')
  AND TO_TIME <= TO_TIMESTAMP('2021-04-15', 'YYYY-MM-DD')
  AND MGA = 995
  AND RE = 10
  AND CONSUMPTION_TYPE = 2
  AND MEASUREMENT_TYPE = 2
ORDER BY TO_TIME;
```

Kód 30: Dotaz scénáře 1 (nový model)

Exekuční plán a statistiky nového modelu

Pro zpracování dotazu se optimalizátor rozhodl využít následující exekuční plán:

```
-----------------------------------------------------------------------------------------------------
| Id | Operation                                  | Name          | Rows | Cost (%CPU)| Pstart | Pstop |
-----------------------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT                           |               |   40 |     6  (17)|        |       |
|  1 |  PARTITION RANGE ITERATOR                  |               |   40 |     6  (17)|      3 |     4 |
|  2 |   SORT ORDER BY                            |               |   40 |     6  (17)|        |       |
|* 3 |    TABLE ACCESS BY LOCAL INDEX ROWID BATCHED | TSP_VAL_CONS… |   40 |     5   (0)|      3 |     4 |
|  4 |     BITMAP CONVERSION TO ROWIDS            |               |      |            |        |       |
|* 5 |      BITMAP INDEX SINGLE VALUE             | BIT_TSP_VAL…  |      |            |      3 |     4 |
-----------------------------------------------------------------------------------------------------
3 - filter("TO_TIME">TIMESTAMP'2021-03-15 00:00:00' AND "TO_TIME"<=TIMESTAMP'2021-04-15 00:00:00'
          AND "VAL_S_QUANTITY"<>'D')
5 - access("OID_MGA"=995 AND "OID_RE"=10 AND "OID_CONSUMPTION_TYPE"=2 AND "OID_MEASUREMENT_TYPE"=2)

-- Statistiky nejlepšího pokusu
consistent gets: 3237
DB time: 62
physical reads: 417
physical reads bytes: 3 416 064
recursive calls: 2825
```

Kód 31: Exekuční plán scénáře 1 (nový model)

Optimalizátor se rozhodl dát přednost nově vytvořenému bitmapovému indexu pro efektivní filtraci obchodních dimenzí, která je velmi levná a vyžadovala až neuvěřitelných 0% procesoru. Na rozdíl od předchozího exekučního plánu, ale bylo nutné výsledek manuálně řadit podle sloupce TO_TIME, což zde představuje nejvíce náročnou operaci, jež vyžadovala 17% procesoru. Optimalizátor se dále správně rozhodl přistoupit pouze k potřebným měsíčním oddílům tabulky.

Čas dotazu

Ačkoliv exekuční plán počítá s vrácením pouze prvních 50 řádků, tak SQL Developer nabízí vestavěnou funkci na postupné vrácení všech řádků dotazu, která ukáže i výsledný celkový čas. Tato funkcionalita je dostupná skrze klávesovou zkratku F5.

Získání všech 2976 řádků trvalo u starého modelu v průměru **23,248 sekund**. U nového modelu stejná situace trvala v průměru **1,379 sekund**, což představuje téměř 17násobné zlepšení. Při zapnutí paralelizace dotazů bylo v novém modelu dosaženo nejlepšího času 1,089 sekund, je ale nutné dodat, že získané časy s paralelizací byly velmi nekonzistentní.

Závěr

Exekuční plán nového modelu je oproti starému daleko více efektivní ve filtraci na základě obchodních dimenzí. O této efektivitě vypovídá i menší počet nutných fyzických čtení z disku a celkový čas databáze. Výkon dotazování dle prvního scénáře je tedy lepší v nově navrženém modelu, především pak při kontinuálním získávání všech řádků dotazu.

6.2.2 Scénář 2

Jako druhý scénář byla zvolena další běžná situace výpočtu týdenní odvozené agregované hodnoty. Cílem testu je výpočet týdenní sumy ze čtvrthodinových dat, skrze filtraci přes 3 ze 4 obchodních dimenzí, tj. výpočet sumy specifického spotřebního typu pro celé MGA. Dotaz by tak měl vrátit pouze 1 řádek. Filtrace podle validity je opět schválně nastavena napříč dvěma oddíly tabulky.

Dotaz starého modelu

Dotaz na hodnoty starého modelu vypadá následovně:

```
SELECT MIN(FROM_TIME), MAX(TO_TIME), SUM(QUANTITY)
FROM USRBUF.O_AF_VAL_CONS_RE
WHERE TO_TIME > TO_TIMESTAMP('2021-03-28', 'YYYY-MM-DD')
  AND TO_TIME <= TO_TIMESTAMP('2021-04-04', 'YYYY-MM-DD')
  AND MGA = 14660000853234864
  AND CONSUMPTION_TYPE = 14660000762936617
  AND MEASUREMENT_TYPE = 14660000853927168;
```

Kód 32: Dotaz scénáře 2 (starý model)

Exekuční plán a statistiky starého modelu

Pro zpracování dotazu se optimalizátor rozhodl využít následující exekuční plán:

```
-------------------------------------------------------------------------------------------------------
| Id | Operation                                  | Name       | Rows  | Cost (%CPU)| Pstart | Pstop |
-------------------------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT                           |            |     1 |  1191  (18)|        |       |
|  1 |  SORT AGGREGATE                            |            |     1 |            |        |       |
|  2 |   PARTITION RANGE ITERATOR                 |            |   489 |  1191  (18)|      1 |     2 |
|* 3 |    TABLE ACCESS BY LOCAL INDEX ROWID BATCHED| TS_145000… |   489 |  1191  (18)|      1 |     2 |
|* 4 |     INDEX SKIP SCAN                        | UQ_145000… |   489 |   700  (30)|      1 |     2 |
-------------------------------------------------------------------------------------------------------
3 - filter("T406VALSTATE_14500000002102116"<>'D' OR "T406VALSTATE_14500000002102116" IS NULL)
4 - access("TO_TIME">TIMESTAMP'2021-03-28 00:00:00' AND "TO_TIME"<=TIMESTAMP'2021-04-04 00:00:00'
          AND "BD_14500000002173483"=14660000853234864 AND "BD_14500000002173487"=14660000762936617
          AND "BD_14500000002173489"=14660000853927168)
    filter("BD_14500000002173483"=14660000853234864 AND "BD_14500000002173487"=14660000762936617
          AND "BD_14500000002173489"=14660000853927168)

-- Statistiky nejlepšího pokusu
consistent gets: 5860
DB time: 558
physical reads: 4476
physical reads bytes: 36 667 392
recursive calls: 1251
```

Kód 33: Exekuční plán scénáře 2 (starý model)

Optimalizátor opět využil index unikátní kombinace v režimu skip scan, který opětovně vyžaduje poměrně velké množství procesorového času. Ani v této situaci optimalizátor nezklamal a využil správně kvartálové oddíly tabulky.

Dotaz nového modelu

Dotaz na hodnoty nového modelu vypadá následovně:

```
SELECT MIN(FROM_TIME), MAX(TO_TIME), SUM(QUANTITY)
FROM USRBUF.N_AF_VAL_CONS_RE
WHERE TO_TIME > TO_TIMESTAMP('2021-03-28', 'YYYY-MM-DD')
  AND TO_TIME <= TO_TIMESTAMP('2021-04-04', 'YYYY-MM-DD')
  AND MGA = 995
  AND CONSUMPTION_TYPE = 2
  AND MEASUREMENT_TYPE = 2;
```

Kód 34: Dotaz scénáře 2 (nový model)

Exekuční plán a statistiky nového modelu

Pro zpracování dotazu se optimalizátor rozhodl využít následující exekuční plán:

```
-----------------------------------------------------------------------------------------------------
| Id | Operation                                | Name           | Rows  | Cost (%CPU)| Pstart| Pstop |
-----------------------------------------------------------------------------------------------------
|  0 | SELECT STATEMENT                         |                |     1 |   891   (1)|       |       |
|  1 |  SORT AGGREGATE                          |                |     1 |            |       |       |
|  2 |   PARTITION RANGE ITERATOR               |                |   489 |   891   (1)|     3 |     4 |
|* 3 |    TABLE ACCESS BY LOCAL INDEX ROWID BATCHED| TSP_VAL_CONS… |   489 |   891   (1)|     3 |     4 |
|  4 |     BITMAP CONVERSION TO ROWIDS          |                |       |            |       |       |
|* 5 |      BITMAP INDEX SINGLE VALUE           | BIT_TSP_VAL…   |       |            |     3 |     4 |
-----------------------------------------------------------------------------------------------------
3 - filter("TO_TIME">TIMESTAMP'2021-03-28 00:00:00' AND "TO_TIME"<=TIMESTAMP'2021-04-04 00:00:00'
          AND "VAL_S_QUANTITY"<>'D')
5 - access("OID_MGA"=995 AND "OID_CONSUMPTION_TYPE"=2 AND "OID_MEASUREMENT_TYPE"=2)

-- Statistiky nejlepšího pokusu
consistent gets: 2108
DB time: 107
physical reads: 1489
physical reads bytes: 12 197 888
recursive calls: 1979
```

Kód 35: Exekuční plán scénáře 2 (nový model)

Stejně jako v prvním scénáři nového modelu se optimalizátor rozhodl pro použití bitmapového indexu. Oproti starému modelu zde došlo k minimálnímu vytížení procesoru a méně přístupům k disku.

Čas dotazu

Získání celkové sumy u starého modelu trvalo v průměru **4,486 sekund**. Tato stejná situace u nového modelu trvala v průměru **1,321 sekundy**, a jde tedy o více než 3násobné zlepšení výkonu dotazu. Paralelizace umožnila tento čas snížit až na 0,711 sekund. Oproti předchozímu scénáři paralelizace dosahovala konzistentních časů.

Závěr

Exekuční plán nového modelu je i pro druhý scénář efektivnější. Nejenže zde dochází k daleko menšímu vytížení procesoru, ale počet fyzických přístupů k disku a celkový čas databáze je také mnohonásobně nižší.

Z pohledu výkonu dotazování je možné považovat nový model za úspěšný. Nejenže je šetrnější k hardwaru, ale zároveň dosahuje i lepších časů.

6.3 Výkon DML operací

Přestože výkon DML operací není pro systém až tak důležitý, jako výkon čtení, tak pro úplnost je vhodné provést test za účelem odhalení potencionálního zhoršení. Testované dotazy jsou zde uvedené pouze pro nový model, dotazy starého modelu byly krom použité tabulky a názvu sloupců totožné. Uvedené metriky přestavují průměr z 5 pokusů.

Tabulka 4: Časová náročnost DML operací

Uvedené metriky jsou v sekundách

	Starý model	Nový model
INSERT	142,212	126,142
UPDATE 1	15,192	8,476
UPDATE 2	24,205	1,363
DELETE	6,132	2,669

INSERT

Uvedená hodnota zde reprezentuje průměrný čas, který byl potřebný pro vložení denního přírůstku hodnotové časové řady, tedy téměř 2 milionů hodnot. V novém modelu došlo k drobnému zlepšení, především díky menšímu počtu sloupců. Vkládání řádků probíhalo skrze již zmiňovaný program, který vznikl za účelem generování a plnění dat.

UPDATE 1

První testovací scénář modifikace hodnot, kde došlo k nastavení pseudonáhodné kvantity pro 4 čtvrthodiny v rámci každého MECu. Celkem bylo naráz modifikováno 80 tisíc řádků. Ačkoliv se nejedná o klasickou modifikaci vyskytující se v systému, tak účelem bylo zjištění, zda obsáhlejší modifikace způsobí výkonnostní problémy. Ani zde nový model nevykazuje známky zpomalení, i přes použití bitmap indexu.

```
UPDATE USRBUF.TSP_VAL_CONS_RE_PT15M
SET VAL_QUANTITY = TRUNC(DBMS_RANDOM.VALUE(100, 999), 6)
WHERE TO_TIME > TO_TIMESTAMP('2021-01-01 00', 'YYYY-MM-DD HH24')
  AND TO_TIME <= TO_TIMESTAMP('2021-01-01 01', 'YYYY-MM-DD HH24');
```

Kód 36: Dotaz scénáře UPDATE 1

UPDATE 2

Druhý testovací scénář modifikace hodnot již více odpovídá běžné operaci v systému. V tomto scénáři došlo k modifikaci celého měsíce čtvrthodinových hodnot pro jeden specifický MEC, tedy celkem 2976 řádků. Rozdíl mezi starým a novým modelem je enormní. Důvodem je to, že pro modifikaci je nutné řádky nejdříve najít. K vyhledání řádků zde dochází skrze nový bitmap index, jež umožňuje efektivnější přístup a ve výsledku tedy i rychlejší modifikaci.

```
UPDATE USRBUF.TSP_VAL_CONS_RE_PT15M
SET VAL_QUANTITY = TRUNC(DBMS_RANDOM.VALUE(100, 999), 6)
WHERE TO_TIME > TO_TIMESTAMP('2021-01-01 00', 'YYYY-MM-DD HH24')
  AND TO_TIME <= TO_TIMESTAMP('2021-02-01 00', 'YYYY-MM-DD HH24')
  AND OID_MGA = 995
  AND OID_RE = 10
  AND OID_CONSUMPTION_TYPE = 2
  AND OID_MEASUREMENT_TYPE = 2;
```

Kód 37: Dotaz scénáře UPDATE 2

DELETE

K „tvrdému“ odstranění řádků hodnotové časové řady v systému téměř nedochází, pro zajímavost byl tento případ ale také zahrnut. Při odstranění platí stejný princip jako v případě scénáře UPDATE 2, tj. řádky je nejdříve nutné vyhledat. Dotaz pro DELETE v novém modelu opět využívá bitmapový index.

```
DELETE
FROM USRBUF.TSP_VAL_CONS_RE_PT15M
WHERE TO_TIME > TO_TIMESTAMP('2021-01-14 00', 'YYYY-MM-DD HH24')
  AND TO_TIME <= TO_TIMESTAMP('2021-01-21 00', 'YYYY-MM-DD HH24')
  AND OID_MGA = 995
  AND OID_RE = 10
  AND OID_CONSUMPTION_TYPE = 2
  AND OID_MEASUREMENT_TYPE = 2;
```

Kód 38: Dotaz scénáře DELETE

Závěr

Nový model dosahuje lepších výsledků ve všech stanovených disciplínách. Nejenže je model méně datově náročný, ale také vykazuje vyšší efektivitu při dotazování. Optimalizaci modelu tak lze považovat za úspěšnou a cíl práce za naplněný.

7 Shrnutí výsledků

Cílem práce byla optimalizace datového modelu časových řad v systému dodávaného společností Unicorn, za účelem dosažení nižší datové náročnosti a lepšího výkonu dotazování. Důvodem pro nutnou optimalizaci byl blížící se přechod na reportování čtyřnásobně většího množství obchodních dat do systému.

V rámci práce byl navržen nový datový model, který v případě nejobjemnější časové řady dosahuje o 43% menší datové náročnosti na diskovou kapacitu úložiště. Dále byla představena možnost dodatečného vylepšení prostřednictvím pokročilých kompresních technik databáze Oracle, pomocí kterých by uvedená časová řada dosahovala až 62% zlepšení v datové náročnosti. Snížení náročnosti bylo dosaženo především prostřednictvím odstranění redundantních sloupců a změnou použitých identifikátorů pro cizí klíče uvnitř tabulek časových řad. Pro koncové uživatele databáze však časové řady nabízí stále stejnou funkcionalitu a existující dotazy externích aplikací není nutné nijak modifikovat.

Navržený datový model je kromě nižší datové náročnosti i mnohonásobně efektivnější v dotazování na časové řady. Za vyšší efektivitu může především definovaní nového indexu, který umožňuje rychlejší filtraci přes obchodní dimenze. Pozitivní dopad na dotazování má i zavedení drobnějšího oddílového rozdělení u tabulek časových řad s nižší granularitou.

Kombinace výše uvedených změn přinesla pozitivní dopad i na vykonávání DML operací v modelu. Znatelné zlepšení výkonu bylo zaznamenáno u vytváření, úpravy a odstranění řádků tabulek časových řad.

Díky navrženým úpravám sice nový model částečně přišel o obecnost předchůdce, ale ve výsledku je pro správce databáze přehlednější a snazší na údržbu.

8 Závěry a doporučení

Teoretická část práce částečně přiblížila problematiku a specifika energetického trhu severských zemí, kam společnost Unicorn dodává systém určený k výpočtu a fakturaci vzniklých odchylek v dodávkách elektřiny. Na základě zjištěných poznatků a interních zdrojů společnosti Unicorn byly následně stanovené požadavky na časové řady vystupující v tomto systému.

Druhá polovina teoretické části byla věnována rozboru různých optimalizačních technik databáze Oracle. Byly zde zmíněny i některé dodatkové funkcionality, které mají velký vliv na efektivitu databáze. Dále byly uvedené různé praktiky a postupy správného návrhu databáze Oracle.

Začátek praktické části práce byl zaměřen na analýzu stávajícího modelu a uvedení jeho nedostatků. Po analýze následoval návrh nového datového modelu, jež řeší problém náhlého zvětšení objemu vstupních dat do systému. Při návrhu modelu byly využité především poznatky získané v teoretické části práce.

Závěrečná část práce byla věnována srovnání stávajícího a nově navrženého modelu ve více disciplínách. Nový datový model byl na základě několika definovaných scénářů vyhlášen za úspěšný, jelikož dosahuje nižší datové náročnosti a současně vyšší efektivity dotazování a zpracování DML operací.

Výběr tématu považuji za vhodný, jelikož mi práce pomohla prohloubit znalosti nejen o databázi Oracle, ale také o relačních databázích obecně. V rámci testování jsem si také vyzkoušel správu databázové instance v cloudu, což považuji za velice cenné zkušenosti do budoucna.

Při zpracování byly vynechány některé detaily jak z oblasti energetiky, tak databáze Oracle. Obě témata jsou velice obsáhlá a jejich kompletní pokrytí je rozsahově spíše na sérii knih. V práci jsem se tedy snažil uvádět především důležité části a doufám, že jsem vybral ty správné, které budou pro čtenáře alespoň částečně zajímavé.

Vhodné rozšíření práce by představovala implementace datového modelu časových řad v jiné databázové technologii, například v NoSQL nebo time series databázi.

9 Seznam použité literatury

1. DUNNING, Ted, B. Ellen FRIEDMAN, Michael Kosta LOUKIDES a Rebecca DEMAREST. *Time series databases: new ways to store and access data*. First edition. Sebastopol, CA: O'Reilly Media, Inc, 2014. ISBN 978-1-4919-1472-4.

2. Počet obyvatel - Metodika. *Počet obyvatel - Metodika* [online]. [vid. 2021-01-24]. Dostupné z: https://www.czso.cz/csu/czso/pocet_obyvatel_m

3. Obyvatelstvo - roční časové řady. *Obyvatelstvo - roční časové řady* [online]. [vid. 2021-01-23]. Dostupné z: https://www.czso.cz/csu/czso/obyvatelstvo_hu

4. TimescaleDB vs. InfluxDB: Purpose-built for time-series data. *Timescale Blog* [online]. 3. srpen 2020 [vid. 2021-02-07]. Dostupné z: https://blog.timescale.com/blog/timescaledb-vs-influxdb-for-time-series-data-timescale-influx-sql-nosql-36489299877/

5. NIELSEN, Aileen. *Practical time series analysis: prediction with statistics and machine learning*. Sebastopol, CA: O'Reilly Media, Inc, 2019. ISBN 978-1-4920-4165-8.

6. MAITRA, Sarit. LSTM to predict Dow Jones Industrial Average Time Series. *Medium* [online]. 10. září 2020 [vid. 2021-02-10]. Dostupné z: https://medium.com/analytics-vidhya/lstm-to-predict-dow-jones-industrial-average-time-series-647b0115f28c

7. DANNECKER, Lars. *Energy time series forecasting: efficient and accurate forecasting of evolving time series from the energy domain*. Wiesbaden: Springer Vieweg, 2015. Research. ISBN 978-3-658-11039-0.

8. Handbook. *eSett* [online]. 19. únor 2015 [vid. 2021-02-13]. Dostupné z: https://www.esett.com/handbook/

9. SALAVEC, Jiří. Trh s elektřinou - specifika, účastnící trhu a rozdělení. *oEnergetice.cz* [online]. [vid. 2021-02-20]. Dostupné z: https://oenergetice.cz/elektrina/trh-s-elektrinou/trh-s-elektrinou

10. Day-ahead market. *Nord Pool* [online]. [vid. 2021-02-25]. Dostupné z: https://www.nordpoolgroup.com/the-power-market/Day-ahead-market/

11. Intraday market. *Nord Pool* [online]. [vid. 2021-02-25]. Dostupné z: https://www.nordpoolgroup.com/the-power-market/Intraday-market/

12. LAGO, Jesus. Introduction to electricity markets, its balancing mechanism and the role of renewable sources. *Jesus Lago* [online]. 24. leden 2017 [vid. 2021-02-25]. Dostupné z: https://jesuslago.com/introduction-to-the-electrical-markets/

13. VAN DER VEEN, Reinier A. C. a Rudi A. HAKVOORT. The electricity balancing market: Exploring the design challenge. *Utilities Policy* [online]. 2016, **43**, 186–194. ISSN 0957-1787. Dostupné z: doi:10.1016/j.jup.2016.10.008

14. Day-ahead and intraday power trading. *Nord Pool* [online]. [vid. 2021-02-24]. Dostupné z: https://www.nordpoolgroup.com/trading/

15. eSett. *Public Data* [online]. [vid. 2021-03-06]. Dostupné z: https://opendata.esett.com/

16. Pro veřejnost. *ČEPS, a.s.* [online]. [vid. 2021-02-23]. Dostupné z: https://www.ceps.cz/cs/pro-verejnost

17. Markets divided into bidding areas. *Nord Pool* [online]. [vid. 2021-02-24]. Dostupné z: https://www.nordpoolgroup.com/the-power-market/Bidding-areas/

18. 15 min Imbalance Settlement Period. *Nordic Balancing Model* [online]. [vid. 2021-02-27]. Dostupné z: https://nordicbalancingmodel.net/roadmap-and-projects/15-min-time-resolution/

19. Nordic regulators urge the Nordic TSOs to continue to strive to avoid delays in the implementation of the 15 minute imbalance settlement period in the Nordics | NordREG. *NordREG* [online]. [vid. 2021-02-27]. Dostupné z: https://www.nordicenergyregulators.org/2019/04/nordic-regulators-urge-the-nordic-tsos-to-continue-to-strive-to-avoid-delays-in-the-implementation-of-the-15-minute-imbalance-settlement-period-in-the-nordics/

20. LONEY, Kevin. *Oracle Database 11g.* [online]. New York, USA: McGraw-Hill Professional Publishing, 2008 [vid. 2021-01-19]. ISBN 978-0-07-159876-7. Dostupné z: http://public.ebookcentral.proquest.com/choice/publicfullrecord.aspx?p=4657376

21. DB-Engines Ranking - Trend Popularity. *DB-Engines - Knowledge Base of Relational and NoSQL Database Management Systems* [online]. [vid. 2021-03-28]. Dostupné z: https://db-engines.com/en/ranking_trend

22. The Data Dictionary. *Database Documentation* [online]. [vid. 2021-03-19]. Dostupné z: https://docs.oracle.com/cd/B28359_01/server.111/b28318/datadict.htm#CNCPT002

23. GREENWALD, Rick, Robert STACKOWIAK a Jonathan STERN. *Oracle Essentials*. Fifth edition. Beijing: O'Reilly, 2013. ISBN 978-1-4493-4303-3.

24. BHATIYA, Rajesh, L. Ashdown A. AGRAWAL a Padmaja POTINENI. Managing Tablespaces. *Oracle Help Center* [online]. [vid. 2021-03-12]. Dostupné z: https://docs.oracle.com/en/database/oracle/oracle-

database/19/admin/managing-tablespaces.html#GUID-5E395CA6-A7AF-4E05-B9C1-E85DE0EA235A

25. Locally vs. Dictionary Managed Tablespaces | Oracle FAQ. *Oracle FAQ* [online]. [vid. 2021-03-19]. Dostupné z: http://www.orafaq.com/node/3

26. Tablespaces, Datafiles, and Control Files. *Database Documentation* [online]. [vid. 2021-03-12]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14220/physical.htm

27. Changing Datafile Size. *Database Documentation* [online]. [vid. 2021-03-18]. Dostupné z: https://docs.oracle.com/cd/B28359_01/server.111/b28310/dfiles003.htm#ADMIN11423

28. Data Blocks, Extents, and Segments. *Database Documentation* [online]. [vid. 2021-03-18]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14220/logical.htm#i4896

29. Logical Storage Structures. *Database Documentation* [online]. [vid. 2021-03-18]. Dostupné z: https://docs.oracle.com/cd/E25054_01/server.1111/e25789/logical.htm

30. HAAN, Lex de, Tim GORMAN, Inger JØRGENSEN a Melanie CAFFREY. *Beginning Oracle SQL: for Oracle database 12c*. Third edition. Berkeley, California? Apress, 2014. The expert's voice in Oracle. ISBN 978-1-4302-6556-6.

31. ROWID Pseudocolumn. *Database Documentation* [online]. [vid. 2021-03-14]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14200/pseudocolumns008.htm

32. Tables and Table Clusters. *Database Documentation* [online]. [vid. 2021-03-14]. Dostupné z: https://docs.oracle.com/cd/E11882_01/server.112/e40540/tablecls.htm#CNCPT010

33. *Oracle HEAP Tables* [online]. [vid. 2021-03-14]. Dostupné z: http://psoug.org/definition/heap.htm

34. Private Temporary Tables in Oracle Database 18c. *oracle-base.com* [online]. [vid. 2021-03-14]. Dostupné z: https://oracle-base.com/articles/18c/private-temporary-tables-18c

35. Refreshing Materialized Views. *Database Documentation* [online]. [vid. 2021-03-15]. Dostupné z: https://docs.oracle.com/database/121/DWHSG/refresh.htm#DWHSG03003

36. Create Materialized View. *Database Documentation* [online]. [vid. 2021-03-15]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_6002.htm

37. Materialized Views. *Database Documentation* [online]. [vid. 2021-03-16]. Dostupné z: https://docs.oracle.com/cd/B10501_01/server.920/a96520/mv.htm

38. HARRISON, Guy. *Oracle Performance Survival Guide: A Systematic Approach to Database Optimization*. Upper Saddle River, NJ: Pearson/Prentice Hall, 2010. ISBN 978-0-13-701195-7.

39. *Automatic Storage Management Administrator's Guide* [online]. [vid. 2021-03-27]. Dostupné z: https://docs.oracle.com/cd/E11882_01/server.112/e18951/asmcon.htm#OSTMG036

40. ALAPATI, Sam R., Darl KUHN a Bill PADFIELD. *Oracle Database 12c Performance Tuning Recipes: A Problem-Solution Approach*. New York: Apress, 2013. The expert's voice in Oracle. ISBN 978-1-4302-6187-2.

41. *Variable Blocksize* [online]. [vid. 2021-03-27]. Dostupné z: https://oss.oracle.com/~mason/blocksizes/

42. *Datetime Datatypes and Time Zone Support* [online]. [vid. 2021-03-21]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14225/ch4datetime.htm#i1005943

43. BURLESON, Donald K. Reverse key indexes with RAC. *Oracle Consulting* [online]. [vid. 2021-03-11]. Dostupné z: http://www.dba-oracle.com/t_rac_tuning_reverse_key_index.htm

44. BAER, Hermann. *The latest in Oracle Partitioning - Part 2: Multi Column List Partitioning* [online]. [vid. 2021-03-11]. Dostupné z: https://blogs.oracle.com/datawarehousing/the-latest-in-oracle-partitioning-part-2%3A-multi-column-list-partitioning

45. Automatic List Partitioning in Oracle Database 12c Release 2 (12.2). *oracle-base.com* [online]. [vid. 2021-03-11]. Dostupné z: https://oracle-base.com/articles/12c/automatic-list-partitioning-12cr2

46. Partitioned Tables And Indexes. *oracle-base.com* [online]. [vid. 2021-03-12]. Dostupné z: https://oracle-base.com/articles/8i/partitioned-tables-and-indexes

47. HODAK, Bill. Oracle Advanced Compression: Reduce Storage, Reduce Costs, Increase Performance. nedatováno, 42.

48. CHRISTMAN, Gregg, Kevin JERNIGAN a Cris PEDREGAL. Compression Best Practices. 2015, 32.

49. CHRISTMAN, Gregg. *Got Indexes? Want to Reduce Your Database Storage? Then You Need to Know About Advanced Index Compression!* [online]. [vid. 2021-03-17]. Dostupné z: https://blogs.oracle.com/dbstorage/got-indexes-want-to-reduce-your-database-storage-then-you-need-to-know-about-advanced-index-compression

50. Index Key Compression. *oracle-base.com* [online]. [vid. 2021-03-17]. Dostupné z: https://oracle-base.com/articles/9i/index-key-compression-9i

51. Influencing the Optimizer. *Database Documentation* [online]. [vid. 2021-03-22]. Dostupné z: https://docs.oracle.com/database/121/TGSQL/tgsql_influence.htm#TGSQL246

52. Managing Optimizer Statistics. *Database Documentation* [online]. [vid. 2021-03-26]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14211/stats.htm#g49431

53. Optimizer Access Paths. *Database Documentation* [online]. [vid. 2021-03-24]. Dostupné z: https://docs.oracle.com/database/121/TGSQL/tgsql_optop.htm#TGSQL228

54. Joins. *Database Documentation* [online]. [vid. 2021-03-24]. Dostupné z: https://docs.oracle.com/database/121/TGSQL/tgsql_join.htm#TGSQL242

55. Using Parallel Execution. *Database Documentation* [online]. [vid. 2021-03-26]. Dostupné z: https://docs.oracle.com/cd/B19306_01/server.102/b14223/usingpe.htm#i1006439

56. Generating and Displaying Execution Plans. *Database Documentation* [online]. [vid. 2021-03-26]. Dostupné z: https://docs.oracle.com/database/121/TGSQL/tgsql_genplan.htm#TGSQL271

57. Using Autotrace in SQL*Plus. *Database Documentation* [online]. [vid. 2021-03-26]. Dostupné z: https://docs.oracle.com/cd/A97630_01/server.920/a96533/autotrac.htm

58. CREATE USER. *Database Documentation* [online]. [vid. 2021-04-05]. Dostupné z: https://docs.oracle.com/database/121/SQLRF/statements_8003.htm#SQLRF01503

59. Moving Data Files Between Oracle ASM Disk Groups Using RMAN. *Database Documentation* [online]. [vid. 2021-04-13]. Dostupné

z: https://docs.oracle.com/database/121/OSTMG/GUID-3B8D0956-0888-452D-A9E4-9FB8D98577E0.htm#OSTMG89997

60. ALL_OBJECTS. *Oracle Help Center* [online]. [vid. 2021-04-14]. Dostupné z: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/refrn/ALL_OBJECTS.html#GUID-AA6DEF8B-F04F-482A-8440-DBCB18F6C976

61. Interval Literals. *Database Documentation* [online]. [vid. 2021-04-16]. Dostupné z: https://docs.oracle.com/database/121/SQLRF/sql_elements003.htm#SQLRF00221

62. Communication guidelines. *eSett* [online]. 1. únor 2016 [vid. 2021-02-27]. Dostupné z: https://www.esett.com/materials/communication-guidelines/

10 Přílohy

Příloha 1: Příklad datového toku bilaterálního obchodu

Příloha 2: Užitečné SQL dotazy

Příloha 3: Příklady řádků starého modelu

Příloha 4: Příklady řádků nového modelu

Příloha 5: Vytvoření primární hodnotové časové řady spotřeby starého modelu

Příloha 6: Vytvoření primární hodnotové časové řady spotřeby nového modelu

Příloha 7: Popis obsahu externí přílohy

Příloha 1: Příklad datového toku bilaterálního obchodu

```
<ScheduleDocument>
    <!--Identifikace dokumentu (GUID)-->
    <DocumentIdentification v="924d5e2822274684a0e3aefe7aa91143"/>
    <!--Identifikace typu MECu (BIT)-->
    <ProcessType v="Z05"/>
    <!--Identifikace odesílatele-->
    <SenderIdentification v="BRP1" codingScheme="A01"/>
    <!--Časový interval pokrývající celý dokument (UTC)-->
    <ScheduleTimeInterval v="2015-11-25T23:00Z/2015-11-26T23:00Z"/>
    <!--Identifikace trhu-->
    <Domain codingScheme="A01" v="10Y1001A1001A91G"/>
    <!--Obchodní data, v dokumentu jich může být i více-->
    <ScheduleTimeSeries>
        <!--Identifikace časové řady MECu-->
        <SendersTimeSeriesIdentification v="TS0001"/>
        <!--MBA ve kterém došlo k obchodu-->
        <Area v="MBA1" codingScheme="A01"/>
        <!--BRP na straně kupujícího-->
        <InParty v="BRP1" codingScheme="A01"/>
        <!--BRP na straně prodávajícího-->
        <OutParty v="BRP2" codingScheme="A01"/>
        <!--Jednotka měření obchodních dat-->
        <MeasurementUnit v="MWH"/>
        <Period>
            <!--Začátek a konec reportovaného intervalu (UTC)-->
            <TimeInterval v="2015-11-25T23:00Z/2015-11-26T23:00Z"/>
            <!--Granularita dat (hodina)-->
            <Resolution v="PT1H"/>
            <!--Každá položka má relativní pozici v intervalu
                a hodnotu časové řady (kvantita obchodované energie)-->
            <Interval>
                <Pos v="1"/>
                <Qty v="10.123456"/>
            </Interval>
            <Interval>
                <Pos v="2"/>
                <Qty v="20.123456"/>
            </Interval>
            ...
        </Period>
    </ScheduleTimeSeries>
</ScheduleDocument>
```

Zdroj: (62)

Příloha 2: Užitečné SQL dotazy

```
-- Zobrazení inicializačních parametrů databáze
SHOW PARAMETERS;

-- Zobrazení specifického parametru databáze, například mód optimalizátoru
SHOW PARAMETER OPTIMIZER_MODE;

-- Získání permanentních parametrů databáze
SELECT * FROM DATABASE_PROPERTIES;

-- Získání parametrů databáze, jež jsou aktivní pro aktuální relaci
SELECT * FROM V$PARAMETER;

-- Získání verze databáze
SELECT * FROM V$VERSION;

-- Získání tabulek s zřetězenými řádky
SELECT TABLE_NAME, CHAIN_CNT, CHAIN_CNT/NUM_ROWS
FROM DBA_TABLES
WHERE CHAIN_CNT > 0 AND NUM_ROWS > 0;

-- Získání velikosti datového bloku a správy místa per tabulkový prostor
SELECT TABLESPACE_NAME, BLOCK_SIZE, EXTENT_MANAGEMENT
FROM DBA_TABLESPACES;

-- Získání počtu bajtů potřebných pro uložení libovolné hodnoty
SELECT VSIZE(42) FROM DUAL;

-- Manuální sběr statistik pro celé schéma
EXECUTE DBMS_STATS.GATHER_SCHEMA_STATS('USRBUF');

-- Promazání mezipaměti pro benchmark a ladění dotazů
ALTER SYSTEM FLUSH SHARED_POOL;
ALTER SYSTEM FLUSH BUFFER_CACHE;

-- Získání celkové velikosti tabulkového prostoru v MB
SELECT ROUND(SUM(BYTES) / (1024 * 1024)) MB
FROM DBA_SEGMENTS
WHERE TABLESPACE_NAME = 'DAE_DATA';

-- Povolení paralelního zpracování dotazů
ALTER SESSION ENABLE PARALLEL QUERY;
```

Zdroj: vlastní zpracování

Příloha 3: Příklady řádků starého modelu

Validita a identifikátory jsou upraveny kvůli formátování.

Tabulka 5: Příklad řádků starého pohledu AF_MGA

OID	MGA	FROM TIME	TO TIME	NAME	CODING SCHEME	INTERNAL ID
1	…546	01.01.2016 00	31.12.2026 00	MGA1	…609	EIC_MGA_1
2	…547	01.01.2021 00	31.12.2021 00	MGA2	…609	EIC_MGA_2

Tabulka 6: Příklad řádků starého pohledu AF_VAL_CONS_RE

OID	MGA	FROM TIME	TO TIME	QUANTITY	QUALITY
1	…546	07.04.2021 00	07.04.2021 01	1,12345678	…309
2	…547	07.04.2021 00	07.04.2021 01	2,12345678	…306

Tabulka 7: Příklad řádků starého pohledu AF_MEC_CONS_RE

OID	MGA	FROM TIME	TO TIME	MEC ID
1	…546	01.01.2016 00	31.12.2026 00	CONS1
2	…547	01.01.2021 00	31.12.2021 00	CONS2

Tabulka 8: Příklad řádků tabulky T101TIME_UNIT

OID	T101NAME	T101CODE	T101TIME_UNIT
1	Hour	HOUR	The basic time unit - hour.
2	Day	DAY	The basic time unit - day.
3	Week	WEEK	A time interval of seven consecutive days.

Tabulka 9: Příklad řádků tabulky T103TIME_UNIT_VALUE

OID	T103FROM_DATE	T103TO_DATE	T101TIME_UNIT
1	07.04.2021 00:00:00	07.04.2021 01:00:00	1
2	07.04.2021 00:00:00	08.04.2021 00:00:00	2
3	05.04.2021 00:00:00	12.04.2021 00:00:00	3

Příloha 4: Příklady řádků nového modelu

Validita a identifikátory jsou upraveny kvůli formátování.

Tabulka 10: Příklad řádků nového pohledu AF_MGA

MGA	BD MGA	FROM TIME	TO TIME	NAME	CODING SCHEME	INTERNAL ID
1	…546	01.01.2016 00	31.12.2026 00	MGA1	3	EIC_MGA_1
2	…547	01.01.2021 00	31.12.2021 00	MGA2	3	EIC_MGA_2

Tabulka 11: Příklad řádků nového pohledu AF_VAL_CONS_RE

OID	MGA	FROM TIME	TO TIME	QUANTITY	QUALITY
1	1	07.04.2021 00	07.04.2021 01	1,12345678	2
2	2	07.04.2021 00	07.04.2021 01	2,12345678	1

Tabulka 12: Příklad řádků nového pohledu AF_MEC_CONS_RE

OID	MGA	FROM TIME	TO TIME	MEC ID
1	1	01.01.2016 00	31.12.2026 00	CONS1
2	2	01.01.2021 00	31.12.2021 00	CONS2

Příloha 5: Vytvoření primární hodnotové časové řady spotřeby starého modelu

```
-- Tabulka časové řady (hodinová verze)
CREATE TABLE USRBUF.TS_14500000002085720
(
    OID NUMBER(22,0) NOT NULL,
    T103TIME_UNIT_VALUE NUMBER(22,0) NOT NULL,
    BD_14500000002173483 NUMBER(22,0) NOT NULL,
    BD_14500000002173485 NUMBER(22,0) NOT NULL,
    BD_14500000002173487 NUMBER(22,0) NOT NULL,
    BD_14500000002173489 NUMBER(22,0) NOT NULL,
    FROM_TIME TIMESTAMP(6) NOT NULL,
    TO_TIME TIMESTAMP(6) NOT NULL,
    COMP_ERR_14500000002102116 NUMBER(1,0), CORR_VALUE_14500000002102116 NUMBER(20,8),
    TS_VALUE_14500000002102116 NUMBER(20,8), T406VALSTATE_14500000002102116 CHAR(1 BYTE),
    COMP_ERR_14500000002102113 NUMBER(1,0), CORR_VALUE_14500000002102113 NUMBER(17,0),
    TS_VALUE_14500000002102113 NUMBER(17,0), T406VALSTATE_14500000002102113 CHAR(1 BYTE)
)
TABLESPACE DAE_DATA
PARTITION BY RANGE (TO_TIME)
(
    PARTITION DAE_DATA_Q_2021_1 VALUES LESS THAN (TIMESTAMP '2021-04-01 00:00:00')
    ROW STORE COMPRESS BASIC TABLESPACE DAE_DATA_Q_2021_1,
    ...
    PARTITION DAE_DATA_MAXVALUE VALUES LESS THAN (MAXVALUE)
    NOCOMPRESS TABLESPACE DAE_DATA_MAXVALUE
);

-- B-tree index primárního klíče a integritní omezení
CREATE UNIQUE INDEX USRBUF.PK_14500000002085720 ON USRBUF.TS_14500000002085720 (OID, TO_TIME)
COMPRESS LOCAL
(
    PARTITION DAE_DATA_Q_2021_1 COMPRESS TABLESPACE DAE_INDX_Q_2021_1,
    ...
    PARTITION DAE_DATA_MAXVALUE NOCOMPRESS TABLESPACE DAE_INDX_MAXVALUE
);

ALTER TABLE USRBUF.TS_14500000002085720 ADD CONSTRAINT PK_14500000002085720
PRIMARY KEY (OID, TO_TIME) USING INDEX USRBUF.PK_14500000002085720;

-- B-tree index unikátní kombinace klíče a integritní omezení
CREATE UNIQUE INDEX USRBUF.UQ_14500000002085720_BD
ON USRBUF.TS_14500000002085720 (TO_TIME, BD_14500000002173483,
BD_14500000002173485, BD_14500000002173487, BD_14500000002173489)
COMPRESS LOCAL
(
    PARTITION DAE_DATA_Q_2021_1 COMPRESS TABLESPACE DAE_INDX_Q_2021_1,
    ...
    PARTITION DAE_DATA_MAXVALUE NOCOMPRESS TABLESPACE DAE_INDX_MAXVALUE
);

ALTER TABLE USRBUF.TS_14500000002085720 ADD CONSTRAINT UQ_14500000002085720_BD
UNIQUE (TO_TIME, BD_14500000002173483, BD_14500000002173485, BD_14500000002173487,
BD_14500000002173489) USING INDEX USRBUF.UQ_14500000002085720_BD;
```

Příloha 6: Vytvoření primární hodnotové časové řady spotřeby nového modelu

```
-- Tabulka časové řady (hodinová verze)
CREATE TABLE USRBUF.TSP_VAL_CONS_RE_PT1H
(
    OID NUMBER(22,0) NOT NULL,
    OID_MGA NUMBER(22,0) NOT NULL,
    OID_RE NUMBER(22,0) NOT NULL,
    OID_CONSUMPTION_TYPE NUMBER(22,0) NOT NULL,
    OID_MEASUREMENT_TYPE NUMBER(22,0) NOT NULL,
    TO_TIME TIMESTAMP(6) NOT NULL,
    VAL_QUANTITY NUMBER(20,7), VAL_S_QUANTITY CHAR(1 BYTE) DEFAULT 'P' NOT NULL,
    VAL_QUALITY NUMBER(22,0), VAL_S_QUALITY CHAR(1 BYTE) DEFAULT 'P' NOT NULL
)
TABLESPACE DAE_DATA
PARTITION BY RANGE (TO_TIME)
(
    PARTITION DAE_DATA_M_2021_1 VALUES LESS THAN (TIMESTAMP '2021-02-01 00:00:00')
    ROW STORE COMPRESS BASIC TABLESPACE DAE_DATA_M_2021_1,
    ...
    PARTITION DAE_DATA_MAXVALUE VALUES LESS THAN (MAXVALUE)
    NOCOMPRESS TABLESPACE DAE_DATA_MAXVALUE
);

-- B-tree index primárního klíče a integritní omezení
CREATE UNIQUE INDEX USRBUF.PK_TSP_VAL_CONS_RE_PT1H ON USRBUF.TSP_VAL_CONS_RE_PT1H (OID,
TO_TIME)
COMPRESS LOCAL
(
    PARTITION DAE_DATA_M_2021_1 COMPRESS TABLESPACE DAE_INDX_M_2021_1,
    ...
    PARTITION DAE_DATA_MAXVALUE NOCOMPRESS TABLESPACE DAE_INDX_MAXVALUE
);

ALTER TABLE USRBUF.TSP_VAL_CONS_RE_PT1H ADD CONSTRAINT PK_TSP_VAL_CONS_RE_PT1H
PRIMARY KEY (OID, TO_TIME) USING INDEX USRBUF.PK_TSP_VAL_CONS_RE_PT1H;

-- B-tree index unikátní kombinace a integritní omezení
CREATE UNIQUE INDEX USRBUF.UQ_TSP_VAL_CONS_RE_PT1H
ON USRBUF.TSP_VAL_CONS_RE_PT1H (TO_TIME, OID_MGA, OID_RE, OID_CONSUMPTION_TYPE,
OID_MEASUREMENT_TYPE)
COMPRESS LOCAL
(
    PARTITION DAE_DATA_M_2021_1 COMPRESS TABLESPACE DAE_INDX_M_2021_1,
    ...
    PARTITION DAE_DATA_MAXVALUE NOCOMPRESS TABLESPACE DAE_INDX_MAXVALUE
);

ALTER TABLE USRBUF.TSP_VAL_CONS_RE_PT1H ADD CONSTRAINT UQ_TSP_VAL_CONS_RE_PT1H
UNIQUE (TO_TIME, OID_MGA, OID_RE, OID_CONSUMPTION_TYPE, OID_MEASUREMENT_TYPE)
USING INDEX USRBUF.UQ_TSP_VAL_CONS_RE_PT1H;

-- Bitmap index obchodních dimenzí
CREATE BITMAP INDEX USRBUF.BIT_TSP_VAL_CONS_RE_PT1H
ON USRBUF.TSP_VAL_CONS_RE_PT1H (OID_MGA, OID_RE, OID_CONSUMPTION_TYPE, OID_MEASUREMENT_TYPE)
LOCAL TABLESPACE DAE_INDX;
```

Příloha 7: Popis obsahu externí přílohy

ZIP archív externí přílohy obsahuje následující složky.

DataGenerator

Složka obsahuje zdrojový kód vytvořeného C# programu pro generování a vkládání dat do časových řad. Generování datových SQL skriptů je závislé na konfiguraci uložené v souboru „ModelConfig.json“. Přiložený konfigurační soubor vygeneruje datové skripty pouze pro 1 den. Pro účely testování byly vygenerovány skripty pro celý rok, tj. 365 dní. Důvodem pro omezení je fakt, že generované skripty pro celý rok vyžadují více než 8 GB diskové kapacity, což nemusí být chtěné. Omezení je možné upravit skrze parametr konfigurace „CountOfDays“. Výsledné skripty jsou generované do složky „output“ nacházející se ve stejné složce, jako spustitelný soubor programu. V debug módu se tedy skripty nachází defaultně na cestě „DataGenerator\bin\Debug\net5.0-windows\output“.

Modely

Ve složce se nachází kompletní SQL skripty pro vytvoření a odstranění starého a nového datového modelu. Nachází se zde také soubor „schema.sql“, jež vytvoří schéma USRBUF. Každý model má vlastní UP a DOWN skript. UP skript vytvoří všechny potřebné tabulkové prostory a přidělí u nich potřebná práva schématu USRBUF. Po spuštění UP skriptu je možné vytvoření všech databázových objektů časových řad skrze jednotlivé skripty (BD, MEC_CONS_RE, VAL_CONS_RE). Pro oba modely existují navíc skripty na vytvoření hodnotových řad s pokročilou kompresí.

Testy

Složka obsahuje všechny získané výsledky při srovnání obou modelů. Nachází se zde excelovský soubor „celkove_vysledky.xlsx“, který obsahuje souhrn všech důležitých metrik, jež jsou uvedeny v této práci. Ve složce jsou dostupné také uvedené dotazy a získané exekuční plány a statistiky dotazů.

Příklady

Složka zahrnuje skripty, které byly uvedené v rámci práce jako příklady, tedy založení tabulek, indexů apod.

Univerzita Hradec Králové
Fakulta informatiky a managementu

Zadání diplomové práce

Autor:	**Bc. Lukáš Chvátal**
Studium:	I1800110
Studijní program:	N1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název diplomové práce:	**Optimalizace ukládání časových řad v energetice**
Název diplomové práce AJ:	Time series storage optimization for energetics

Cíl, metody, literatura, předpoklady:

Energetický sektor často ukládá data ve formě časových řad, které vyžadují speciální datové modely. Časové řady v energetice mají různou granularitu, od dat z měřících čidel se sekundovou granularitou, až po obchodní data, například s denní granularitou. Cílem diplomové práce je optimalizace stávajícího datového modelu, který již není vyhovující pro aktuální požadavky.

DP ve spolupráci s firmou Unicorn Systems a.s.

Osnova:

1. Úvod
2. Časová řada
3. Časová řada v energetice
4. Možnosti ukládání časové řady
5. Oracle
6. Stávající datový model
7. Optimalizace datového modelu
8. Shrnutí výsledků
9. Závěry a doporučení
10. Seznam použité literatury

1. DUNNING, Ted, B. Ellen FRIEDMAN, Michael Kosta LOUKIDES a Rebecca DEMAREST. Time series databases: new ways to store and access data. First edition. Sebastopol, CA: O'Reilly Media, Inc, 2014. ISBN 978-1-4919-1472-4.
2. GREENWALD, Rick, Robert STACKOWIAK a Jonathan STERN. Oracle Essentials. Fifth edition. Beijing: O'Reilly, 2013. ISBN 978-1-4493-4303-3.
3. ALAPATI, Sam R., Darl KUHN a Bill PADFIELD. Oracle Database 12c performance tuning recipes: a problem-solution approach. New York: Apress, 2013. The expert's voice in Oracle. ISBN 978-1-4302-6187-2.
4. KLEPPMANN, Martin. Designing data-intensive applications: the big ideas behind reliable, scalable, and maintainable systems. Beijing: O'Reilly, 2017. ISBN 978-1449373320.
5. DATE, C. J. SQL and Relational Theory: How to Write Accurate SQL Code. Third edition. Sebastopol, CA: O'Reilly Media, Inc, 2015. Theory in practice. ISBN 978-1-4919-4117-1.

Garantující pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	Ing. Barbora Tesařová, Ph.D.
Datum zadání závěrečné práce:	14.1.2018