TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics and Interdisciplinary Studies

Návrh a implementace datové a aplikační vrstvy databáze pro vzdálený přístup, webové aplikace a reporting

Design and implementation of data and business logic layer for remote database access, web applications and reporting

STUDIJNÍ PROGRAM N2612 – ELEKTROTECHNIKA A INFORMATIKA
STUDY PROGRAMME N2612 – ELECTROTECHNOLOGY AND INFORMATICS

STUDIJNÍ OBOR 1802T007 – INFORMAČNÍ TECHNOLOGIE
STUDY BRANCH 1802T007 – INFORMATION TECHNOLOGY

DIPLOMOVÁ PRÁCE | DIPLOMA THESIS

Autor práce | Author — Bc. Pavel Polívka
Vedoucí práce | Thesis supervisor — Ing. Jan Kraus, Ph.D.

LIBEREC 2013

Tento list nahraďte
originálem zadání.

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Datum:

Podpis:

Abstrakt

Cílem práce je vytvoření datové vrstvy pro databázi vizualizační aplikaci ENVIS pro Microsoft SQL Server 2012. Tato vrstva zbavuje ostatní aplikace, které potřebují přístup k datům závislosti na dynamických knihovnách aplikace ENVIS.

Tato datová vrstva je realizována pomocí uložených procedur a je aplikována v několika testovacích aplikacích, které vznikly jako příklady jejího použití.

Klíčová slova

Microsoft SQL Server, ENVIS, Uložené procedury, T-SQL, CLR

Abstract

This paper focuses on designing logical layer for database of visualization aplication ENVIS for Microsoft SQL Server 2012. This layer removes requirement for ENVIS dynamic libraries in other aplications that need acces to this database.

This logical layer is implemented using stored procedures and is applied in several test applications that were created as examples of its use.

Keywords

Microsoft SQL Server, ENVIS, Stored procedures, T-SQL, CLR

Typografické konvence

V textu jsou použity různé styly textu a formátování, které pomáhají rozlišit různé typy informací:

Pro normální text je použito běžné proporcionální patkové písmo.

Kurzíva je používána v poznámkách, které obsahují doplňující informace k textu. Taktéž je užita v názvech ovládacích prvků, aplikací apod.

```
Zdrojovy kod je oznacovan takto.
```

Obsah

Seznam zkratek

T-SQL	Transact Structured Query Language
CLR	Common Language Runtime
DB	Databáze (database)
.NET	.NET Framework
MSSQL	Microsoft SQL Server
DLL	Dynamic-link library
LINQ	Language Integrated Querys
VS2010	Visual Studio 2010
SSRS	SQL Server Reporting Services
WPF	Windows Presentation Foundation

1. Úvod

Tato práce se zabývá návrhem a implementací datové a aplikační vrstvy pro vzdálený přístup k datům v databázi měření elektrických veličin. Tato databáze je součástí aplikace *ENVIS* od firmy *KMB*, která zajišťuje vizualizaci daných naměřených veličin. Tato aplikace používá pro databáze Microsoft SQL Server. V této práci je pro jednoduchost použita nejnovější verze MSSQL 2012.

Datová vrstva bude sloužit jako náhrada složitého kódu pro zpracování a rozkódování dat databáze, který je v současnosti součástí aplikace *ENVIS* a probíhá na straně klienta. Přesunutím těchto operací na stranu serveru má mnoho výhod, mezi ně patří potencionální zrychlení provádění kódu (server může být rychlejší, než klient), nutnost stahovat jen potřebná data a ne i data pro různá nastavení a modifikátory atd. Další velkou výhodou je možnost přenositelnosti kódu. V současnosti, pokud by měla vznikat jednoduší aplikace, webová stránka, report, atd., které by potřebovali tato data, musí vznikat v *.NETu*, *C#* a aby mohli použít *DLL* knihovnu *Envisu*, pro přístup z pracování dat. Přesun těchto operací na server, vznik těchto aplikací usnadní a dovolí i vzniku aplikací mimo strukturu Windows a *.NET*.

Tato datová vrstva je realizována pomocí uložených procedur (stored procedures). Microsft SQL Server nabízí několik možností vytvoření uložené procedury, jednou je vytvoření přímo pomocí *TSQL* jazyka (Microsoft úprava normálního SQL) a druhou je pomocí takzvaného *CLR*, které umožňuje uložit *.NET* kód jako proceduru, nebo jinou *SQL* strukturu. Součástí práce je testovací aplikace (Stored Tester), která testuje efektivnost těch dvou postupů.

Testovací aplikace umí volat vytvořené procedury ve smyčce a zaznamenávat čas potřebný pro jejich vykonání. Také umí provádět stejné operace s daty jako samotné

procedury, aby bylo možné ověřit rychlost procedur vůči rychlosti operací v klientovi. Naměřené rychlosti je možné uložit jako soubor dat pro program *MATLAB*, který umožňuje následné zpracování.

Datová vrstva je omezena na strukturu v databázi, která uchovává data pro přístroje *SMV*, *SMP* a *SMPQ* od společnosti *KMB Systems*. Jednotlivé uložené procedury mají předponu *akd* aby byly odlišeny od ostatních procedur.

Pro demonstraci funkčnosti vytvořené datové vrstvy vzniklo několik aplikací, které by normálně musely používat *DLL* aplikace *ENVIS*. První je aplikace pro Microsoft Reporting Server, která zobrazuje protokol o stavu databáze, jenž umí zobrazovat i *ENVIS* . Druhá aplikace se jmenuje *Power Checker*. Ta testuje takzvané čtvrthodinové maximum, zobrazuje data do grafů, tabulek a exportuje je do programu *MATLAB* nebo formátu *CSV*.

Aplikace jsou součástí CD přílohy.

2. Envis

2.1 Struktura databáze

Na obrázku 2.1 je znázorněná zjednodušená struktura databáze, kterou používá aplikace *ENVIS*.

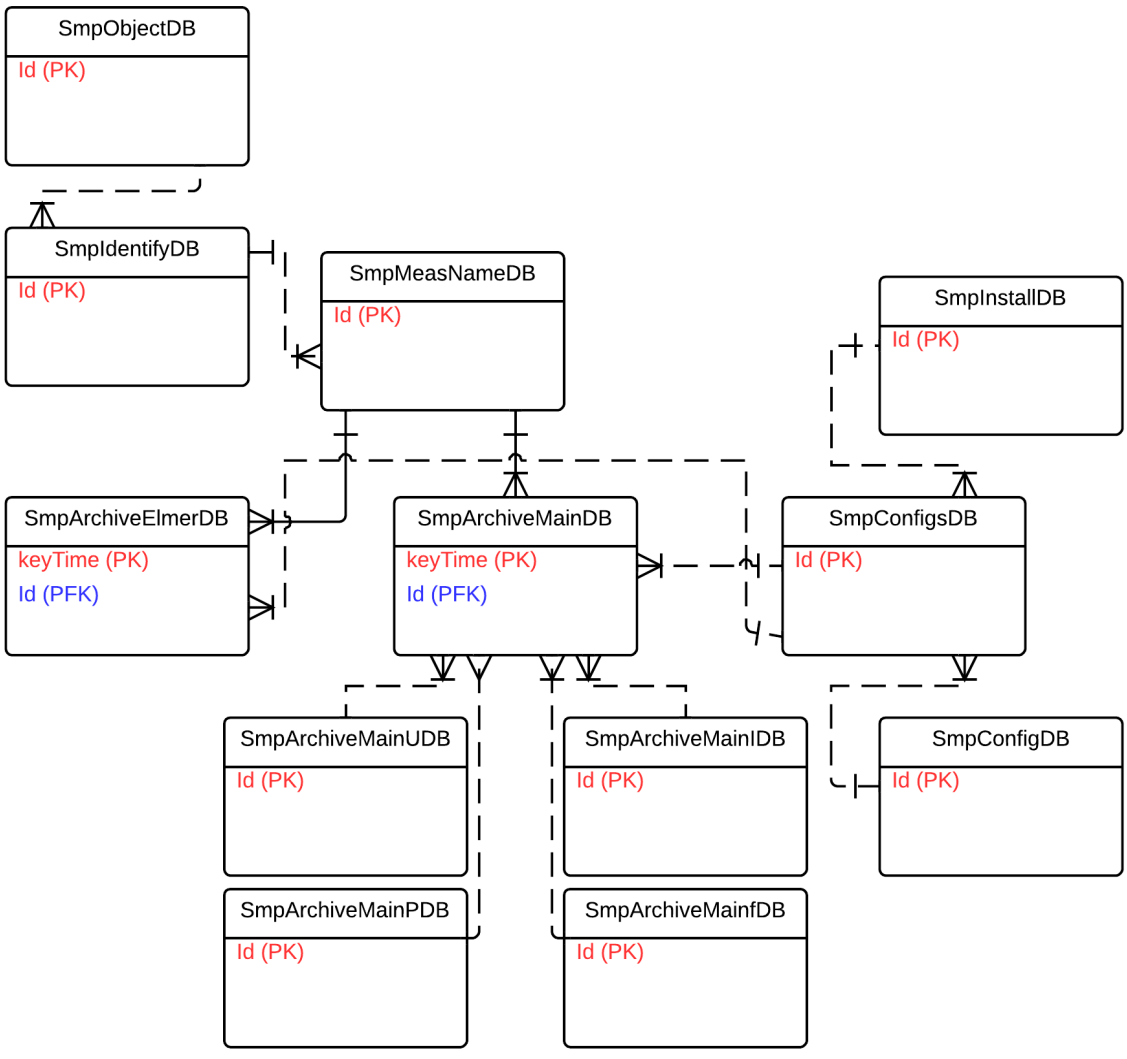
První tabulkou je *SmpObjectDB*, ve které jsou uchovávány informace o jednotlivých objektech, kde jsou přístroje použity. Tabulka obsahuje pouze celočíselný primární klíč a řetězec reprezentující jméno objektu. Samotné přístroje jsou ukládány v tabulce *SmpIdentifyDB*, kde se kromě cizího klíče z tabulky objektů nachází takové informace jako výrobní číslo, číslo softwaru nebo přesný typ zařízení. Další tabulkou je soubor jednotlivých měření, jenž je k nalezení v tabulce *SmpMeasNameDB*. Zde je každé měření označeno svým unikátním identifikátorem, je přiřazeno ke konkrétnímu přístroji a pro jednoduchou orientaci pojmenováno.

Velmi důležitou částí celé databáze jsou takzvané archivy (tabulky *SmpArchiveMainDB*, *SmpArchiveElmerDB*, atd.). Archivy obsahují zakódované hodnoty z měření. Elmer zaznamenává stav elektroměru atd. Nejdůležitějším archivem je hlavní archiv. Jeden záznam v hlavním archivu reprezentuje stovky až tisíce jednotlivých měření různých veličin v časovém úseku. Všechny tyto hodnoty jsou zde zakódovány a uloženy jako binární data. Toto je ovšem nepraktické pro rychlý přístup k datům. Proto hlavní archiv obsahuje soubor referencí na tabulky obsahující různé statistické hodnoty pro každou veličinu v daném časovém úseku, který reprezentuje jeden záznam. Například tabulka *SmpArchiveMainUDB* slouží k ukládání minimální, maximální a průměrné hodnoty napětí, *SmpArchiveMainIDB* pro proud atd.

V tabulce *SmpConfigsDB* se nachází seznam referencí na tabulky, jež uchovávají

různá nastavení. Například *SmpConfigDB* obsahuje nastavení konkrétního přístroje, jako třeba informace o časovém pásmu měření atd. *SmpInstallDB* obsahuje konstanty, pomocí kterých se převádí uložené hodnoty na skutečné jednotky.

Na obrázku 2.1 není rozkreslena celá databáze, to bohužel z důvodů množství tabulek není možné. Nachází se v ní více archivů, více statistických tabulek k hlavnímu archivu, dokonce i více tabulek pro nastavení. Pokud v průběhu práce narazíme na tabulku, která není v obrázku nebo v této kapitole popsána, bude její zasazení dovysvětleno.



Obrázek 2.1: Struktura databáze

2.1.1 Tabulka SmpMainArchive

SmpMainArchiveDB je nejdůležitější archiv celé *ENVIS* databáze. Naměřené hodnoty se neukládají do databáze po jedné (je jich stovky až tisíce), ale po větších částech ohraničené nějakým časovým intervalem. Každý záznam z hlavního archivu reprezentuje jednu takovou část. Jako součást primárního klíče je počáteční čas měření (*keyTime*), je ukládán i konečný čas (*endTime*). Druhou částí primárního klíče je identifikátor konkrétního měření (*keymeansName*), tento identifikátor je v této práci označován jako *identify*. Dále je zde několik referencí do tabulek uchovávající informace o nastavení *ENVISu* i jednotlivých přístrojů (*conf*, *additional* atd.).

Všechny naměřené hodnoty z tohoto časového období jsou statisticky zpracovány a uloženy jako průměr, minimum atd. do samostatných tabulek (viz obrázek 2.1). Hlavní archiv obsahuje reference na tyto jednotlivé tabulky.

Někdy ovšem nejsou statistická data za určitý časový úsek dostatečná. Proto se do hlavního archivu ukládají skutečné naměřené hodnoty po určitém intervalu. Tyto data jsou uložená jako *VARBINARY(MAX)* ve sloupci *Data*. Počet všech hodnot pro záznam je v *RecordCount*. Jednotlivé hodnoty jsou uloženy jako binární data, proto není snadné je z databáze číst. Musí se přečíst všechna data, a následně konkrétní hodnotu rozkódovat. Proto je potřeba znát v jakém pořadí (popřípadě která)data jsou ukládána. V rámci optimalizace nemusí být vždy uloženy i nuly pro veličiny, které nebyly naměřeny, informace, co bylo a co nebylo naměřeno, jsou uloženy u nastavení měření a musí se brát v úvahu při čtení konkrétních hodnot. Přesný popis uložení hodnot je k nalezení v [3] nebo ve zdrojovém kódu procedury *smp_main_archive_detail* popsané v 3.3.3.

2.1.2 Tabulka SmpArchiveElmer

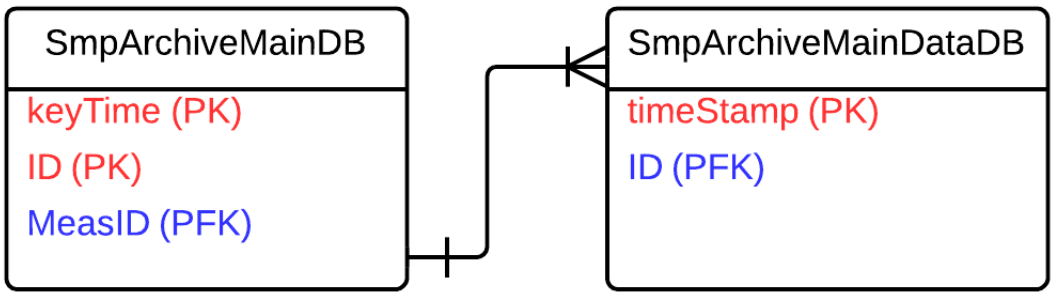
Archiv elektroměru *SmpArchiveElmerDB* je další důležitý archiv. Není ani zdaleka tak komplexní jako hlavní archiv, obsahuje však nějaké společné rysy. Primární klíč je u obou archivů stejný (*keyTime* a *keymeasName*). Reference pro nastavení jsou také stejné. Jelikož se z elektroměru jen přečte hodnota a uloží a časové interva-

ly jsou větší než v hlavních archivu (kde se provádí měření v intervalech v řádu sekund). Neobsahuje archiv elektroměru žádný koncový čas, ani statistická zpracování. Každá naměřená hodnota pro každou fázi se rovnou ukládá ke klíči. Hodnoty, které elektroměr měří jsou popsány v [3].

2.1.3 Návrh na úpravu databáze ENVIS

V této části se práce zaměřuje na nastínění jiného řešení ukládání dat do databáze aplikace *ENVIS*. Především části hlavního archivu (2.1.1). Hlavní problém přístupu k datům v hlavním archivu je, že konkrétní naměřená data jsou uložena do binárních dat, s kterými si *T-SQL* dotaz nedokáže poradit.

Jedno z možných řešení je rozdělit hlavní archiv na dvě samostatné tabulky.



Obrázek 2.2: Přehled archivu - CLR vs. Klient

První by byla původní hlavní archiv (*SmpMainArchiveDB*), kde by zůstala všechna data kromě sloupce data *VARBINARY(MAX)*. Do primárního klíče by přibyla položka *ID*, která by přesně identifikovala každý konkrétní řádek měření. Současný primární klíč již plně identifikuje řádek a pro současný model je dostatečný, jednoatributový identifikátor je potřeba pro následné indexování.

Druhá by byla tabulka, která by uchovávala konkrétní data (třeba *SmpMainArchiveDataDB*). Data by se neukládala jako binární data, ale jako konkrétní hodnoty. Jeden řádek by byl jeden časový záznam (třeba tisíc záznamů pro jeden záznam z hlavního archivu). Jeden z atributů by byl cizí klíč na nový atribut *ID* z původního hlavního archivu, ten by také byl součástí primárního klíče této tabulky. Další částí

primárního klíče by byla časová známka konkrétního záznamu.

Procházení této druhé tabulky nebude příliš efektivní (příliš mnoho záznamů), proto se na ní aplikuje několik indexů. *MSSQL 2012* podporuje několik indexů, pro potřebu zrychlení budou stačit dva základní *CLUSTERED* a *NONCLUSTER*. Tyto indexy prakticky vytvoří *B+ strom* z tabulky, podle atributu na kterém byl index vytvořen. Tento strom urychluje nalezení konkrétní hodnoty atributu. Rozdíl mezi *CLUSTERED* a *NONCLUSTERED* je v tom, že *CLUSTERED* vytváří strom na celých konkrétních řádcích. V databázi může být pouze jeden a díky tomu, že pracuje s celými řádky prakticky řadí tabulku vzestupně/sestupně podle daného atributu. *NONCLUSTERED* index vytvoří strom do samostatné struktury vedle a vytváří jí pouze z referencí na paměť kde je řádek uložen, to znamená že, procházení podle tohoto indexu potřebuje jednu operaci navíc (přející na cíl reference). *NONCLUSTERED* indexů může být na jedné tabulce několik.

Tabulka *SmpMainArchiveDataSMP* se bude procházet prakticky pouze podle dvou atributů, podle ID řádku z hlavního archivu a občasně podle data (uvnitř konkrétního *ID*). Pro rychlejší procházení tabulky by optimální bylo vytvořit *NONCLUSTERED* index na *ID* a *CLUSTERED* index na časovém záznamu. Počet různých ID v tabulce je několikanásobně menší než časů, takže pomalejší procházení zde nebude takový problém. Navíc jakmile se najde první vyhovující ID, díky uspořádání podle času (díky *CLUSTERED* indexu) bude další potřebný záznam nalezen prakticky instantně. Díky těmto indexům se nebudou procházet zbytečná data.

Potencionální problém tohoto řešení spočívá především v tom, že údržba stromových indexů je procesorově náročná pokud nad databází probíhá mnoho *INSERT/UPDATE* operací. Datová tabulka rozhodně při vytvoření indexů nebude konečná, každopádně frekvence ukládání nových dat je podstatně menší než frekvence jejich čtení. Ale je potřeba počítat že operace ukládání nových dat se výrazně prodlouží (i přes to že dnes se data musí serializovat do binárních dat).

Další možný problém je, že tato neserializovaná data plus data z indexů budou na disku zabírat mnohem více místa, než současná binární data. Zde spočívá nebezpečí v tom, že bezplatná verze *MSSQL* serveru, která je instalována společně s aplikací

ENVIS má velikost databáze na disku omezenou na 4GB, ale maximální množství využívané paměti RAM 1GB ([10]), takže se sem větší databáze nemusejí vejít. Novější verze tohoto serveru již mají tyto limity posunuté nahoru.

Na závěr této části je důležité zmínit, že tento postup úpravy databáze nebyl aplikován, není tedy změřen jeho dopad na výsledný výkon databáze. Je zcela možné, že po jeho aplikaci by se zjistilo, že je tento způsob pomalejší než současná serializace a následná deserializace na straně klienta.

3. Datová vrstva

3.1 Uložené procedury

Uložená procedura je skupina jednoho nebo více T-SQL dotazů (nebo reference na .NET CLR metodu). Je uložená přímo v databázi, takže se k ní lze chovat jako k jakémukoliv jinému objektu databáze (index, pohled, atd.). Uložená procedura umí zpracovávat vstupní parametry a vracet data volacímu programu pomocí výstupních parametrů. Obsahuje příkazy, které provádějí operace nad databází, mezi tyto příkazy může patřit i volání dalších procedur. Vrací výsledky pro každý *SELECT* příkaz, jenž je volaný uvnitř procedury a vrací stavovou hodnotu, která indikuje, zda se vykonávání povedlo či ne. [6]

Uložené procedury budou v této práci použity jako náhrada kódu, jež vykonává aplikace *ENVIS* během zpracování dat. To, že data jsou zpracována na serveru a do aplikace jsou posílána již hotová data (ne veškerá data potřebná pro jejich modifikaci), sníží množství přenášených dat, a tím potencionálně urychlí zpracování. Navíc umožní takto zpracovaná data použít i v jiných aplikacích než *ENVIS*.

Další významnou vlastností procedur je zvýšení bezpečnosti, procedura může přistupovat k datům, kterým třeba uživatel posléze nemusí mít přístup. Což zvedá zabezpečení. Pro tuto práci tato vlastnost nemá signifikantní význam.

Při prvním zavolání procedury si *MSSQL* server pro proceduru vytvoří takzvaný spouštěcí plán. Ten zvyšuje rychlost provádění procedury při dalším spuštění. Takže by procedury měly být prováděny rychleji. Pokud dojde k významnějším změnám v tabulkách nebo ve struktuře, tak musí být plán znovu vytvořen.

3.1.1 T-SQL

Jedna ze dvou metod jak v *MSSQL* vytvořit uloženou proceduru je pomocí *T-SQL* jazyka. Syntaxe vytvářejícího dotazu je následující:

```
CREATE PROC [ EDURE ] [ owner. ] procedure_name [ ; number ]
    [ { @parameter data_type }
        [ VARYING ] [ = default ] [ OUTPUT ]
    ] [ ,...n ]

[ WITH
    { RECOMPILE | ENCRYPTION | RECOMPILE , ENCRYPTION } ]

[ FOR REPLICATION ]
AS sql_statement [ ...n ]
```

Přesný popis dotazu je k nalezení v [7] a příklad takto vytvořené procedury je k nalezení v kapitole 3.3.4.

3.1.2 CLR - Common Language Runtime

Další z metod jak vytvořit uloženou proceduru, je pomocí *CLR*. *MS Sql Server 2012* umožňuje integrovat *CLR* komponentu z *.NET frameworku*. V *CLR* se uložené procedury píší pomocí *.NET* jazyků (v této práci byl zvolen *C#*, ale může se používat *Visual Basic* nebo *C++*).

Visual Studio obsahuje již předpřipravený projekt pro *CLR*, ve vlastnostech projektu se nastaví na jakou cílovou konkrétní databázi se má výsledné *DLL* rozbalit. [8] Knihovnu lze do databáze rozbalit i manuálně pomocí *SQL Managment Studia*.

Jednotlivé procedury (nebo triggery, funkce, atd.) jsou psány jako statické funkce bez návratové hodnoty (pokud se jedná o uložené procedury). S databází se dá pracovat jakýmkoliv způsobem, který je v *.NETu* k dispozici (*SQL Client*, *ADO.NET*, *LINQ*), připojovací řetězec je k naší konkrétní databázi k dispozici v objektu *SqlContext*. Označení atributů za výstupní se v *C#* dosahuje použitím modifikátoru ***out*** nebo ***ref*** v *C++* pomocí pointerů atd.

Každá funkce musí mít označeno, co reprezentuje na databázovém serveru. Pokud se jedná o uloženou proceduru je označení následující.

```
[Microsoft.SqlServer.Server.SqlProcedure]
```

Možnost používat *CLR* se musí aktivovat pomocí procedury sp_configure.

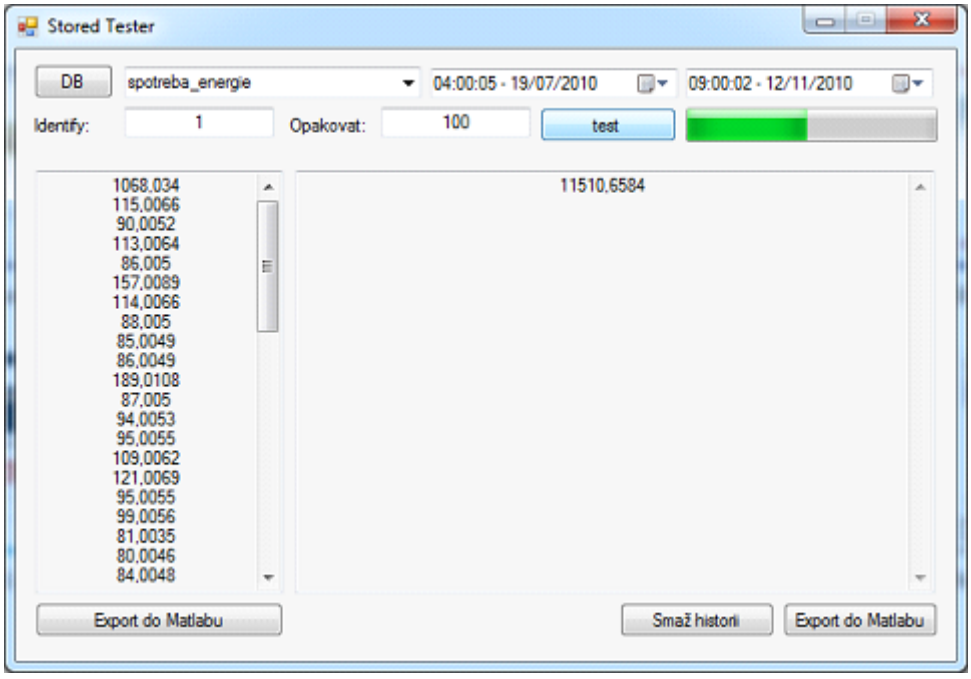
```
EXEC sp_configure 'clr enabled' , '1' ;
reconfigure;
```

Přesný popis tvorby uložených procedur pomocí *CLR* je k nalezení v [9] a příklad v kapitole 3.3.4.

3.2 Testování

Pro vykonání testů, které porovnají časový rozdíl ve zpracování *T-SQ*L a *CLR* procedur a zároveň jejich efektivitu proti kódu na straně klienta, vznikla testovací aplikace Stored Tester. Ta má možnost ve velkém množství volat různé procedury i implementovat různý kód pro zpracování dat na straně klienta.

3.2.1 Testovací aplikace Stored Tester



Obrázek 3.1: Hlavní okno aplikace Stored Tester

V horní části hlavního okna se nachází ovládací prvky pro nastavení konkrétního testování. Tlačítko DB otevře jednoduchý dialog pro výběr připojení na konkrétní

server a konkrétní databázi. V select boxu vedle tohoto tlačítka se vybírá, jaký kód se bude během testu provádět (volání procedury, nebo nějakých jiných funkcí atd.). Dva *datetime pickery* slouží k výběru časového časového intervalu pro, který má být vykonávání kódu omezeno. Položka identify specifikuje konkrétní měření. A položka opakovat, kolikrát se má daný kód zopakovat během jednoho testování. Tlačítko test spustí testování a *progress bar* ukazuje postup konkrétního spuštěného testu.

V dolním levém sloupci jsou vypsány časy jednotlivých opakování v rámci testu v milisekundách. V pravém sloupci jsou uloženy konečné časy pro všechny proběhlé testy (opět v milisekundách). Oba tyto sloupce mají tlačítko pro export hodnot do *MATLAB* souboru. Sloupec se záznamy o všech provedených měření má tlačítko pro vynulování historie.

3.2.2 Metodika měření

Pro jednotlivé testy byly vytvořeny *T-SQL*, *CLR* procedury a k nim kód vykonávající stejnou úlohu přímo v klientovi. Jednotlivé procedury jsou součástí AKD a jsou popsány v kapitole 3.3.

U testů je použito 100 opakování. Před každým z opakování je zaznamenán přesný čas, následně je zaznamenán po dokončení prováděného kódu (před provedením změn do grafického rozhraní). Ukládány jsou jak časy jednotlivých opakování, tak jejich konečný součet.

Během testování na počítači běží pouze operační systém, MS SQL Server a aplikace Stored Tester. Všechny testy jsou provedené na stejném SW a HW za maximálně podobných podmínek.

V diagramu je znázorněn medián, minimum a maximum, horní a dolní kvartil atd. Přesný popis i s příkladem je k nalezení na stránce 38 v [11].

3.2.3 Testovací data

Jako testovací data pro menší archiv jsou použita demo data dodávaná v instalaci aplikace *ENVIS*. Pro větší data je speciální archiv. Oba soubory jsou přiloženy v CD příloze.

3.2.4 Použitý software a hardware

Testy jsou prováděny na počítači s operačním systémem Windows 7 SP1. Pro databáze je použit Microsoft SQL Server 2012.

Pro testování je použita tato sestava:

Procesor - 2,4 GHz Intel Core i3 370M, 2 jádra, 4 thready, 3MB L3 cache

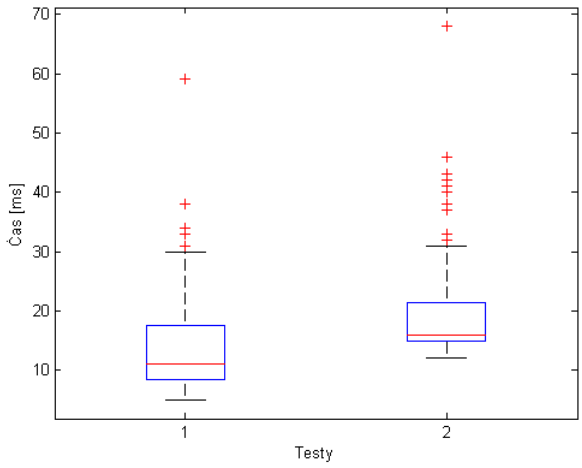
Paměť - 4 GB DDR3

Pevný disk - 500 GB SATA, 7200 RPM

3.2.5 Jednotlivé testy

TSQL vs. CLR

Tento test porovnává efektivnost procedur vytvořených pomocí T-SQL 3.1.1 a pomocí CRL 3.1.2.



Obrázek 3.2: Přehled archivu - TSQL vs. CLR

Na obrázku 3.2 je zobrazena výkonnost procedury *akd_overview_all* pro mezní časové rozhraní a měření číslo 1. Toto nastavení se opakuje i ostatních testů. V prvním sloupci je *TSQL* a v druhém *CLR* metoda.

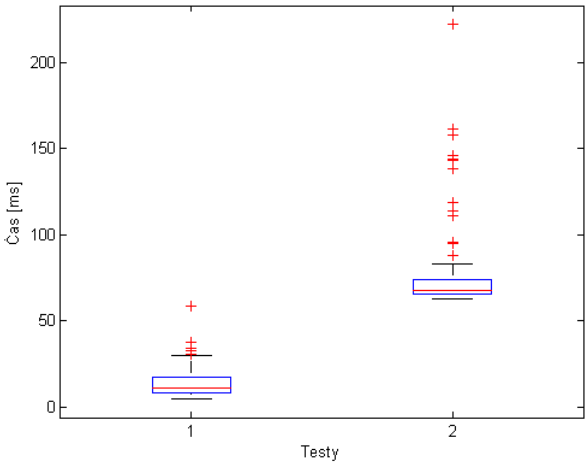
Graf stejného testu pro proceduru *akd_energy_consuption* je k nalezení v příloze A.1.2 V příloze A.1.1 je graf shrnující obě použité procedury.

Stejné testy z kategorie TSQL vs. CLR pro velká data jsou k nalezení v přílohách A.2.1, A.2.2 a A.2.3.

Z grafů je patrné že procedury napsané v TSQL jsou efektivnější než procedury v CLR na velkých i malých datech.

TSQL vs. Klient

Tento test porovnává efektivitu TSQL procedury oproti té samé úloze vykonávané přímo v klientovi.



Obrázek 3.3: Přehled archivu - TSQL vs. Klient

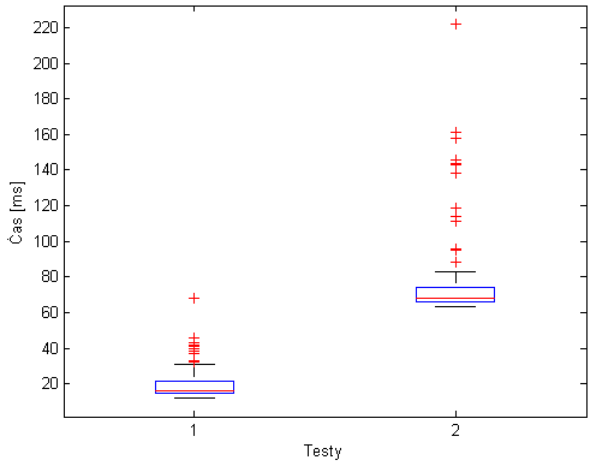
Na obrázku 3.3 je v prvním sloupci zobrazena výkonnost *TSQL* procedury *akd_overview_all* pro mezní časové rozhraní a měření číslo 1. V druhém sloupci jsou časy potřebné pro vykonání adekvátních operací přímo v klientovi.

Graf stejného testu pro proceduru *akd_energy_consuption* je k nalezení v příloze A.1.3. A grafy pro velká data jsou k nalezení v přílohách A.2.4 a A.2.5.

Z grafů je patrné, že procedury napsané v *TSQL* jsou efektivnější než vykonávání adekvátního kódu v klientovi na velkých i malých datech.

CLR vs. Klient

Tento test porovnává efektivitu *CLR* procedury oproti té samé úloze vykonávané přímo v klientovi.



Obrázek 3.4: Přehled archivu - CLR vs. Klient

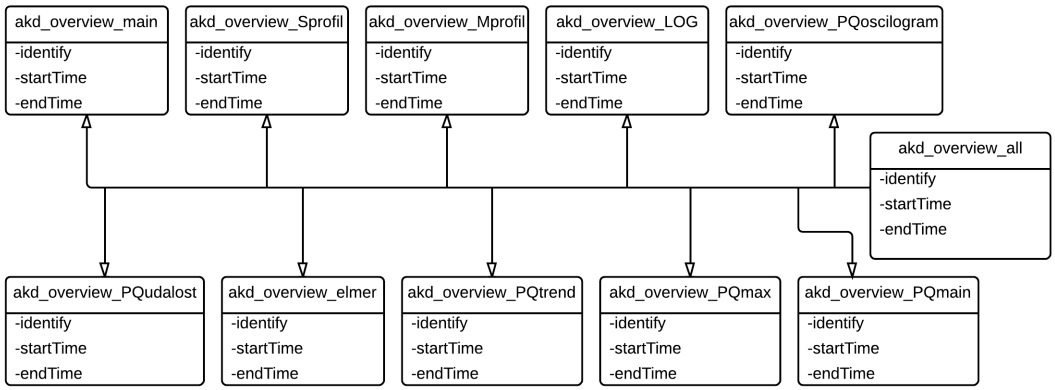
Na obrázku 3.4 je v prvním sloupci zobrazena výkonnost *CLR* procedury *akd_overview_all* pro mezní časové rozhraní a měření číslo 1. V druhém sloupci jsou časy potřebné pro vykonání adekvátních operací přímo v klientovi.

Graf stejného testu pro proceduru *akd_energy_consuption* je k nalezení v příloze A.1.4. A grafy pro velká data jsou k nalezení v přílohách A.2.6 a A.2.7.

Z grafů je patrné, že procedury napsané v *CLR* jsou efektivnější než vykonávání adekvátního kódu v klientovi na velkých i malých datech.

3.3 AKD

3.3.1 Přehled archivů - akd_overview



Obrázek 3.5: akd_overview - struktura

Tyto procedury vrací přehled o jednotlivých archivech. Pro každý archiv vrací kolik obsahuje záznamů, minimální a maximální data v daném časovém rozmezí pro konkrétní přístroj. Celková procedura akd_overview vrací statistiku pro všechny neprázdné archivy v daném časovém intervalu pro konkrétní přístroj.

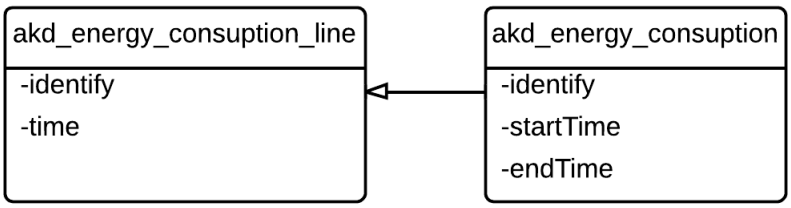
```
EXECUTE akd_overview_all '1','2010-11-05␣10:15:00.000','2010-11-12␣
    10:15:00.000'
```

Každá statistika pro jednotlivé archivy je zpracována ve vlastní proceduře, pokud je potřeba statistika jen pro hlavní archiv, stačí procedura *akd_overview_main* pro archiv elektroměru *akd_overview_elmer* atd.

V jednotlivých procedurách jsou použity i pomocné procedury z kapitoly 3.3.4 na obrázku 3.5 však z důvodů přehlednosti nejsou zobrazeny.

3.3.2 Spotřeba energie

Komplexní statistika - akd_energy_consumption



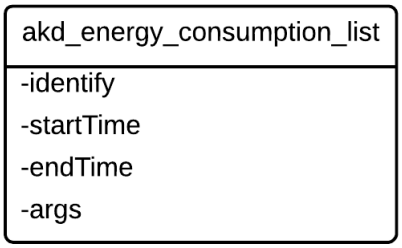
Obrázek 3.6: akd_energy_consumption - struktura

Tato procedura vrací komplexní statistiku spotřebované energie. Pro konkrétní přístroj v daném časovém rozmezí vrátí počáteční a konečnou hodnotu na elektroměru pro import (kWh), export (kWh), kapacitní (kvarh) i induktivní (kvarh). V dalším řádku je jejich rozdíl. Na dalších řádcích jsou rozvedeny přesné nárůsty hodnot po dvou hodinách.

```
EXECUTE akd_energy_consumption '1','2010-11-05␣
   10:15:00.000','2010-11-12␣10:15:00.000'
```

Procedura *akd_energy_consumption* vrací hodnoty pro jeden záznam v archivu elektroměru. V archivu elektroměru nejsou typicky záznamy ukládány po dvou hodinách, ale spíše v řádech minut. Musí se vždy od řádku na konci dvouhodinového intervalu odečíst ten co je na začátku.

Zjednodušená statistika - akd_energy_consumption_list



Obrázek 3.7: akd_energy_consumption_list - struktura

Tato procedura vrací stav elektroměru pro konkrétní přístroj v daném časovém rozmezí. Pro každý uložený záznam v archivu elektroměru, který spadá do daného časového období, zobrazí konkrétní čas a hodnoty veličin vybraných v argumentu *args*.

```
EXECUTE akd_energy_consumption_list '1','2010-11-05␣
   10:15:00.000','2010-11-12␣10:15:00.000','I,E'
```

Parametr *args* není *case sensitive*. Jeho povolené hodnoty jsou I, E, L, C, IT, ET, LT, CT. V příkladu volání procedury je použita hodnota *'I,E'*. Procedura tedy zobrazí hodnoty elektroměru pro Import (kWh) a Export (kWh).

Měření - akd_energy_consumption_identifylist

Tato procedura nemá žádný vstupní parametr.

Ve většině procedur je jeden ze vstupních parametrů *identify* značí to identifikátor konkrétního měření. Pokud uživatel zná číslo měření, není to problém, pokud ale má být AKD použito i pro aplikace, kde uživatel číslo měření nemusí znát, musí zde být i možnost vrátit seznam uložených měření.

```
EXECUTE akd_energy_consumption_identifylist
```

ENVIS sice obsahuje i tabulku se seznamem všech měření (*SmpIdentifyDB*), viz obrázek 2.1, ta ale obsahuje seznam všech měření použitých napříč všemi archivy. Proto tato procedura projde celý archiv elektroměru a vrátí všechny *identify*, které jsou zde uložené. Takhle má uživatel k dispozici přesný seznam měření, která ukládala i hodnoty elektroměru.

Časy - akd_energy_consumption_datetable

Tato procedura má jediný vstupní parametr a tím je *identify*, identifikátor měření.

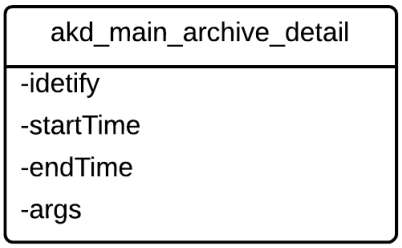
Ve většině procedur jsou vstupní parametry pro začáteční a konečný čas. Tyto časy samozřejmě mohou být libovolné. Pokud by však nějaká aplikace chtěla omezit výběr časů jen na ty, které jsou fyzicky uložené v databázi, musí zde být procedura co vrací tyto časy.

```
EXECUTE akd_energy_consumption_datetable '1'
```

Procedura projde celý archiv elektroměru a vrátí nejmenší a největší datum. Parametr *identify* jde zde aby se tyto data dala omezit jen pro konkrétní měření.

3.3.3 Hlavní archiv

Detaily archivu - akd_main_archive_detail



Obrázek 3.8: akd_main_archive_detail - struktura

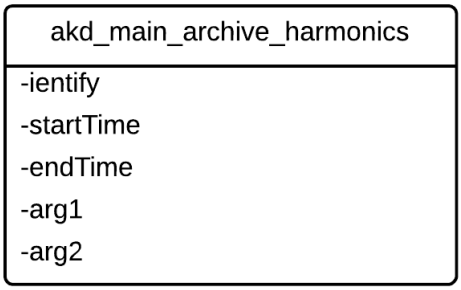
Tato procedura vrací hodnoty zakódované v binárních datech uložených v *MainArchivu* (viz. 2.1.1) v daném časovém rozmezí pro konkrétní měření. Tato procedura je psaná v *CLR*, i když práce v *CLR* podle testů (3.2.1) není tak efektivní jako v *T-SQL* práce se zakódovanými daty je v *C#* mnohem proveditelnější.

Jelikož každý záznam archivu reprezentuje desítky až stovky reálných hodnot, na začátku procedury se rozhoduje, které záznamy spadají do časového rozmezí a které ne. Poté se provede jeden delší dotaz, který projde první reálný záznam z prvního řádku archivu. Tento dotaz slouží k propočítání pozic jednotlivých hodnot v binárních datech. Následuje mechanické čtení dat podle jednotlivých parametrů.

```
EXECUTE akd_main_archive_detail '1','2010-11-05␣
   10:15:00.000','2010-11-12␣10:15:00.000','AVG_ULN1,AVG_ULN2'
```

Parametr *args* není case sensitive. Jeho povolené hodnoty jsou uvedeny v příloze B.3.1. *X* u jednotlivých hodnot značí proměnou hodnotu, za kterou se můžou dosadit čísla 1,2,3 a 4. V příkladu volání procedury je použita hodnota *'AVG_ULN1,AVG_ULN2'*. Procedura tedy zobrazí hodnoty průměrného napětí pro první a druhou fázi.

Harmonické - akd_main_archive_harmonics



Obrázek 3.9: akd_main_archive_harmonics - struktura

Procedura *akd_main_archive_detail* sice vrací hodnoty zakódované v binárních datech hlavního archivu, neumí však vrátit hodnoty harmonických. Výpis harmonických je totiž mírně specifičtější úloha. Harmonická není typicky pouze jedna hodnota, takže nejde udělat jednoduché mapování na "sloupečky". Proto vznikla tato druhá procedura, která slouží k výpisu harmonických pro dané časové rozmezí a konkrétní měření.

Celá procedura je opět psaná v *CLR* a průběh je velmi obdobný (určení záznamů, první průzkumný dotaz atd.) Jelikož je počet výsledných harmonických proměnný, procedura vždy zvládne vykreslit pouze jeden konkrétní problém (na rozdíl od ostatních podobných procedur, co mohly vykreslovat více argumentů najednou). Pokud jsou počítány jen liché harmonické, výsledná tabulka se adaptuje a vykreslí pouze nenulové (liché) hodnoty. Obdobně pro pouze sudé.

```
EXECUTE akd_main_archive_harmonics '1','2010-11-05␣
   10:15:00.000','2010-11-12␣10:15:00.000','HARMUR',1
```

Parametr *arg1* není case sensitive. Jeho povolené hodnoty jsou HARMUR, HARMIR, INTERHARMUR, INTERHARMIR, HARMUHELR. Parametr vždy zpracuje pouze jednu hodnotu. Pokud jich je zadáno více najednou je použita ta první a ostatní jsou ignorovány.

Parametr *arg2* se zadává jako číslo v rozmezí 1 až 4. Určuje pro kterou fázi se mají zobrazit harmonické. V příkladu je použito *HARMUR 1*. Takže by se zobrazily harmonické pro první fázi na napětí (*u1*).

Měření - akd_main_archive_identifylist

Tato procedura nemá žádný vstupní parametr.

Ve většině procedur je jeden ze vstupních parametrů *identify* značí to identifikátor konkrétního měření. Pokud uživatel zná číslo měření, není to problém, pokud ale má být AKD použito i pro aplikace, kde uživatel číslo měření nemusí znát, musí zde být i možnost vrátit seznam uložených měření.

```
EXECUTE akd_main_archive_identifylist
```

Envis sice obsahuje i tabulku se seznamem všech měření (*SmpIdentifyDB*), viz obrázek 2.1, ta ale obsahuje seznam všech měření použitých napříč všemi archivy. Proto tato procedura projde celý hlavní archiv a vrátí všechny *identify*, které jsou zde uložené. Takhle má uživatel k dispozici přesný seznam měření, která ukládala i hodnoty elektroměru.

Časy - akd_main_archive_datetable

Tato procedura má jediný vstupní parametr a tím je *identify*, identifikátor měření.

Ve většině procedur jsou vstupní parametry pro začáteční a konečný čas. Tyto časy samozřejmě mohou být libovolné. Pokud by však nějaká aplikace chtěla omezit výběr časů jen na ty, které jsou fyzicky uložené v databázi, musí zde být procedura co vrací tyto časy.

```
EXECUTE akd_main_archive_datetable '1'
```

Procedura projde celý hlavní archiv a vrátí nejmenší a největší datum. Parametr *identify* jde zde aby se tyto data dala omezit jen pro konkrétní měření.

3.3.4 Pomocné procedury

Ve zde zpracovávaných procedurách je potřeba pomocné procedury, jež nalezené hodnoty převádí na skutečné jednotky, nebo převádí čas do místního časového pásma. Ty jsou jednoduše popsány v této podkapitole.

Převod na reálné hodnoty

Hodnoty v databázi nejsou uloženy v běžně používaných jednotkách (napětí ve voltech, proud v ampérech atd.), jsou uloženy zakódované pro šetření paměti v databázi a pro snadnější počítání s danými jednotkami. Konstanty, které jsou potřeba pro převod na skutečné hodnoty, jsou k nalezení v tabulce *SmpInstallConfigDB*. Přesný popis jednotlivých převodů je k nalezení v [3].

Následuje jednoduchý příklad v T-SQL:

```
CREATE PROCEDURE real_value_u
@id integer,
@vstup integer,
@vystup float OUTPUT
AS
DECLARE @idd integer
SELECT @idd=conf FROM SmpArchiveMainDB WHERE keymeasName=@id
DECLARE @konstanta integer
SELECT @konstanta=MTN FROM SmpInstallConfigDB WHERE Id=@idd
IF @konstanta = 65535
BEGIN
 SELECT @vystup = @vstup / 40
END
ELSE
BEGIN
 SELECT @vystup = @vstup * @konstanta / 40
END
```

Tato procedura převádí vstupní hodnotu *@vstup* napětí na výslednou hodnotu *@vystup* ve voltech. Parametr *@id* ukazuje z jakého měření je převáděná hodnota.

Podobné procedury jsou vytvořeny jak v *T-SQL*, tak *CLR* pro všechny veličiny jež jsou v databázi zpracovány. Hotové jsou k nalezení na CD příloze.

Převod času

V archivech je jedním z klíčů počáteční čas daného měření. Tento čas je ukládán jako *GMT* (Greenwich Mean Time) čas, protože je aplikace Envis používána celosvětově. V tabulce *SmpConfigDB* je uloženo nastavení pro čas, které říká ve kterém časovém pásmu (popřípadě modifikace pro letní čas) se má čas zobrazovat.

Následuje příklad v C# *CLR*:

```
[Microsoft.SqlServer.Server.SqlProcedure]
public static void convert_date(SqlInt32 identify,ref SqlDateTime cas)
{
using (SqlConnection connection = new SqlConnection("context
    connection=true"))
```

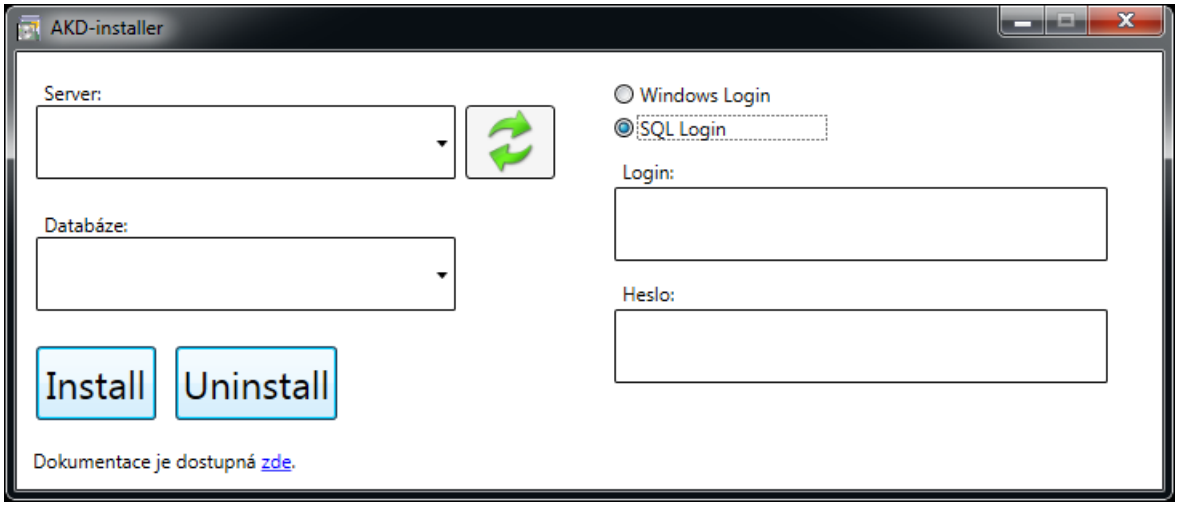
```
  {
            connection.Open();
            SqlCommand command = new SqlCommand("SELECT
               TimeZone,SummerTime FROM
            SmpConfigDB WHERE Id="+identify+"", connection);
            SqlDataReader reader = command.ExecuteReader();
            reader.Read();
            SqlInt32 timezone =
               SqlInt32.Parse(reader["TimeZone"].ToString());
            SqlInt32 summertime =
               SqlInt32.Parse(reader["SummerTime"].ToString());
            reader.Close();

            timezone -= 12;
            if (summertime != 1) timezone += 1;
            int t = Int32.Parse(timezone.ToString());
            cas = SqlDateTime.Add(cas, new TimeSpan(t, 0, 0));
        }

}
```

Tato procedura převádí *cas* do konkrétního časového pásma. Parametr *identify* ukazuje, z jakého měření pochází časová hodnota a tím specifikuje přesné nastavení.

3.3.5 Instalační aplikace AKD installer



Obrázek 3.10: Hlavní okno aplikace AKD-installer

Jelikož jsou některé procedury vytvořené pomocí *CLR*, je instalace celé *AKD* datové vrstvy poněkud složitější než provedení pár SQL dotazů. Proto existuje tato jednoduchá instalační aplikace. V *combolistu* server se vybere server na kterém je *ENVIS*

databáze nainstalována. Tlačítko pro obnovení aktualizuje seznam všech serverů. Následně se vybere konkrétní databáze a *login* metoda.

Důležité je pro instalaci a odinstalaci konkrétní verze AKD použít vždy instalační aplikaci ke konkrétní verzi. To znamená, že pro novější verzi nebude starší odinstalace fungovat zcela správně (neodstraní nové metody, které ve staré nebyly implementovány) a naopak novější odinstalace se bude pokoušet odstraňovat metody co ještě neexistují.

Všechny v *T-SQL* napsané procedury jsou ve složce *SCRIPTS* u instalační aplikace. *CLR* procedury jsou v té samé složce pod názvem *Procedury.dll*. Při instalaci se nejprve zavolá *T-SQL* kód, poté se uloží *.dll* do instance MSSQL serveru a nainstalují se i procedury z něj. *CLR* procedury nejsou zpětně editovatelné v *Managment studiu* jako normální uložené procedury. V seznamu procedur se zobrazují s malým zámečkem, a pokud je potřeba je změnit, musí se AKD odinstalovat, upravit kód ve *VS2010*, vytvořit nové *.dll* a znovu nainstalovat.

Celá AKD instalační aplikace v poslední aktuální verzi je k nalezení na CD příloze. Novější verze budou dostupné na vyžádání.

4. Aplikační vrstva

V této části práce bude demonstrováno použití datové vrstvy *AKD* popsané v kapitole 3.3. Bude demonstrováno použití *Microsoft Report Serveru* ve kterém se pro jednotlivé části reportů používají *AKD* uložené procedury.

4.1 SQL Server Reporting Services - Reporty

SQL Server Reporting Services (SSRS) poskytuje celou škálu nástrojů pro tvorbu, distribuci a správu reportů. Dále poskytuje mnoho programovacích nástrojů, které umožňují rozšířit a upravit konkrétní reporty o dodatečné funkce.

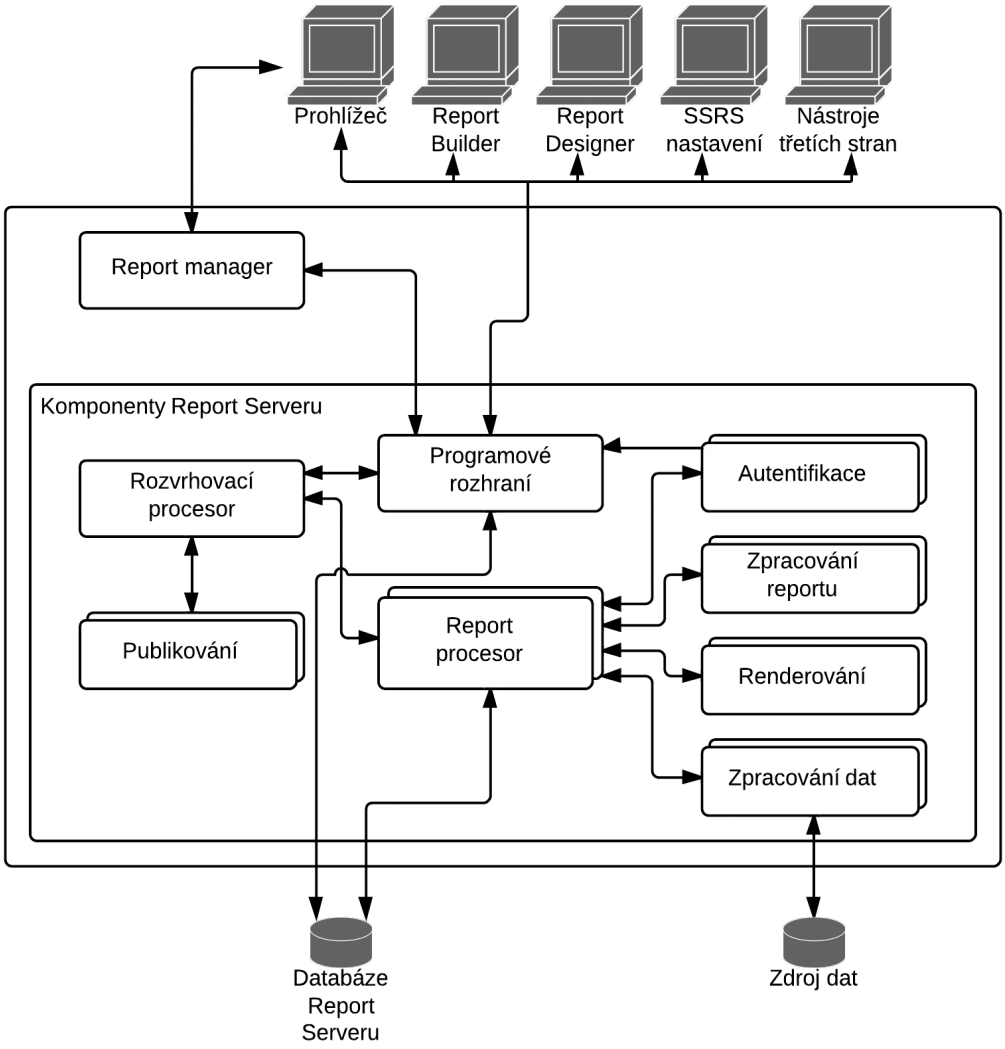
SSRS je serverově založená platforma, která funguje s velkým množství různých datových zdrojů. Mezi ně patří Microsoft SQL Server, další relační a multidimenzionální databázové servery nebo XML datové soubory. Poskytuje také sadu API volání, která vývojářům umožňuje nasadit konkrétní reporty do vlastních aplikací, umožňuje hostování reportů na webové stránce, nebo export do *PDF*.

Vytvořené interaktivní reporty mohou být tabulkové, grafové, mohou obsahovat obrázky nebo mapy atd. Reporty se dají okamžitě publikovat, načasovat jejich vytvoření, nebo se mohou generovat na požádání. Pomocí *Microsoft SharePoint*, jde vytvořit emailové upozornění na nový report.

Nástroje SSRS pracují uvnitř *Visual Studia* a jsou plně integrovaná s nástroji a komponenty SQL Serveru.

Více je k dispozici v [12].

4.1.1 Microsoft Report Server



Obrázek 4.1: Schéma Report Serveru

Microsoft Report Server je hlavní částí SSRS. Skládá se ze dvou procesních jednotek a kolekce rozšíření, které se starají o autentizaci, zpracování dat, renderování, rozvrhování a publikování. Může běžet ve dvou módech, nativním módu a *SharePoint* módu. Plné detailní popisy jsou k dispozici v [13].

Postup instalace serveru a všech problémů s tím spojených je k nalezení v [14].

Nativní mód

V tomto módu je server stand-alone aplikace, která poskytuje všechny potřebné moduly pro tvorbu, management, zpracování a publikování reportu. Tento mód je výchozí.

Schéma třívrstvého nativního módu je zobrazeno na obrázku 4.1. První vrstvou je datová vrstva, sem spadá konkrétní databáze Report Serveru a všechny použité zdroje dat. Konkrétní komponenty serveru jsou prostřední vrstva. Klientské aplikace a vestavěné nástroje jdou třetí vrstvou. Obrázek dále ukazuje cesty požadavků a data mezi jednotlivými komponenty.

Report Server je implementován jako *Microsoft Windows service* který hostuje web server a všechny potřebné komponenty.

SharePoint mód

V *SharePoint* módu server běží jakou součást *SharePoint* serverové farmy. Větší část komponent co běží v nativním módu nahrazují nové *SharePoint* komponenty.

V této práci je použit nativní mód, konkrétní vlastnosti tohoto módu jsou k nalezení v [13].

4.1.2 Vytvoření základního tabulkového reportu

Tato podkapitola slouží jako velmi jednoduchý návod na vytvoření z reportu pomocí SSRS. Plný návod je k nalezení v [15].

Vytvoření projektu

Pro vytvoření Report Serveru projektu se spustí *SQL Server Data Tools*, ten prakticky otevře *Visual Studio*, a přidá do něj několik nových předinstalovaných projektů. Report Server projekt je v záložce *Business Intelligence*. Při prvním spuštění *Data Tools* se zobrazí *Business Intelligence Settings*, což je dialog ve kterém se nastavují různé hodnoty pro prostředí.

Po založení projektu se do něj musí přidat konkrétní report. Ten se přidává jako všechny položky ve Visual Studiu (pravým tlačítkem v *Solution Exploreru*, *Add New Item*, *Report*). Jméno reportu by mělo končit zkratkou *.rdl*.

Nastavení datových zdrojů

Aby měl report informativní smysl, musí obsahovat nějaká data. Ta se získávají z jednotlivých datových zdrojů. Po přidání nového reportu se zobrazí panel *Report Data* na tomto panelu se editují zdroje pro celý report. Nový datový zdroj se přidá pravým tlačítkem *New*, *Data Source*. Dále se postupuje přímo podle návodu na obrazovce, většinou následuje jednoduché připojení na MSSQL.

Nastavení datasetu pro report

Další důležitá část je takzvaný *dataset*. Jedná se kolekci konkrétních dat z daného zdroje získaných nějakým dotazem, uloženou procedurou atd. Tato data jsou poté konkrétně zobrazována v reportu.

Nový *data set* se opět zakládá v *Report Data* panelu. V okně po jeho založení se vybírá z kterého zdroje jsou data použita a následně jestli se jedná o dotaz nebo o proceduru. Obě varianty mají možnost jak textového tak grafického designéru dotazu, popřípadě volání procedury.

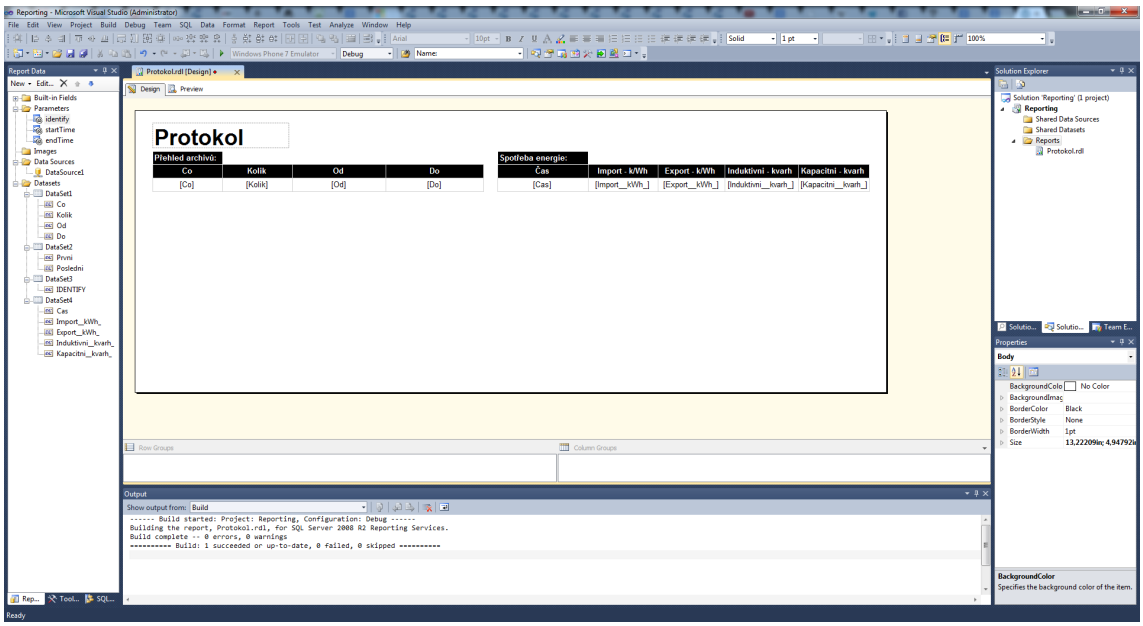
Přidání tabulky do reportu

Poté co je hotový dataset, zbývá už jen vytvořit konkrétní report. Z *Report Data* panelu se dají jednotlivé věci přetahovat do záložky s vytvořeným reportem (*.rdl* soubor). Dají se sem umístit jednotlivé datové regiony z datasetu, různé vstupní pole, nadpisy, obrázky.

Položky ve kterých se vyskytuje více řádků, jsou nazývány datové regiony a jsou označeny hranatou závorkou zleva a zprava. V každém report by měl každý region použit minimálně jednou. Může se ale použít víckrát, pokud třeba pro jeden výstup je potřeba vykreslit tabulku i graf.

Přetažením *datasetu* do reportu se vytvoří tabulka podobná té co je v programu *Microsoft Excel*, která obsahuje základní nadpisy a všechny regiony. Tato tabulka má i velmi podobné grafické formátování jako tabulka v *Excelu*.

Výslednou tabulku se dá prohlédnout v záložce *Preview*.



Obrázek 4.2: Tvorba reportu

Publikování

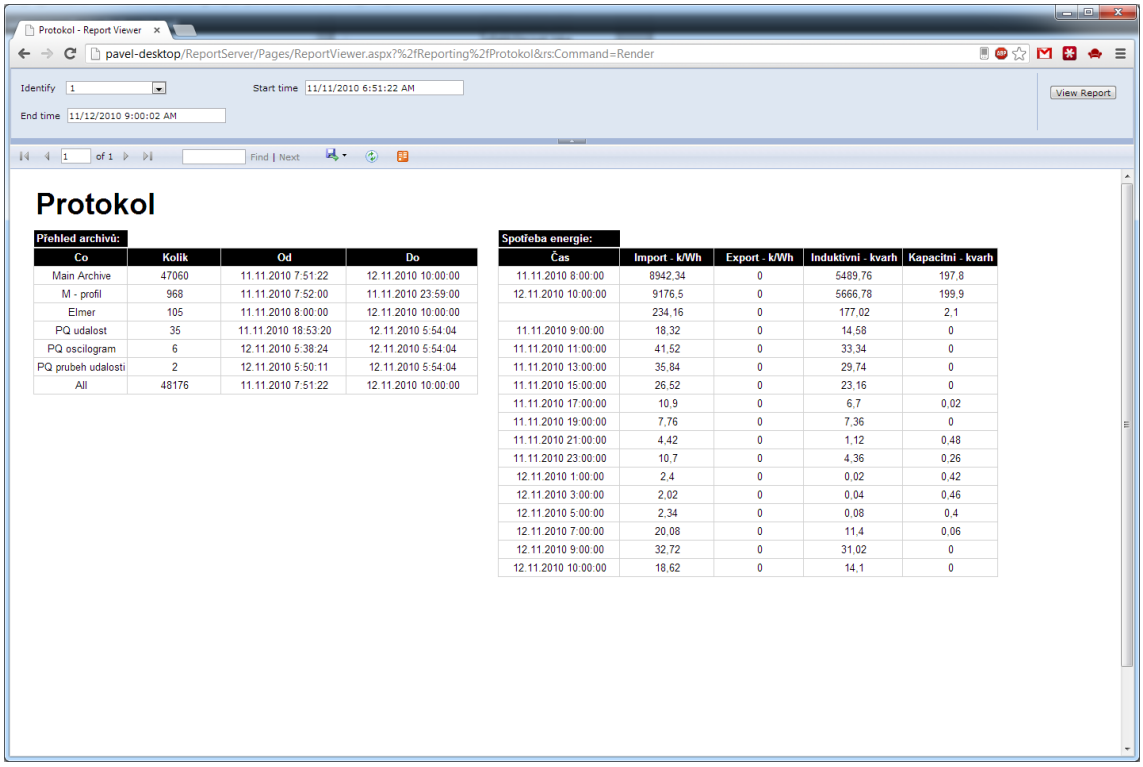
Pokud je Report Server správně nastavený (první nastavení je poněkud složité, ale podle postupů v [13], [14], [15] by to nemělo být nemožné), základní publikování je velmi jednoduché. Pravé tlačítko myši na projekt v *Solution Exploreru* a kliknutí na *Deploy*. Toto publikuje report na web server hostovaný Report Serverem.

Komplexnější nastavení publikace se nastavují pomocí vlastností projektu, popřípadě přímo vlastností Report Serveru.

4.1.3 Tvorba protokolu aplikace ENVIS

Aplikace *ENVIS* poskytuje možnost vytvořit si protokol, který obsahuje různé informace o naměřených hodnotách (obsahy archivů, spotřeba energie, atd..). Tato data se získávají z databáze a následně se na straně klienta zpracovávají, poté je *ENVIS*

zpracuje do grafických tabulek a uloží do *PDF*.



Obrázek 4.3: ENVIS protokol

Pomocí návodu v 4.1.2 a *AKD* uložených procedur *akd_overview* (3.3.1) a *akd_energy_consumption* (3.3.2) se dá vytvořit základní tabulkový report, který zvládá prakticky totožné věci jako report v aplikaci *ENVIS*. Výsledný report je na obrázku 4.3. Tato webová stránka umožňuje procházet jednotlivé stránky reportu, export do *PDF* i *CSV* má i speciální verzi pro tisk.

V návodu 4.1.2 není popsáno jak se do reportu přidávají vstupní parametry pro uložené procedury (*identify*, *startTime*, *endTime*). U uložených procedur se vstupní parametry vygenerují samy, podle jejich konkrétních požadavků v předpisu procedury. U dotazů se musí konkrétní parametry vytvořit v sekci *parameters* v panelu *Report Data*. Parametry jsou vidět na obrázku 4.2.

Tyto vstupní parametry jsou ve výsledném webovém reportu reprezentovány vrchním formulářem (viz obrázek 4.3). Při odeslání formuláře (stisknutím tlačítka *View Report*) se v základu vygeneruje nový report pomocí zadaných vstupních parametrů.

4.2 Aplikace

V této části je popsána jednoduchá aplikace, která demonstruje funkčnost rozhraní *AKD*.

4.2.1 Power Checker

Tato aplikace za pomoci *AKD* datové vrstvy (3.3) kontroluje takzvané čtvrthodinové maximum. Zobrazuje tabulky a grafy konkrétních hodnot, umožňuje jejich export do *MATLABu*, *CVS* a *PNG*.

Aplikace je vyvíjena na frameworku *.NET 4.5*, technologií *WPF*. To znamená že je k dispozici pouze pro platformu *Windows*. Není ale závislá na žádných dynamických knihovnách aplikace *ENVIS*, kdyby byla psaná v multiplatformním jazyce, mohla by fungovat i na jiných platformách.

Pro demonstraci funkčnosti jsou výsledky této aplikace i data které zpracovává k nalezení na CD příloze.

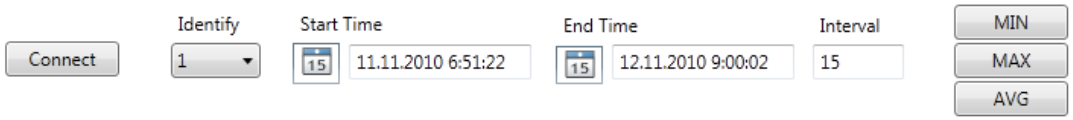
Čtvrthodinové maximum

V průmyslových podnicích představujících možný větší odběr příkonu je obvykle z provozních důvodů sledovaný časový úsek stanoven na 15 minut – čtvrthodinu. Čtvrthodinovým (technickým) maximem (1/4max) se tedy rozumí hodnota průměrného čtvrthodinového elektrického příkonu, kterou smí odběratel za sledovaný časový úsek nejvýše odebrat.

Tato hodnota je stanovena i smluvně, dodavatelem je režim 1/4max sledován a překročení smluvního limitu je dodavatelem penalizováno. Každý provozovatel se proto snaží smluvní hodnotu nepřekračovat, a to buď vhodným rozložením energetické náročnosti spotřeby (někdy i za cenu negativních zásahů do výrobního procesu – snížením současného výkonového odběru, tzn. násilným vypínáním spotřebičů, strojů), nebo instalací vhodných regulačních prvků – regulátorů čtvrthodinového maxima.
Předchozí dva odstavce citovány z [16].

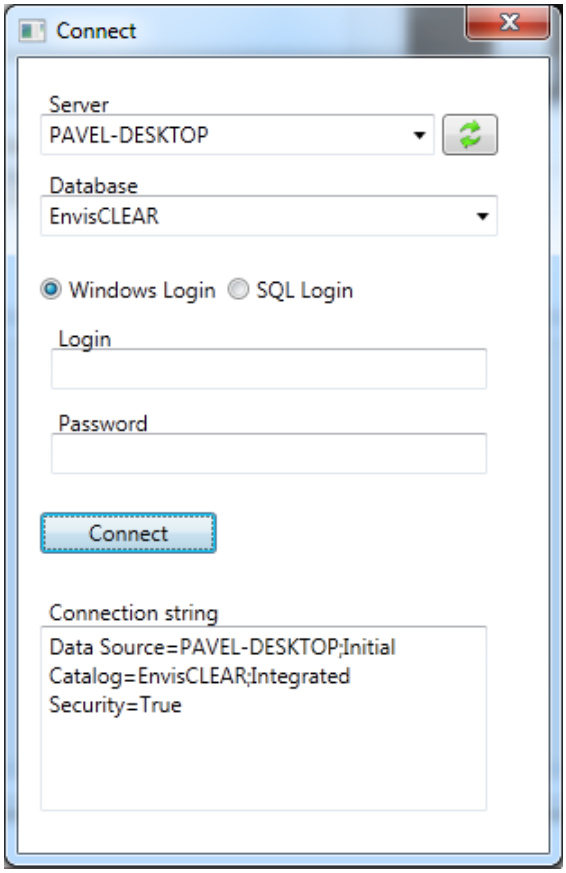
Popis aplikace

Čtvrthodinové maximum se počítá jako klouzavý průměr každých 15 minut ve vybraném intervalu z součtu hodnot příkonu. Proto aplikace používá metodu *akd_main_archive_detail* (popsaná v 3.3.3).



Obrázek 4.4: Power Checker Toolbar

Na obrázku 4.4 je ukázáno počáteční nastavení aplikace *Power Checker*. Tlačítko *Connect* slouží k vybrání správného *MSSQL* serveru a databáze. Dialog který se po jeho stisknutí zobrazí je na obrázku 4.5. Tento dialog vygeneruje připojovací řetězec a uloží ho, poté se může zavřít.



Obrázek 4.5: Power Checker Připojení

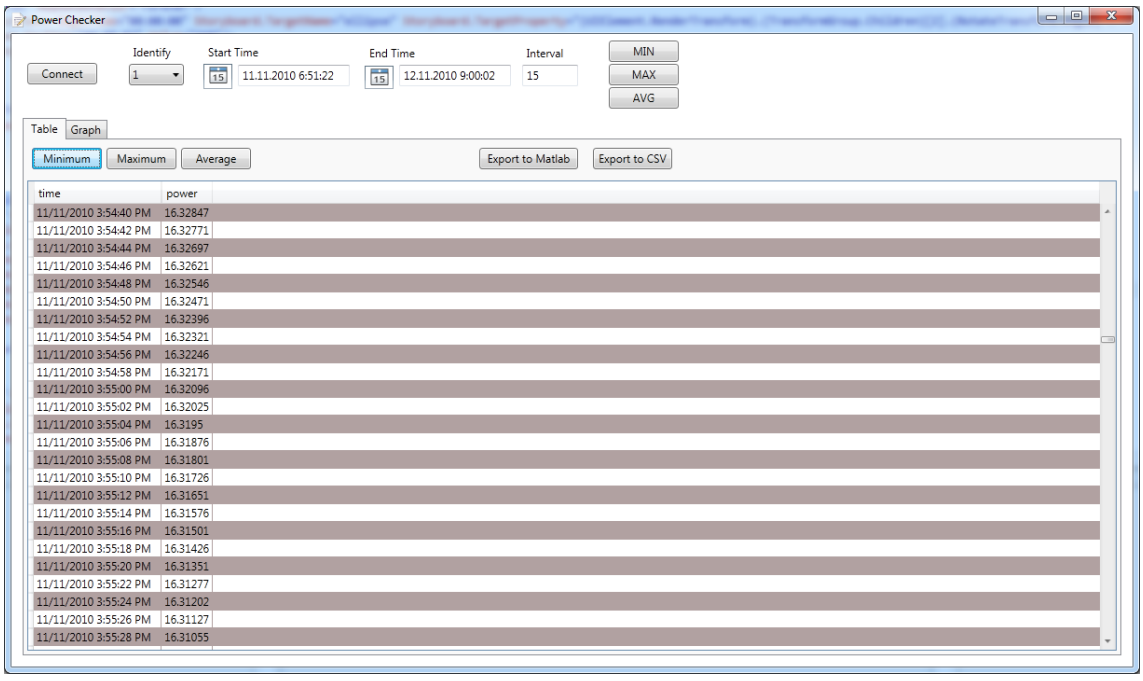
Následující tři prvky na obrázku 4.4 slouží k výběru již dobře známých parametrů. *Identify* k výběru konkrétního měření, *Start Time* určuje počáteční čas zkoumaného úseku a *End Time* konečný čas.

Textové pole *Interval* je v základu nastaveno na hodnotu 15 minut (**čtvrthodinové** maximum). Pokud by ale zkoumaný interval byl smluvně nastaven jiný, umožňuje aplikace tuto hodnoty měnit.

Tlačítka *MIN*, *MAX* a *AVG* slouží k výběru konkrétních dat. V databázi jsou pro konkrétní měření (které nějaký čas trvá) uchovávány minimální, maximální a průměrná naměřená hodnota. Pro potřeby čtvrthodinového maxima se používá tlačítko *AVG*. Stisk jednoho z těchto tlačítek spustí funkčnost aplikace, zavolá se konkrétní procedura, data se stáhnou a zobrazí do tabulky.

V prostředí s tabulkou se s nimi dá dělat různé statistické operace, popřípadě je exportovat do *MATLABu* nebo *CSV*. V prostředí s grafem se vykreslí graf, který umožňuje export do *PDF*.

Výsledná aplikace



Obrázek 4.6: Power Checker Table View

Obrázek 4.7: Power Checker Graph View

5. Závěr

Pomocí uložených procedur byla navržena datová vrstva AKD (3.3), ta usnadňuje přístup k datům v databázi aplikace *ENVIS* na technologii Microsoft SQL Server 2012. Tato vrstva se skládá z deseti hlavních procedur a dvaceti pěti pomocných procedur. Díky této vrstvě je umožněno používat komprimovaná binární data v databázi bez závislosti na dynamických knihovnách aplikace *ENVIS*.

Microsoft SQL Server má k dispozici dvě metody jak vytvořit uložené procedury *T-SQL* a *CLR* (3.1). Pro otestování efektivnosti jednotlivých metod a otestování případného zrychlení po přesunu výpočtů na stranu databázového serveru z klienta vznikla testovací aplikace *Stored Tester*. Tato aplikace umožňuje opakované volání uložených procedur, má hotové rozhraní pro výpočty na straně klienta a zaznamenává časy jednotlivých volání. Umožňuje zaznamenané časy exportovat do programu *MATLAB*.

Díky aplikaci *Stored Tester* bylo odhaleno, že uložené procedury vytvořené pomocí *T-SQL* jsou efektivnější, než uložené procedury vytvořené pomocí *CLR*. Proto je datová vrstva *AKD* navržena především pomocí *T-SQL* procedur. Tato technologie však neumožňuje použití práci ze zakódovanými binárními daty v databázi aplikace *ENVIS* (2.1.1). Proto procedury, které s těmito daty musí pracovat jsou vytvořeny pomocí méně efektivní technologie *CLR*.

Protože datová vrstva obsahuje i uložené procedury vytvořené pomocí *CLR* je její zavedení na MSSQL server složitější operace, proto vznikla instalační aplikace. Ta umožňuje jednoduchou instalaci *AKD* na databázový server, pomocí jednoduchého grafického prostředí. Tato instalační aplikace je k nalezení na CD příloze.

Dále byly vytvořeny testovací aplikace, které demonstrují nasazení datové vrstvy

v reálném prostředí mimo aplikaci *ENVIS*.

První tato aplikace vznikla pro Microsoft Report Server, jedná se o webovou stránku, která simuluje vytváření protokolu aplikace *ENVIS*. Pomocí *AKD* procedur shrnuje stav jednotlivých archivů a stav elektroměru za určité časové období pro konkrétní přístroj. (4.1).

Druhá aplikace je normální desktopová aplikace vytvořená pomocí technologie *WPF* na frameworku *.NET 4.5*. Její účel je kontrolovat takzvané čtvrthodinové maximum u spotřeby elektrické energie. (4.2.1).

Obě aplikace jsou plně funkční a jejich výstupy odpovídají výstupům aplikace *ENVIS*.

Dodatky

Použitá literatura

[1] Mike Hotek: Microsoft SQL Server 2008. Computer Press, Brno, 2009. ISBN 978-80-251-2309-6

[2] Vidya Vrat Agarwal, James Huddleston: Databáze v C# 2008. Computer Press, Brno, 2009. ISBN 978-80-251-2309-6

[3] KMB Systems. Manual SMV, SMP, SMPQ [online]. KMB Systems, Liberec, 2009 [cit. 2011-04-08]. Dostupné z: http://www.kmb.cz/07/doc/SMV_SMP_SMPQ-Manual-v4-cze.pdf

[4] Christian Nagel: C# 2008 Programujeme profesionálně. Computer Press, Brno, 2009. ISBN 978-80-251-2401-7

[5] Robert E. Walters, Michael Coles, Robert Rae, Fabio Ferracchiati, Donald Farmer: Mistrovství v Microsoft SQL Server 2008. Computer Press, 2009. ISBN 978-80-251-2329-4

[6] MICROSOFT. Stored Procedures (Database Engine) [online]. 2011 [cit. 2012-04-28]. Dostupné z: http://msdn.microsoft.com/en-us/library/ms190782.aspx

[7] MICROSOSFT. CREATE PROCEDURE [online]. 2007 [cit. 2012-04-28]. Dostupné z: http://msdn.microsoft.com/en-us/library/aa258259(v=sql.80).aspx

[8] MICROSOSFT. How to: Deploy SQL Server CLR Integration Database Project Items to a SQL Server [online]. 2007 [cit. 2012-04-28]. Dostupné z: http://msdn.microsoft.com/en-us/library/dahcx0ww.aspx

[9] MICROSOSFT. CLR Stored Procedures [online]. 2007 [cit. 2012-04-28]. Dostupné z: http://msdn.microsoft.com/en-us/library/ms131094.aspx

[10] Limitations of SQL Server Express. DOTNET SPIDER. [online]. [cit. 2013-04-29]. Dostupné z: http://www.dotnetspider.com/tutorials/SqlServer-Tutorial-158.aspx

[11] BUDÍKOVÁ, Marie. Statistika II: distanční studijní opora [online]. Brno, 2006 [cit. 2012-05-02]. Dostupné z:

http://econ.muny.cz/data/PMSTII/PMSTII_dso.pdf. Skripta. Masarykova universita.

[12] MICROSFT. Reporting Services (SSRS). [online]. 2012 [cit. 2013-04-09]. Dostupné z: http://msdn.microsoft.com/en-us/library/ms159106.aspx

[13] MICROSOFT. Reporting Services Report Server (SSRS). [online]. 2012. [cit. 2013-04-09]. Dostupné z: http://msdn.microsoft.com/en-us/library/ms157231.aspx

[14] MICROSOFT. How to Configure Reporting Services [online]. 2007. [cit. 2013-04-09]. Dostupné z: http://msdn.microsoft.com/en-us/library/aa545752(v=cs.70).aspx

[15] MICROSOFT. Create a Basic Table Report (SSRS Tutorial) [online]. 2012. [cit. 2013-04-09]. Dostupné z: http://msdn.microsoft.com/en-us/library/3b539b4b-26f2-4c0b-b506-80f175679a46

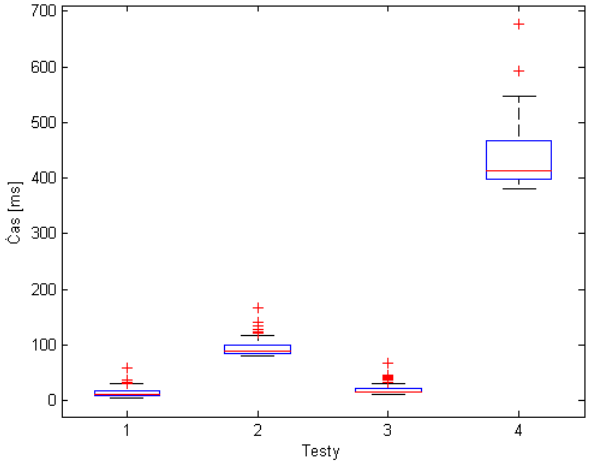
[16] MAJDA, František. Čtvrthodinové maximum. [online]. 2012. [cit. 2013-04-20]. Dostupné z: http://www.odbornecasopisy.cz/index.php?id_document=38165

Seznam obrázků

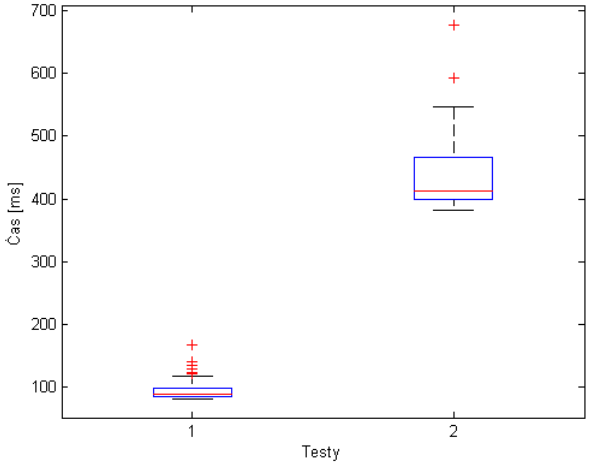
A. Dodatečné grafy

A.1 Malá data

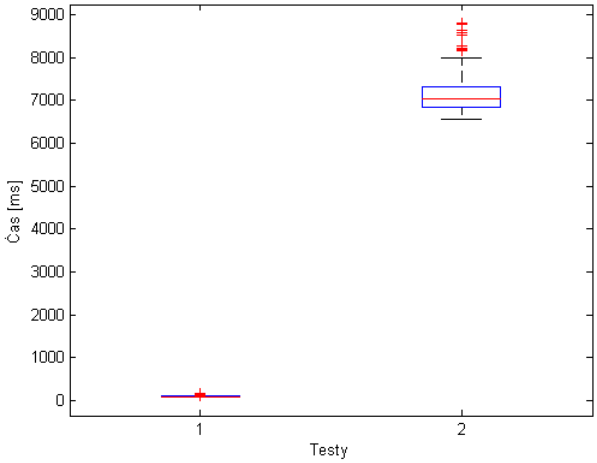
A.1.1 Přehled archivů, spotřeba energie TSQL vs. CLR



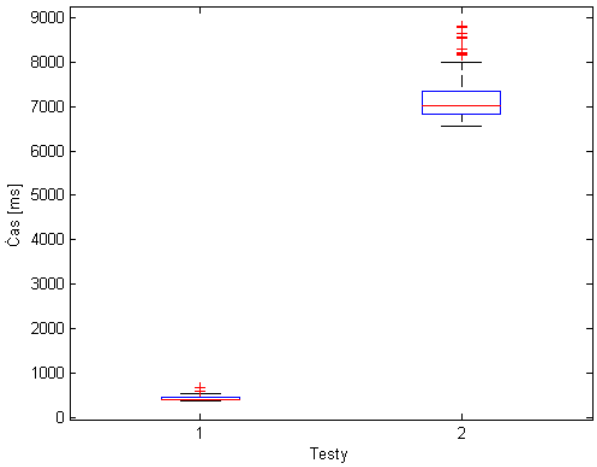
A.1.2 Spotřeba energie - TSQL vs. CLR



A.1.3 Spotřeba energie - TSQL vs. Klient

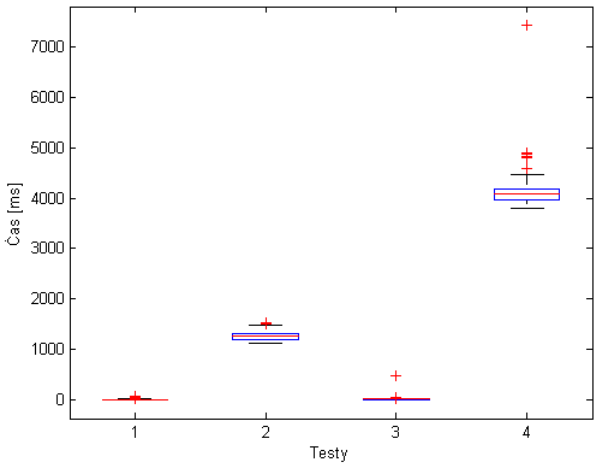


A.1.4 Spotřeba energie - CLR vs. Klient

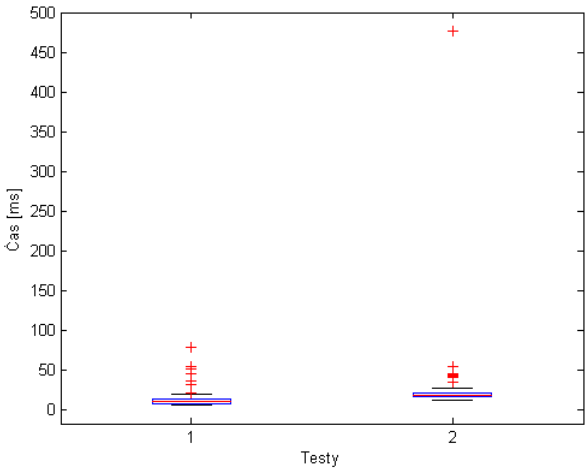


A.2 Velká data

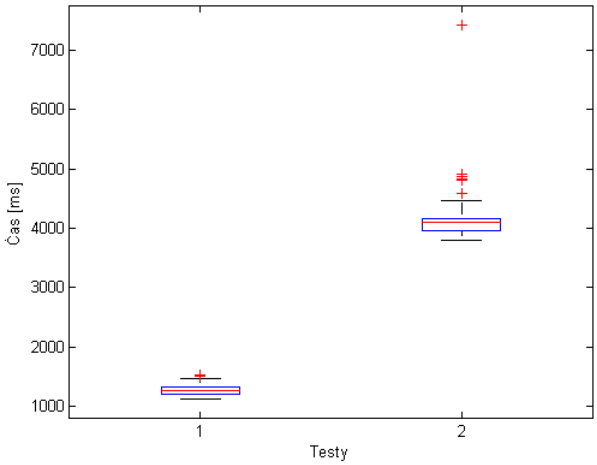
A.2.1 Přehled archivů, spotřeba energie TSQL vs. CLR



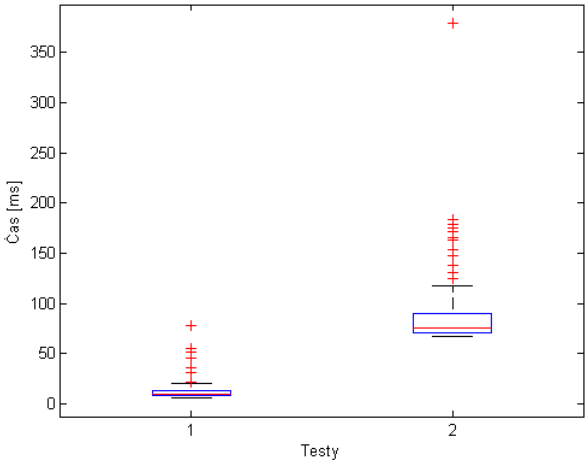
A.2.2 Přehled archivů - TSQL vs. CLR



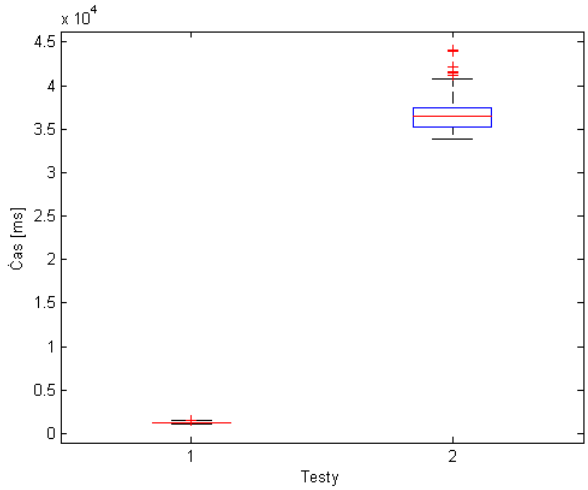
A.2.3 Spotřeba energie - TSQL vs. CLR



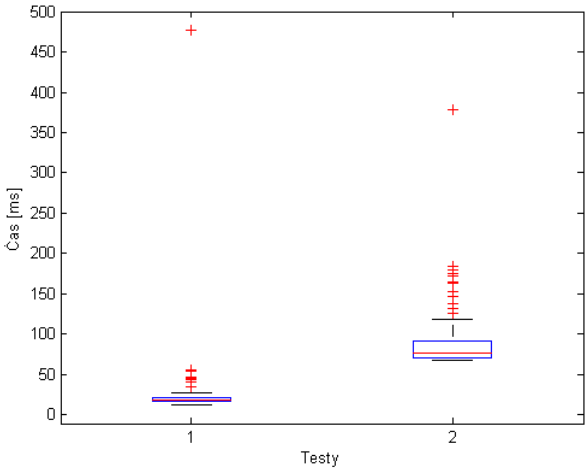
A.2.4 Přehled archivů - TSQL vs. Klient



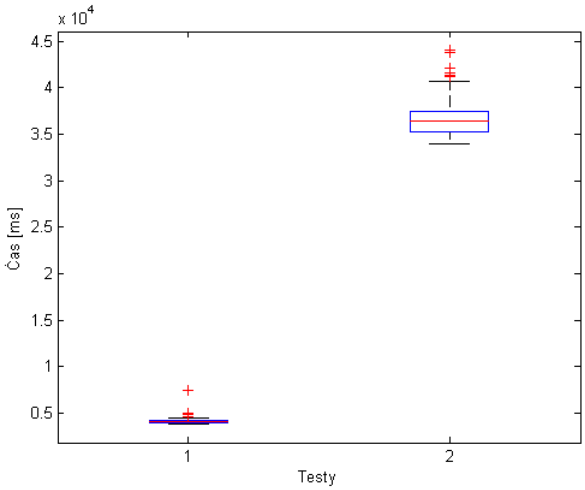
A.2.5 Spotřeba energie - TSQL vs. Klient



A.2.6 Přehled archivů - CLR vs. Klient



A.2.7 Spotřeba energie - CLR vs. Klient



B. AKD - Technická dokumentace

B.1 Poznámky

Vstupní parametry identify a time korespondují s primárními klíči z SmpArchiveMainDB (popřípadě jinými archivy SmpArchiveElmerDB atd..)
identify = keyMeasureName je číslo daného měření
time = keyTime čas ve kterém je měření uloženo
Čas zadávat ve tvaru: RRRR-MM-DD HH:mm:ss.sss

B.2 Spotřeba energie - Elektroměr

B.2.1 AKD_ENERGY_CONSUMPTION_LIST - akd_energy_consumption_list

- identify (int)
- starttime (datetime)
- endtime (datetime)
- args (nvarchar 4000)

Vrátí tabulku obsahující stav elektroměru v daném časovém rozhraní. Seznam požadovaných hodnot je zadán pomocí parametru args.

TIME (datetime) | ARGS (double)

Args se zadává jako řetězec. Jednotlivé veličiny jsou oddělené čárkou.
Povolené args jsou: I, E, L, C, IT, ET, LT, CT
Pro vrácení hodnot Importu a Exportu by args tedy bylo 'I,E'.
Args není case sensitive.

B.2.2 AKD_ENERGY_CONSUMPTION - akd_energy_consumption

- identify (int)
- starttime (datetime)

- endtime (datetime)

Vrátí tabulku shrnující informace o spotřebě energie z dat elektroměru.
První řádek jsou hodnoty ze start time. Druhý z end time. Třetí je rozdíl prvních dvou. Další řádky jsou hodnoty relativní změny v časovém intervalu 2 hodin.

CAS(datetime)	IMPORT(double)	EXPORT(double)
INDUKTIVNI (double)	KAPACITNI (double)	

B.2.3 AKD_ENERGY_CONSUMPTION_LINE - akd_energy_consumption_line

- identify (int)
- time (datetime)

Vrací hodnoty elektroměru v konkrétním čase.

CAS(datetime)	IMPORT(double)	EXPORT(double)
INDUKTIVNI (double)	KAPACITNI (double)	

B.2.4 AKD_ENERGY_CONSUMPTION_DATETABLE - akd_energy_consumption_datetable

- identify (int)

Vrací minimální a maximální datum pro dané identify v elektroměru.

Prvni (datetime)	Posledni (datetime)

B.2.5 AKD_ENERGY_CONSUMPTION_IDENTIFYLIST - akd_energy_consumption_identifylist

Vrací seznam všech identify v elektroměru.

IDENTIFY (int)

B.3 Hlavní archiv

B.3.1 AKD_MAIN_ARCHIVE_DETAIL - akd_main_archvie_detail

- identify (int)
- starttime (datetime)
- endtime (datetime)
- args (nvarchar 4000)

Vrátí tabulku obsahující rozkódovaná podrobná data z hlavního archivu v daném časovém rozhraní. Seznam požadovaných hodnot je zadán pomocí paramtru args.

TIME (datetime) | ARGS (double)

Args se zadává jako řetězec. Jednotlivé veličiny jsou oddělené čárkou.
Argumenty fungují pro 4 hodnoty (v napětí třeba u1,u2,u3,u4).
Args není case sensitive.
Povolné args jsou:
X má hodnoty 1,2,3,4
Napětí
AVG_ULNX, MIN_ULNX, MAX_ULNX
AVG_ULLX, MIN_ULLX, MAX_ULLX
Proud
AVG_ILX, MIN_ILX, MAX_ILX
THD
AVG_UTHDX, MIN_UTHDX, MAX_UTHDX
AVG_ITHDX, MIN_ITHDX, MAX_ITHDX
Výkony
AVG_PX, MIN_PX, MAX_PX
AVG_PminusX, MIN_PminusX, MAX_PminusX
AVG_PfhX, MIN_PfhX, MAX_PfhX
AVG_PfhminusX, MIN_PfhminusX, MAX_PfhminusX
AVG_QX, MIN_QX, MAX_QX
AVG_QminusX, MIN_QminusX, MAX_QminusX
AVG_QfhdX, MIN_QfhdX, MAX_QfhdX
AVG_QfhdminusX, MIN_QfhdminusX, MAX_QfhdminusX
AVG_SX, MIN_SX, MAX_SX
AVG_DX, MIN_DX, MAX_DX
Frekvence
AVG_F, MAX_F, MIN_F, AVG_TEMP, MAX_TEMP, MIN_TEMP, F_MOSTLY, F_ALWAYS, F_ABOVE, F_BELOW

B.3.2 AKD_MAIN_ARCHIVE_DATETABLE - akd_main_archive_datetable

- identify (int)

Vrací minimální a maximální datum pro dané identify v hlavních archivu.

Prvni (datetime) | Posledni (datetime)

B.3.3 AKD_MAIN_ARCHIVE_IDENTIFYLIST - akd_main_archive_identifylist

Vrací seznam všech identify v hlavním archivu.

IDENTIFY (int)

B.3.4 AKD_MAIN_ARCHIVE_HARMONICS - akd_main_archive_harmonics

- identify (int)
- starttime (datetime)
- endtime (datetime)
- args (nvarchar 4000)
- args (int)

Vrátí tabulku obsahující rozkódovaná data harmonických z hlavního archivu v daném časovém rozhraní. Args1 určuje, které harmonické (proud, napětí) a args2 určuje kterou hodnotu konkrétně (u napětí třeba u1,u2,u3 atd...). Args2 je zadáváno pouze číslem.

TIME (datetime) | Všechny existující harmonické pro kombinaci args1 a args2 (double)

Args1 se zadává jako řetězec. Vždy akceptuje jenom jednu možnost (pokud jich bude zadáno více, bere se celý řetezec jako jedna možnost).
Args1 není case sensitive.
Povolené hodnoty pro args1 jsou:

- HARMUR
- HARMIR
- INTERHARMUR
- INTERHARMIR
- HARMUHELR

Args2 se zadává jako číslo. Většinou v rozmezí 1 až 4.
Třeba u napětí je 1 pro u1 atd...