**Univerzita Pardubice**
**Fakulta ekonomicko-správní**

# Analytické funkce Oracle

**Josef Pirkl**

**Bakalářská práce**
**2008**

Poděkování

Děkuji Ing. Miloslavu Hubovi, PhD. za hodnotné rady a odborné vedení během mé práce.

## SOUHRN

Práce popisuje principy činnosti analytických funkcí, rozdíl řešení problémů klasickým způsobem a za pomoci analytických funkcí. Pro porovnání CPU trvání dotazu provádí testování trvání pomocí klasického SQL dotazu a analytické funkce, kdy oba tyto přístupy vrací stejnou výslednou množinu dat. Následně je experiment rozšířen i na sledování vyhodnocení dotazů interním Oracle analyzátorem s porovnáním očekávaných nákladů těchto operací. Závěrečná část se zaměřuje na hledání alternativ analytických funkcí v konkurečním prostředí Microsoft SQL serveru.

## KLÍČOVÁ SLOVA

analytické funkce, Oracle, SQL, rychlost odezvy

## TITLE

Analytic functions in Oracle

## ABSTRACT

The work describes the principles of the analytic functions operation, the difference between problems solving in a classic way and by using analytic functions. For comparison of CPU time of demand the testing of the demand time is executed by a classic SQL demand and analytic function, when both these ways return the same final set of data. Then the experiment is also extended for following up the demands evaluation by the internal analyzer Oracle with the comparison of the expected costs of these operations. The final part is focused on the finding of analytic functions alternatives in the competitive environment of Microsoft SQL server.

## KEYWORDS

analytical functions, Oracle, SQL, response swiftness

# Obsah

# Úvod

Ačkoli je jazyk SQL i přes svou jednoduchost velmi mocným nástrojem, existují dotazy, které se s jeho pomocí řeší velmi komplikovaně. Typickým příkladem může být výpis, který bude vedle hodnot prodejů obsahovat také jejich průběžnou kumulaci s celkovou hodnotou na posledním řádku. S pomocí klasického SQL je tento problém řešitelný pouze se zavedením poddotazu, který bude umístěn v rámci základního příkazu SELECT. Analytické funkce (AF) však přinášejí funkcionalitu, s níž lze tento a jiné požadavky vyřešit mnohem efektivněji, zejména z hlediska složitosti dotazu.

Analytické funkce tedy **představují určité rozšíření možností jazyka SQL**, pomocí kterých je možné řešit problémy, jejichž řešení by bez jejich použití bylo příliš komplikované. Vedle toho, že SQL kód opticky zjednodušují, jsou obecně výsledky dotazu s jejich použitím vráceny mnohem rychleji [4].

Za příchodem analytických funkcí do systému Oracle stojí stále se zvyšující potřeba rozšířit možnosti Business Intelligence (BI) [10]. BI může být definováno jako manipulace s daty, potřebnými k získání požadované informace pro obchodní a jiná rozhodnutí, přičemž kritickou roli v této fázi sehrává datová analýza [3]. Úkolem BI je poskytnout analytické nástroje pro podporu Data Warehousingu na bázi klasických transakčních OLTP systémů [10].

Cílem této práce je popsat principy činnosti analytických funkcí, rozdíl řešení problémů klasickým způsobem[1] a za pomoci analytických funkcí. Pro porovnání trvání dotazu je provedeno testování trvání pomocí klasického SQL dotazu a analytické funkce, kdy oba tyto přístupy vrací stejnou výslednou množinu dat. Následně je experiment rozšířen i na sledování vyhodnocení dotazů interním Oracle analyzátorem s porovnáním očekávaných nákladů těchto operací. Závěrečná část se zaměřuje na hledání alternativ analytických funkcí v konkurečním prostředí Microsoft SQL serveru.

[1] "Klasickým způsobem" jsou míněné SQL dotazy bez použití analytických funkcí.

# 1. Analytické funkce

První verzí Oracle, ve které se poprvé objevily analytické funkce, byla verze 8i [4]. Historie jejich zavádění byla:

- v **Oracle 8i Release 1** (rok 1999 [13]) se objevuje rozšíření GROUP BY sekce o volitelné CUBE a ROLLUP [4],
- **Oracle 8i Release 2** (rok 2000 [13]) přináší další skupiny analytických funkcí, které nárokově vyhovují stále rostoucím potřebám Bussiness intelligence. Tyto nové funkce byly začleněny do standardu SQL-99 [4],
- **Oracle 9i** (rok 2001 [13]) je významně rozšířena o analytické funkce pořadí, agregační funkce postavené nad oknem, funkce LAG/LEAD, FIRST/LAST [4],
- **Oracle 10g** (rok 2004 [13]) obsahuje již téměř 100 analytických funkcí použitelných v různých kombinacích [2].

Hlavní výhody, které používání analytických funkcí přináší, lze formulovat následovně [4]:

- lepší výkon dotazu související s jeho optimalizací,
- vyšší produktivita vývojáře, jenž dostává prostředky, pomocí nichž může provádět dotazy, jejichž vytvoření bylo předchozími prostředky velmi obtížné,
- minimální nároky na zaškolení, které souvisí s tím, že analytické funkce jsou přímo začleněny do již známého SQL a funkcionalita je pouze zvyšována.

## 1.1 Nové termíny analytických funkcí

Analytické funkce rozšiřují klasický jazyk SQL. S jejich nástupem se začínají používat některé nové techniky a termíny, které s jejich zavedením souvisí [8]:

- Oblast (angl. partition)
- Okno (angl. window)
- Aktuální řádek (angl. current row)

## Oblast

Každá výsledná množina dat dotazu SQL může být pomocí analytických funkcí dále volitelně rozdělena do tzv. *Oblastí*. Tyto *Oblasti* jsou definovány až po vytvoření skupin vzniklých dle bloku GROUP BY dotazu [8], což je jeden z nejdůležitějších přínosů zavedení analytických funkcí. Výsledky, které jsou navráceny v rámci výpočtu agregačních funkcí GROUP BY skupinování, lze pak vybranými analytickými funkcemi dále zpracovávat. Jestliže není žádná oblast explicitně definována, pracuje se se všemi vrácenými daty tak, jako by se jednalo o oblast jedinou [7].

Dotaz 1 demonstruje význam *Oblasti*. Ačkoli je agregační funkcí SUM provedeno vyčíslení pole PRICE_TOTAL za jednotlivá období a segmenty prodeje; volitelným definováním *Oblasti* dle pole PERIOD je pro každý řádek navíc vyčíslena celková suma za každé období.

```
select a.*, sum( price_total ) over ( partition by period ) as
price_total_period
from
(
select period, segment, sum( price_total ) as price_total
from ba.sale
where
period like '2007%'
group by period, segment
order by period, segment
) a
```

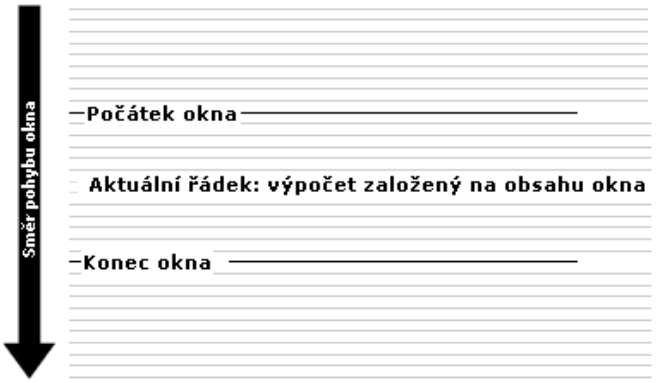
```
PERIOD SEGMENT PRICE_TOTAL            PRICE_TOTAL_PERIOD
------ ------- ---------------------- ----------------------
200701 A       1889257                6966219
200701 B       2477991                6966219
200701 C       2598971                6966219
200702 A       2663161                9795931
200702 B       3950639                9795931
200702 C       3182131                9795931
(výpis byl zkrácen)
21 rows selected
```

Dotaz 1: Suma za období a prodejní segmenty v roce 2007[2]

## Okno

*Okno* představuje náhled na skupinu řádků. Jeho velikost je udána počátečním a koncovým řádkem. Pevně ji lze nastavit přes celou oblast, výhoda *Okna* spočívá však v dynamice, kterou může být jeho velikost popsána [8]. V takovém případě pak lze definovat velikost *Okna* například rozsahem 3 řádky před a 3 řádky za aktuální řádek. Takto určeného *Okna* může být využito pro výpočty pohyblivých agregací, kde typickým příkladem jsou klouzavé průměry. Vzhled *Okna* ilustruje Obrázek 1.

[2] Zdroj: Vlastní.



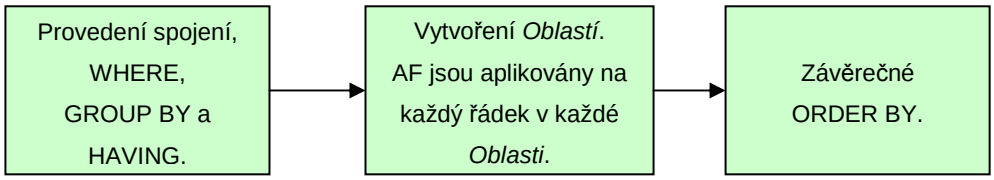

Obrázek 1: Definovaná oblast pohyblivého okna v řádcích [8]

### Aktuální řádek

Všechny výpočty analytických funkcí jsou založeny na hodnotě a pozici aktuálního řádku v *Oblasti* [8]. Tento aktuální řádek pak slouží jako výchozí bod určující startovací a konečný řádek pro *Okno*.

## 1.2 Zpracování analytických funkcí

Zjistí-li Oracle při zpracování SQL dotazu, že byla použita analytická funkce, má zpracování tohoto dotazu určitá specifika. Nad každým SQL dotazem dochází nejdříve k provedení všech spojení, podmínek výběru WHERE, vytváření skupin GROUP BY, případně pak i k aplikaci podmínek na GROUP BY skupiny dle klauzule HAVING [4].
S použitím GROUP BY dochází k rozdělení výsledků dotazu do jednotlivých skupin řádků [3]. Následně jsou ale na takto vráceném výsledku dle pokynů analytických funkcí vytvořeny jednotlivé *Oblasti*, v jejichž rámci jsou tyto funkce aplikovány na každý jejich řádek. Zde tedy dochází jakémusi dalšímu skupinování, resp. další hlubší práci se skupinami [4]. Nakonec je aplikováno závěrečné setřídění dat ORDER BY. Vypočítané hodnoty analytických funkcí zůstávají však uložené u řádků v původním pořadí, tzn. předchozí případné ORDER BY nezmění jejich pozici [4]. Postup zpracování AF od okamžiku zadání dotazu reprezentuje obrázek Obrázek 2.




Obrázek 2: Procesní tok zpracování AF [8]

## 1.3 Stručný přehled skupin analytických funkcí

V rámci analytických funkcí Oracle lze definovat několik kategorií [8]:

- funkce pořadí (angl. ranking functions),
- agregační funkce nad oknem (angl. windowing aggregate functions),
- LAG/LEAD funkce,
- FIRST/LAST funkce,
- rozšíření funkcionality GROUP BY skupinování,
- další skupiny funkcí (jako jsou funkce pro zobrazení podílu ze sumy hodnot, pro podporu výpočtu parametrů lineárních regresí, funkce testování statistických hypotéz, atd.).

### Funkce pořadí

Tyto analytické funkce obsahují zabudované mechanismy pro určení pořadí hodnot [8]. To je účelné právě v prostředí BI.

Dotaz 2 zobrazuje způsob, jakým lze určit pořadí výše prodejů jednotlivých období roku 2007. Pro funkci RANK je zadáno, že k určení pořadí má být využito pole PRICE_TOTAL, a to v sestupném pořadí dle výše pole za každé období. Tím je zajištěno, že funkce RANK vrátí hodnotu 1 pro nejvyšší hodnotu prodeje, 2 pro následující, atd. Celkové třídění výstupního souboru je pak dle období uskutečnění prodeje.

```
select a.*, rank() over( order by price_total desc ) as rank
from
(
select period, sum( price_total ) as price_total
from pirkl.sale
where
period like '2007%'
group by period
) a
order by period
```

```
PERIOD     PRICE_TOTA RANK
---------- ---------- ----------
    200701    6576758          7
    200702   10012391          1
    200703    8500420          3
    200704    8323706          4
    200705    7262579          6
    200706    9380109          2
    200707    7364287          5
7 rows selected.
```

Dotaz 2: Pořadí měsíčních prodejů s AF RANK[3]

[3] Zdroj: Vlastní.

## Agregační funkce nad oknem

Tyto analytické agregační funkce umožňují provádět výpočty kumulativních, pohyblivých a centrovaných agregací [8].

Dotaz 3 definuje nový sloupec PRICE_TOTAL_KUM jako kumulační pro pole PRICE_TOTAL. Pořadí načítání hodnot není v rámci sekce OVER změněno, zůstává vzestupné. Hodnota ve sloupci PRICE_TOTAL_KUM tak bude stoupat s každým obdobím.

```
select a.*, sum( a.price_total ) over( order by period ) price_total_kum
from
(
select period, sum( price_total ) as price_total
from ba.sale
where
period like '2007%'
group by period
) a
order by period
```

```
PERIOD PRICE_TOTAL            PRICE_TOTAL_KUM
------ ---------------------- ----------------------
200701 6966219                6966219
200702 9795931                16762150
200703 8518085                25280235
200704 7646368                32926603
200705 8115355                41041958
200706 9925669                50967627
200707 10858281               61825908
7 rows selected
```

Dotaz 3: Použití AF SUM pro kumulační součet měsíčních prodejů[4]

## LAG/LEAD funkce

Pomocí funkcí LAG / LEAD je možné přistupovat k řádkům výpisu v určité konkrétní relativní vzdálenosti od řádku současného [8]. Tak je například možné získat hodnotu u řádku předchozího či následujícího.

Výhodu této funkcionality lze mimo již zmíněné vyšší přehlednosti spatřit v tom, že oproti dohledání hodnoty poddotazem nedochází k vytvoření nového spojení na stejnou tabulku, čímž je zvýšena i rychlost zpracování.

Pomocí přístupu k relativní pozici současného řádku je prezentován výpočet vývoje řetězového indexu měsíčních prodejů. Dotaz 4 za pomocí funkce LAG přistupuje k předchozímu řádku a získává jeho hodnotu. Následně je vypočítán řetězový index[5].

[4] Zdroj: Vlastní.

[5] Uvedený příklad pro zjednodušení neošetřuje variantu, kdy dělitel je roven nule.

```
select a.*, round( a.price_total / previous, 4 ) as index_value
from
(
select a.*, lag( price_total, 1 ) over ( order by period ) as previous
from
(
select period, sum( price_total ) as price_total
from ba.sale
where
period like '2007%'
group by period
) a
) a
```

```
PERIOD PRICE_TOTAL           PREVIOUS             INDEX_VALUE
------ -------------------- -------------------- -----------
200701 6966219
200702 9795931               6966219               1,4062
200703 8518085               9795931               0,8696
200704 7646368               8518085               0,8977
200705 8115355               7646368               1,0613
200706 9925669               8115355               1,2231
200707 10858281              9925669               1,094
7 rows selected
```

Dotaz 4: Řetězcový index měsíčních prodejů s AF LAG[6]

## FIRST/LAST funkce

FIRST/LAST skupina funkcí dovoluje určit pořadí v rámci datové množiny a následně pak pracovat s nejlépe/nejhůře ohodnoceným záznamem tohoto pořadí. Tyto funkce ohodnotí určitý sloupec A a vrací výsledek agregační funkce nad sloupec B právě dle tohoto ohodnoceného pořadí. U dotazů tohoto typu tak dochází ke zvýšení výkonnosti, protože se zabraňuje dalším dotazům do tabulky nebo poddotazům [8]. Typickým příkladem pro tento typ dotazu je požadavek na zjištění nejvyšší minimální ceny za každou kategorii produktu [3].

Dotaz 5 vypisuje ve sloupci PRICE_TOTAL celkovou hodnotu prodeje za období každého měsíce. Následně dochází k použití analytické funkce FIRST.

- Závorky uvedené za sekcí KEEP specifikují, že se prohledává první den každého období.
- Následně je z tohoto prvního dne každého období zobrazena minimální a maximální hodnota, která vstoupila do výpočtu agregační funkce.

[6] Zdroj: Vlastní.

```
select period, sum( price_total ) as price_total,
min( price_total ) keep ( dense_rank first order by date_ ) as min_first_day,
max( price_total ) keep ( dense_rank first order by date_ ) as max_first_day
from ba.sale
where
period like '2007%'
group by period
```

```
PERIOD PRICE_TOTAL       MIN_FIRST_DAY          MAX_FIRST_DAY
------ ----------------- ---------------------- --------------
200701 6966219           4671                   24270
200702 9795931           119                    28852
200703 8518085           0                      36420
200704 7646368           10                     98920
200705 8115355           38                     42495
200706 9925669           2096                   66858
200707 10858281          0                      56000
7 rows selected
```

Dotaz 5: Minimální a maximální hodnoty prvního dne pomocí AF FIRST[7]

Výsledky Dotaz 5 tak ukazují, že za první den, spadající do období 200701, je minimální nalezená hodnota 4671; maximální nalezená hodnota pak 24270.

## Rozšíření funkcionality GROUP BY

Přináší lepší podporu agregací [4] a mezisoučtů do GROUP BY členění.

Dotaz 6 ukazuje výpočet mezisoučtu po každém období a za všechny prodejní segmenty tohoto období. Na závěr je pak uveden celkový součet objemu těchto prodejů. K tomuto dotazu je použito ROLLUP rozšíření bloku GROUP BY.

```
select period, segment, sum( price_total ) as price_total
from ba.sale
where
period in ( '200706', '200707' )
group by rollup( period, segment )
```

```
PERIOD SEGMENT PRICE_TOTAL
------ ------- ----------------------
200706 A       3234617
200706 B       3191298
200706 C       3499754
200706         9925669
200707 A       3426543
200707 B       3610315
200707 C       3821423
200707         10858281
               20783950
9 rows selected
```

Dotaz 6: Mezisoučty s využitím GROUP BY ROLLUP[7]

[7] Zdroj: Vlastní.

# 2. Porovnání rychlosti odezvy analytických funkcí s klasickým přístupem

Kritériem pro testování je délka trvání provedení dotazu. Tento časový interval je v rámci tohoto testu vyjádřen ve vteřinách tzv. CPU času. Takto vyjádřený čas není ovlivněn případným čekáním na obsazené systémové prostředky. V případě vzniku existence velkého rozdílu mezi CPU časem a skutečným uplynulým časem, nasvědčuje toto o častém čekání na přidělení systémových zdrojů [7].

Během těchto testů je vyšetřována agregační funkce SUM proti agregační analytické funkci SUM na dotazech, které vrací shodný výsledek.

Jsou provedena tato měření:

- srovnání klasické agregační funkce SUM proti analytické funkci SUM, bez použití indexů nad tabulkou (varianta A - B),
- srovnání klasické agregační funkce SUM proti analytické funkci SUM, s použitím indexů nad tabulkou (varianta C - D),
- srovnání analytické funkce SUM na tabulce bez indexů/s indexy (varianta B -D).

## 2.1 Trvání SUM proti AF SUM – varianty bez indexů

### 2.1.1 Varianta A – klasická agregační funkce SUM bez indexů

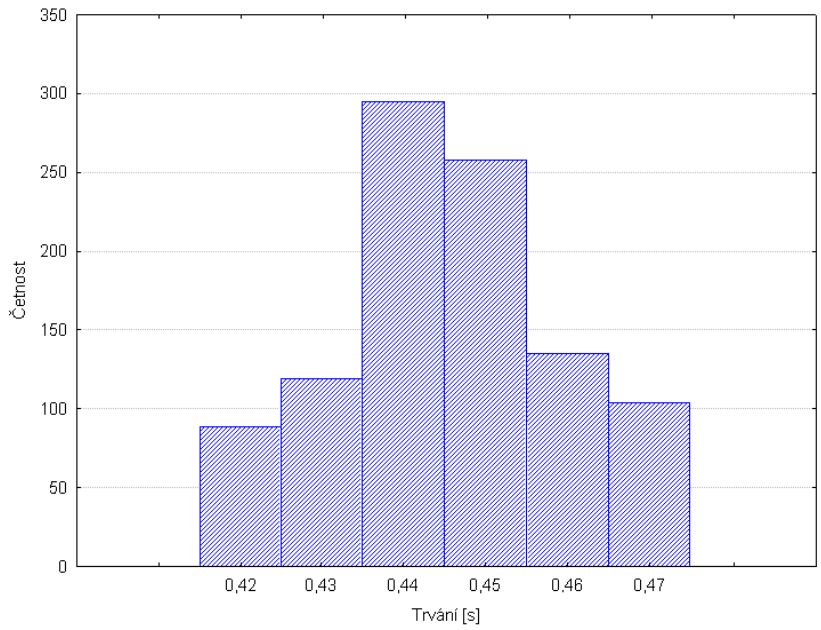
Pro měření CPU trvání v této variantě byl použit Dotaz 7.

```
select a.*, ( select sum( price_total ) from sale
              where
              period = a.period
            ) as total_period
from
(
select period, segment, sum( price_total ) as price_total
from sale
group by period, segment
order by period, segment
) a
```

Dotaz 7: Dotaz pro klasickou agregační funkci SUM poddotazem[8]

Histogram četností naměřených CPU trvání pro 1000 provedených měření ukazuje Graf 1.

[8] Zdroj: Vlastní.




Graf 1: Histogram naměřených CPU trvání pro Dotaz 7[9] (1000 měření)

Ze získaných měření byly vypočítány tyto charakteristiky vzorku dat (Tabulka 1).

Tabulka 1: Varianta A - charakteristiky získané ze vzorku dat (1000 měření)[10]

| Charakteristika | Hodnota |
|---|---|
| $\bar{t}_{BEZ\_INDEXU}$ (průměr) | 0,4454 [s] |
| $s_{BEZ\_INDEXU}$ (směrodatná odchylka) | 0,0138 [s] |

### 2.1.2 Varianta B – AF SUM bez indexů

Ve variantě B je místo poddotazu s agregační funkcí SUM použita analytická funkce SUM. Vrácená data (nikoli naměřená trvání) jsou shodná, jako u dotazu varianty A. Variantu B představuje Dotaz 8.

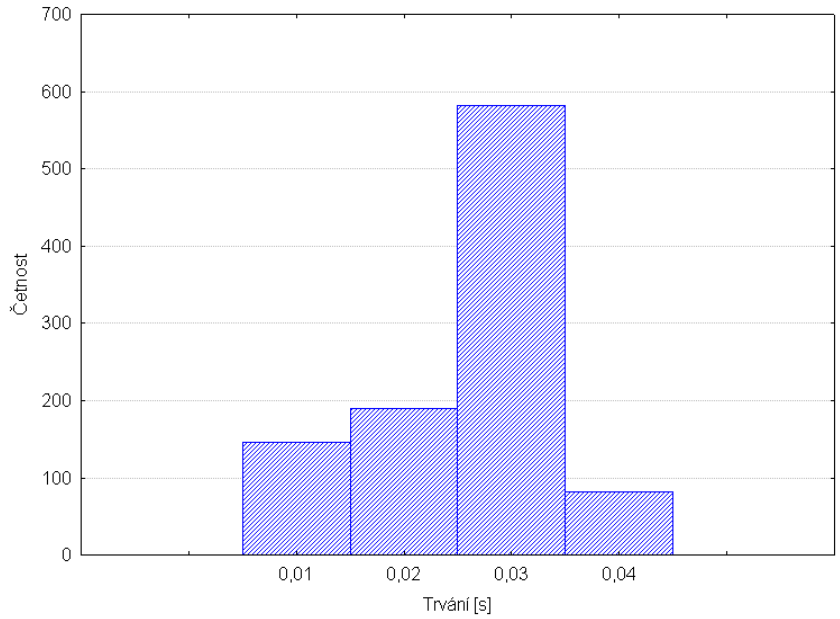
```
select a.*, sum( price_total ) over( partition by period )
from
(
select period, segment, sum( price_total ) as price_total
from sale
group by period, segment
order by period, segment
) a
```

Dotaz 8: Dotaz pro agregační AF SUM[10]

Histogram četností naměřených CPU trvání pro 1000 provedených měření ukazuje Graf 2. Následně vypočítané charakteristiky tohoto vzorku dat pak Tabulka 2.

[9] Zdroj: Dávkové měření CPU trvání přes vlastní SQL skript.

[10] Zdroj: Vlastní.




Graf 2: Histogram naměřených CPU trvání pro Dotaz 8 [11] (1000 měření)

Tabulka 2: Varianta B - charakteristiky získané ze vzorku dat (1000 měření)[12]

| Charakteristika | Hodnota |
|---|---|
| $\bar{t}_{AF\_BEZ\_INDEXU}$ (průměr) | 0.026 [s] |
| $s_{AF\_BEZ\_INDEXU}$ (směrodatná odchylka) | 0.0083 [s] |

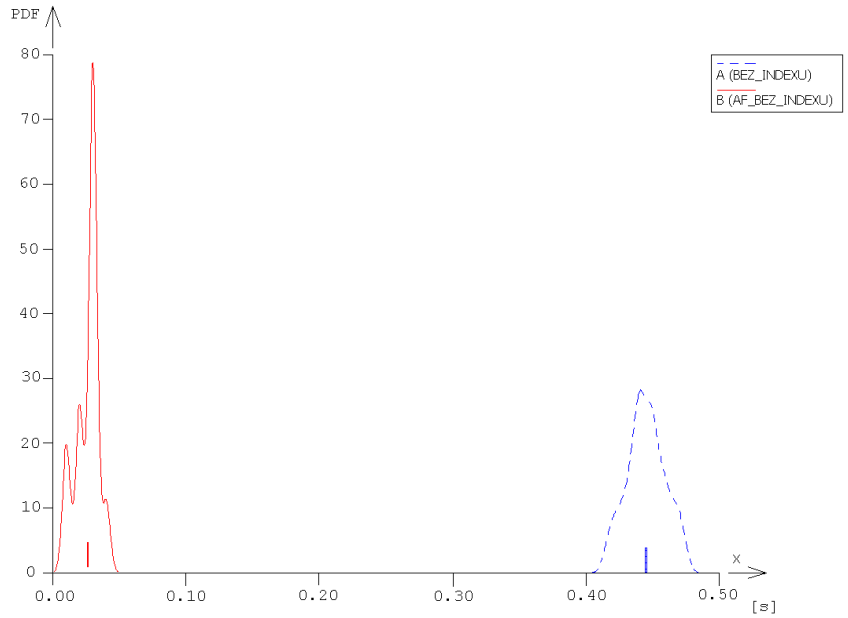
### 2.1.3 Časové porovnání obou variant A - B (bez indexu)

Porovnání středních hodnot vzorků varianty A (*BEZ_INDEXU*) a varianty B (*AF_BEZ_INDEXU*) naznačuje, že použití analytických funkcí je zřejmě rychlejší, nežli zpracování metodou klasickou. Aby bylo možné toto tvrzení podpořit či vyvrátit, byla ve statistickém nástroji QC.Expert[13] provedena pro oba tyto výběry funkce *Porovnání 2 výběrů*. Výsledky tohoto porovnání jednoznačně dospěly k závěru, že oba vzorky jsou rozdílné jak z hlediska shody rozptylů, tak z hlediska shody průměrů (protokol porovnání viz příloha C), přičemž při porovnání bylo užito i robustních testů [9] a testování shody K-S. Diference jsou vizuálně viditelné ve grafu jádrových odhadů pravděpodobnosti pro oba výběry (Graf 3), také ve znázornění příslušných distribučních funkcí (Graf 4).

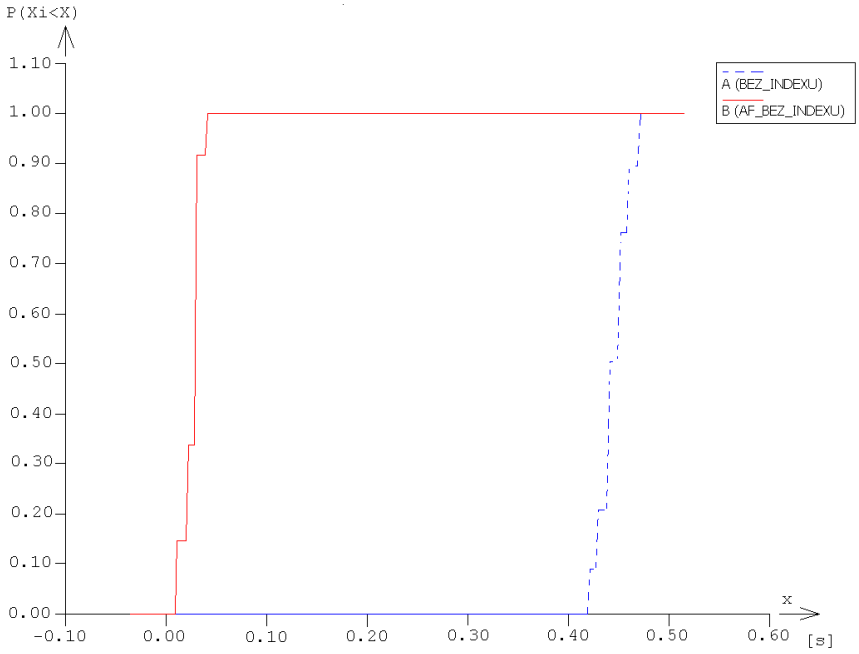
[11] Zdroj: Dávkové měření CPU trvání přes vlastní SQL skript.

[12] Zdroj: Vlastní.

[13] QC.Expert (ADSTAT) firmy TriloByte STATISTICAL SOFTWARE.




Graf 3: Jádrové odhady hustoty pravděpodobnosti pro porovnávané výběry (A-B)[14]




Graf 4: Empirické distribuční funkce pro porovnávané výběry (A-B)[14]

V tomto případě lze říci, že bez přítomnosti indexů nad tabulkou jsou střední hodnoty vzorků A a B rozdílné, přičemž varianta analytických funkcí (B) je výrazně rychlejší než varianta klasická (A) .

[14] Zdroj: Vlastní.

## 2.2 Trvání SUM proti AF SUM – varianty s indexy

Zajímavým faktorem při porovnání rychlosti dotazu může být její ovlivnění existujícími indexy nad tabulkami. Při použití stejných dotazů, které byly uvedeny v kapitole 2.1, jsou stejná měření provedena i nad tabulkou *SALE_I* s indexem nad poli *PERIOD* a *SEGMENT*.

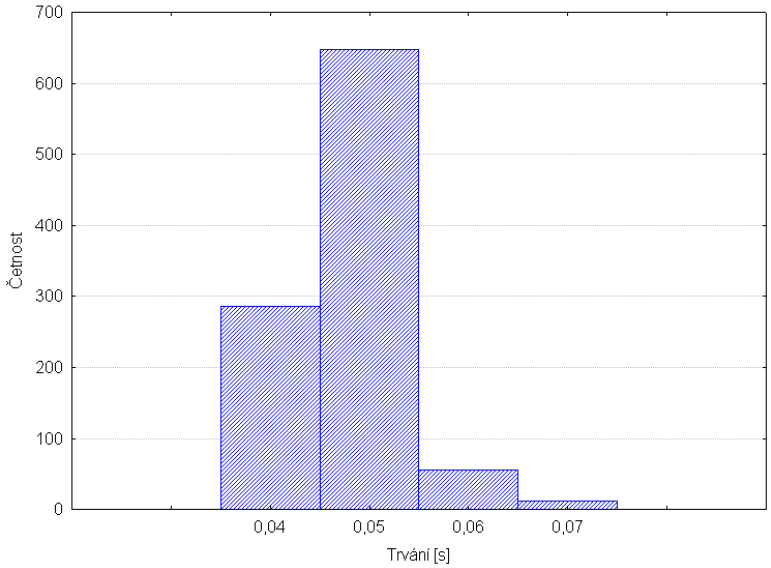
### 2.2.1 Varianta C - klasická agregační funkce SUM s indexy

Pro měření v této variantě byl použit stejný dotaz, jako v kapitole 2.1. Jediný rozdíl spočívá v tom, že je postaven nad tabulkou SALE_I (Dotaz 9).

```
select a.*, ( select sum( price_total ) from sale_i
              where
              period = a.period
            ) as total_period
from
(
select period, segment, sum( price_total ) as price_total
from sale_i
group by period, segment
order by period, segment
) a
```

Dotaz 9: Klasická agregační funkce SUM poddotazem (s indexy)[15]

Histogram četností naměřených CPU trvání pro 1000 provedených měření zobrazuje Graf 5. Vypočítané charakteristiky tohoto vzorku dat pak Tabulka 3.




Graf 5: Histogram naměřených CPU trvání pro Dotaz 9[16] (1000 měření)

[15] Zdroj: Vlastní.

[16] Zdroj: Dávkové měření CPU trvání přes vlastní SQL skript.

Tabulka 3: Varianta C - charakteristiky ze získaného vzorku dat (1000 měření)[17]

| Charakteristika | Hodnota |
|---|---|
| $\bar{t}_{S\_INDEXY}$ (průměr) | 0.0479 [s] |
| $s_{S\_INDEXY}$ (směrodatná odchylka) | 0.0058 [s] |

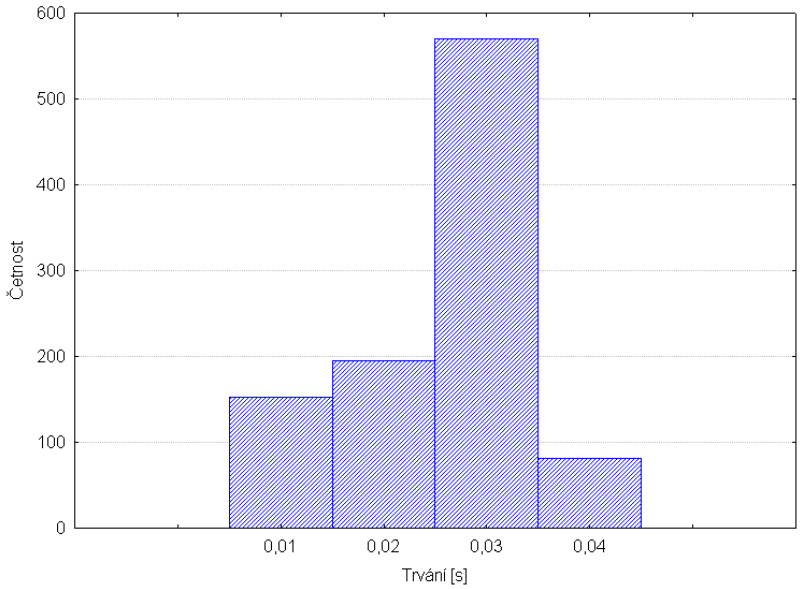
### 2.2.2 Varianta D - AF SUM s indexy

Varianta D využívá analytickou funkci SUM nad tabulkou s indexy. Pro měření CPU trvání byl položen Dotaz 10.

```
select a.*, sum( price_total ) over( partition by period )
from
(
select period, segment, sum( price_total ) as price_total
from sale_i
group by period, segment
order by period, segment
) a
```

Dotaz 10: Dotaz pro agregační AF SUM (s indexy)[17]

Histogram četností naměřených CPU trvání pro 1000 provedených měření zobrazuje Graf 6. Vypočítané charakteristiky tohoto vzorku dat pak Tabulka 4.




Graf 6: Histogram naměřených CPU trvání pro Dotaz 10[18] (1000 měření)

[17] Zdroj: Vlastní.

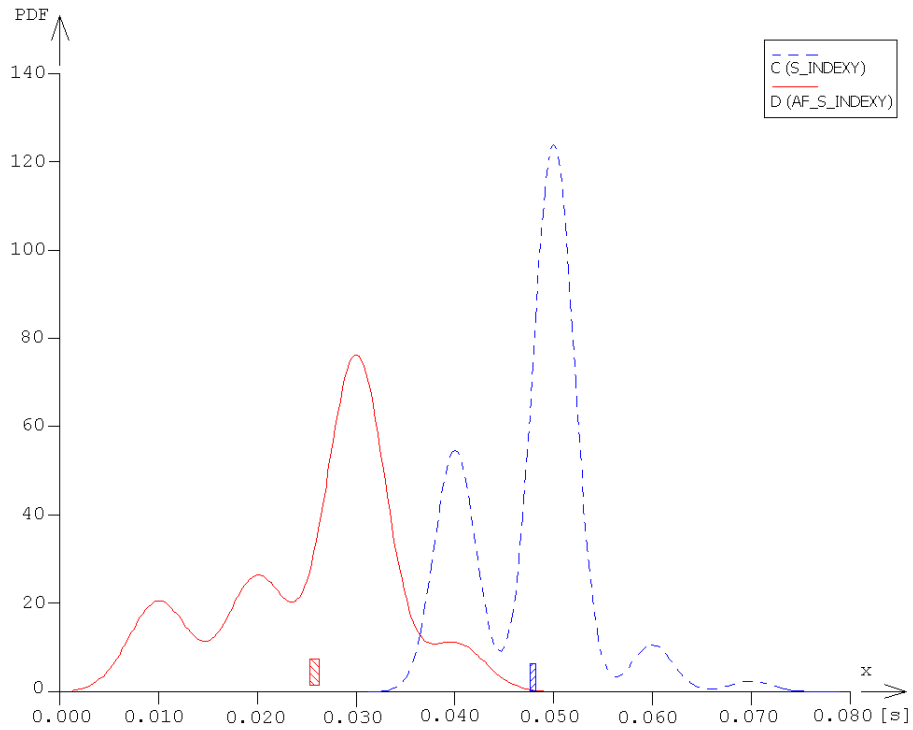
[18] Zdroj: Vlastní.

Tabulka 4: Varianta D - charakteristiky ze získaného vzorku dat (1000 měření)[19]

| Charakteristika | Hodnota |
|---|---|
| $\bar{t}_{AF\_S\_INDEXY}$ (průměr) | 0.0258 [s] |
| $s_{AF\_S\_INDEXY}$ (směrodatná odchylka) | 0.0084 [s] |

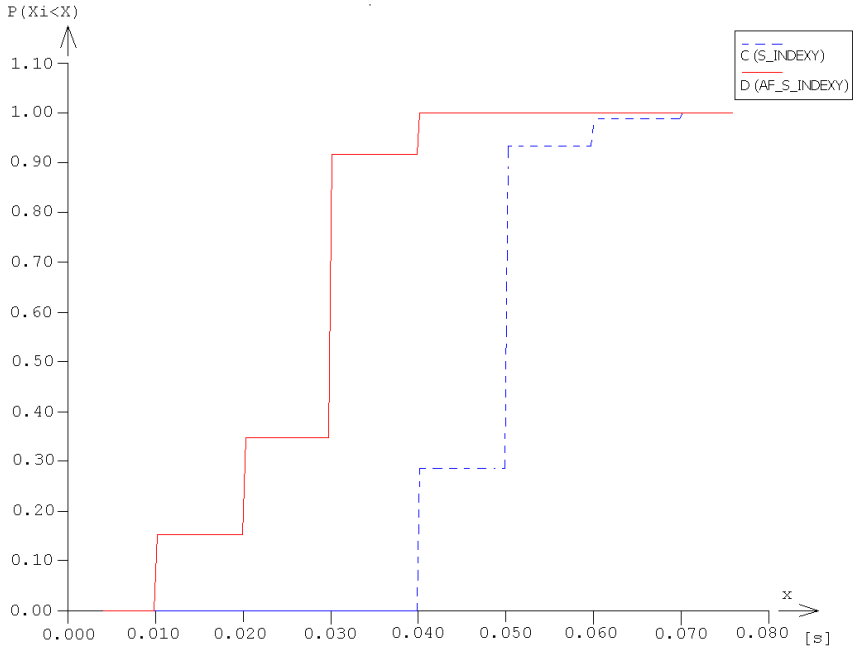
### 2.2.3 Časové porovnání obou variant s indexy

Přidání indexu do dotazu výrazným způsobem ovlivnilo výkon klasické funkce SUM, která je na testovaném vzorku rychlejší oproti stejné variantě bez použití indexů. Střední hodnota vzorku analytické funkce SUM (D) je však i přesto nižší. Statistickým porovnáním obou výběrů provedeném v nástroji QC.Expert byly vzorky C a D označeny i v tomto případě jako rozdílné (protokol porovnání viz příloha C). Zřejmou rozdílnost ukazují i grafické náhledy na jádrové hustoty a empirické funkce obou vzorků v Graf 7 a Graf 8.




Graf 7: Jádrové odhady hustoty pravděpodobnosti pro porovnávané výběry (C-D)[19]

[19] Zdroj: Vlastní.




Graf 8: Empirické distribuční funkce pro porovnávané výběry(C-D)[20]

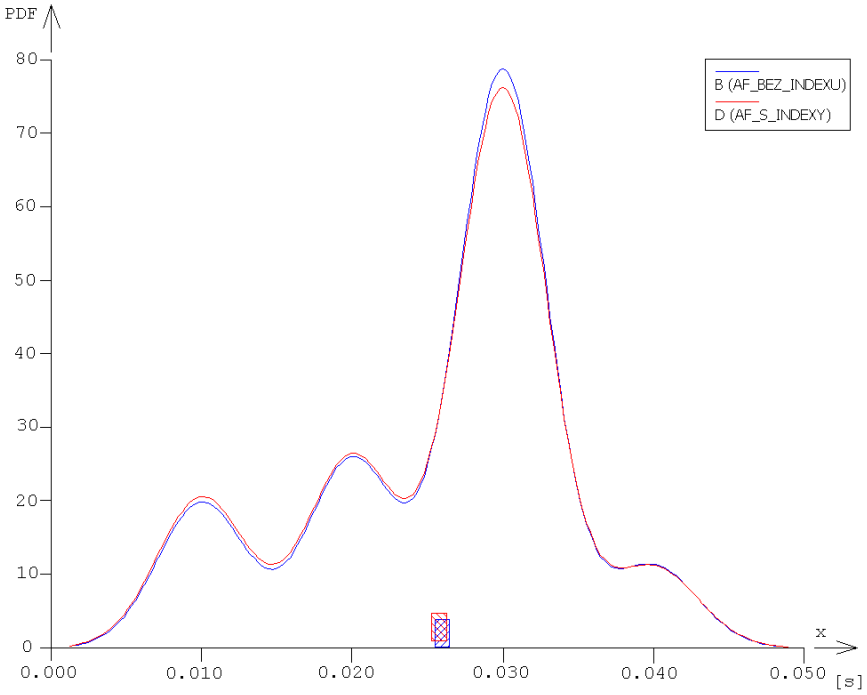
V tomto případě lze tedy říci, že za přítomnosti indexů nad tabulkou jsou střední hodnoty vzorků C a D rozdílné. Varianta D (*AF S INDEXY*) je rychlejší nežli vzorek s měřeními klasické funkce (*S INDEXY*).

## 2.3 Trvání dotazu AF SUM u variant bez indexů – s indexy

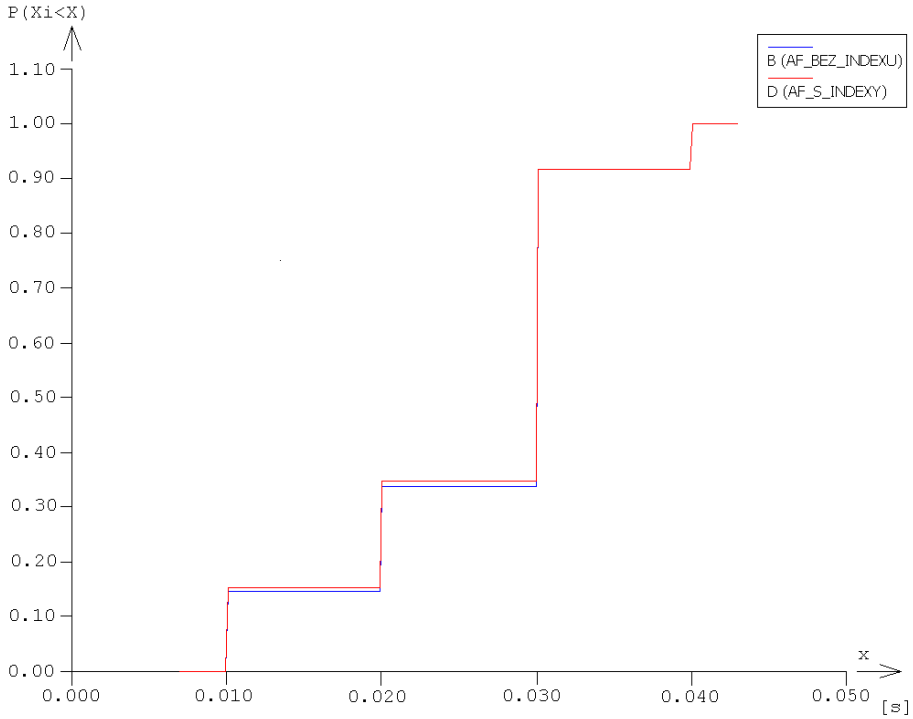
Ze získaných měření (varianta B proti D) je možné provést srovnání výsledků obou měření pro dotazy, které využívají analytické funkce z pohledu přítomnosti indexů nad databázovou tabulkou. Lze tak porovnat, zda přítomnost indexů měla/neměla vliv na CPU trvání dotazu.

Porovnáním vzorků variant B a D proti sobě bylo zjištěno (protokol porovnání viz příloha C), že oba vzorky jsou shodné. Lze tedy prohlásit, že přítomnost indexů neměla v tomto případě na výkon analytické funkce vliv. Graficky ukazuje jádrový odhad hustoty pravděpodobnosti i empirickou distribuční funkci pro oba vzorky Graf 9 a Graf 10.

[20] Zdroj: Vlastní.




Graf 9: Jádrové odhady hustoty pravděpodobnosti pro porovnávané výběry (B-D)[21]




Graf 10: Empirické distribuční funkce pro porovnávané výběry (B-D)[21]

[21] Zdroj: Vlastní.

## 2.4 Trvání dotazu v závislosti na velikosti vzorku dat

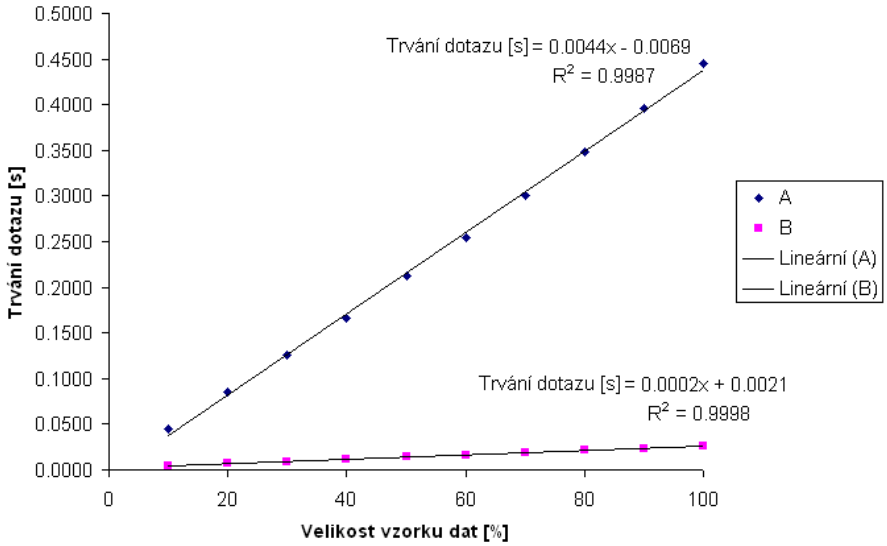
Pro zjištění, jakým způsobem ovlivňuje množství dat rychlost dotazu, bylo provedeno náhodné vytvoření vzorků dat o velikosti 10, 20, 30, 40, 50, 60, 70, 80 a 90 % z původních tabulek SALE a SALE_I, které původně obsahovaly 38 850 záznamů. K vytvoření vzorků bylo použito rozšíření SAMPLE příkazu SELECT.

Pro takto získané vzorky bylo provedeno měření rychlosti dotazu za použití shodných postupů jako v kapitolách 2.1, 2.2. Naměřené průměrné hodnoty zobrazuje tabulka Tabulka 5.

Tabulka 5: Naměřené průměrné hodnoty trvání dotazu [s] [22]

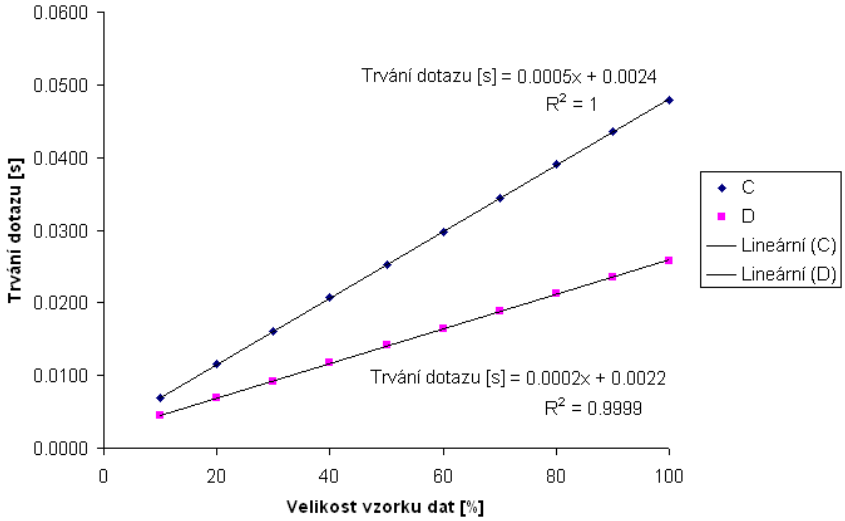
| Varianta | Velikost vzorku [%] | | | | |
|---|---|---|---|---|---|
| | **10** | **20** | **30** | **40** | **50** |
| **A** | 0.0448 s | 0.0851 s | 0.1251 s | 0.1657 s | 0.2126 s |
| **B** | 0.0045 s | 0.0068 s | 0.0092 s | 0.0118 s | 0.0142 s |
| **C** | 0.0069 s | 0.0115 s | 0.016 s | 0.0207 s | 0.0253 s |
| **D** | 0.0045 s | 0.0069 s | 0.0092 s | 0.0117 s | 0.0141 s |
| | **60** | **70** | **80** | **90** | **100** |
| **A** | 0.2542 s | 0.3013 s | 0.3481 s | 0.3955 s | 0.4454 s |
| **B** | 0.0164 s | 0.0189 s | 0.0211 s | 0.0234 s | 0.026 s |
| **C** | 0.0297 s | 0.0344 s | 0.039 s | 0.0435 s | 0.0479 s |
| **D** | 0.0164 s | 0.0189 s | 0.0212 s | 0.0235 s | 0.0258 s |

Grafické zobrazení pro varianty dotazů A-B ukazuje Graf 11, pro varianty C-D pak Graf 12. Výsledné grafy naznačují, že velikost vzorku výsledek měření neovlivňuje.




Graf 11:Trvání dotazu v závislosti na velikosti vzorku (varianty A-B)[22]

[22] Zdroj: Vlastní.




Graf 12:Trvání dotazu v závislosti na velikosti vzorku (varianty C-D)[23]

## 2.5 Shrnutí dílčích závěrů

Při tomto testování bylo zjištěno, že použití analytické funkce SUM je výrazně rychlejší než klasický přístup s využitím poddotazů. Přítomnost indexů neměla na analytickou funkci SUM vliv, vyhodnocení klasické funkce se urychlilo výrazně. Přesto klasický dotaz s indexy nedosahoval rychlosti analytické funkce s indexy.

Nelze říci, zda podobné chování platí i pro ostatní analytické funkce. Dále je třeba vzít na vědomí, že byla testována pouze jedna konkrétní množina dat, o daném rozsahu, složení sloupců a dané velikosti.

[23] Zdroj: Vlastní.

# 3. Porovnání způsobu vyhodnocení analytických funkcí s klasickým přístupem

Testovacím kritériem této části je náklad dotazu (angl. cost), který je analyzován dle navrácené hodnoty přes EXPLAIN plán dotazu. Náklad dotazu je hodnota, vyjadřující odhad očekávaných zdrojů, které budou potřebné pro vykonání dotazu s každým jednotlivým plánem vykonávaného dotazu [1]. Do této hodnoty jsou zahrnuty prováděné vstupně-výstupní operace, CPU a operace s pamětí [1]. Čím je hodnota nákladu dotazu vyšší, tím jsou vyšší i potřebné očekávané zdroje [1].

Dále je proveden rozbor plánu dotazu, kde jsou sledovány rozdíly, jak v jednotlivých případech optimalizátor Oracle dotaz zpracovává. Porovnávány jsou výsledkově srovnatelné dotazy; v prvním případě bez použití analytických funkcí, ve druhém případě s jejich použitím.

## 3.1 Vyhodnocení dotazu s nejjednodušším SUM

### 3.1.1 Varianta A – agregační funkce SUM, bez třídění

Předpokládá se tabulka SALE s prodeji za období, kdy jsou vypsány všechny řádky tabulky. Ke každému řádku je dohledávána celková hodnota pole PRICE_TOTAL. Třídění výstupní množiny řádků není použito (Dotaz 11). Získaný *Explain Plan* tohoto dotazu ukazuje Tabulka 6 a Obrázek 3.

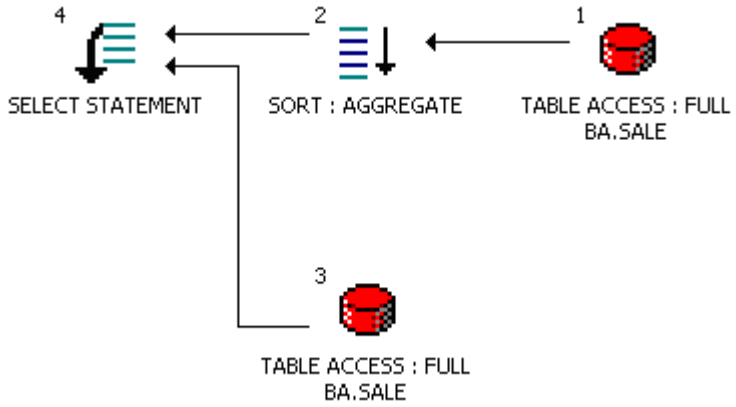
```
select period, ( select sum( price_total ) from sale ) as price_total
from sale
```

Dotaz 11: Vrácení součtu za pole PRICE_TOTAL poddotazem[24]

Tabulka 6: Získaný Explain Plan pro Dotaz 11[24]

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      | 38744 |   264K|   103   (1)| 00:00:02 |
|   1 |  SORT AGGREGATE    |      |     1 |     5 |            |          |
|   2 |   TABLE ACCESS FULL| SALE | 38744 |   189K|   104   (2)| 00:00:02 |
|   3 |  TABLE ACCESS FULL | SALE | 38744 |   264K|   103   (1)| 00:00:02 |
-------------------------------------------------------------------------------
```

[24] Zdroj: Vlastní.




Obrázek 3: Grafické znázornění EXPLAIN plánu pro Dotaz 11[25]

*Explain Plan* ukazuje, že nejdříve je přečten každý řádek tabulky SALE. Následně jsou řádky setříděny pro podporu skupinových agregačních operací jako SUM, MIN, AVG, atd. Znovu je čten každý řádek tabulky SALE. Celkový náklad tohoto dotazu je 103.

### 3.1.2 Varianta B – AF SUM, bez třídění

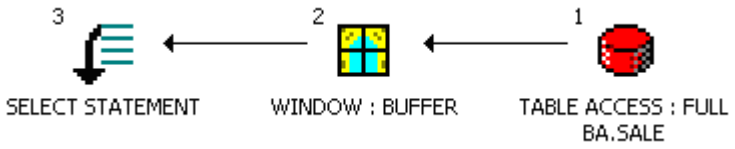
Varianta B řeší stejný dotaz jako varianta A, avšak s využitím analytické funkce SUM (Dotaz 12). Získaný *Explain Plan* tohoto dotazu ukazuje Tabulka 7 a Obrázek 4.

```
select period, segment, sum( price_total ) over () as price_total
from sale
```

Dotaz 12: Vrácení součtu za pole PRICE_TOTAL pomocí AF SUM[25]

Tabulka 7: Získaný Explain Plan pro Dotaz 12[25]

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      | 38744 |   529K|   104   (2)| 00:00:02 |
|   1 |  WINDOW BUFFER     |      | 38744 |   529K|   104   (2)| 00:00:02 |
|   2 |   TABLE ACCESS FULL| SALE | 38744 |   529K|   104   (2)| 00:00:02 |
-------------------------------------------------------------------------------
```




Obrázek 4: Grafické znázornění EXPLAIN plánu pro Dotaz 12[25]

Celkový náklad tohoto dotazu je 104. Ten vzniká v rámci plného přístupu k tabulce SALE, kde se stejně jako u varianty A přistupuje ke všem 38744 řádkům. Následně je

[25] Zdroj: Vlastní.

vytvořeno *Okno* využité pro analytickou agregační funkci SUM. Nakonec je uplatněn příkaz SELECT a data jsou navrácena.

### 3.1.3 Shrnutí dílčích závěrů

Z hlediska získané hodnoty nákladů dotazů nebyl mezi variantami pozorován významnější rozdíl. Poměr nákladu dotazů mezi variantami A:B je 103:104. U varianty B je dobře vidět zpracování analytické funkce, kde je po přístupu k tabulce provedena část *Okna*, a z ní je navrácen výsledek, přesně dle procesního toku popsaného v kapitole 1.2.

## 3.2 Vyhodnocení dotazu s nejjednodušším SUM s ORDER BY

Následující testování přidává navíc k oběma v předchozí části definovaným dotazům, které jinak zůstávají stejné, sekci ORDER BY, jenž zajišťuje třídění výsledné množiny dat dle období.

### 3.2.1 Varianta C - agregační funkce SUM, tříděno

Tato varianta s poddotazem přidává oproti předchozí variantě A sekci ORDER BY (Dotaz 13). Získaný *Explain Plan* tohoto dotazu ukazuje Tabulka 8 a Obrázek 5.

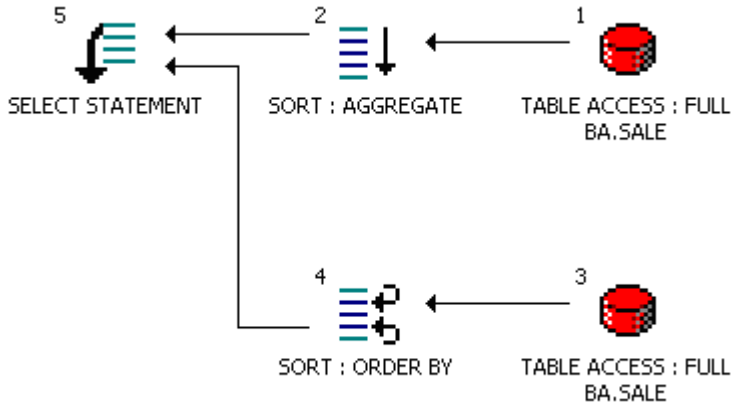
```
select period, ( select sum( price_total ) from sale ) as price_total
from sale
order by period
```

Dotaz 13: Vrácení celkového součtu poddotazem, tříděno[26]

Tabulka 8: Získaný Explain Plan pro Dotaz 13[26]

| Id | Operation | Name | Rows | Bytes | TempSpc | Cost (%CPU) | Time |
|---|---|---|---|---|---|---|---|
| 0 | SELECT STATEMENT | | 38744 | 264K | | 243 (3) | 00:00:03 |
| 1 | SORT AGGREGATE | | 1 | 5 | | | |
| 2 | TABLE ACCESS FULL | SALE | 38744 | 189K | | 104 (2) | 00:00:02 |
| 3 | SORT ORDER BY | | 38744 | 264K | 1224K | 243 (3) | 00:00:03 |
| 4 | TABLE ACCESS FULL | SALE | 38744 | 264K | | 103 (1) | 00:00:02 |

[26] Zdroj: Vlastní.




Obrázek 5: Grafické znázornění EXPLAIN plánu pro Dotaz 13[27]

*Explain Plan* ukazuje, že celkový náklad tohoto dotazu je 243. Dvakrát tu dochází k přístupu ke všem řádkům v tabulce SALE: jednou k zajištění podpory třídění dotazu, podruhé k vrácení jednoho řádku, který je výsledkem použité agregační funkce [1].

### 3.2.2 Varianta D - AF SUM, tříděno

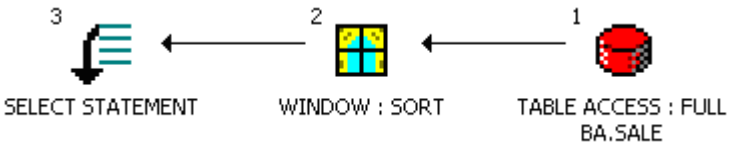
Stejná situace je řešena pomocí analytické funkce SUM, nyní bez poddotazu (Dotaz 14). Získaný *Explain Plan* tohoto dotazu ukazuje Tabulka 9 a Obrázek 6.

```
select period, segment, sum( price_total ) over () as price_total
from sale
order by period
```

Dotaz 14: Vrácení celkového součtu pomocí AF SUM[27]

Tabulka 9: Získaný Explain Plan pro Dotaz 14[27]

```
-------------------------------------------------------------------------
|Id| Operation          | Name | Rows   |Bytes| Cost (%CPU)| Time     |
-------------------------------------------------------------------------
|0| SELECT STATEMENT    |      | 38744  | 529K|   104   (2)| 00:00:02 |
|1|  WINDOW SORT        |      | 38744  | 529K|   104   (2)| 00:00:02 |
|2|   TABLE ACCESS FULL| SALE  | 38744 | 529K|   104   (2)| 00:00:02 |
-------------------------------------------------------------------------
```




Obrázek 6: Grafické znázornění EXPLAIN plánu pro Dotaz 14[27]

[27] Zdroj: Vlastní.

Přidání sekce ORDER BY náklad dotazu neovlivnilo jako u varianty C. Celkově zůstal na hodnotě 104. Jediný rozdíl je v použití WINDOW SORT oproti WINDOW BUFFER ve stejném dotazu, kde není požadováno setřídění výstupní množiny.

### 3.2.3 Shrnutí dílčích závěrů

Při použití dotazu s ORDER BY vzniká významný rozdíl v nákladech obou dotazů (testovaná varianta C - D). Zatímco dotaz bez analytické funkce má náklad 243; u dotazu s jejím využítím, který poskytuje stejnou výslednou množinu, je hodnota nákladu rovna 104. V tomto případě lze říci, že použití analytické funkce bylo z hlediska vyhodnocení nákladů dotazu výhodnější.

U testovaných variant bez ORDER BY (A - B) nebyl v nákladech dotazů pozorován výraznější rozdíl.

# 4. Identifikace alternativ analytických funkcí v Microsoft SQL Serveru

## 4.1 Analytické funkce Microsoft SQL Serveru

Analytické funkce začaly do Transact-SQL (T-SQL) pronikat až ve verzi 2005 [11]. Jedná se zejména o funkce pořadí ROW_NUMBER, RANK, DENSE_RANK a NTILE [11]. Seznam dostupných analytických funkcí v Microsoft SQL Serveru 2005 zobrazuje Tabulka 10.

Tabulka 10: AF v MS SQL 2005 a jejich Oracle protějšek [8], [11].

| **Dostupné AF MS SQL 2005** | **Druh funkce** | **Protějšek Oracle** |
|---|---|---|
| ROW_NUMBER | Funkce pořadí | ROW_NUMBER |
| RANK | Funkce pořadí | RANK |
| DENSE_RANK | Funkce pořadí | DENSE_RANK |
| NTILE | Funkce pořadí | NTILE |
| Agregační AF (například AF SUM) | Agregační funkce nad oknem | Odpovídající skupina funkcí |
| PIVOT | Relační operátor | Neexistuje |

## 4.2 Funkce pořadí

Všechny čtyři funkce pořadí, které jsou zabudovány v T-SQL, jsou identické s funkcemi, které přináší Oracle [8]. Oracle však navíc mimo tyto analytické funkce nabízí v této kategorii také funkci CUME_DIST a PERCENT_RANK [8].

## 4.3 Agregační funkce nad oknem

Ačkoli T-SQL podporuje agregační funkce nad oknem, syntaxe, jenž může být užita, je výrazně slabší nežli u Oracle. Především není možné v rámci těchto analytických funkcí provést definování OVER klauzule přes ORDER BY. To dosvědčuje i syntaxe pro tuto skupinu analytických funkcí uvedená v Tabulka 11 a Tabulka 12.

Tabulka 11: MS SQL - syntaxe klauzule OVER pro AF nad oknem [12]

```
<OVER_CLAUSE> :: = OVER ( ( PARTITION BY value_expression , ... ( n ) ) )
```

Tabulka 12: Oracle - syntaxe OVER pro agregační funkce nad oknem [8]

```
analytic_function(( arguments ))
OVER (analytic_clause)
where analytic_clause =
( query_partition_clause )
( order_by_clause ( windowing_clause ) )
```

```
and query_partition_clause =
PARTITION BY
{ value_expr(, value_expr )...
| ( value_expr(, value_expr )... )
}
and windowing_clause =
{ ROWS | RANGE }
{ BETWEEN
{ UNBOUNDED PRECEDING
| CURRENT ROW
| value_expr { PRECEDING | FOLLOWING }
}
AND
{ UNBOUNDED FOLLOWING
| CURRENT ROW
| value_expr { PRECEDING | FOLLOWING }
}
| { UNBOUNDED PRECEDING
| CURRENT ROW
| value_expr PRECEDING
}
}
```

Oracle syntaxe nejen že nabízí definici OVER klauzule přes třídění ORDER BY, s využitím rozsahu ROWS | RANGE je možné definovat i klouzavou oblast, nad kterou budou analytické funkce uplatněny [8].

## 4.4 Podpora křížových tabulek

Pomocí operátoru PIVOT je možné T-SQL vytvářet křížové dotazy, kdy je výstup směřován do jednotlivých sloupců. Lze tak umístit například období do řádků a celkovou sumu za jednotlivé segmenty do sloupců, jak to ilustruje Dotaz 15.

```
SELECT period, (A) as 'segment_a',  (B) as 'segment_b', (C) as 'segment_c'
FROM
( select period, segment, sum( price_total ) as price_total
  from dbo.sale
  group by period, segment
) a
PIVOT ( sum( price_total ) FOR segment IN ( (A), (B), (C) ) ) as PIVOT_TABLE
where
period like '2007%'
order by period
```

```
period segment_a     segment_b              segment_c
------ ------------- ---------------------- ----------
200701 1889257       2477991                2598971
200702 2663161       3950639                3182131
200703 2575838       2971715                2970532
200704 2505355       2606748                2534265
200705 2488192       3003020                2624143
200706 3234617       3191298                3499754
200707 3426543       3610315                3821423
(7 row(s) affected)
```

Dotaz 15: Funkcionalita operátoru PIVOT Microsoft SQL Serveru 2005[28]

PIVOT operátor, který je používán pro tvorbu křížových tabulek, nemá v Oracle databázi protějšek. K dosažení této funkcionality v Oracle je zapotřebí použít rozšířené OLAP nástroje [2]. Vedle operátoru PIVOT zavádí Microsoft i reverzní operátor UNPIVOT, s jehož pomocí je možné data ze sloupců tabulky dostat opět do řádků.

Ačkoli v Oracle operátor PIVOT neexistuje, neznamená to, že by nebylo možné se dopracovat ke stejnému výsledku. Pro tento účel je možné použít příkaz CASE [8], který rozliší jednotlivé prodejní segmenty[29]. Výsledek Dotaz 16 je pak stejný jako při použití Microsoft PIVOT.

```
select period,
sum( case when segment = 'A' then price_total end ) as segment_a,
sum( case when segment = 'B' then price_total end ) as segment_b,
sum( case when segment = 'C' then price_total end ) as segment_c
from ba.sale
where
period like '2007%'
group by period
order by period
```

Dotaz 16: Simulace PIVOT dotazu přes Oracle CASE[30]

[28] Zdroj: Vlastní.

[29] Příkaz CASE existuje i v MS-SQL.

[30] Zdroj: Vlastní.

# Závěr

V rámci omezeného rozsahu bakalářské práce se při testování pracovalo pouze s jednou analytickou funkcí SUM, data měla určitou velikost a podobu, spojení mezi tabulkami nebylo používáno. Z tohoto důvodu je třeba upozornit na to, že závěry přijaté pro funkci SUM nemusí zákonitě platit i pro ostatní analytické funkce, i když se to předpokládá.

Přesto však z testování CPU trvání dotazu mezi klasickou funkcí SUM a analytickou funkcí SUM vyplývá, že použití analytické funkce SUM je výrazně rychlejší než u klasického přístupu s využitím poddotazů. Přítomnost indexů neměla na analytickou funkci SUM vliv, klasický přístup ale urychlila výrazně.

Z hlediska nákladů dotazu[31] byla analytická funkce SUM výhodnější, avšak pouze při použití třídění výsledné množiny dat přes ORDER BY. U testovaných variant bez ORDER BY nebyl v nákladech dotazů pozorován výraznější rozdíl.

V Microsoft SQL Serveru byly nalezeny některé analytické funkce Oracle, obecně je však možné říci, že v rámci Oracle je podpora analytických funkcí vyšší a jejich možnosti jsou i bohatší.

Určitou nevýhodu analytických funkcí lze spatřovat ve složitosti syntaxe zápisu, která je odlišná, a na první pohled může paradoxně působit spíše nepřehledně. Důvodem může být skutečnost, že jde pouze o zvyk naučit se s funkcemi nativně pracovat.

Využívání analytických funkcí vyžaduje vyšší formu abstrakce problému ještě před jeho řešením.

[31] Náklady dotazu viz kapitola 3.

# Seznam použitých zkratek

| | |
|---|---|
| AF | Analytické funkce (angl. Analytical Functions) |
| BI | Business Intelligence |
| CBO | Optimalizátor dle nákladů dotazu (angl. Cost-Based Optimizer) |
| CPU | Centrální procesorová jednotka (angl. Central processor unit) |
| OLAP | Analytické zpracování dat (angl. Online analytical processing) |
| OLTP | Online transakční zpracování dat (angl. Online transaction processing) |
| SQL | Strukturovaný dotazovací jazyk (angl. Structured query language) |

# Seznam použité literatury


[1] CHAN, I., *Oracle Database Performance Tuning Guide, 10g Release 2 (10.2)*. 1th ed., Redwood City : Oracle Corporation, 2005. B14211-01.

[2] GORNSTEIN, D. *Features, strengths and weakness comparison between MS SQL 2005 and Oracle 10g databases,* (online). WisdomForce Technologies, Inc, 2004 (cit. 2007-07-15). Dostupný z WWW: http://www.wisdomforce.com/dweb/resources/docs/MSSQL2005_ORACLE10g_compare.pdf.

[3] GREENWALD, R., STACKOWIAK, R., DODGE, G., KLEIN, D., SHAPIRO, B., CHELLIAH, CH. *Professional Oracle Programming*. 1th ed., Indianapolis: Wiley Publishing, Inc, 2005. ISBN 0-7645-7482-5.

[4] HAYDU, J. Analytic SQL Features in Oracle9i. 1th ed., Redwood Shores: Oracle Corporation, 2001.

[5] KUBANOVÁ, J. *Statistické metody pro ekonomickou a technickou praxi*. 2. vyd. Bratislava: STATIS, 2002. ISBN 80-85659-37-9.

[6] KUBANOVÁ, J., LINDA, B. *Kritické hodnoty a kvantily vybraných rozdělení pravděpodobnosti*. 1. vyd. Pardubice: Univerzita Pardubice, 2006. ISBN 80-7194-852-7.

[7] KYTE, T. *Expert One-on-One Oracle*. Berkeley: Apress, 2003. ISBN 1590592433.

[8] LANE, P., SCHUPMANN, V., STUART, I. *Oracle Database Data Warehousing Guide, 10g Release 2 (10.2)*. 1th ed., Redwood City : Oracle Corporation, 2005. B14223-02.

[9] MELOUN, M., MILITKÝ, J. *Statistické zpracování experimentálních dat*. 1. vyd. Praha: PLUS spol. s.r.o., 1994. ISBN 80-85297-56-6.

[10] PALINSKI, J. *Oracle SQL and PL/SQL Handbook: A Guide for Data Administrators, Developers, and Business Analysts*. Addison Wesley Professional, 2002. ISBN 0-201-75294-8.

[11] RANKINS, R., BERTUCCI, P., GALLELLI, CH., SILVERSTEIN, A. *Microsoft SQL Server 2005 Unleashed.* 1th ed., Indianapolis: Sams Publishing, 2007. ISBN 0-672-32824-0.

[12] Transact-SQL Reference (Transact-SQL), (online). Microsoft Corporation, ( cit. 2007-10-01). Dostupné z: <http://msdn2.microsoft.com/en-us/library/ms189826.aspx>.

[13] What is Oracle history ?, (online). (cit. 2008-02-15). Dostuné z: <http://www.orafaq.com/faq/what_is_oracles_history>.

# Seznam obrázků

# Seznam tabulek

# Seznam grafů

# Seznam dotazů

# Seznam příloh

**Příloha A**

# Použité softwarové nástroje

Testování probíhalo na stanici HP COMPAQ, Intel Core 2 CPU, 1.86 GHz, 3 GB RAM, 250 GB HD, Windows XP, Service Pack 2.

Jako databáze byla použita:

- Oracle Database 10g Express Edition Release 10.2.0.1.0 (dostupná volně na webu na adrese *http://www.oracle.com/technology/software/products/database/xe/htdocs/102xewinsoft.html*)
- Microsoft SQL Server Express 2005 (dostupná volně na webu na adrese http://msdn2.microsoft.com/en-us/express/bb410792.aspx)

Testovací databáze implicitně využívá optimalizační režim ALL_ROWS, který určuje, že se použije CBO, bez ohledu na přítomnost statistik, ale s cílem využití minima zdrojů k dokončení dotazu [1].

K pokládání dotazů, zobrazování jejich výsledků a k rozborům nákladů každého z nich byly použity nástroje:

- pro Oracle klient Toad verze 9.0.1 společnosti Quest Software.
- pro SQL Server klient Microsoft SQL Server Management Studio Express, verze 9.00.3042.00.

K získání 1000 měření CPU trvání jednotlivých dotazů byly použity vlastní SQL skripty, které jsou součástí přiloženého CD (příloha D).

Pro statistická porovnání vzorků, výpočty jejich průměrů a směrodatných odchylek, výstupy statistických grafů, byl použit softwarový nástroj QC.Expert (ADSTAT) firmy TriloByte STATISTICAL SOFTWARE (www.trilobyte.cz).

K vyjádření trasovacích charakteristik dotazů byl použit nástroj Oracle TKPROF.

Popis datových tabulek, na kterých probíhalo testování analytických funkcí, je uveden v příloze B.

# Příloha B

## Testovací data

Pro testování obou metod přístupu byla použita tato testovací data:

### Tabulka SALE

Tabulka SALE představuje řádky prodejů za jednotlivé období.

Tabulka: Struktura databázové tabulky SALE.

| Název sloupce | Typ | Popis |
|---|---|---|
| PERIOD | VARCHAR2(6) | Období ve tvaru YYYYMM. |
| ITEM | VARCHAR2(34) | Číslo položky. |
| SBJ_ID | NUMBER(10) | ID klíč odběratele. |
| SEGMENT | VARCHAR2(1) | Segment prodeje (A,B,C). |
| DATE_ | DATE | Datum prodeje. |
| QUANTITY | NUMBER(9,3) | Množství, které bylo prodáno. |
| UNIT | VARCHAR2(2) | Měrná jednotka (KS). |
| PRICE | FLOAT(0) | Cena za měrnou jednotku. |
| PRICE_TOTAL | FLOAT(0) | Celková cena. |

Tabulka SALE neobsahuje žádný index.

Počet řádků tabulky: 38 850.

Pro orientaci je uveden celkový počet řádků tabulky SALE v jednotlivých obdobích.

```
select substr( period, 1, 4 ), count( * ) as row_count
from pirkl.sale
group by rollup( substr( period, 1, 4 ) )
```

```
SUBS  ROW_COUNT
---- ----------
2001       4988
2002       5439
2003       5319
2004       5808
2005       6230
2006       6719
2007       4347
          38850
```

### Tabulka SALE_I

Tato tabulka je naprosto identická s tabulkou SALE, rozdíl spočívá pouze v přítomnosti indexu nad polem PERIOD.

# Příloha C

## QC.Expert - porovnání výběrů "Bez indexu - AF bez indexu" (varianty A-B)

Porovnání dvou výběrů

| | | |
|---|---|---|
| Název úlohy : | List1 | |
| Data: | Všechna | |
| Hladina významnosti : | 0.05 | |
| Porovnávané sloupce : | BEZ INDEXU | AF BEZ INDEXU |
| Počet dat : | 1000 | 1000 |
| Průměr : | 0.44543 | 0.026 |
| Směr. odchylka : | 0.013842207 | 0.008346836 |
| Rozptyl : | 0.000191607 | 6.96697E-05 |
| Korel. koef. R(x,y) : | -0.024951679 | |
| Test shody rozptylů | | |
| Poměr rozptylů : | 2.750216954 | |
| Počet stupňů volnosti : | 999 | 999 |
| Kritická hodnota : | 1.10696448 | |
| Závěr : | **Rozptyly jsou ROZDÍLNÉ** | |
| Pravděpodobnost : | 1.27543E-55 | |
| Robustní test shody rozptylů | | |
| Poměr rozptylů : | 2.750216954 | |
| Redukované stupně volnosti : | 403 | 403 |
| Kritická hodnota : | 1.173638464 | |
| Závěr : | **Rozptyly jsou ROZDÍLNÉ** | |
| Pravděpodobnost : | 1.23817E-23 | |
| Test shody průměrů pro SHODNÉ rozptyly | | |
| t-statistika : | 820.5582538 | |
| Počet stupňů volnosti : | 1998 | |
| Kritická hodnota : | 1.961152015 | |
| Závěr : | **Průměry jsou ROZDÍLNÉ** | |
| Pravděpodobnost : | 0 | |
| Test shody průměrů pro ROZDÍLNÉ rozptyly | | |
| t-statistika : | 820.5582538 | |
| Redukované stupně volnosti : | 1641 | |
| Kritická hodnota : | 1.961410659 | |
| Závěr : | **Průměry jsou ROZDÍLNÉ** | |
| Pravděpodobnost : | 0 | |
| Test dobré shody rozdělení dvouvýběrový K-S test | | |
| Diference DF : | 1 | |

# QC.Expert - porovnání dvou výběrů "S indexy - AF s indexy" (varianty C-D)

Porovnání dvou výběrů

| | | |
|---|---|---|
| Název úlohy : | List1 | |
| Data: | Všechna | |
| Hladina významnosti : | 0.05 | |
| Porovnávané sloupce : | S INDEXY | AF S INDEXY |
| Počet dat : | 1000 | 1000 |
| Průměr : | 0.04793 | 0.02581 |
| Směr. odchylka : | 0.005886404 | 0.008450758 |
| Rozptyl : | 3.46497E-05 | 7.14153E-05 |
| Korel. koef. R(x,y) : | 0.040782914 | |
| Test shody rozptylů | | |
| Poměr rozptylů : | 2.061062947 | |
| Počet stupňů volnosti : | 999 | 999 |
| Kritická hodnota : | 1.10696448 | |
| Závěr : | **Rozptyly jsou ROZDÍLNÉ** | |
| Pravděpodobnost : | 5.67524E-30 | |
| Robustní test shody rozptylů | | |
| Poměr rozptylů : | 2.061062947 | |
| Redukované stupně volnosti : | 359 | 359 |
| Kritická hodnota : | 1.184902952 | |
| Závěr : | **Rozptyly jsou ROZDÍLNÉ** | |
| Pravděpodobnost : | 5.76222E-12 | |
| Test shody průměrů pro SHODNÉ rozptyly | | |
| t-statistika : | 67.92019769 | |
| Počet stupňů volnosti : | 1998 | |
| Kritická hodnota : | 1.961152015 | |
| Závěr : | **Průměry jsou ROZDÍLNÉ** | |
| Pravděpodobnost : | 0 | |
| Test shody průměrů pro ROZDÍLNÉ rozptyly | | |
| t-statistika : | 67.92019769 | |
| Redukované stupně volnosti : | 1784 | |
| Kritická hodnota : | 1.96129462 | |
| Závěr : | **Průměry jsou ROZDÍLNÉ** | |
| Pravděpodobnost : | 0 | |
| Test dobré shody rozdělení dvouvýběrový K-S test | | |
| Diference DF : | 0.918 | |

# QC.Expert - porovnání dvou výběrů "AF bez indexů - AF s indexy" (varianty B-D)

Porovnání dvou výběrů

| | | |
|---|---|---|
| Název úlohy : | List1 | |
| Data: | Všechna | |
| Hladina významnosti : | 0.05 | |
| Porovnávané sloupce : | AF BEZ INDEXU | AF S INDEXY |
| Počet dat : | 1000 | 1000 |
| Průměr : | 0.026 | 0.02581 |
| Směr. odchylka : | 0.008346836 | 0.008450758 |
| Rozptyl : | 6.96697E-05 | 7.14153E-05 |
| Korel. koef. R(x,y) : | -0.030652839 | |
| Test shody rozptylů | | |
| Poměr rozptylů : | 1.025056034 | |
| Počet stupňů volnosti : | 999 | 999 |
| Kritická hodnota : | 1.10696448 | |
| Závěr : | **Rozptyly jsou SHODNÉ** | |
| Pravděpodobnost : | 0.320813498 | |
| Robustní test shody rozptylů | | |
| Poměr rozptylů : | 1.025056034 | |
| Redukované stupně volnosti : | 463 | 463 |
| Kritická hodnota : | 1.161082036 | |
| Závěr : | **Rozptyly jsou SHODNÉ** | |
| Pravděpodobnost : | 0.36431824 | |
| Test shody průměrů pro SHODNÉ rozptyly | | |
| t-statistika : | 0.50584004 | |
| Počet stupňů volnosti : | 1998 | |
| Kritická hodnota : | 1.961152015 | |
| Závěr : | **Průměry jsou SHODNÉ** | |
| Pravděpodobnost : | 0.61302475 | |
| Test shody průměrů pro ROZDÍLNÉ rozptyly | | |
| t-statistika : | 0.50584004 | |
| Redukované stupně volnosti : | 1998 | |
| Kritická hodnota : | 1.961152015 | |
| Závěr : | **Průměry jsou SHODNÉ** | |
| Pravděpodobnost : | 0.61302475 | |
| Test dobré shody rozdělení dvouvýběrový K-S test | | |
| Diference DF : | 0.012 | |

**Příloha D**

# Obsah DVD s elektronickou formou dat

1. Kompletní použitá databáze Oracle 10g Express (\data\oracle).

2. Získané naměřené hodnoty CPU trvání dotazů (\statistic\source_data.xls).

3. SQL skripty použité k získání 1000 měření CPU trvání dotazů (\sql).

4. Výsledky porovnání dvou výběrů přes nástroj QC.Expert (\statistic\qcexpert\comparing).