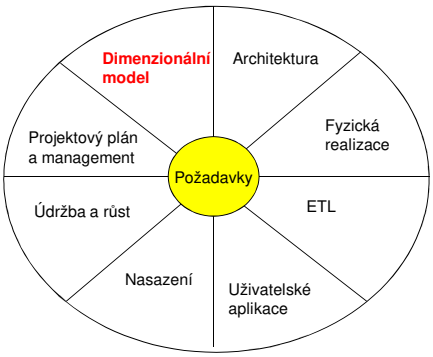
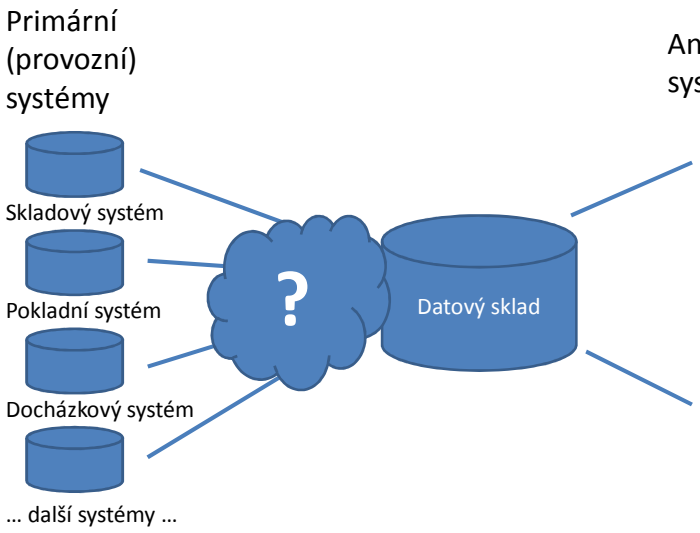
Databázové systémy a metody zpracování dat

6.přednáška

Datová kostka – jiný pohled na dimenzionální modelování





3

Různé přístupy k datům

- Provozní databázové systémy (např. pokladní systém) mají jiné požadavky na data se kterými pracují než analytické systémy, ve kterých hledají uživatelé smysl dat
- OLTP – online transakční zpracování
  - Vytváření a modifikace záznamů
- OLAP – online analytické zpracování
  - Reporty a analýzy

4

Porovnání OLTP a OLAP

- Detailní data
- Význam ve chvíli zpracování
- Častá změna dat
- Transakční orientace
- Výkonnost je důležitá
- Vysoká dostupnost je důležitá
- Redundance dat je nežádoucí
- Slouží technicko-hospodářským pracovníkům
- Předem známy požadavky na zpracování

- Agregovaná data
- Zpracování za období
- Data téměř neměnná
- Orientace na analýzu
- Výkonnost není tak důležitá
- Na vysoké dostupnosti příliš nezáleží
- Redundance dat je běžná
- Slouží především analytikům a manažerům
- Většina požadavků není předem známa

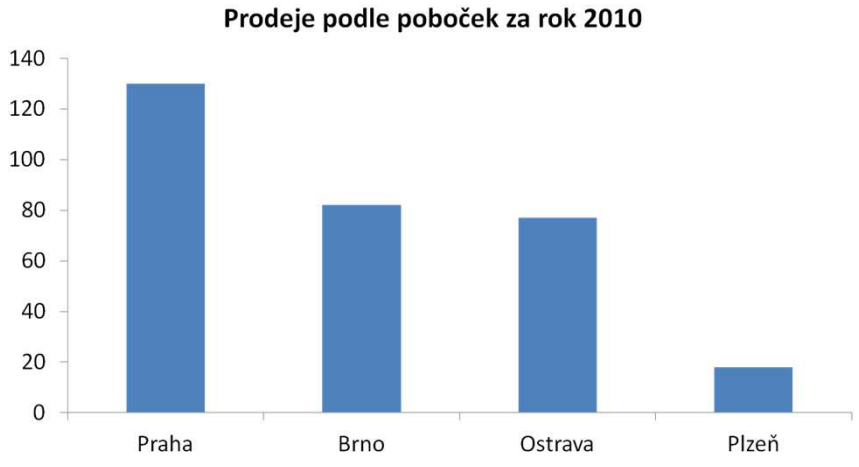
5

Datová analýza

- Formulace dotazu
  - Získání relevantních dat z databáze
- Získání výsledků
  - Získání agregovaných hodnot
- Vizualizace výsledků
  - Zobrazení v 2D a 3D objektech
  - Co nejvíce závislostí najednou
- Analýza výsledků a formulace nového dotazu

6

Histogram



7

Křížová tabulka

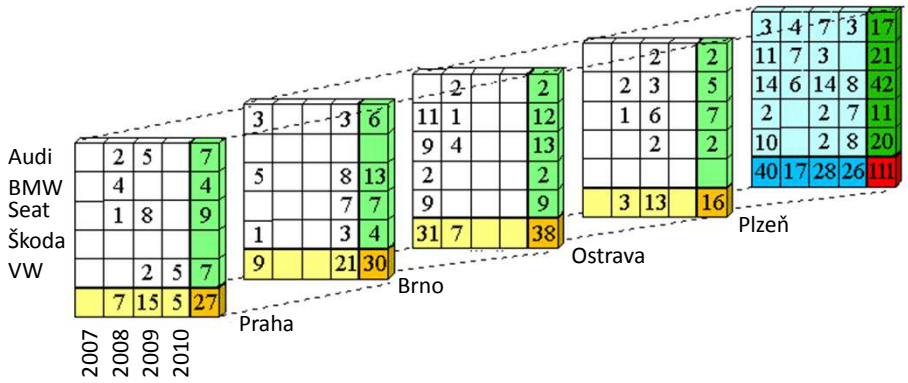
- Obsahuje:
  - Skalární data
  - Data seskupená přes jednu dimenzi
  - Data seskupená přes dvě dimenze

	Praha	Brno	Ostrava	Všechny pobočky
Škoda	28	15	13	56
VW	22	13	9	44
Audi	32	8	3	43
Všechny značky	82	36	25	143

8

Datová kostka

- Rozšíření křížové tabulky do prostoru



9

Operace s datovou kostkou

- Kostka je pro uživatele interaktivní, není statická
- Nejčastější operace
  - Slice (krájení) – omezení některé dimenze na podmnožinu o jednom prvku
  - Dice (kostkování) – omezení některé dimenze na podmnožinu o dvou a více prvcích
  - Drill Down – pohyb hierarchií dimenze od celku k detailům
  - Drill Up – pohyb hierarchií dimenze od detailů k celku (např. měsíc->rok)

10

1. Relační online analytické zpracování (ROLAP)

- Servery ROLAP jsou umístěny mezi relačním back-endovým serverem a klientskými klientskými nástroji. K ukládání a správě dat ve skladu používá relační nebo rozšířený DBMS. ROLAP má v zásadě 3 hlavní komponenty: databázový server, server ROLAP a front-endový nástroj.

11

Výhody ROLAP

- ROLAP se používá pro zpracování velkého množství dat.
- Nástroje ROLAP nepoužívají předem vypočítané datové kostky.
- Data lze efektivně ukládat.
- ROLAP může využívat funkce inherentní v relační databázi.

12

Nevýhody ROLAP

- Výkon ROLAP může být pomalý.
- V ROLAP je obtížné udržovat agregované tabulky.
- Omezeno funkcemi SQL.

13

2. Multidimenzionální online analytické zpracování (MOLAP)

- MOLAP nepoužívá relační databázi k úložišti. Ukládá se do optimalizovaného vícerozměrného pole.
- Využití úložiště může být u multidimenzionálních datových úložišť nízké.
- Mnoho serverů MOLAP zpracovává husté a řídké datové sady pomocí dvou úrovní reprezentace datového úložiště.
- MOLAP má 3 komponenty: databázový server, server MOLAP a front-endový nástroj.

14

Výhody MOLAP

- MOLAP se v zásadě používá pro složité výpočty.
- MOLAP je optimální pro operace, jako jsou řezy a kostky.
- MOLAP umožňuje nejrychlejší indexování předem vypočítaných souhrnných dat.

15

Nevýhody MOLAP

- MOLAP nedokáže zpracovat velké množství dat.
- V MOLAP vyžaduje další investice.
- Bez opětovné agregace je obtížné změnit dimenzi.

16

3. Hybridní online analytické zpracování (HOLAP)

- Hybrid je kombinací obou ROLAP a MOLAP. Nabízí funkce ROLAP i MOLAP, jako je rychlejší výpočet MOLAP a vyšší škálovatelnost ROLAP. Agregace jsou uloženy samostatně v úložišti MOLAP. Jeho server umožňuje ukládat velké objemy dat podrobných informací.

17

Výhody HOLAP -

- HOLAP poskytuje funkce jak MOLAP, tak ROLAP.
- HOLAP poskytuje rychlý přístup na všech úrovních agregace.

Nevýhody HOLAP -

architektura HOLAP je velmi složitá na pochopení, protože podporuje jak MOLAP, tak ROLAP.

18

Používané způsoby uchovávání multidimenzionálních dat v databázi

- Multidimenzionální OLAP (MOLAP) – data uložena v specializovaných multidimenzionálních databázích
  - Největší výkon při dotazování
  - Vhodné pro malé/střední databáze
  - Složité, malý počet dimenzí, prakticky nemožný update
- Relační OLAP (ROLAP) – originální data zůstávají v původních tabulkách, jsou použity speciální relační tabulky k uložení agregací
  - Vhodné pro velké distribuované databáze

19

Hlavní otázky

- Jak získávat z relační databáze multidimenzionální data?
- Jak mít data v databázi uložená?
- Jak co nejvíce urychlit výpočet dat?

- Ve výkladu není řešeno:
  - Grafické zobrazení dat z databáze v OLAP nástrojích
  - Multidimenzionální databáze

20

0-D: Agregační funkce

- Pomocí použití agregačních funkcí
- Kolik se prodalo celkem aut?

```
SELECT SUM(pocet) FROM prodeje
```

185

- Kolik se prodalo celkem Škodovek?

```
SELECT SUM(pocet) FROM prodeje
WHERE znacka = 'Škoda'
```

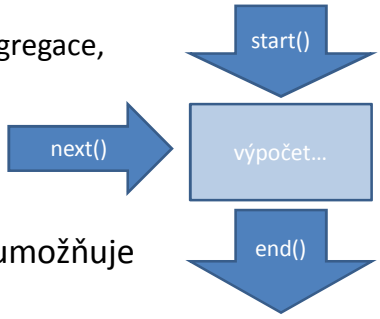
88

- Malá analytická hodnota

21

Agregační funkce

- Implementace agregačních funkcí
  - Zavolání funkce start() na začátku agregace
  - Zavolání funkce next(value) pro každý řádek dat
  - Zavolání funkce end() na konci agregace, vrací vypočtenou hodnotu



- Většina databázových systémů umožňuje psát vlastní agregační funkce

22

1-D: GROUP BY

- Použití agregačních funkcí a GROUP BY
- Kolik se prodalo Škodovek na jednotlivých pobočkách

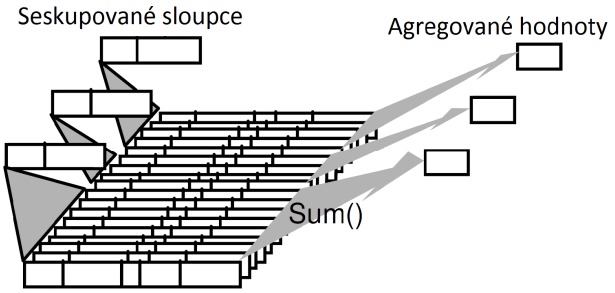
```
SELECT pobocka, SUM(pocet)
FROM prodeje
WHERE znacka = 'Škoda'
GROUP BY pobocka
```

pobocka	SUM(pocet)
Praha	130
Brno	82
Ostrava	77
Plzeň	32

23

GROUP BY

- Vyhodnocování dotazu s GROUP BY
  - Výsledek dotazu (řádky, které splňují podmínky z WHERE) je setříděn podle sloupců přes které je seskupováno
  - Jsou vytvořeny disjunktní skupiny (každou skupinu tvoří řádky se stejnými hodnotami seskupovaných sloupců)
  - Na každou skupinu se aplikuje agregační funkce, která vrací pro každou skupinu jednu hodnotu



24

2-D – N-D: ???

- Jak jednoduše získat data seskupená podle více dimenzí?
  - Pomocí známých prostředků nelze jednoduchým způsobem
- Např. pro získání dat pro křížovou tabulku s dimenzemi A, B můžeme získat jedním dotazem neseskupená data, druhým seskupená podle dimenze A, třetím podle dimenze B, čtvrtým podle dimenzí A a B
- Množství dotazů roste exponenciálně s počtem dimenzí

25

GROUP BY a UNION

```
SELECT znacka, pobocka, pocet
  FROM prodeje
UNION
SELECT 'ALL', pobocka,
  SUM(pocet) FROM prodeje
  GROUP BY pobocka
UNION
SELECT znacka, 'ALL',
  SUM(pocet) FROM prodeje
  GROUP BY znacka
UNION
SELECT 'ALL', 'ALL', SUM(pocet)
  FROM prodeje
```

znacka	pobocka	pocet
Škoda	Praha	28
Škoda	Brno	15
Škoda	Ostrava	13
VW	Praha	22
VW	Brno	13
VW	Ostrava	9
Audi	Praha	32
Audi	Brno	8
Audi	Ostrava	3
ALL	Praha	82
ALL	Brno	36
ALL	Ostrava	25
Škoda	ALL	56
VW	ALL	44
Audi	ALL	43
ALL	ALL	143

26

GROUP BY a UNION

- Nevýhody
  - Pro rostoucí velikost dimenze roste počet dotazů a sjednocení – je potřeba 2^N sjednocení (N je počet dimenzí přes které je seskupováno)
  - Náročnost výpočtu – v každém poddotazu se počítají data znova, dala by se ale již částečně využít spočítaná data
  - Dlouhý zápis dotazu, špatná údržba, náchylnost k chybám…
  - ALL jsou řetězcové hodnoty
- Motivace
  - Rozšíření SQL o operátory umožňující jednodušší zápis a rychlejší výpočet hodnot

27

Operátor CUBE

- Nový seskupovací operátor přidaný do standardu SQL 99
- Syntaxe:
  - `GROUP BY CUBE sloupec1, sloupec2, ...`
- Provádí seskupení podle všech možných podmnožin předaných sloupců (2^N různých agregací)
- Výpočet pomocí CUBE je rychlejší (2 a více-krát) než dotaz pomocí sjednocení dotazů

28

Operátor CUBE - příklad

- CUBE (znacka, pobocka, rok) vytvoří následující seskupení:

  (znacka, pobocka, rok)
  (znacka, pobocka)
  (znacka, rok)
  (pobocka, rok)
  (znacka)
  (pobocka)
  (rok)
  ()

- `SELECT znacka, pobocka, SUM(pocet) FROM prodeje GROUP BY CUBE znacka, pobocka`

29

Operátor ROLLUP

- Podobný operátoru CUBE
- Někdy nemusí dávat všechny kombinace seskupení smysl, např:
  - Seskupovat podle měsíce neseskupovat podle dne
- Produkuje hierarchii podle pořadí parametrů
  - Může se seskupovat podle daného sloupce pouze pokud není seskupeno podle sloupce uvedeného v zápisu dříve

30

Operátor ROLLUP - příklad

- ROLLUP (rok, mesic, den) vytvoří následující seskupení:

  (rok, mesic, den)
  (rok, mesic)
  (rok)
  ()

31

Podpora CUBE a ROLLUP

- Součástí standardu SQL 99
- Jsou podporovány např. v těchto SŘBD:
  - MS SQL Server
  - Oracle
  - DB2
- V MS SQL Serveru 2005 a starším je mírně odlišná syntaxe
  - GROUP BY <seznam sloupců> WITH [CUBE|ROLLUP]
  - Tato syntaxe byla použita v návrhu standardu

32

GROUPING SETS

- Nově v SQL 2006
  - Podpora v MS SQL Serveru, Oracle, DB2
- Možnost přímo určit množiny sloupců, přes které se bude seskupovat
  - Např.

```
GROUP BY GROUPING SETS((A,B), (C,D))
```

- Množina sloupců nemusí být vyjmenována přímo, může jít např. o výsledek operátoru CUBE
  - Např.

```
GROUP BY GROUPING SETS((A,B), CUBE(C,D))
```

33

Skládání operátorů CUBE, ROLLUP a GROUP BY

- Operátory tvoří hierarchii s následující algebrou:
  - CUBE(ROLLUP) = CUBE
  - CUBE(GROUP BY) = CUBE
  - ROLLUP(GROUP BY) = ROLLUP
- ROLLUP (stat,kraj), ROLLUP(rok,mesic) vytvoří následující seskupení:
  (stat, kraj, rok, mesic)
  (stat, kraj, rok)
  (stat, kraj)
  (stat, rok, mesic)
  (stat, rok)
  (stat)
  ()
- U operátoru CUBE by nemělo skládání smysl, protože CUBE(a,b), CUBE(c,d) by bylo to samé jako CUBE(a,b,c,d)
- Nejsilnější varianta:
  - `GROUP BY <sloupce> ROLLUP <sloupce> CUBE <sloupce>`

34

Hodnota ALL

- Co vlastně hodnota ALL znamená?
- Každá hodnota ALL reprezentuje množinu, přes kterou byla agregovaná hodnota spočítána
  - Např. ALL(model) = {Škoda, VW, Audi}
- Nebylo vhodné přidávat do SQL nové klíčové slovo pro hodnotu ALL – podobné problémy jako s NULL
- Stačí jedno NULL (jak pro původní NULL hodnotu, tak i pro novou ALL) a možnost rozeznat, jestli jde o ALL hodnotu a nebo o běžnou NULL hodnotu
- Přidána nová funkce GROUPING()

35

Funkce GROUPING

- Indikuje, zda hodnota v daném sloupci je agregovaná nebo ne
  - Vrací 1 pokud je hodnota agregovaná
  - Vrací 0 jinak

```
SELECT
CASE
WHEN GROUPING(model)=1
THEN 'ALL'
…
```

36

Funkce GROUPING_ID

- Přidána později (podporována minimálně v MS SQL Serveru a v Oracle)
- Vrací hodnotu úrovně seskupování jako bitovou mapu (i-tý bit je nastaven na 1 pokud bylo seskupováno podle i-tého sloupce)
  - GROUPING_ID (<seznam sloupců>)

```
SELECT A, B, GROUPING_ID(A, B)
FROM tabulka
GROUP BY CUBE A, B
```

Seskupováno podle sloupců	Bitová hodnota GROUPING_ID(A,B)	Návratová hodnota GROUPING_ID(A,B)
-	00	0
A	10	2
B	01	1
A, B	11	3

37

Multidimenzionální databázový model v relační databázi

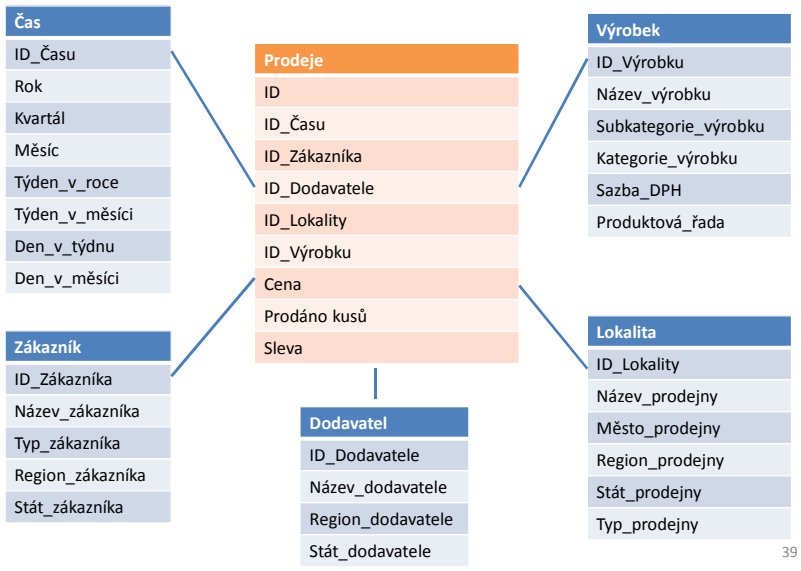
- Metriky – ukazatele, které měříme – např. počet prodaných kusů, čistý zisk, marže apod.
- Dimenze – kategorie, pro které se provádí měření – např. čas, skupiny produktů apod.

- 2 druhy tabulek
  - Tabulka faktů – obsahuje hodnoty metrik (fakty) a odkazy na dimenzní tabulky
  - Dimenzní tabulka – obsahuje elementy dimenzí
  - Nejnižší úroveň detailů v tabulce faktů musí odpovídat nejnižší úrovni detailu všech dimenzí

- Různé databázové modely
  - Základní schémata: schéma hvězdy a sněhové vločky
  - Složená schémata: schéma souhvězdí a sněhové bouře – složená z hvězd/vloček, více faktových tabulek

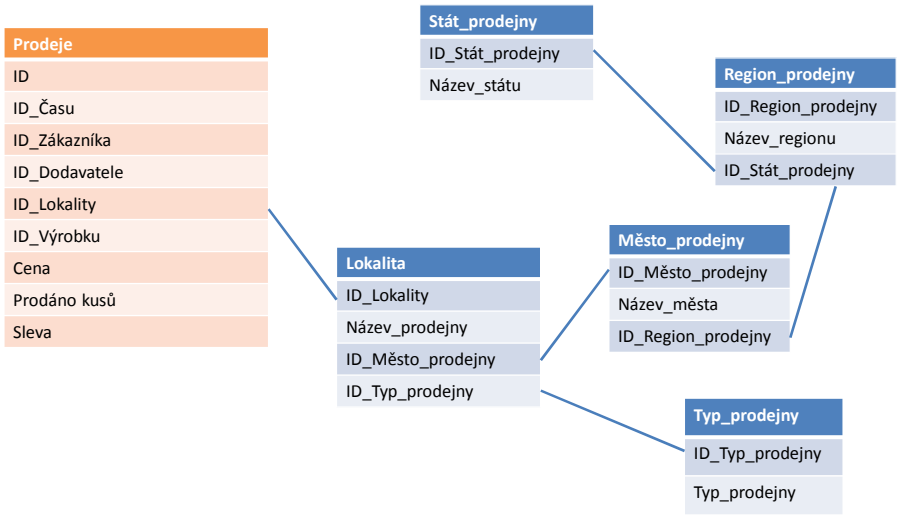
38

Schéma hvězdy (star schema)



39

Schéma sněhové vločky (snowflake schema)



40

Porovnání

Schéma hvězdy

- Data jsou denormalizovaná
- Větší nároky na místo
- Jednoduchost
- Rychlost – málo spojení
- Data na více místech – složitější aktualizace dat

Schéma sněhové vločky

- Data jsou ve 3NF
- Úspora místa
- Složitější dotazy
- Nižší výkon – hodně spojení
- Jednoduchá aktualizace dat
- Snadnější změny hierarchií

Většinou se používá kombinace obou schémat – částečná denormalizace

41

Typy agregačních funkcí

- Snaha o eliminaci duplicitních výpočtů dat
- U některých agregačních funkcí lze využít pro výpočet agregované hodnoty použít výsledky výpočtů sub-agregovaných dat

- Dělení agregačních funkcí na
  - Distributivní
  - Algebraické
  - Holistické

42

Distributivní agregační funkce

- Hodnotu funkce F je možné spočítat aplikací funkce G na sub-agregáty vypočítané funkcí F
- F(X) = G(F(X))
- Příklady
  - COUNT()
  - SUM()
  - MAX()
  - MIN()
  - …

	Praha	Brno	Ostrava	
Škoda	28	15	13	56
VW	22	13	9	44
Audi	32	8	3	43
	82	36	25	143

82+36+25 = 143 nebo 56+44+43 = 143

- Pro SUM, MAX, MIN platí F=G, pro COUNT G=SUM

43

Algebraické agregační funkce

- Hodnotu funkce F je možné spočítat aplikací funkce H na všechny výsledky funkce G
- F(X) = H(G(X))
- Výsledek G(X) musí být konstantní velikosti

- Příklady
  - AVG()
  - Směrodatná odchylka

44

Příklad – AVG()

	Praha	Brno	Ostrava	
Škoda	32	15	13	20 (60,3)
VW	20	13	9	14 (42,3)
Audi	32	11		21.5 (43,2)
	28 (84,3)	13 (39,3)	11 (22,2)	18.125 (145,8)

Výpočet možný na základě doplňujících informací (sumy a počtu hodnot) k vypočtenému výsledku sub-agregací.

45

Holistické agregační funkce

- Neexistuje konstantní počet hodnot popisující výsledek sub-agregované hodnoty, ze kterých by bylo možné vypočítat hodnotu funkce F
- Příklady
  - MEDIAN()
  - MODUS()
- Není známa efektivnější metoda výpočtu agregace než pomocí GROUP BY nad neagregovanými daty (vůbec se nepoužívají sub-agregáty)
- Snaha o co nejmenší používání, více se používají pro výpočet hodnoty různé aproximační techniky

46

Holistický princip, holistický přístup (Holistic Principle, Approach)

- **Holistický** přístup znamená celostní pohled na systém.
- Vlastnosti systému nelze určit jen pomocí vlastností jeho částí.
- Naopak celek ovlivňuje podobu a fungování svých částí.

47

Agregační funkce při modifikaci dat

- Často se používají triggery nad základními tabulkami k zajištění výpočtu při změně dat
- Náročnost výpočtu agregačních funkcí při modifikaci dat se může lišit od výběru dat

- MAX() je při použití v SELECT dotazu distributivní
- MAX() již nemusí být distributivní v případě modifikace dat
  - V případě vložení nových dat je výpočet jednoduchý – budeme propagovat novou hodnotu dokud bude větší než maximální aktuální hodnota
  - V případě smazání největší hodnoty v kostce je nutné přepočítat celou kostku
  - MAX() je distributivní pro SELECT a INSERT, ale holistická pro DELETE
- COUNT(), SUM() jsou algebraické pro INSERT i DELETE, údržba krychle není tak nákladná jako např. u MAX()

48

Zdroje

- J. Gray a kolektiv: Data cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals, New Orleans, March 1996
- http://msdn.microsoft.com/en-us/library/ms130214.aspx - MS SQL Server Books Online
- R.Vieira: Professional Microsoft SQL Server 2008 Programming, Wrox, Hoboken, 2009

"ISO/IEC DIS 9075-15 Information technology -- Database languages – SQL -- Part 15: Multi-dimensional arrays (SQL/MDA)". Retrieved 2018-05-27.

49