Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Bakalářská práce

Optimalizace procesů centrálního datového skladu v oblasti Process Control Center

Eduard Hübner

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačních technologií

Provozně ekonomická fakulta

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Hübner Eduard

Informatika

Název práce

Optimalizace procesů centrální datového skladu v oblasti Process Control Center

Anglický název

Optimalization of the central data warehouse in the field of Process Control Center

---

Cíle práce

Bakalářská práce je tematicky zaměřena na problematiku optimalizace procesů centrálního datového skladu (CDS).
Hlavním cílem práce je řešení problému plánování běhu procesů centrálního datového skladu, konkrétně jeho součásti Process Control Center (PCC) a jeho praktická aplikace. Dílčí cíle bakalářské práce jsou:
-analýza aplikace teorie grafů při řízení procesů
-návrh optimalizovaného algoritmu
-vytvoření řešení v SQL

Metodika

Metodika řešení problematiky dané práce je založena na studiu a analýze odborných zdrojů, či již hotových a realizovaných popsaných řešeních. Vlastní řešení je realizováno návrhem nového algoritmu pro běh procesů aplikace PCC a jeho praktickou aplikací v jazyce SQL. Na základě výsledků úspěšnosti implementace budou formulovány závěry bakalářské práce.

Harmonogram zpracování

1) Příprava a studium odborných informačních zdrojů, upřesnění dílčích cílů práce a volba postupu řešení: 06/2011
2) Zpracování přehledu řešení problematiky dle informačních zdrojů: 07/2011 – 08/2011
3) Zahájení pracovního poměru, tvorba řešení reálného problému: 09/2011- 12/2011
4) Tvorba finálního dokumentu bakalářské práce, implementace řešení: 01/2012 – 02/2012
5) Odevzdání bakalářské práce a teze: 03/2012

Rozsah textové části

30 - 40 stran

Klíčová slova

Process Control Center (PCC), Structured Query Language (SQL), Centrální datový sklad (CDS), Teorie grafů

Doporučené zdroje informací

Křenek, Miroslav. PCC_provozni_prirucka:ČSOB, 2007. Verze 00_07

Mehlhorn, Kurt. Algorithms and Data Structures: The Basic Toolbox, Berlin : Springer-Verlag Berlin Heidelberg, 2010. ISBN 978-3-642-09682-2

Nielsen, Paul. Microsoft SQL Server 2008 Bible, Indianapolis, Wiley Publishing, 2009. ISBN: 978-0-470-25704-3

Lacko Ľuboslav.Business Intelligence v SQL Serveru 2008, Praha: Computer Press a.s. 2009. ISBN: 978-80-251-2887-9

Tvrdíková, Milena. Aplikace moderních informačních technologií v řízení firmy, Praha: Grada Publishing a.s. 2008. ISBN: 978-80-247-2728-8

Vedoucí práce

Hesová Ivana, Ing.

Termín odevzdání

březen 2012

doc. Ing. Zdeněk Havlíček, CSc.
Vedoucí katedry

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE
Provozně ekonomická fakulta

prof. Ing. Jan Hron, DrSc., dr.h.c.
Děkan fakulty

V Praze dne 21.11.2011

Čestné prohlášení

Prohlašuji, že svou diplomovou práci "Optimalizace procesů centrálního datového skladu v oblasti Process control center" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autoruvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 28.3.2012 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Ivaně Hesové, za odborné vedení a cenné připomínky při zpracování této práce. Dále bych rád poděkoval kolegům z Centrálního datového skladu ČSOB, a.s, za poskytnutí odborných materiálů, konzultace a pomoc při orientaci v rozebírané problematice.

Eduard Hübner

Optimalizace procesů centrálního datového skladu v oblasti Process Control Center

---

Optimalisation of the central data warehouse in the field of Process Control Center

Souhrn

Tato práce se zabývá problematikou optimalizace běhu procesů při provozu centrálního datového skladu, konkrétně jeho řídícího programu Process Control Center. Práce je rozdělená do dvou částí.

První, teoretická část pojednává o problematice informačně technologického prostředí (Corporate Information Factory) a jejích komponentách, především o datových skladech. Jsou zde popsány nejdůležitější mechanismy datového skladu. Dále tato část popisuje řešení datového skladu ve společnosti ČSOB. Závěrem je uvedeno stručné pojednání o teorii grafů, které slouží k zorientování se v termínech z teorie grafů, souvisejících s touto prací.

Druhá, prakticky zaměřená část práce, pojednává o problému výpočtu priorit pro zpracování úloh. V této části se nachází popisy zdrojových SQL tabulek, algoritmu využívaného pro výpočet priorit a popisy používaných Perl skriptů. První skriptem je současný využívaný skript. Druhým skriptem je autorem upravený skript pro ilustrační účely práce. V závěru jsou uvedeny výsledky testování nového skriptu, jejich zhodnocení a možnosti jejich využití.

Summary

Topic of this thesis is optimisation of process run in Data Warehouse, namely its control program Process Control Center. The work is divided in to two parts.

The theoretical part contains problematic of Corporate Information Factory and its components, especially Data Warehouses. There are described crucial mechanisms of Data Warehouse. Furthermore this part describes example of Data Warehouse in company ČSOB, a.s. In the end of this part there is a short introduction to graph theory which is necessary for understanding this thesis.

Second part is practical. It describes the problem of priority scheduling for jobs of daily ETL. In this part there is description of used SQL tables, algorithm used for priority scheduling and description of used Perl scripts. First script is the currently used. Second script is modified by the author for purpose of an illustration of this thesis. In the end there is a summary of new script testing and evaluation of contribution of the new script and work overall to ČSOB.

Klíčová slova: PCC, SQL, CDS, Perl, kritická cesta, teorie grafů, strom.

Keywords: PCC, SQL, CDS, Perl, Critical Path, Graph Theory, Tree.

Obsah

1 Úvod

Téma problematiky Information Factory a datových skladů je zajímavé z důvodu jedinečnosti a důležitosti jeho obsahu. Unikátnost projektu je dána rozsáhlostí, ovlivněnou velikostí společnosti ČSOB a počtem jednotlivých systémů, ze kterých je sběr do datového skladu prováděn.

S rozvojem možností výpočetní techniky jde ruku v ruce rozvoj bankovních systémů. Dnes již nestačí skladovat pouze základní informace o účtech, jako jsou jejich zůstatky, či informace o jejich vlastnících. Obrovské spektrum bankovních služeb jako jsou kreditní karty, hypotéky, korporátní služby, ale zároveň vnitrobankovní operace potřebují být uloženy a zaznamenány na přehledném a přístupném místě.

Pro tyto potřeby byla vytvořena koncepce centrálních datových skladů, v tomto případě založených na SQL databázi. Sběr a zpracování dat je prováděn tzv. ETL programy (extract, transofrm load), kterých je velké množství a při jejich zpracování musí být dodržovány závislosti mezi nimi. Celé zpracování řídí aplikace PCC (Process Control Center).

V případě rozsáhlých subjektů není možné vystačit si pouze s jedním systémem produkujícím homogenní formát dat, snadno uložitelný a zpracovatelný. V případě ČSOB hovoříme o desítkách systémů, běžících na různých standardech, jejichž data jsou mezi sebou provázaná.

O chápání klíčového významu těchto datových zdrojů pro firmu svědčí i skutečnost, že všechny aktivity a procesy související s provozem datového skladu jsou v ČSOB soustředěny ve finanční divizi.

2 Cíle a Metodika

2.1 Cíl Práce

Cílem této práce je optimalizace procesů potřebných k zpracování operací v centrálním datovém skladu v ČSOB, a.s., konkrétně algoritmu sloužícímu k řízení pořadí zpracování, který je součástí řídící aplikace PCC (Process Control Center). Výsledkem optimalizace je úprava stávajícího algoritmu. Tato úprava přinese možnost manuálního zásahu do pořadí zpracování jednotlivých úloh. To poslouží k časové úspoře a efektivnější práci s daty, závisejících na úlohách, které je potřeba urychlit.

Dílčím cílem práce je studie a rozbor principu fungování tzv. Corporate Information Factory (jejímž základem je datový sklad) a principu fungování datového skladu a jeho řídících procesů v ČSOB, a.s. V závěru práce je provedeno zhodnocení přínosu upraveného algoritmu. Práce by zároveň měla sloužit i jako metodický materiál pro zaměstnance a partnery ČSOB, kteří přijdou s CDS do styku.

2.2 Metodika

V úvodní části práce jsou zpracována teoretická východiska pro řešení vybraného problému, která byla získána studiem odborných publikací, internetových zdrojů a interních materiálů ČSOB, a.s. Na základě získaných informací bylo provedeno třídění dat a jejich uspořádání.

V praktické části byla provedena analýza již hotových fungujících řešení, vytvořených v jazyce Perl a SQL. SQL data není třeba pro potřeby práce modifikovat, upraven byl pouze algoritmus a skript v jazyce Perl. Jazyk Perl je vhodný pro práci s databázemi a tvorbu krátkých skriptů. V tomto jazyce byl napsán i původně používaný výchozí skript. Pro úpravy a analýzu algoritmu byla využita metodika teorie grafů.

Na závěr bude provedeno zhodnocení a popis výsledků, dosažených úpravou algoritmu. Dále byl zhodnocen přínos práce pro ČSOB, a.s.

3 Úvod do Corporate Information Factory

3.1 Historie a vývoj

Po dlouhá léta firmy a organizace toužily po nástroji, který by jim umožnil sběr dat ze všech dostupných zdrojů a dal jim k dispozici co největší objem klíčových dat o klientech. Tato data mají význam především v oblastech, jako jsou PR, marketing či zákaznický servis. Výsledná data složená až z desítek informačních systémů mají daleko větší vypovídající hodnotu a umožňují přesnější adresnost nabídky.

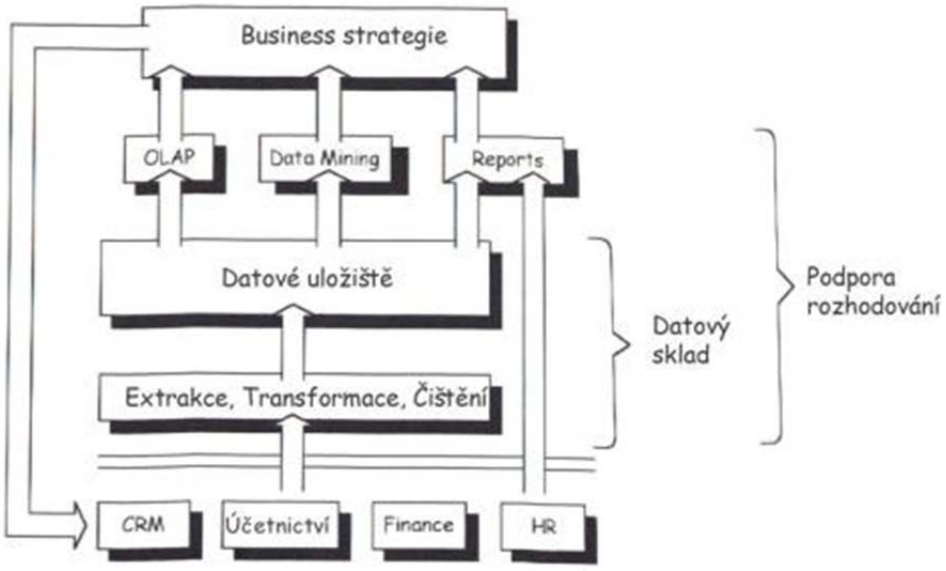
Tyto funkce s nástupem informačních technologií ze začátku zastávaly databáze, kterých ale s postupem času začalo přibývat a zároveň se přímo úměrně začaly zvětšovat objemy dat, které jsou v databázích obsaženy. Stávající situace vyžadovala řešení v podobě komplexní správy dat plynoucích z těchto zdrojů, kterým je právě Corporate Information Factory.[1]

3.2 Business Intelligence

Business inteligenci (BI) můžeme definovat jako set matematických modelů a analytických metod, které využívají dostupná data k vytvoření informací, potřebných jako podklady k rozhodovacímu procesu. [2]

Jak již bylo uvedeno výše, údaje nabyté pomocí BI najdou své využití především v rozhodovacím procesu. Nejmarkantněji se tato data dají využít v dlouhodobých plánováních, které se dají předvídat právě na základě modelových situací z historie. [3]

[1] Imhoff, s. 31 - 32
[2] Vercellis, s. 4
[3] Giovinazzo s. 3 - 4



Obrázek 1 Koloběh business inteligence. [4]

3.3 Datový sklad

Datový sklad je možné označit za srdce Business Inteligence, z laického pohledu je možno ho přirovnat k jakémukoliv normálnímu skladu, např. materiálu, kde musíme provádět velké množství doprovodných činností k uložení věcí samotných.

Komplex datového skladu můžeme rozdělit na několik základních komponent:

Datové uložiště

-hardware a software

-databáze a datový model

ETL

Využití dat

3.3.1 ETL

Systém Extract - Transform – Load je základním způsobem plnění datového skladu. Správně navržený ETL systém zajišťuje kvalitu, konzistenci dat a doručení výsledků pro jejich přehlednou prezentaci, což usnadňuje práci jak tvůrcům aplikací, tak koncovým uživatelům. ETL je rozhodujícím faktorem datového skladu. Přestože uživatel jeho funkci nijak nevnímá, systém dokáže využít kolem 70% dostupných výpočetních zdrojů.[5]

[4] Giovinazzo, s. 2
[5] Kimball, s.8.

Hlavními činnostmi ETL systému jsou čištění dat, jejich konsolidace a transformace do jednotné formy pro koncové uživatele. Pro tuto činnost dnes existuje spousta nástrojů například Informatica PowerCenter, SAS ETL Studio, Microsoft SQL server Integration Services, opensource Pentaho Kettle a mnoho dalších.[6]

3.3.1.1 Extract

Data poskytují základní zdrojové systémy formou extraktů, čtení z databází atd. Pro zachycení všech zdrojových dat je třeba nastavit podmínky jejich načtení. Může se jednat o časový interval, vynucené načtení, načtení po každé transakci. U datových skladů se nejčastěji jedná o interval časový v řádu dní, týdnů či měsíců.[7]
Pokud jsou data načtena, přichází na řadu jejich samotné zpracování. Zdrojové materiály musí být nejdříve seskupeny, zkontrolována jejich úplnost. V případě, že není k dispozici požadovaná dávka dat, mohou čekat na své další zpracování v tzv. stage sektoru. [8]
Stage sektor je určen ke kontrole kvality dat, pokud nesplňují požadovaná kritéria, do procesu uložení nepokračují. Další možností je seřazení dat pro zrychlení procesu uložení. O tuto činnost se stará i samotný ukládací nástroj, ale v případě větších objemů dat je výhodné toto seřazení provést.[9]

3.3.1.2 Transform

Jelikož data pochází z heterogenních systémů, které se liší jak přesností a objemem dat, které produkují, tak rozličnými formáty, závislými na programovacích jazycích a alfanumerických standardech a jsou ukládána do jednotného datového modelu datového skladu, je nutné je transformovat.[10]
Prvním zástupcem tohoto procesu je čištění dat. Základními předpoklady pro vznik nekvalitních dat jsou: nedostatečné porozumění dostupným datům, neznalost dat, se kterými pracujeme, neochota zabývat se nápravou špatných dat – špatná zpětná vazba, nezájem odpovědných osob.[11]

---

[6] www.etltools.net
[7] Imhoff, s. 213.
[8] Giovinazzo, s. 14. – 17.
[9] Imhoff, s. 220. – 221.
[10] Imhoff, s. 47
[11] Imhoff, s. 220

Čištění samotné je proces o různých stupních sofistikovanosti. Může probíhat jako prostá kontrola formátů dat, záporných hodnot atd. Nicméně čištění dat pro velké datové sklady vyžaduje vyšší stupeň profesionality. Nejčastěji jsou využívány samostatné procesy vlastní tvorby, či od externích dodavatelů. Tyto procesy jsou nazývány jako hygienické. Přestože některé čistící systémy mohou být na velice profesionální úrovni, nikdy se určitému stupni znečištění dat nevyhneme a nezbývá než ho akceptovat. [12]

Po čištění přichází na řadu transformace. Tento důležitý proces konsoliduje všechny možné formáty a označení do normované podoby. Nejlépe si ilustrujeme variabilitu dat na následující tabulce.

	Poukazy	Objednávky	Inventář
Popis	Jméno zákazníka— I.B.M.	Jméno zákazníka— IBM	Jméno zákazníka— International Business Machines
Kódování	Pohlaví— 1 = muž 2 = žena	Pohlaví— M = muž F = žena	Pohlaví— X = muž Y = žena
Jednotky	Délka kabelu— centimetry	Délka kabelu— yardy	Délka kabelu— palce
Formátování	Key— Character (10)	Key— Integer	Key— Pic '999999'

Obrázek 2 Problémy integrace [13]

Z tabulky je zřejmé kolik možných variant nám mohou systému doručit. Při nakládání s těmito daty je nejlepší určit jednotný systém jednotek a formátů, ve kterém bude převáděn. [14]

[12] Imhoff, s.220 - 221
[13] Giovinazzo, s. 17
[14] Giovianazzo, s. 16. – 19.

3.3.1.3 Load

Poslední částí systému ETL je samotné uložení dat do datového skladu. Tento proces je, co se týče výpočetního výkonu, nejnáročnější. Existuje více možností, jak tento proces provést, pro ilustraci jsou uvedeny základní části, které by neměly být při plánování opomenuty.[15]

Záloha příchozích dat. Při operacích s velkými objemy dat mohou nastat chyby ať již s poškozenými zdrojovými daty, které nebyly odhaleny dříve, či chyby hardwarové.[16]

Stanovení priorit zdrojů. Některé zdroje dat musí být načteny dříve pro zajištění integrity dat. Nemůže například načítat informace o transakcích provedených klientem dříve, než načteme informace o klientu samotném. Toto pravidlo platí i při ukládání do databáze. Tento krok je z velké míry eliminován využitím sofistikovaných ETL nástrojů.[17]

Nezměnitelnost dat. Hlavním rozdílem mezi datovým skladem a ostatními transakčně orientovanými systémy je stálost dat. Datový sklad se v tomto ohledu chová jako databáze sloužící pouze ke čtení. Zápis do databáze je samozřejmě možný, ale je prováděn automaticky. Jakýkoliv přepis dat tedy není možný, povolen je pouze nový zápis. Tento mechanizmus poskytuje možnost procházet data ve skladu podle času jejich pořízení – datový sklad obsahuje historii jednotlivých datových atributů. [18]

[15] www.etltools.org
[16] Imhoff, s. 224.
[17] Imhoff, s. 224.
[18] Giovinazzo, s. 18.-20.

4 Teorie Grafů

Teorie grafů je jedním z nejvýznamnějších zástupců diskrétní matematiky. Znalost teorie grafů je nezbytná ve většině oblastí moderní informatiky.

„Graf je dvojice (V,E), kde V je množina, nazýváme ji množinou vrcholů, a E je množina některých dvojic prvků množiny V, nazýváme ji množinou hran. Je-li E množina uspořádaných dvojic, říkáme, že graf G je orientovaný, je-li množina neuspořádaných dvojic, říkáme, že graf G je neorientovaný. Je-li G graf, pak symboly V(G) a E(G) označujeme jeho množinu vrcholů a hran.“ [19]

4.1 Reprezentace grafů

Teorie grafů je ve velkém množství využívána k řešení problémů v informatice, proto je potřeba mít k dispozici reprezentaci grafu vhodnou k počítačovému zpracování. Mezi vhodné reprezentace není možné počítat znázornění grafu pomocí obrázku, přestože je pro uživatele přehledný a přívětivý. Ke znázornění struktury grafu stačí určení krajních uzlů a hran. Obrázková metoda obsahuje zbytečně velké množství neužitečných dat o obrázku samotném například informace o tvaru, grafických prvcích. Tato data se mohou lišit dle jednotlivých grafických nástrojů a přitom reprezentují stejnou situaci. Nicméně vizuální podobu grafu je možné použít u speciální grafové problematiky, jako jsou například planární grafy či propojování prvků na masce integrovaného obvodu.[20]

První možností reprezentace grafu je reprezentace maticová. Její výhoda spočívá v propojení teorie grafů a lineární algebry. Druhou variantou je zobrazení spojové, je to reprezentace výhodnější a to díky menší asymptotické paměťové složitosti a především díky menším časovým nárokům, potřebným k dosažení řešení dané úlohy.[21]

[19] Turzík, s. 44
[20] Kolář, s. 65
[21] Kolář, s. 65

4.2 Prohledávání grafů

4.2.1 Prohledávání do šířky

Jedním z nejjednodušších a zároveň nejpotřebnějších algoritmů je prohledávání grafu. Základní variantou je prohledávání do šířky. Z jeho myšlenky vycházejí další algoritmy jako například Djikstrův algoritmus hledání nejkratší cesty, či Jarníkův-Primův algoritmus hledání minimální kostry grafu.

Při zadaném grafu G=<H,U> a vyznačeném uzlu s, jsou prohledávány do šířky hrany vedoucí ke každému dosažitelnému uzlu z uzlu s. „Při prohledávání je zároveň získána vzdálenost od uzlu s ke každému dosažitelnému uzlu a jako vedlejší produkt je získán strom prohledávání do šířky obsahující všechny s dosažitelné uzly. Tento strom je označován jako BF-strom. Pro každý uzel tohoto stromu je cesta z kořene s do uzlu v zároveň nejkratší cestou z s, do v, v grafu G. BF-strom je tedy současně stromem nejkratších cest z uzlu s do všech dosažitelných uzlů.“[22]

4.2.2 Prohledávání do hloubky

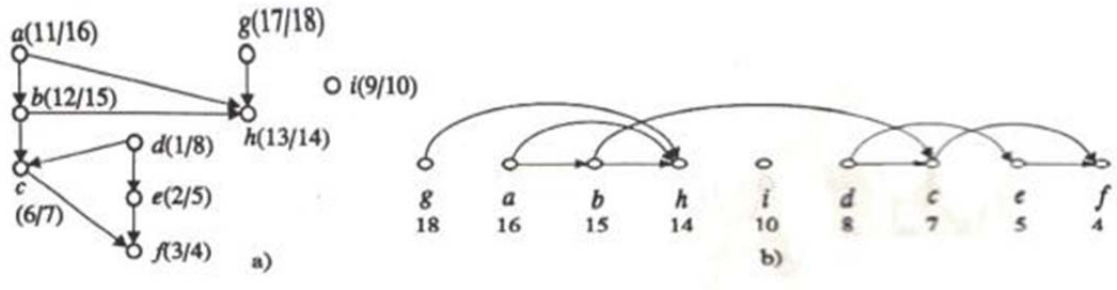
Dalším základním algoritmem je prohledávání do hloubky. Algoritmus probírá hrany vycházející z posledně nalezeného uzlu v, který má ještě nějaké neprobrané hrany. Když probere všechny jeho hrany, vrátí se zpátky k uzlu, z něhož se do uzlu *v* dostal, a z něho pokračuje po další dosud neprobrané hraně. Takovým způsobem postupuje až do objevení všech uzlů dosažitelných z výchozího uzlu. Pokud existuje další neobjevený uzel, je určen jako uzel výchozí a prohledávání do hloubky pokračuje z něho[23]

4.3 Topologické uspořádání

Vedlejším produktem prohledávání grafu do hloubky je topologické uspořádání. „Topologickým uspořádáním se rozumí takové uspořádání uzlů, v němž je pro každou hranu (u,v) uzel *u* před uzlem *v*. Takové uspořádání lze zavést pouze pro acyklické grafy,

[22] Kolář, s. 74
[23] Kolář, s. 79

neboť libovolný cyklus (včetně smyčky) by takové uspořádání znemožnil.“[24]



Obrázek 3 Topologické uspořádání [25]

4.4 Vzdálenosti v orientovaných a ohodnocených grafech

Pro určování vzdáleností v grafech je třeba stanovit jejich ohodnocení. Jako kritérium, podle kterého vzdálenosti stanovujeme, nemusí sloužit pouze délky na skutečných křivkách. Mohou to být i údaje o času, náročnosti, ztrátách, či jakákoliv jiná kvantifikovatelná veličina. Ve většině případů, jsou hodnoty kladné, ale je možné se setkat i s hodnotami zápornými. Přítomnost záporných hodnot je jedním z faktorů určujících vhodný algoritmus pro výpočet vzdálenosti. [26]

4.5 Acyklické grafy a digrafy

„Orientovaný graf G je acyklický právě tehdy, je-li acyklický každý jeho podgraf $G - \{u\}$, kde u je libovolný kořen nebo list grafu G“ [27]

Pro acyklický digraf dále platí:[28]

- Digraf $G = (V,H)$ je orientovaný strom
- V digrafu $G = (V,H)$ existuje pro každé $u,v \in V$, $u \neq v$ jediná u-v polocesta
- Digraf G = (V,H) je neorientovaně souvislý a každá orientovaná hrana množiny H je mostem (Mostem v orientovaném digrafu rozumíme takovou orientovanou hranu, po jejímž vyjmutí stoupne počet komponent digrafu)
- Digraf G = (V,H) je neorientovaně souvislý a $|H| = |V| - 1$
- V digrafu $G = (V,H)$ platí $|H|=|V|-1$ a G neobsahuje polocyklus

[24] Kolář, s. 84
[25] Kolář, s. 122
[26] Kolář, s. 111. - 112.
[27] Kolář, s. 85
[28] www.studentmatematiky.own.cz/g/grkap5-nopics.pdf

Orientované acyklické digrafy jsou v anglické terminologii označovány jako DAG (directed acyclic graph).

4.6 Kritická cesta

Metodu kritické cesty je využívána při řízení projektů složených z více dílčích činností. Průběh projektu je zadán acyklickým digrafem, kde vrcholy grafů určují délku trvání činnosti a hrany určují závislosti mezi jednotlivými činnostmi.
Pro zjištění kritické cesty slouží velké množství algoritmů, jako jsou Dijkstrův, Bellman-Fordův. Tyto algoritmy jsou poměrně složité kvůli operacím se zápornými cykly. Tyto cykly odpadají u acyklických grafů, kde můžeme využít jejich topologického uspořádání. „Pokud existuje orientovaná cesta z uzlu *u* do uzlu *v,* pak se uzel *u* nachází před uzlem *v* v topologickém uspořádání. To znamená, že stačí procházet uzly v tomto pořadí a pro každý uzel relaxovat všechny z něj vycházející hrany“[29]

[29] Kolář, s. 118-121

5 Central Data Store v ČSOB

Obsah této kapitoly byl čerpán z Provozní příručky aplikace Process Control Center, z prezentace Central Data Store (CDS) - Information Factory a z informacích poskytnutých spolupracovníky z útvaru CDS, zejména od RNDr. Josefa Hlavicy.

5.1 Základní informace o CDS

Československá obchodní banka, a. s. je se svými více než třemi miliony klientů jedním z největších hráčů na českém finančním trhu, z čehož plyne nutnost implementace pokročilého řešení Business intelligence, jejímž základem je Central Data Store (CDS) Information Factory.

CDS není pouze datovým skladem, ale jak bylo již zmíněno, vysoce automatizovanou Information Factory. CDS se stará o sběr dat ze zdrojových systémů, jejich transformaci, konsolidaci a uložení. Dále vytváří reporty, distribuuje data pro uživatele, datamarty či další aplikace.

Hlavními komponentami CDS jsou:

- PCC- Process Control Centrum, systém pro řízení a monitoring
- Teradata – SQL databáze pro uložení dat v CDS
- Informatica – ETL nástroj
- MicroStrategy – OLAP nástroj
- Business Objects – nový OLAP nástroj
- DPM – Distribuční portál MIS, nástroj pro tvorbu a distribuci reportů z CDS vytvořených SQL příkazy.

Podpůrnými komponentami CDS jsou:

- Power Designer – nástroj pro tvorbu datamodelů
- Metadata Manager – nástroj ze systému Informatica pro správu metadat
- DSA – Development Support Application – podpůrný prostředek pro ETL správce a vývojáře - archivace datových struktur, generování tabulek atd.
- Documentation – nástroj pro zpracování a distribuci CDS dokumentace.

Sdílenými komponentami dle standardů ČSOB jsou:

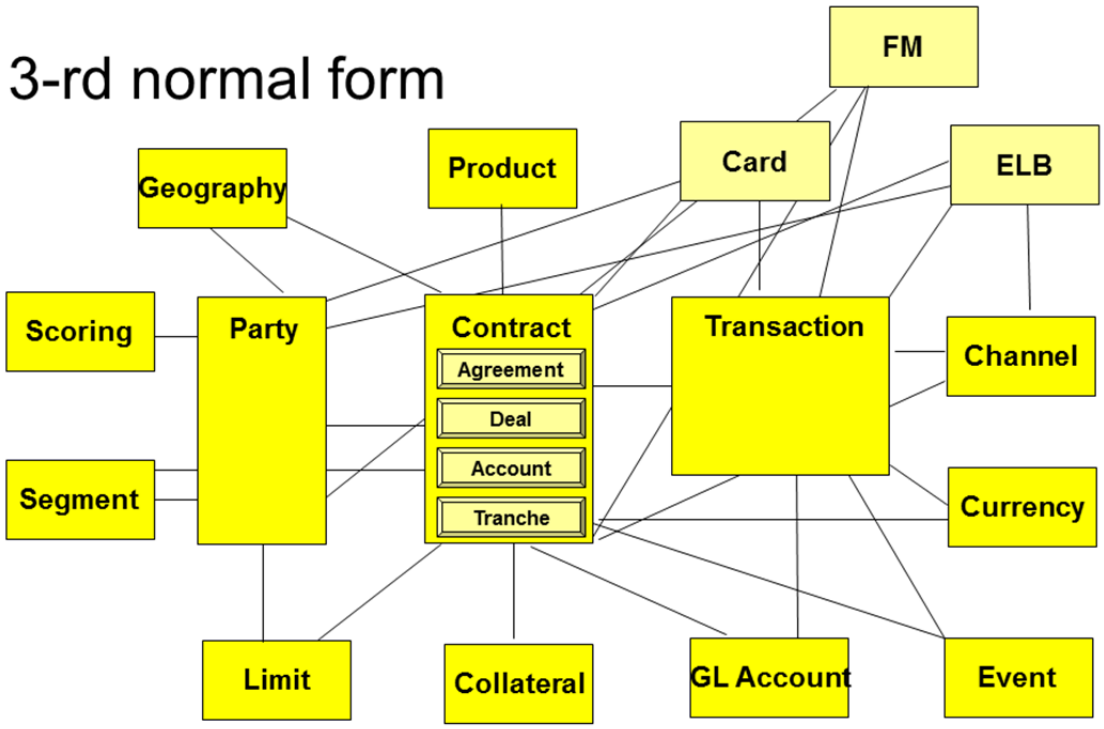
- ITIM – role manager, systém pro správu přístupových oprávnění
- NETCOOL – systém pro hlášení operačních zpráv
- Oracle – SQL databáze pro metadata Business objektů

5.1.1 Sklad

Sklad samotný archivuje data z 77 zdrojových systémů a z více než 2000 zdrojových tabulek a souborů. Počet cílových tabulek je přes 400 a zabírají více než 4 TB místa na discích. Dále sklad obsahuje 40 datamartů, které mají přes 6500 tabulek a 3,7 TB místa na discích, přičemž všechny údaje neustále narůstají.
Počet ETL jobů přesahuje 20 000, počet uživatelů CDS je 200. CDS produkuje přes 3000 statických reportů a přes 12 000 Microstrategy reportů pro 1300 uživatelů.

5.1.1.1 Datový model

Datový model jádra CDS – targetu je ve třetí normálové formě a vychází ze standardního modelu pro finanční informatiku. Obsahuje základní entity Party – Klient, Contract – Účet, Transaction – Transakce. Dalšími entitami jsou entity typu Segment, rozlišující korporátní či privátní bankovnictví, Scoring, určující hodnocení klienta, či Currency – Měna, uvádějící měnu, ve které je vedena transakce a účet, vedení informací o měně je u klienta zbytečné. Entitou společnou pro účet, klienta i transakce je například Card, určující číslo karty, která byla použita pro transakci.

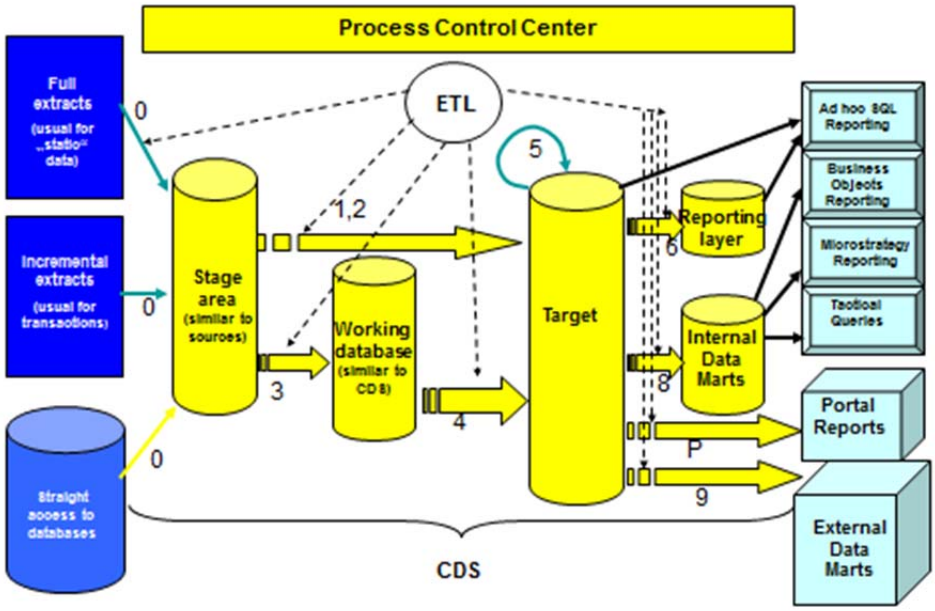


Obrázek 4 Datový model CDS[30]

[30] ČSOB, CDS s.12

5.2 PCC

Pro provedení ETL procesu používá CDS více systémů. Pro extrakci a transformaci dat se používá systém Informatica. Pro uložení dat nástroje systému Teradata (FastLoad, MultiLoad). K řízení a monitoringu běhu procesů ETL slouží nástroj PCC.



Obrázek 5 Fáze ETL v CDS [31]

5.2.1 Fáze PCC

- 0 – načítání extraktů nebo tabulek v databázích, kontrola, nahrání do Stage area
- 1 – vytvoření operačních klíčů v Target
- 2 – transformace a uložení dat do Target (zůstatky, transakce)
- 3 – transformace a uložení dat do Working (pomalu se měnící data)
- 4 – konsolidace historie v Target
- 5 – transformace výstupů ETL
- 6 – vytvoření reportovací vrstvy
- 8 – nahrání do datamartů
- 9 – přímý zápis dat do externích databází
- P – pravidelný reporting
- S – statistika systému Teradata – pro optimalizaci výkonu

[31] ČSOB, CDS, s. 19.

5.2.2 Kontrola běhu PCC

PCC je základní řídící aplikací CDS. Každá úloha ETL má svůj řádek v tabulce CTRL_JOB_ALL s jejím názvem a tzv. command_line obsahujícím informace jak úlohu v systému odstartovat. Dále obsahuje pomocné parametry. Jméno úlohy je tvořeno ve formátu pSsssTtttAA podle CDS ETL metodologie: p = fáze ETL, sss = číslo systému, tttt = číslo tabulky v datovém modelu, AA = zkratka facety (oblasti) v datovém modelu.
V dalších řídících tabulkách PCC jsou uloženy důležité parametry pro řízení chodu úkolu:

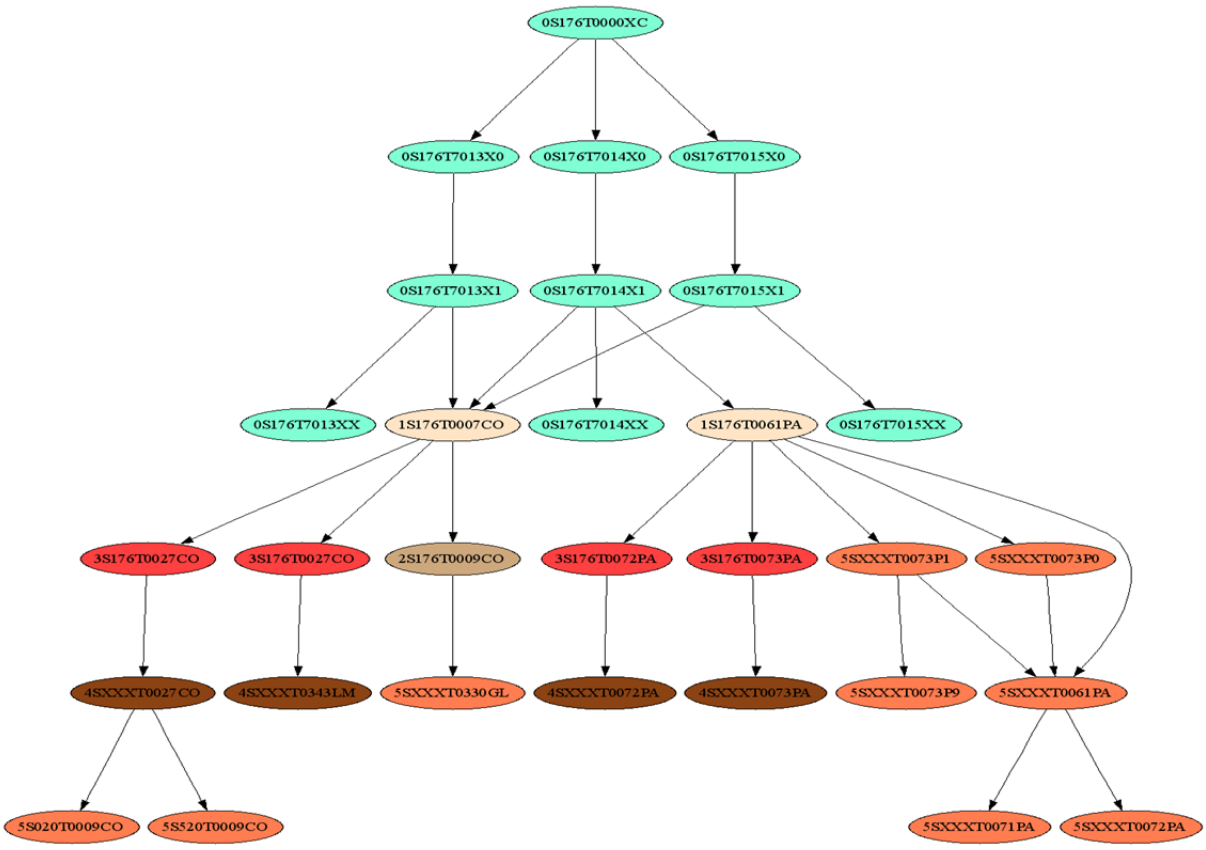
- Kalendáře – obsahují informace o dnech, týdnech, kdy v měsíci atd. se mají úlohy zpracovávat
- Závislosti úloh – obsahuje vztahy mezi úlohami a určuje, které musí být zpracovány, aby následující úlohy mohly proběhnout.

EOD. Každodenní zpracování CDS se nazývá EOD – end of the day. Zahrnuje systémové procesy, jako jsou zálohy či archivace, ETL a výstupy. Výstupy mohou být plnění datamartů, či tvorba reportů.

Obrázek 6 Schéma zpracování EOD.[32]

PCC během startu CDS EOD zvolí úlohy, které budou k danému datu probíhat. Vybrané úlohy a jejich závislosti je možné zobrazit formou orientovaného stromového grafu. Přiložený graf ilustruje strukturu úloh, které poběží v daný den. Graf ukazuje pouze výsek celkového denního zpracování, konkrétně start. Ve skutečnosti je graf daleko více rozvětven a končí opět jednou úlohou.
Barvy v následujícím stromovém grafu ilustrují fáze zpracování:
Zelená – fáze 0
Žlutá – fáze 1
Okrová – fáze 2
Růžová – fáze 3
Hnědá – fáze 4
Oranžová – fáze 5

[32] ČSOB, CDS s.44



Obrázek 7 Příklad části stromového grafu úloh.[33]

Podle priorit a závislostí jednotlivých úloh engine vybírá ty, které mají splněny všechny podmínky pro svůj start. PCC obsahuje systém statistik pro jednotlivé úlohy, které jsou využívány pro budoucí stanovení pořadí běhu úloh. Zároveň obsahuje systém pro dynamické přepočítání priorit k dosáhnutí nejlepších časů. Supervizoři CDS mají možnost zasahovat do běhu úloh a jejich úkony jsou archivovány.

Za normálních okolností je pořadí běhu určeno tímto enginem a operace mohou probíhat zcela automaticky. Problém nastává, je-li třeba dokončit nějakou úlohy prioritně.

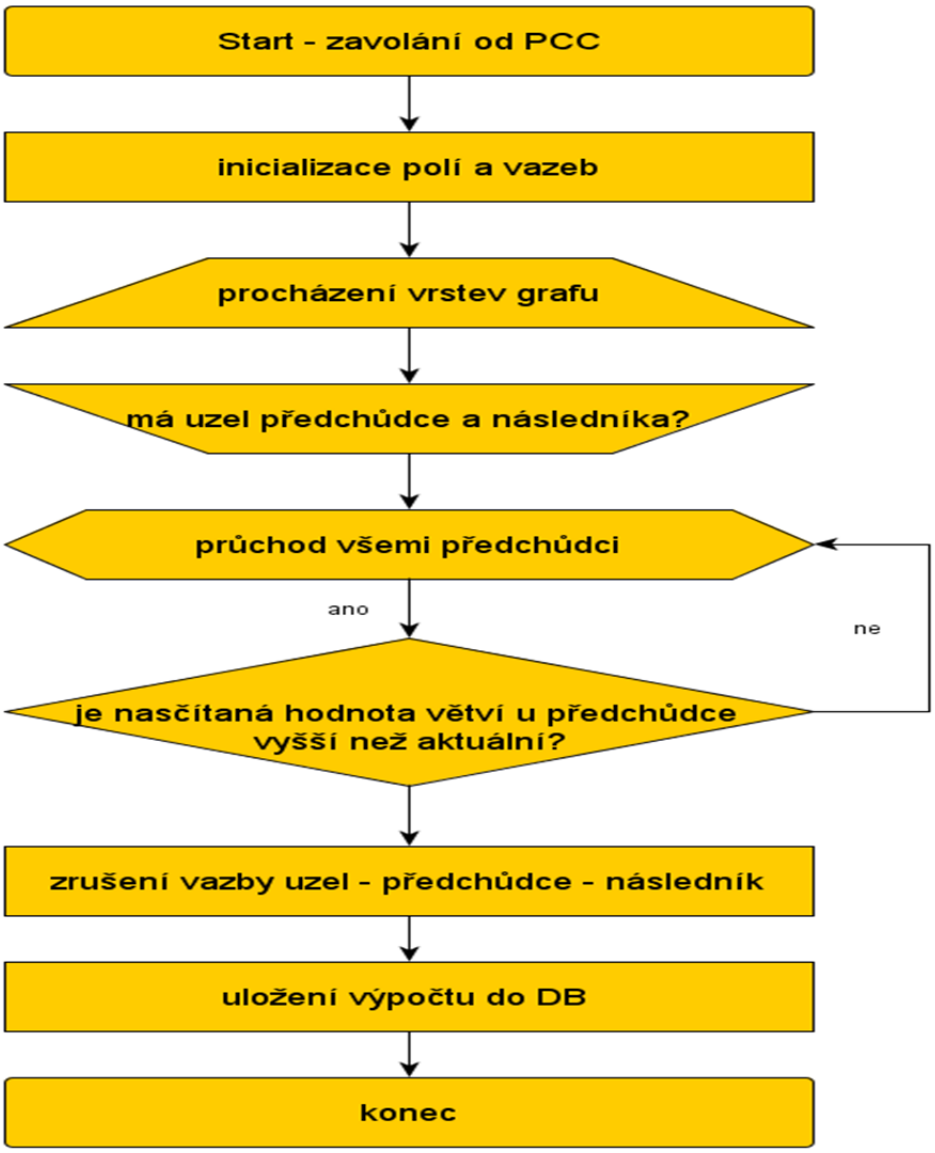
[33] ČSOB CDS, s.24

6 Praktická část

Obsah této kapitoly byl čerpán z Provozní příručky aplikace Process Control Center, z prezentace Central Data Store (CDS) - Information Factory a z informacích poskytnutých spolupracovníky z útvaru CDS, zejména od RNDr. Josefa Hlavicy. Původní skript byl vytvořen Ing. Tomášem Naxerou.

6.1 Priorita v PCC

Priorita každé úlohy se v PCC počítá na základě statistik z předchozích zpracování. Tyto statistiky jsou uvedeny v SQL tabulce SESS_JOBS_STATISTICS. Algoritmus se spouští po dobu celého EOD s periodicitou 15 minut.
Algoritmus počítá priority úloh na základě předpokládané době trvání, kterou poskytuje tabulka s údaji o trvání běhů minulých. Výsledkem je ohodnocení všech vrcholů a upřednostnění cesty, která je časově nejdelší. Je tedy upřednostněna kritická cesta.
Algoritmus funguje na principu grafového procházení do hloubky. Celá procedura spočívá v načtení údajů ze zdrojových SQL tabulek do formátu pro zpracování algoritmem, jeho běhu a uložení výsledků.
Při svém běhu, algoritmus prochází každý uzel, který nemá žádného rodiče a jeho hodnota je přičtena k jeho dětem. Takto získáme ohodnocené vrcholy pro běh zpracování.



Obrázek 8 Algoritmus plánování úloh EOD

6.2 SQL část

Parametry pro běh PCC jsou uloženy v SQL tabulkách. V následující části jsou popsány jen ty tabulky, které jsou využívány skriptem pro výpočet priority.

```
-- CTRL_JOB_ALL
CREATE TABLE [dbo].[CTRL_JOB_ALL]( -- CTRL_JOB_ALL
CREATE TABLE [dbo].[CTRL_JOB_ALL](
	[job_name_id] [char](12) NOT NULL,
	[priority] [int] NULL,
	[cmd_line] [varchar](1024) NOT NULL,
	[cont_anyway] [char](1) NOT NULL,
	[max_runs] [smallint] NOT NULL,
	[always_restart] [char](1) NOT NULL,
	[status_begin] [smallint] NULL,
	[waiting_hr] [smallint] NULL,
	[engine_id] [smallint] NULL,
	[job_desc] [varchar](255) NULL,
	[author] [varchar](50) NULL,
	[deadline_hr] [int] NULL,
	[note] [varchar](2000) NULL,
 CONSTRAINT [PK_CTRL_JOB_ALL] PRIMARY KEY CLUSTERED
(
	[job_name_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON, FILLFACTOR = 90) ON [PRIMARY]
) ON [PRIMARY]
```

Základní tabulkou s údajemi o všech úlohách je CTRL_JOB_ALL. Obsahuje následující instance:

job_name_id - identifikátor úlohy

cmd_line – popis startu úlohy

priority – priorita úlohy

job_desc - stručný popis k úloze, co dělá a kdo je autor

waiting_hr – slouží pro zpoždění spuštění úlohy. Počítá se od 00:00h dne, za který probíhá zpracování

deadline_hr – udává požadovaný čas dokončení zpracování úlohy

```
--SESS_JOB_ALL
CREATE TABLE [dbo].[SESS_JOB_ALL](
	[job_name_id] [char](12) NOT NULL,
	[status] [smallint] NOT NULL,
	[last_update] [datetime] NULL,
	[business_dt] [datetime] NOT NULL,
	[priority] [int] NULL,
	[cmd_line] [varchar](1024) NOT NULL,
	[cont_anyway] [char](1) NOT NULL,
	[res_ind_id] [smallint] NOT NULL,
	[n_run] [smallint] NOT NULL,
	[max_runs] [smallint] NOT NULL,
	[always_restart] [char](1) NOT NULL,
	[waiting_hr] [smallint] NULL,
	[application_id] [smallint] NULL,
	[engine_id] [smallint] NULL,
 CONSTRAINT [PK_SESS_JOB_ALL] PRIMARY KEY CLUSTERED
(
	[job_name_id] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON, FILLFACTOR = 90) ON [PRIMARY]
) ON [PRIMARY]
```

Tabulka SESS_JOB_ALL je inicializována při startu každodenního zpracování CDS. Postupně je plněna daty, která jsou upravována inicializačními procedurami. Až když je tabulka vyplněna, může začít samotné zpracování CDS.

job_name_id – identifikátor úlohy

status – udává stav úlohy. Při inicializaci se nastavuje 0 – úloha bude zpracována, nebo 20 – úloha bude přeskočena. Konečných statusů je mnoho, mezi nejdůležitější patří např:

2 – úspěšný konec po prvním běhu

5 – úspěšný konec při opakovaném běhu

6 – selhání úlohy, ale je pokračováno dále ve zpracování, jako by úloha odjela dobře

business_dt – datum ke kterému probíhá zpracování úlohy

priority – priorita vypočtená algoritmem

waiting_hr – hodina, ke které je nejdříve možné začít zpracovávat úlohu

```
--SESS_JOB_DEPENDENCY
CREATE TABLE [dbo].[SESS_JOB_DEPENDENCY](
	[job_name_id] [char](12) NOT NULL,
	[parent_job_name_id] [char](12) NOT NULL
) ON [PRIMARY]
```

Tabulka SESS_JOB_DEPENDENCY obsahuje pouze dvě instance, ale jedná se o jedny z nejdůležitějších, neboť odkazují na id rodičů a dětí jednotlivých úloh.

job_name_id – id úlohy

parent_job_name_id – obsahuje id všech aktuálně nastavených rodičovských úloh

Při skončení úlohy jsou z tabulky vymazány všechny řádky, kde je job_name_id úlohy ve sloupci parent_job_name_id. Úloha s job_name_id se může rozjet teprve, když jsou dokončeny všechny k ní příslušné rodičovské, tedy v tabulce SESS_JOB_DEPENDENCY neexistuje ani jeden řádek, kde by se ve sloupci job_name_id vyskytovalo job_name_id úlohy.

```
--SESS_JOB_STATISTICS
CREATE TABLE [dbo].[SESS_JOB_STATISTICS](
	[job_name_id] [char](12) NOT NULL,
	[business_dt] [datetime] NULL,
	[first_start_ts] [datetime] NULL,
	[last_start_ts] [datetime] NULL,
	[last_status_ts] [datetime] NULL,
	[end_ts] [datetime] NULL,
	[last_status] [smallint] NULL,
	[n_restart] [smallint] NULL,
	[avg_previous_duration] [int] NULL,
	[avg_previous_end_tm] [int] NULL,
	[ignore_stat] [smallint] NULL
) ON [PRIMARY]
```

Tabulka SESS_JOB_STATISTICS obsahuje důležité údaje, které jsou pro algoritmus klíčové.

avg_previous_duration – délka zpracování úlohy při minulém běhu systému

avg_previous_end_tm –čas dokončení při minulém běhu

6.3 PERL skript

Skript sloužící k výpočtu priorit úloh se nazývá SetPriority.pl. Je napsán v jazyce PERL, který kombinuje snadný přístup k databázím s klasickými nástroji jazyků typu C. Samotný princip algoritmu pro výpočet priorit je již uveden výše. Zde je algoritmus uveden do širšího kontextu celého skriptu.

Prvním krokem skriptu je definice proměnných. Dále jsou formátovány výstupy logovacího souboru, jedná se zde o sjednocení formátů dat a časů, které jsou porovnávány s lokálním časem SQL.

Druhým krokem je definice SQL příkazů. Jsou zde upravovány parametry pro spouštění s nepravidelným během.

Třetím krokem je načtení záznamů z databáze a jejich uspořádání do definované struktury. Tato část skriptu připravuje data do formy vhodné pro zpracování algoritmem. Dále se nastavují časové limity zpracování, dle požadovaného času ukončení, určeného hodnotou LimitTm.

6.3.1 Skript pro algoritmus

Čtvrtý krok je výpočet algoritmu samotného. Nejdříve skript prochází strom odspoda vzhůru a počítá délku větví z důvodu hlídání limitu. Dalším krokem je for cyklus, který prochází vytvořenou strukturu, dle následujícího klíče: Pro každou úroveň odpovídající hodnotě pomocné proměnné i, je spuštěn cyklus foreach, kontrolující, zda má nějaká úloha svého následovníka a nemá předchůdce. Pokud ano, je zjištěna délka větve, na které se úloha nachází. Následuje podmínka if, sčítající délku následníka a současnou délku větve. Pokud je tento součet větší než současná délka větve, je hodnota zvětšena o délku následníka.

Následně je zrušena zkoumaná vazba následník – předchůdce a předchůdce – následník. Dalším krokem jsou výpočty odhadu konce zpracování. Je vypočtena nejdelší větev. Je třeba mít na paměti, že tento skript běží neustále až do konce zpracování všech úloh. Proto je zde nutné zohledňovat i již hotové úlohy, které jsou od jednotlivých větví odečteny. Zároveň jsou zjišťována překročení stanovených limitů, což slouží k úpravě priorit.

Výsledkem těchto kroků skriptu jsou výpisy: Odhadovaného konce dle nejdelší větve, první úlohy na nejdelší větvi, maximální délka větve, minimální délka větve, maximální překročení limitu, minimální překročení limitu, nejhlubší úroveň závislosti – nejhlubší vnoření.

Závěrečným úkonem skriptu je export jeho výsledků do databáze a do PCC.

6.4 Upravený skript pro testovací účely

Jelikož úprava celého výše uvedeného skriptu by byla neúměrně složitá a pro účely této práce zbytečná, je samotný skript sloužící k testování ořezán a upraven pouze k účelu ilustrování dané situace.

První úpravou je definice zdrojových dat. Místo SQL tabulek jsou definována pole:

hash_strom – asociativní pole sloužící k uspořádání uzlů do stromové struktury

hash_vaha – asociativní pole přiřazující váhy jednotlivým uzlům

hash_priorita – asociativní pole sloužící k přiřazení priority pro daný uzel

vazby – pole definující vazby předek – potomek

Prvním krokem je definování stromové struktury hash_strom. Nejdříve je nastavena váha pro každý uzel a proměnná pro budoucí výpočet délky větví. Následuje přiřazení priority – prvek výrazně modifikující předchozí algoritmus. Na závěr jsou nastaveny odkazy na předchůdce a následníky.

Výsledek těchto kroků je pro lepší přehlednost vytištěn na obrazovku terminálu v podobě vytvořené datové struktury. Jednotlivé řádky výpisu popisují vlastnosti uzlů. První sloupec obsahuje číslo uzlu rodiče, druhý číslo uzlu, třetí váhu a čtvrtý číslo dítěte, na které odkazuje.

```
Výpis vazeb a vah uzlů:
                    Uzel 1   vaha:29   2:C-->
                    Uzel 1   vaha:29   3:C-->
<--P:1              Uzel 2   vaha:20
                    Uzel 2   vaha:20   4:C-->
                    Uzel 2   vaha:20   5:C-->
<--P:1              Uzel 3   vaha:25
                    Uzel 3   vaha:25   5:C-->
                    Uzel 3   vaha:25   6:C-->
<--P:2              Uzel 4   vaha:12
                    Uzel 4   vaha:12   7:C-->
<--P:2              Uzel 5   vaha:21
<--P:3              Uzel 5   vaha:21
                    Uzel 5   vaha:21   7:C-->
                    Uzel 5   vaha:21   8:C-->
<--P:3              Uzel 6   vaha:18
                    Uzel 6   vaha:18   8:C-->
<--P:4              Uzel 7   vaha:23
<--P:5              Uzel 7   vaha:23
                   Uzel 7   vaha:23   9:C-->
<--P:5              Uzel 8   vaha:15
<--P:6              Uzel 8   vaha:15
                   Uzel 8   vaha:15   9:C-->
<--P:7              Uzel 9   vaha:19
<--P:8              Uzel 9   vaha:19
```

Po rozřazení do struktury v asociativním poli hash_strom následuje výpočet algoritmu. Prvním krokem je for cyklus, který prochází strukturu po jednotlivých patrech. Uvnitř tohoto cyklu běží cyklus foreach, kontrolující, zda daný uzel má uzel rodičovský a nemá uzel dětský. Tento krok zajišťuje, že je struktura procházena od finální úlohy k začátku. Pokud má úloha nějaké předchůdce, algoritmus všechny projde a vypočte nejhlubší vnoření tj. počet vrstev struktury.

V dalším kroku je počítána výsledná váha uzlu. Je proveden součet váhy rodiče, uzlu a priority. Pokud je tento součet větší než váha rodičovského uzlu, je nastaven jako nová nejvyšší hodnota délky.

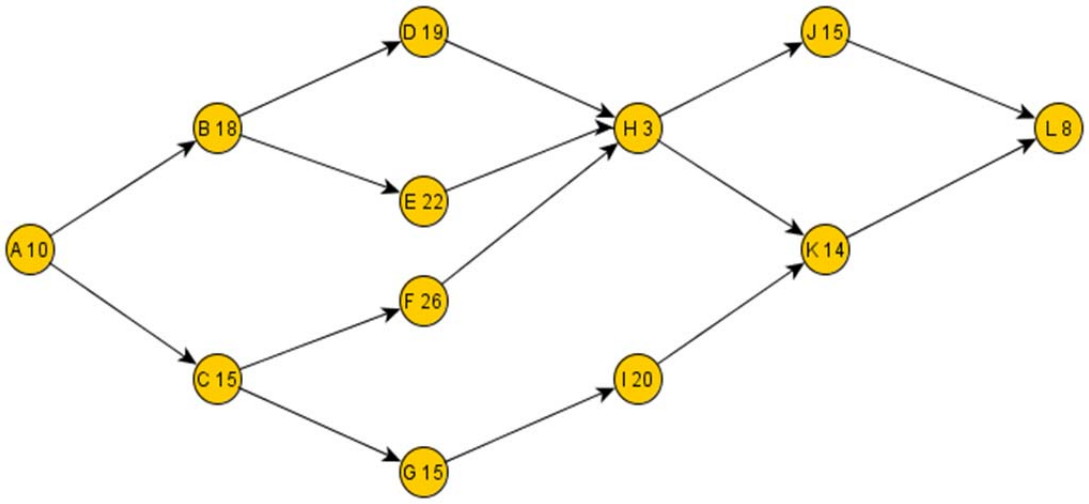
Po provedení součtů je vymazána vazba předchůdce – následník a zároveň následník – předchůdce. Dále jsou uzly seřazeny dle délky a vytištěny do souboru. Z tohoto souboru je možné vyčíst jak použitou prioritu, tak její vliv na změnu pořadí uzlů.

Finálním výstupem je tabulka, kde je uvedeno pořadí uzlu, jeho název a váha, délka cesty k uzlu, priorita nastavená u uzlu a na závěr samotná suma, určující dobu zpracování.

```
Vypis uzlu s nascitanymi delkami - setrideno dle delky
1. Uzel a          vaha:10          delka:82          priorita:0          suma:10
```

6.5 Výsledky

Graf, na kterém je prováděn pokus má 12 uzlů a 15 hran. Cílem experimentu je zvýhodnění jednoho uzlu o prioritu a sledování změny času, který je zapotřebí pro zpracování daného uzlu.



Obrázek 9 Ilustrační graf úloh

Jak je uvedeno v ukázkovém výstupu zpracovaném výše, výchozí délka kritické větve bez započítání priorit je 82 a kritickým uzlem je uzel A. Suma času, potřebná k zpracování celého algoritmu zůstává neměnná během celého experimentu a činí 185. Měnit se bude pouze suma času sledovaného uzlu.

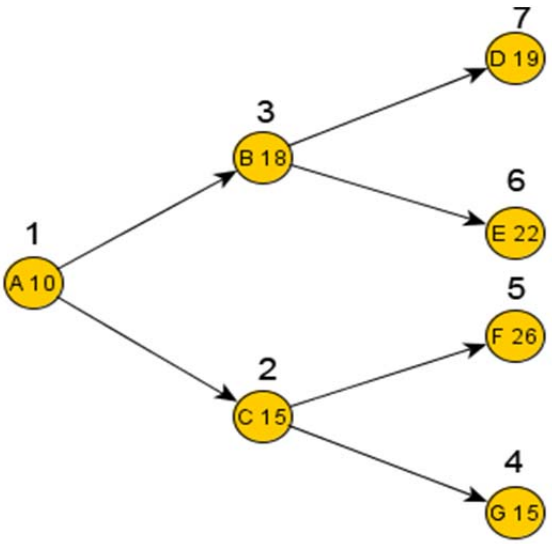
6.6 Testování 1

6.6.1 Výchozí stav

Prvním uzlem, který bude urychlen, je uzel D. Součet původní délky činil 82, proto bude přidávána hodnota odpovídající pěti procentům z prvotní délky tj. 4.

V základní formě je uzel číslo D zpracován jako sedmý, s časem 125.

Vypis uzlu s nascitanymi delkami - setrideno dle delky				
1. Uzel a	vaha:10	delka:82	priorita:0	suma:10
2. Uzel c	vaha:15	delka:72	priorita:0	suma:25
3. Uzel b	vaha:18	delka:66	priorita:0	suma:43
4. Uzel g	vaha:15	delka:57	priorita:0	suma:58
5. Uzel f	vaha:26	delka:52	priorita:0	suma:84
6. Uzel e	vaha:22	delka:48	priorita:0	suma:106
7. Uzel d	**vaha:19**	**delka:45**	**priorita:0**	**suma:125**
8. Uzel i	vaha:20	delka:42	priorita:0	suma:145
9. Uzel h	vaha:3	delka:26	priorita:0	suma:148
10. Uzel j	vaha:15	delka:23	priorita:0	suma:163
11. Uzel k	vaha:14	delka:22	priorita:0	suma:177
12. Uzel l	vaha:8	delka:8	priorita:0	suma:185

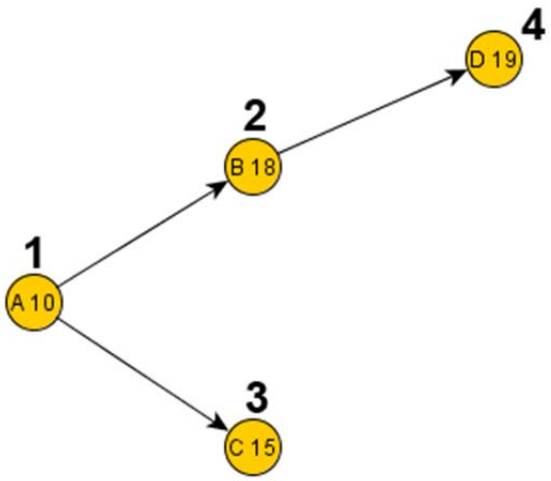


Obrázek 10 Stav pořadí zpracování při prioritě d = 0

6.6.2 Změna +12

Při navýšení hodnoty priority na 12, algoritmus zařadí uzel D pro zpracování na čtvrté místo s časem zpracování 62.

Vypis uzlu s nascitanymi delkami - setrideno dle delky				
1. Uzel a	vaha:10	delka:85	priorita:0	suma:10
2. Uzel b	vaha:18	delka:75	priorita:0	suma:28
3. Uzel c	vaha:15	delka:72	priorita:0	suma:43
4. Uzel d	**vaha:19**	**delka:57**	**priorita:12**	**suma:62**
5. Uzel g	vaha:15	delka:57	priorita:0	suma:77
6. Uzel f	vaha:26	delka:52	priorita:0	suma:103
7. Uzel e	vaha:22	delka:48	priorita:0	suma:125
8. Uzel i	vaha:20	delka:42	priorita:0	suma:145
9. Uzel h	vaha:3	delka:26	priorita:0	suma:148
10. Uzel j	vaha:15	delka:23	priorita:0	suma:163
11. Uzel k	vaha:14	delka:22	priorita:0	suma:177
12. Uzel l	vaha:8	delka:8	priorita:0	suma:185

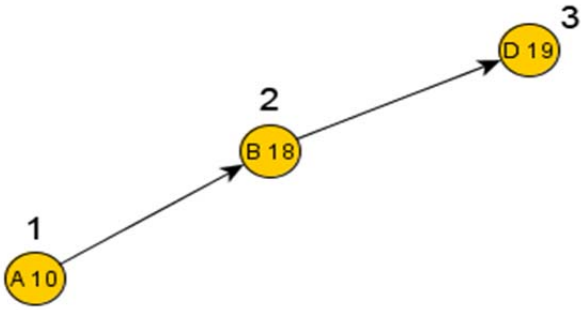


Obrázek 11 Stav pořadí zpracování při prioritě d = 12

6.6.3 Změna +28

Při navýšení priority na 28 zařadí algoritmus uzel D na třetí místo zpracování s časem 47.

Vypis uzlu s nascitanymi delkami - setrideno dle delky				
1. Uzel a	vaha:10	delka:101	priorita:0	suma:10
2. Uzel b	vaha:18	delka:91	priorita:0	suma:28
3. Uzel d	**vaha:19**	**delka:73**	**priorita:28**	**suma:47**
4. Uzel c	vaha:15	delka:72	priorita:0	suma:62
5. Uzel g	vaha:15	delka:57	priorita:0	suma:77
6. Uzel f	vaha:26	delka:52	priorita:0	suma:103
7. Uzel e	vaha:22	delka:48	priorita:0	suma:123
8. Uzel i	vaha:20	delka:42	priorita:0	suma:145
9. Uzel h	vaha:3	delka:26	priorita:0	suma:148
10. Uzel j	vaha:15	delka:23	priorita:0	suma:163
11. Uzel k	vaha:14	delka:22	priorita:0	suma:177
12. Uzel l	vaha:8	delka:8	priorita:0	suma:185



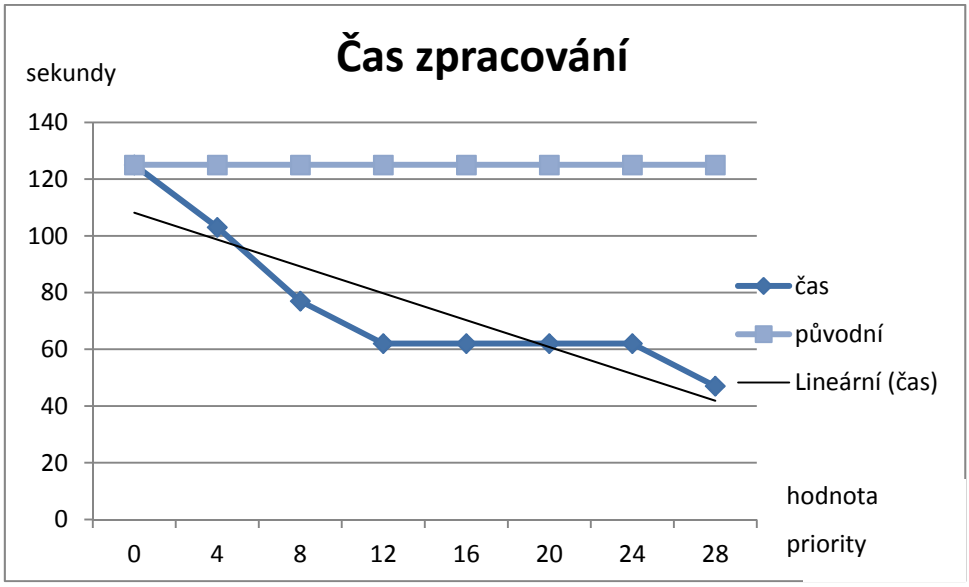
Obrázek 12 Stav pořadí zpracování při prioritě d = 28

Z provedených měření vyplývá, že pro posunutí ze sedmého místa zpracování, kterému odpovídá čas 125, na třetí místo, kterému odpovídá čas 47, je třeba prioritu navýšit o 28 tj. 34.15% původní délky.

Zároveň je nutné zvážit, zda nebude zpomalena úloha jiná. Z pohledu maximální optimalizace a neprodlužování zpracování jiných větví vychází nejlépe priorita 12, která dosáhne času zpracování 62 tj. více než polovina původního času.

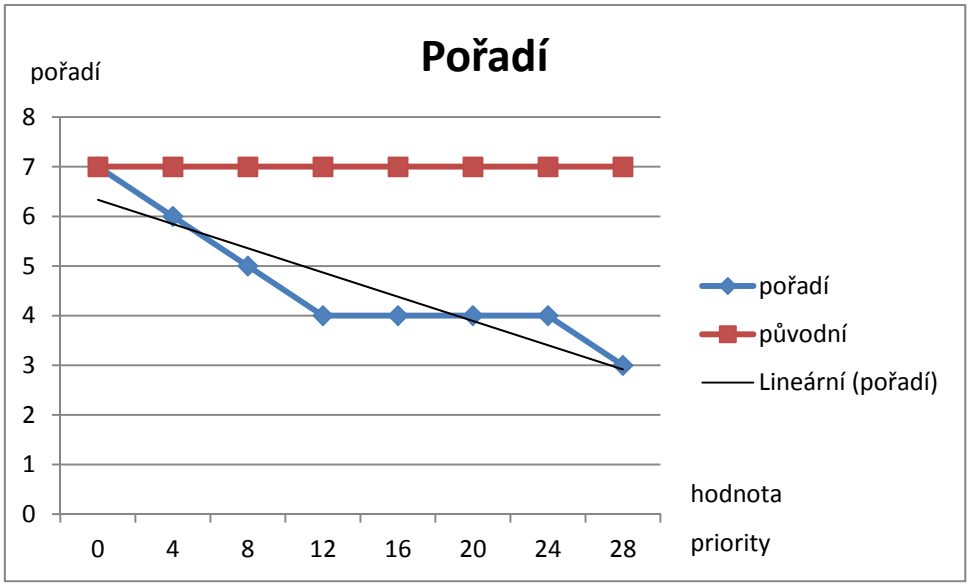
změna	změna %	pořadí	čas
0	0,00%	7	125
4	4,88%	6	103
8	9,76%	5	77
12	14,63%	5	62
16	19,51%	4	62
20	24,39%	4	62
24	29,27%	4	62
28	34,15%	3	47

Z grafu vyplývá, že nejstrmější vývoj času zpracování je při počátečním navyšování priority z hodnoty 0 na hodnotu 12. Dále je delší dobu přínos nulový a teprve změna z 24 na 28 přinese další zlepšení. Ze spojnice grafu je patrné, nejvyšší přínos má priorita 12, protože se bod jí náležící nachází nejníže pod lineární spojnicí.



Obrázek 13 Souhrn vývoje času zpracování v závislosti na hodnotě priority

Co se vývoje pořadí týče, křivka kopíruje hodnoty z času. Je zde též patrný strmý vývoj způsobený zvyšováním priority z hodnoty 0 na hodnotu 12. Dále se pořadí nezlepšuje až do nastavení priority na hodnotu 28, tehdy dojde k posunu uzlu d na třetí místo.



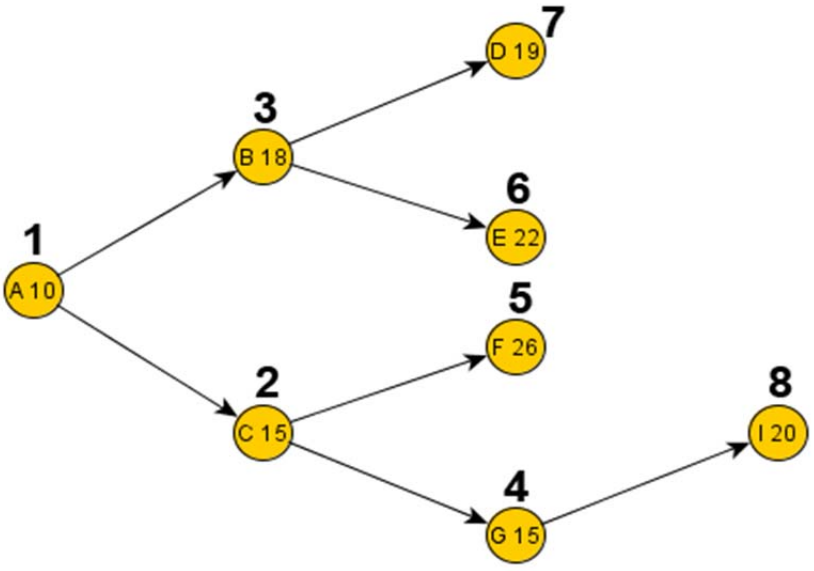
Obrázek 14 Souhrn vývoje pořadí zpracování v závislosti na hodnotě priority

6.7 Testování 2

6.7.1 Výchozí stav

Druhým uzlem který bude urychlen bude uzel I. Nachází se podobně jako uzel D v druhé polovině zpracování, kontrétně na osmém místě. Cílem je opět urychlit čas jeho zpracování. Pro srovnatelný výsledek bude opět hodnota priority činit 4.

Vypis uzlu s nascitanymi delkami - setrideno dle delky				
1. Uzel a	vaha:10	delka:82	priorita:0	suma:10
2. Uzel c	vaha:15	delka:72	priorita:0	suma:25
3. Uzel b	vaha:18	delka:66	priorita:0	suma:43
4. Uzel g	vaha:15	delka:57	priorita:0	suma:58
5. Uzel f	vaha:26	delka:52	priorita:0	suma:84
6. Uzel e	vaha:22	delka:48	priorita:0	suma:106
7. Uzel d	vaha:19	delka:45	priorita:0	suma:125
8. Uzel i	**vaha:20**	**delka:42**	**priorita:0**	**suma:145**
9. Uzel h	vaha:3	delka:26	priorita:0	suma:148
10. Uzel j	vaha:15	delka:23	priorita:0	suma:163
11. Uzel k	vaha:14	delka:22	priorita:0	suma:177
12. Uzel l	vaha:8	delka:8	priorita:0	suma:185

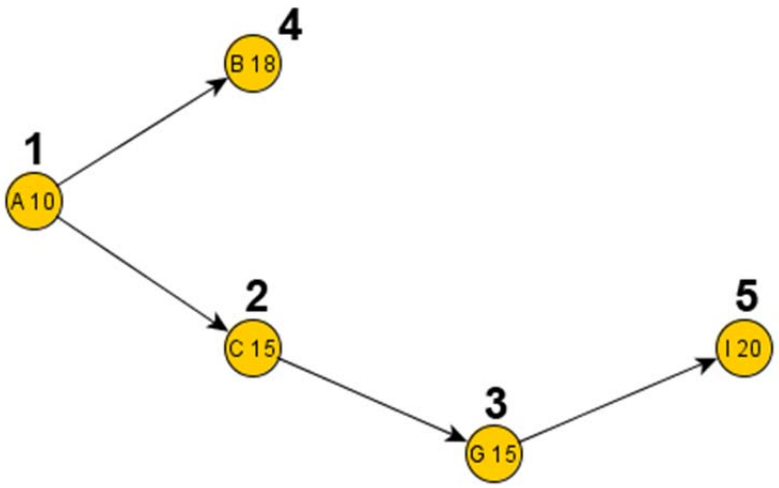


Obrázek 15 Stav pořadí zpracování při prioritě i = 0

6.7.2 Změna +12

Navýšení priority na hodnotu 12 přináší posun zpracování uzlu na páté místo s časem zpracování 78.

Vypis uzlu s nascitanymi delkami - setrideno dle delky				
1. Uzel a	vaha:10	delka:94	priorita:0	suma:10
2. Uzel c	vaha:15	delka:84	priorita:0	suma:25
3. Uzel g	vaha:15	delka:69	priorita:0	suma:40
4. Uzel b	vaha:18	delka:66	priorita:0	suma:66
5. Uzel i	**vaha:20**	**delka:54**	**priorita:12**	**suma:78**
6. Uzel f	vaha:26	delka:52	priorita:0	suma:104
7. Uzel e	vaha:22	delka:48	priorita:0	suma:126
8. Uzel d	vaha:19	delka:45	priorita:0	suma:145
9. Uzel h	vaha:3	delka:26	priorita:0	suma:148
10. Uzel j	vaha:15	delka:23	priorita:0	suma:163
11. Uzel k	vaha:14	delka:22	priorita:0	suma:177
12. Uzel l	vaha:8	delka:8	priorita:0	suma:185

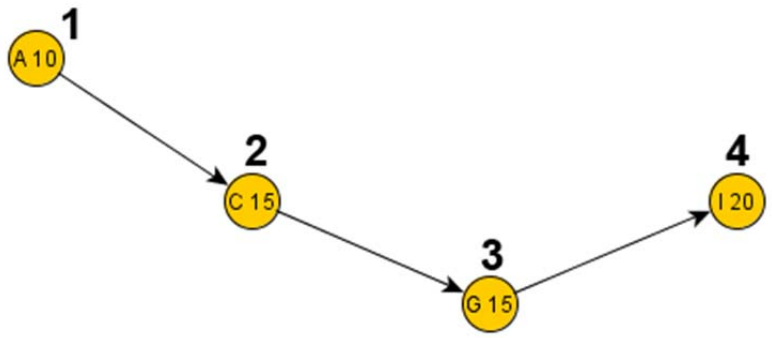


Obrázek 16 Stav pořadí zpracování při prioritě i = 12

6.7.3 Změna +28

Při navýšení priority na hodnotu 12 bude uzel I zpracován jako čtvrtý s časem 60. Z grafu je zřejmé, že lepšího výsledku není možné dosáhnout.

Vypis uzlu s nascitanymi delkami - setrideno dle delky				
1. Uzel a	vaha:10	delka:94	priorita:0	suma:10
2. Uzel c	vaha:15	delka:84	priorita:0	suma:25
3. Uzel g	vaha:15	delka:69	priorita:0	suma:40
4. Uzel i	**vaha:20**	**delka:70**	**priorita:28**	**suma:60**
5. Uzel b	vaha:18	delka:66	priorita:0	suma:78
6. Uzel f	vaha:26	delka:52	priorita:0	suma:104
7. Uzel e	vaha:22	delka:48	priorita:0	suma:126
8. Uzel d	vaha:19	delka:45	priorita:0	suma:145
9. Uzel h	vaha:3	delka:26	priorita:0	suma:148
10. Uzel j	vaha:15	delka:23	priorita:0	suma:163
11. Uzel k	vaha:14	delka:22	priorita:0	suma:177
12. Uzel l	vaha:8	delka:8	priorita:0	suma:185

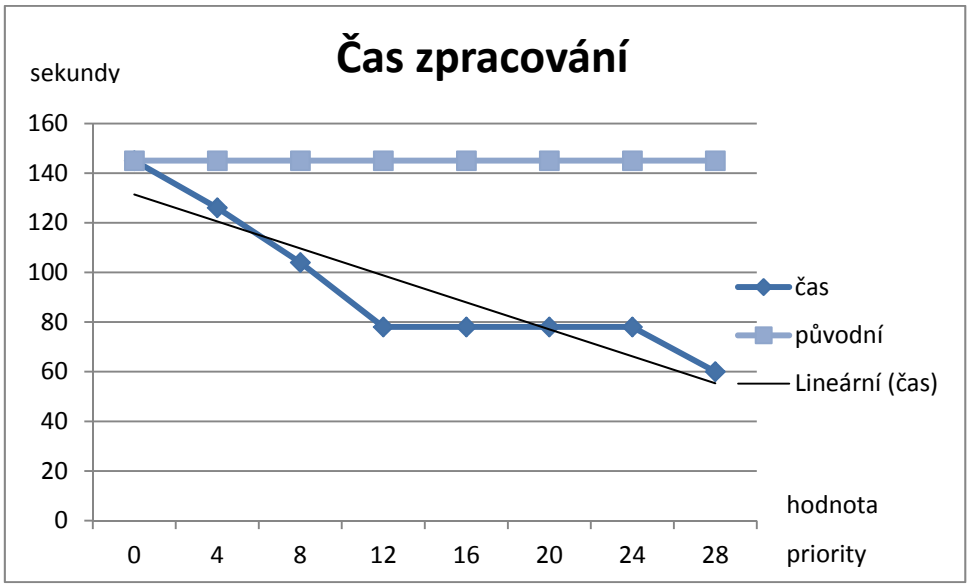


Obrázek 17 Stav pořadí zpracování při prioritě i = 28

Z provedených měření vyplývá, že nejrychleji může být uzel I zpracován s časem 60 a to v případě nastavení priority na hodnotu 28. Hodnota této priority tvoří 34.15% z hodnoty původní délky.

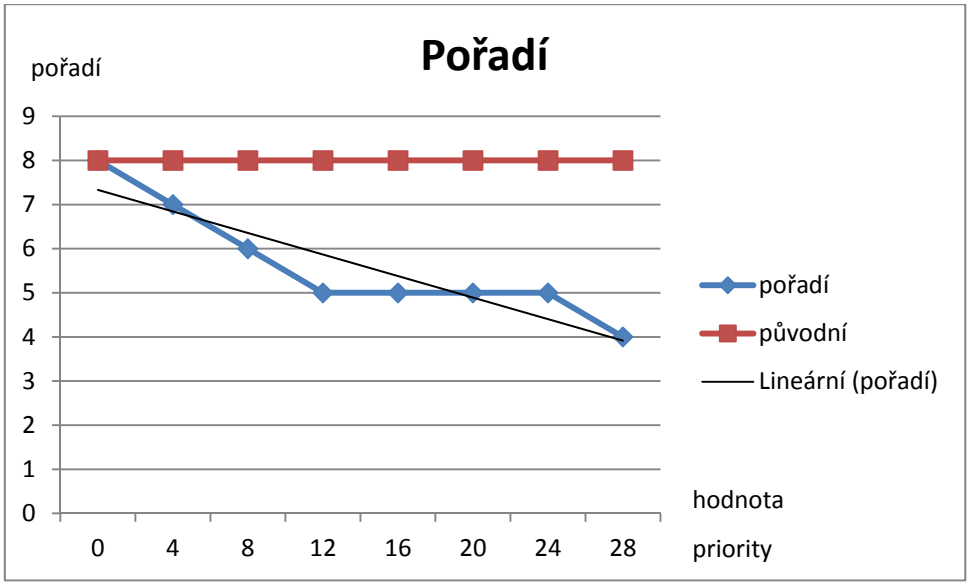
změna	změna %	pořadí	čas
0	0,00%	8	145
4	4,88%	7	126
8	9,76%	6	104
12	14,63%	5	78
16	19,51%	5	78
20	24,39%	5	78
24	29,27%	5	78
28	34,15%	4	60

Křivka času zpracování je opět jako v minulém pozorování na začátku strmě klesající do priority o hodnotě 12. Další zlepšení času zpracování přináší až zvýšení priority na hodnotu 28.



Obrázek 18 Vývoj času zpracování v závislosti na prioritě

Křivka pořadí zpracování koresponduje s křivkou času zpracování. Nejlepším pořadím je pozice 4, které lze dosáhnout navýšením hodnoty priority na 28. Zároveň je patrné, že nejhlouběji leží pod lineární spojnicí priorita 12, která znamená zlepšení pořadí zpracování o tři místa, tudíž se jeví jako nejvýhodnější v případě urychlování dalších uzlů.



Obrázek 19 Vývoj pořadí zpracování v závislosti na prioritě

Z porovnání výsledků obou pokusů je patrné, že se algoritmus chová stále stejně předvídatelně. Algoritmus je vždy „účinnější“ ze začátku a ke konci je sklon křivek pořadí a času zpracování pozvolnější. Upravený skript je tedy možné použít, vždy je ale nutno u manuálního zvyšování priority zvážit nakolik je možné zpomalit zpracování uzlů ostatních.

změna	změna %	pořadí	čas
0	0,00%	8	145
4	4,88%	7	126
8	9,76%	6	104
12	14,63%	5	78
16	19,51%	5	78
20	24,39%	5	78
24	29,27%	5	78
28	34,15%	4	60

změna	změna %	pořadí	čas
0	0,00%	7	125
4	4,88%	6	103
8	9,76%	5	77
12	14,63%	4	62
16	19,51%	4	62
20	24,39%	4	62
24	29,27%	4	62
28	34,15%	3	47

7 Závěr

Cílem práce byla optimalizace procesů v systému Process Control Center, které řídí denní zpracování v datovém skladu. Pro uvedení do této složité problematiky byla práce rozložena do dvou částí – do části teoretické a praktické.

V teoretické části byla shrnuta základní problematika, týkající se datových skladů a řízení jejich provozu. Dále byl proveden úvod do metodiky teorie grafů, která bezprostředně s procesy datového skladu souvisí.

Praktická část práce byla zhotovena na příkladu Centrálního datového skladu v Československé obchodní bance, a.s. Vzhledem k počtu klientů, zaměstnanců a informačních systémů je Centrální datový sklad pravděpodobně jedním z největších firemních skladů na českém trhu.

Jak již bylo řečeno, cílem této práce byla úprava procesů, které mají na starost zpracování dat v datovém skladu. Součástí nástroje Process Control Center je skript, starající se o plánování běhu zpracování jednotlivých úloh.

Při obrovském množství úloh se může stát, že některý uživatel systému bude požadovat aby jeho data byla zpracována prioritně. Tento požadavek nebylo možné splnit, proto bylo potřeba navrhnout úpravu stávajícího skriptu, potažmo jeho algoritmu do podoby, která by umožňovala manuální zadávání priorit.

Jelikož je současně používaný skript velmi složitý – musí zohledňovat velké množství aspektů jako jsou časové limity atd. Bylo nutné ho upravit do podoby, ve které bude možné pracovat s jednoduše zadatelnými daty. Původně byla vstupní data, jako jsou závislosti a hodnoty uzlů, získávána z SQL tabulek. To by nebylo praktické pro demonstrativní účely práce, proto byl skrip upraven aby bylo možné závislosti a hodnoty uzlů získávat manuálně. Dále byla do skriptu přidána možnost manuálního nastavení priority pro uzel, který je potřeba dokončit prioritně.

S takto upraveným skriptem již bylo možné provádět testovací měření. Cílem úprav algoritmu bylo urychlování požadovaných uzlů. Byla provedena dvě měření. Měření probíhala na stejném grafu, ale urychlován byl vždy jiný uzel. Prvním uzlem, který byl urychlován, byl uzel D. Pokud by algoritmus pracoval bez zásahu v podobě priority, byl by zpracován jako v pořadí sedmý s časem 125 sekund. Postupným navyšováním priority bylo dosaženo posunutí na třetí místo v pořadí zpracování s časem 47 sekund. Hodnota priority

byla zvednuta na 34,71% hodnoty délky větve výchozího výpočtu. Časová úspora pro uzel D tedy činí 78 sekund, to znamená zrychlení o 266%.
Druhé sledování bylo provedeno na uzlu I. Uzel I by byl bez nastavení priority zpracován jako osmý v pořadí v čase 145 sekund. Postupným navyšováním priority na číslo 28 tj. 34,71% délky nejdelší větve výchozího zpracování se podařilo uzel I dostat až na čtvrté místo v pořadí zpracování s časem 60 sekund. To znamená úsporu 85 sekund tj. 242% zrychlení.
Ze srovnání vyplývá, že nastavení hodnoty priority je nejvhodnější na hodnoty odpovídající 30-40% délky nejdelší větve ve výchozím propočtu.
Hlavním přínosem nového algoritmu je fakt, že dokáže přinést velké urychlení zpracování požadovaných uzlů. Doba zpracování celého běhu se sice nemění, ale v případě vznesení požadavku na urgentní zpracování určité oblasti dat je možné s novým algoritmem dosáhnout velmi dobrých výsledků. Rychleji získaná potřebná data slouží k zeefektivnění práce zaměstnanců a k větší spokojenosti klientů ČSOB, a.s.
Vedlejším produktem práce je materiál, který může sloužit pro prezentaci problematiky CDS a jeho části, týkající se plánování běhu úloh. Vytvořený materiál tak může sloužit dále zaměstnancům, případně externím spolupracovníkům, jako úvod do problematiky CDS.

8 Seznam zdrojů

IMHOFF, Claudia. Building the Customer-Centric Enterprise. New York: Wiley, 2001. ISBN 0-471-31981-3.

GIOVINAZZO, W. Object-oriented data warehouse design: building a star schema. Upper Saddle River, NJ: Prentice Hall, 2000, 349 s. ISBN 01-308-5081-0.

VERCELLIS, Carlo. Business intelligence: data mining and optimization for decision making. Chichester, U.K.: Wiley, 2009, 417 s. ISBN 04-705-1139-7.

KIMBALL, Ralph a Joe CASERTA. The data warehouse ETL toolkit: practical techniques for extracting, cleaning, conforming, and delivering data. Indianapolis, IN: Wiley, c2004, 491 s. ISBN 07-645-6757-8.

KOLÁŘ, Josef. Teoretická informatika. Praha, 2009. ISBN 9788090085381.

TURZÍK, Daniel a Pavla PAVLÍKOVÁ. Diskrétní matematika. Vyd. 1. Praha: Vydavatelství VŠCHT, 2007, 106 s. ISBN 978-80-7080-667-8 (BROž.).

Studentmatematiky.own.cz. Studentmatematiky [online]. [cit. 2012-03-07]. Dostupné z: http://www.studentmatematiky.own.cz/

ČSOB, a.s. Provozní příručka aplikace Process Control Center. 00_07. Praha, 2011.

ČSOB, a.s. Central Data Store (CDS) - Information Factory. Praha, 2011.

9 Seznam obrázků