OLAP-КУБ И ЕГО ВЫЧИСЛЕНИЕ

Незнание кубизма не освобождает от Малевича.
Г. Александров

Технологии анализа данных

Содержание

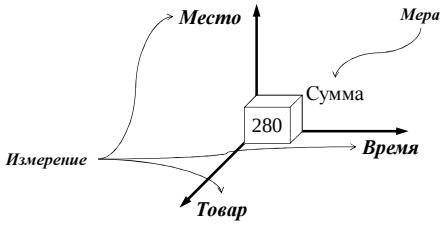
2

- Куб данных
- OLAP-куб
- Вычисление OLAP-куба

OLAP, многомерные данные

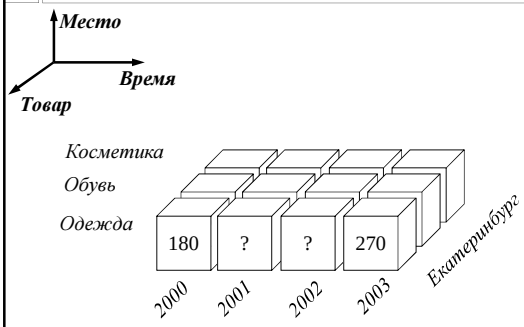
3

- *OLAP (On-line Analytical Processing)* – оперативная аналитическая обработка данных, поддерживающая многомерную модель данных.



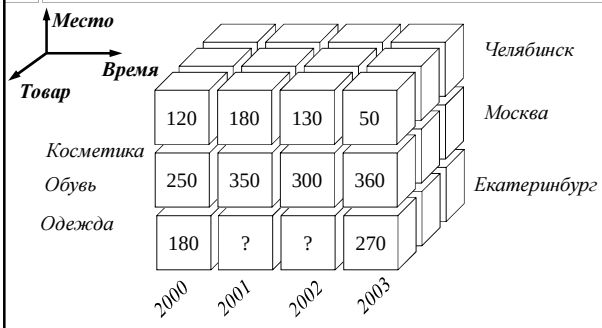
Куб данных

4



Куб данных

5



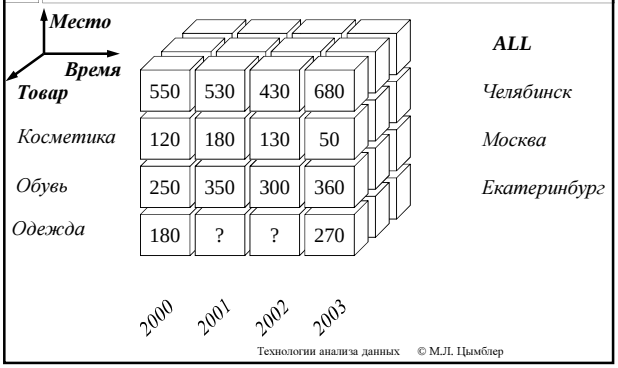
OLAP-куб

6

- *OLAP-куб* представляет собой куб данных, в котором каждое измерение дополняется значением *ALL* и полученные таким образом новые точки пространства вычисляются с помощью заданной *агрегатной функции*.
- Агрегатные функции
  - *Дистрибутивные*
    - count(), sum(), min(), max() и др.
  - *Алгебраические*
    - avg(), stddev() и др.
  - *Холистические меры*
    - median(), mode() и др.

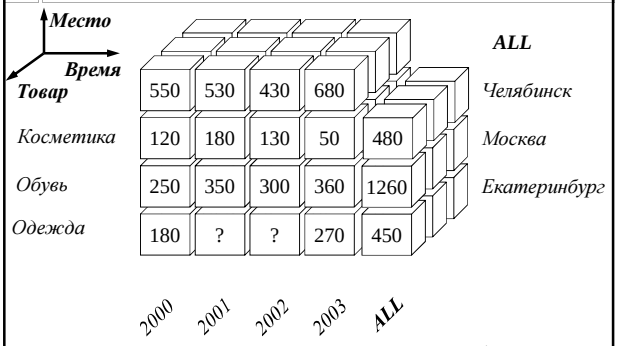
OLAP-куб

7



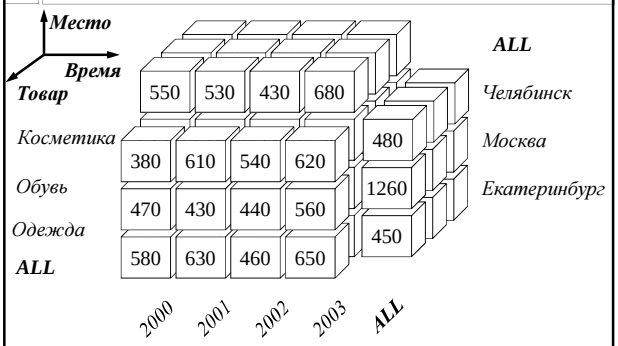
OLAP-куб

8



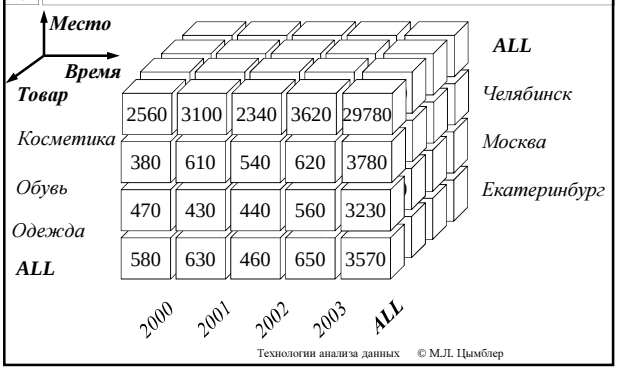
OLAP-куб

9



OLAP-куб

10



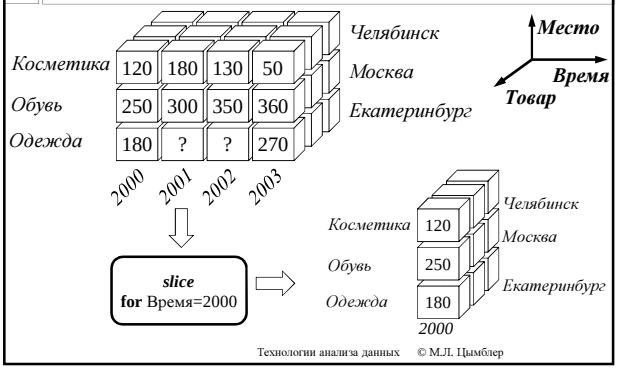
OLAP-операции (построение нового OLAP-куба)

11

- *Срез (slice and dice)*
  - проекция и/или отбор
- *Агрегация (roll-up, drill-up)*
  - вычисление меры при продвижении измерения снизу вверх по иерархии
- *Детализация (drill-down, roll-down)*
  - вычисление меры при продвижении измерения сверху вниз по иерархии
- *Вращение (pivot)*
  - изменение порядка представления (визуализации) измерений.

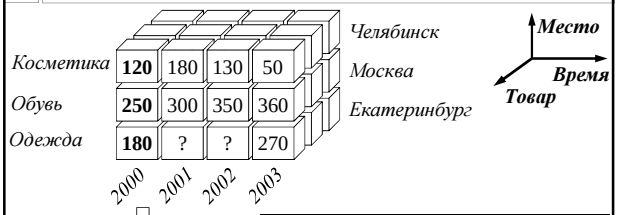
Срез (slice)

12



Срез (slice)

13



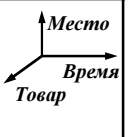
slice
for Время=2000

2000	*Екатеринбург*	*Москва*	*Челябинск*
Косметика	100	240	210
Обувь	320	170	320
Одежда	**180**	**250**	**180**

Вращение (pivot)

14

2000	*Екатеринбург*	*Москва*	*Челябинск*
Косметика	100	240	210
Обувь	320	170	320
Одежда	180	250	120

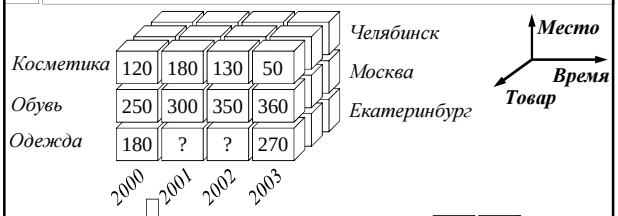


pivot

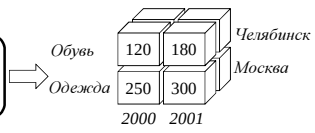
2000	*Одежда*	*Обувь*	*Косметика*
Екатеринбург	180	320	100
Москва	250	170	240
Челябинск	120	320	210

Срез (dice)

15

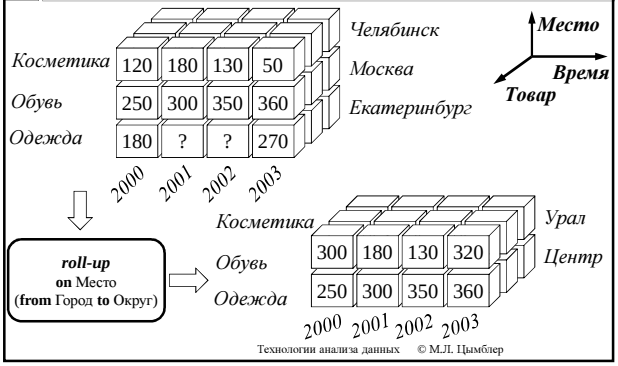


dice
for (Время=2000 or Время=2001) and (Место=Челябинск or Место=Москва) and (Товар=Обувь or Товар=Одежда)



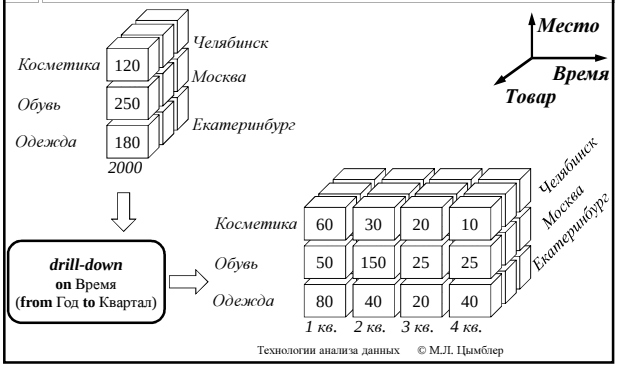
Агрегация (roll-up)

16



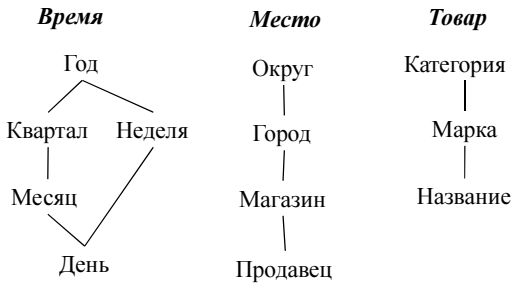
Детализация (drill-down)

17



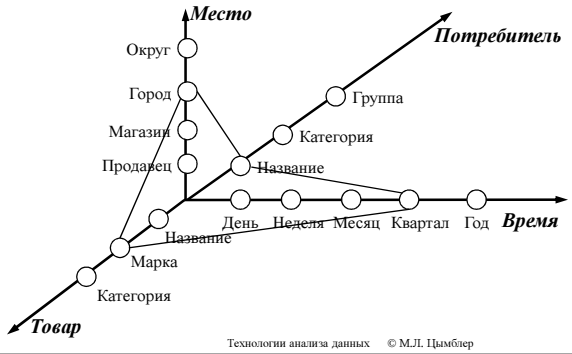
Иерархия в измерениях

18



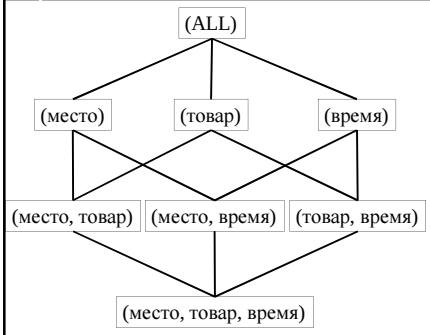
Модель OLAP-запросов

19



Решетки кубоидов

20



0-D (apex)
group by ()

1-D
group by (место)
group by (товар)
group by (время)

2-D
group by (место, товар)
group by (место, время)
group by (товар, время)

3-D (base)
group by
(место, товар, время)

ЧаВо о кубах

21

- Объем куба
  $V_{куб} = \prod_{i=1}^{n} d_i$
- Объем OLAP-куба
  $V_{OLAP-куб} = \prod_{i=1}^{n} (m + d_i)$
- Количество кубоидов в OLAP-кубе
  $Q_1 = 2^n \qquad Q_L = \prod_{i=1}^{n} (1 + L_i)$
- Как визуализировать k-мерный куб (k>3)?
  Как набор из d_k (k-1)-мерных кубов.

Вычисление OLAP-куба

22

- SQL
  - ROLLUP
  - CUBE
- Индексирование данных
- Материализация кубоидов

OLAP-куб и SQL

23

- ROLLUP BY
  - вычисление агрегата меры для каждого указанного измерения
  - вычисление частичных итогов (справа налево в списке группируемых измерений)
  - вычисление общего итога
- CUBE BY
  - вычисление агрегата меры для всех возможных комбинаций указанных измерений

ROLLUP BY

24

```
select Время, Место, Товар,
   sum(Сумма) as Прибыль
from Продажи
rollup by (Время, Место, Товар)
```

```
select Время, Место, Товар,
   sum(Сумма) as Прибыль
from Продажи
group by (Время, Место, Товар)
union
select Время, Место, ",
   sum(Сумма) as Прибыль
from Продажи
group by (Время, Место)
union
select Время, ", ",
   sum(Сумма) as Прибыль
from Продажи
group by (Время)
union
select ", ", ", sum(Сумма) as Прибыль
from Продажи
```

ROLLUP BY

25

Время	Место	Товар	Сумма
2000	Челябинск	Одежда	100
2000	Челябинск	Косметика	120
2000	Москва	Одежда	250
2000	Москва	Косметика	75
2001	Челябинск	Одежда	230
2001	Челябинск	Косметика	310
2001	Москва	Одежда	170
2001	Москва	Косметика	350

ROLLUP BY

26

```
select
   Время, Место, Товар,
   sum(Сумма) as Прибыль
from Продажи
rollup by (Время,
          Место, Товар)
```

Время	Место	Товар	Прибыль
2000	**Челябинск**	**Одежда**	100
2000	Челябинск	**Косметика**	120
2000	**Челябинск**	[NULL]	**220**
2000	**Москва**	**Одежда**	250
2000	Москва	**Косметика**	75
2000	**Москва**	[NULL]	**325**
2000	[NULL]	[NULL]	**545**
2001	**Челябинск**	**Одежда**	230
2001	Челябинск	**Косметика**	310
2001	**Челябинск**	[NULL]	**540**
2001	**Москва**	**Одежда**	170
2001	Москва	**Косметика**	350
2001	**Москва**	[NULL]	**520**
2001	**[NULL]**	**[NULL]**	**1 060**
[NULL]	**[NULL]**	**[NULL]**	**1 605**

CUBE BY

27

Время	Место	Товар	Сумма
2000	Челябинск	Одежда	100
2000	Челябинск	Косметика	120
2000	Москва	Одежда	250
2000	Москва	Косметика	75
2001	Челябинск	Одежда	230
2001	Челябинск	Косметика	310
2001	Москва	Одежда	170
2001	Москва	Косметика	350

CUBE BY

28

```
select
   Время, Место, Товар,
   sum(Сумма) as Прибыль
from Продажи
cube by (Время,
         Место, Товар)
```

Время	Место	Товар	Прибыль
2000	**Челябинск**	**Одежда**	100
2000	Челябинск	**Косметика**	120
2000	**Челябинск**	**[NULL]**	**220**
2000	**Москва**	**Одежда**	250
2000	Москва	**Косметика**	75
2000	**Москва**	**[NULL]**	**325**
2000	[NULL]	**Одежда**	**350**
2000	[NULL]	**Косметика**	**195**
2000	**[NULL]**	**[NULL]**	**545**
2001	**Челябинск**	**Одежда**	230
2001	Челябинск	**Косметика**	310
2001	**Челябинск**	**[NULL]**	**540**
2001	**Москва**	**Одежда**	170
2001	Москва	**Косметика**	350
2001	**Москва**	**[NULL]**	**520**

CUBE BY

29

```
select
   Время, Место, Товар,
   sum(Сумма) as Прибыль
from Продажи
cube by (Время,
         Место, Товар)
```

Время	Место	Товар	Прибыль
[NULL]	**Челябинск**	**Одежда**	330
[NULL]	Челябинск	**Косметика**	430
[NULL]	**Челябинск**	**[NULL]**	**760**
[NULL]	**Москва**	**Одежда**	420
[NULL]	Москва	**Косметика**	425
[NULL]	**Москва**	**[NULL]**	**845**
[NULL]	[NULL]	**Одежда**	**750**
[NULL]	[NULL]	**Косметика**	**855**
[NULL]	**[NULL]**	**[NULL]**	**1 605**

Обработка кубов данных: битовые индексы

30

RID	Время	Место	Товар	Сумма
R1	2000	Челябинск	Одежда	100
R2	2000	Челябинск	Косметика	120
R3	2001	Москва	Одежда	250
R4	2001	Москва	Косметика	75
R5	2001	Екатеринбург	Одежда	230
R6	2002	Челябинск	Обувь	310

Данные

RID	2000	2001	2002
R1	1	0	0
R2	1	0	0
R3	0	1	0
R4	0	1	0
R5	0	1	0
R6	0	0	1

Индекс
по измерению *Время*

Обработка кубов данных: битовые индексы

31

RID	Время	Место	Товар	Сумма
R1	2000	Челябинск	Одежда	100
R2	2000	Челябинск	Косметика	120
R3	2001	Москва	Одежда	250
R4	2001	Москва	Косметика	75
R5	2001	Екатеринбург	Одежда	230
R6	2002	Челябинск	Обувь	310

Данные

RID	Чел	Мск	Ект
R1	1	0	0
R2	1	0	0
R3	0	1	0
R4	0	1	0
R5	0	0	1
R6	1	0	0

Индекс
по измерению *Место*

Обработка кубов данных: битовые индексы

32

RID	Время	Место	Товар	Сумма
R1	2000	Челябинск	Одежда	100
R2	2000	Челябинск	Косметика	120
R3	2001	Москва	Одежда	250
R4	2001	Москва	Косметика	75
R5	2001	Екатеринбург	Одежда	230
R6	2002	Челябинск	Обувь	310

Данные

RID	Одеж	Косм	Обувь
R1	1	0	0
R2	0	1	0
R3	1	0	0
R4	0	1	0
R5	1	0	0
R6	0	0	1

Индекс
по измерению *Товар*

Обработка кубов данных: join-индексы

33

RID	Время	Место	Товар	Сумма
R1	2000	Челябинск	Одежда	100
R2	2000	Челябинск	Косметика	120
R3	2001	Москва	Одежда	250
R4	2001	Москва	Косметика	75
R5	2001	Екатеринбург	Одежда	230
R6	2002	Челябинск	Обувь	310

Данные

Место	RID
Екатеринбург	R5
Москва	R3
Москва	R4
Челябинск	R1
Челябинск	R2
Челябинск	R6

Индекс
по измерению *Место*

Обработка кубов данных: join-индексы

34

RID	**Время**	**Место**	**Товар**	**Сумма**
R1	2000	Челябинск	Одежда	100
R2	2000	Челябинск	Косметика	120
R3	2001	Москва	Одежда	250
R4	2001	Москва	Косметика	75
R5	2001	Екатеринбург	Одежда	230
R6	2002	Челябинск	Обувь	310

Данные

Товар	RID
Косметика	R4
Косметика	R6
Обувь	R6
Одежда	R1
Одежда	R3
Одежда	R5

Индекс
по измерению *Товар*

Обработка кубов данных: join-индексы

35

RID	**Время**	**Место**	**Товар**	**Сумма**
R1	2000	Челябинск	Одежда	100
R2	2000	Челябинск	Косметика	120
R3	2001	Москва	Одежда	250
R4	2001	Москва	Косметика	75
R5	2001	Екатеринбург	Одежда	230
R6	2002	Челябинск	Обувь	310

Данные

Товар	**Место**	RID
Косметика	Москва	R4
Косметика	Челябинск	R2
Обувь	Челябинск	R6
Одежда	Екатеринбург	R5
Одежда	Москва	R3
Одежда	Челябинск	R1

Индекс
по группе измерений
Товар, Место

Материализация кубоидов

36

- *Без материализации*
  - Вычисление агрегатов по запросу
    - Требует много времени
- *Полная материализация*
  - Предварительное вычисление всех кубоидов
    - Требует много пространства для хранения результатов
- *Частичная материализация*
  - Избирательное предварительное вычисление кубоидов
    - Критерий материализации кубоида
    - Эффективное использование материализованных кубоидов в обработке запросов
    - Эффективное обновление материализованных кубоидов

Обработка OLAP-запросов с помощью кубоидов

37

- Определить операции, которые должны быть выполнены над имеющимися кубоидами
  - OLAP-операции → SQL-операции (dice→select+project)
- Определить материализованные кубоиды, над которыми необходимо выполнить операции
  - Выбрать наименее затратный способ выполнения операций

Обработка OLAP-запросов с помощью кубоидов

38

- CUBE(Время, Товар, Место; sum(Сумма))
  - Время: день < месяц < квартал < год
  - Товар: название < марка < тип
  - Место: улица < город < округ < страна
- Запрос: {марка, округ} where год=2004
- Материализованные кубоиды:
  - C1: {год, название, город}
  - C2: {год, марка, страна}
  - C3: {год, марка, округ}
  - C4: {название, округ} where год=2004
- Какой кубоид можно (нужно) использовать для обработки запроса?

Обработка OLAP-запросов с помощью кубоидов

39

- CUBE(Время, Товар, Место; sum(Сумма))
  - Время: день < месяц < квартал < год
  - Товар: название < марка < тип
  - Место: улица < город < округ < страна
- Запрос: {марка, округ} where год=2004
- Материализованные кубоиды:
  - C1: {год, название, город}
  - C2: {год, марка, страна}
  - C3: {год, марка, округ}
  - C4: {название, округ} where год=2004
- Какой кубоид *можно* использовать для обработки запроса?
  - Не C2: C2.страна > округ
  - C1, C3, C4:
    - то же множество (надмножество) атрибутов, что и в запросе;
    - выборка в запросе подразумевает выборку в кубоиде

Обработка OLAP-запросов с помощью кубоидов

40

- CUBE(Время, Товар, Место; sum(Сумма))
  - Время: день < месяц < квартал < год
  - Товар: название < марка < тип
  - Место: улица < город < округ < страна
- Запрос: {марка, округ} where год=2004
- Материализованные кубоиды:
  - C1: {год, название, город}
  - C2: {год, марка, страна}
  - C3: {год, марка, округ}
  - C4: {название, округ} where год=2004
- Какой кубоид *нужно* использовать для обработки запроса?
  - Не C1: C1.название < Q.марка, C1.город < Q.округ
  - Если с измерением *название* ассоциировано не много различных значений измерения *год*, а для значений измерения *марка* – несколько значений измерения *название*, то C3 < C4 и следует предпочесть C3.
  - Однако, если для C4 существуют эффективные индексы, то лучше взять C4.

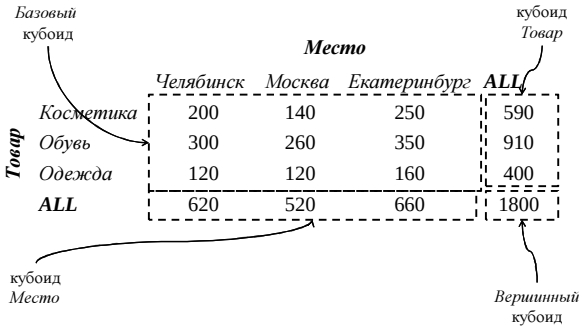
Типы ячеек OLAP-куба

41

- *Базовые* и *агрегатные*
  - *Базовая* – принадлежащая базовому кубоиду
    - 3-D: (2001, Обувь, Челябинск, 180)
  - *Агрегатная* – не принадлежащая базовому кубоиду.
    - 2-D: (2001, Обувь, *, 480), (*, Обувь, Москва, 350)
    - 1-D: (2001, *, *, 1380), (*, Обувь, *, 1260)
    - 0-D: (*, *, *, 25600)
- *Предки* и *потомки*
  - *c1*=(2001, *, *, 1380)
  - *c2*=(2001, Обувь, *, 480)
  - *c3*=(2001, Обувь, Челябинск, 180)
  - *c1* – предок *c2* и *c3*, *c3* – потомок *c2* и *c1*; *c2* – родительская для *c3*, *c3* – дочерняя для *c2*.

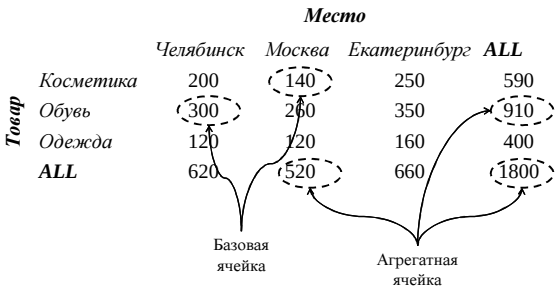
Типы ячеек OLAP-куба

42



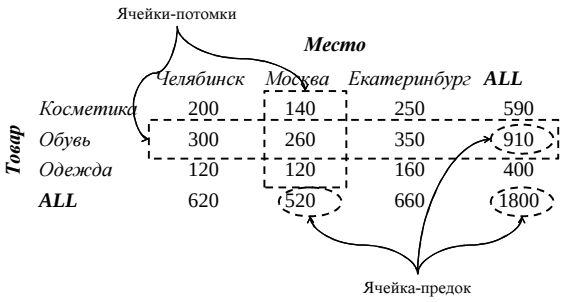
Типы ячеек OLAP-куба

43



Типы ячеек OLAP-куба

44



Типы OLAP-кубов

45

- *Полный куб (full cube)* – материализуются все ячейки всех кубоидов.
- *Куб-айсберг (iceberg cube)* – материализуются только ячейки, удовлетворяющие определенному условию.
- *Замкнутый куб (close cube)* – материализуются только ячейки-предки, которые имеют меру, большую, чем ячейки-потомки.
- *Каркасный куб (shell cube)* – материализуются только кубоиды с ограниченным количеством измерений.

Полный куб

46

- Вычисление полного куба имеет экспоненциальную сложность.
- Алгоритмы вычисления полного куба важны:
  - могут быть использованы для вычисления куба меньшего размера, который является полным кубом для данного подмножества измерений и/или значений измерений
  - помогают разработать эффективные методы вычисления частичных кубов.
- Альтернатива – *частичная материализация*:
  - вычисление *подмножества кубоидов* куба данных
  - вычисление *подкубов*, состоящих из подмножеств ячеек из различных кубоидов.

Куб-айсберг

47

- select месяц, город, категория, count(*)
  from Продажи
  cube by месяц, город, категория
  having count(*) >= *min_sup*

условие айсберга

- Куб-айсберг позволяет избавиться от ячеек, не несущих полезной информации.

Общие стратегии вычисления куба

48

- Сортировка, хеширование, группировка
- Одновременная агрегация и кэширование промежуточных результатов
- Агрегация от наименьшего потомка
- Принцип Apriori

Сортировка, хеширование, группировка ячеек куба

49

- Агрегация выполняется над ячейками, разделяющими общее множество значений атрибутов измерения (-ний). Поэтому важно сгруппировать эти данные, чтобы облегчить вычисления при агрегации.
  - При вычислении суммы продаж по *городу*, *дню* и *названию товара* наиболее эффективно выполнить сортировку ячеек по городу, затем по дню, и затем сгруппировать их по названию товара.

Одновременные вычисления и кэширование промежуточных рез-тов

50

- Вычисление агрегатов более высоких уровней иерархии путем использования ранее вычисленных агрегатов более низкого уровня иерархии, а не базовых ячеек.
  - При вычислении суммы продаж по городу можно использовать результат вычислений продаж по городу и дню.
- Одновременная агрегация предварительно сохраненных промежуточных результатов сокращает количество операций с диском.

Агрегация от наименьшего потомка

51

- Если имеется несколько вычисленных дочерних кубоидов, то вычисление родительского кубоида выгоднее проводить, используя наименьший из них.
  - Если имеются кубоиды *C1{город, год}* и *C2{город, товар}*, то вычисление кубоида *C{город}* лучше проводить на основе кубоида *C1*, поскольку различных наименований товара больше, чем различных значений лет.

Принцип Apriori

52

- Если данная ячейка не превосходит минимальную поддержку, то ни один ее потомков также не превосходит минимальную поддержку.
  - Если условие айсберга нарушено для некоторой ячейки, то оно будет нарушено для любого ее потомка.
  - Меры, поддерживающие данное свойство, называют *антимонотонными*.

Методы вычисления кубов

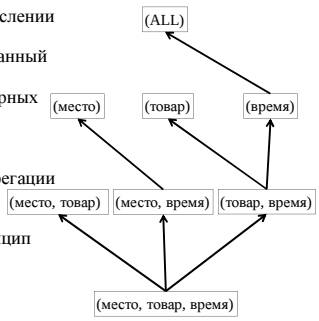
53

- Сверху вниз
  - Multiway
- Снизу вверх
  - BUC
  - H-cube
- Интеграция
  - Star-cubing

Multiway array aggregation

54

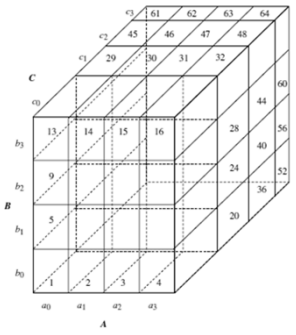
- Используется в MOLAP и вычислении полного куба
- Алгоритм "снизу-вверх", основанный на массивах
- Использует разбиение многомерных массивов на фрагменты
- Одновременная агрегация по нескольким измерениям
- Промежуточные результаты агрегации используются повторно для вычисления кубоидов-предков
- Невозможно использовать принцип *Apriori*



Multiway array aggregation

55

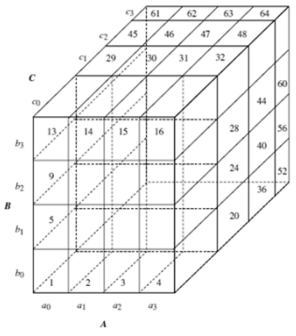
- Разбить массив на подкубы меньшего размера, достаточного для размещения в памяти.
- Адресация (chunkID, offset).
- Вычислить агрегаты путем посещения ячеек куба в таком порядке, который минимизирует количество посещений каждой ячейки.



Multiway array aggregation

56

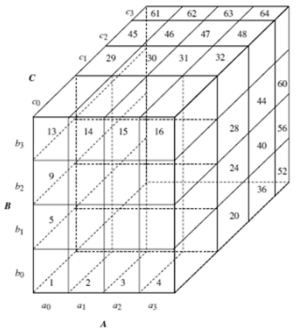
- Пусть вычисляется фрагмент b_0c_0 кубоида *BC*.
- b_0c_0 размещается в *буфере фрагментов* и вычисляется путем обработки фрагментов 1-4.
- Затем в буфер фрагментов можно поместить следующий фрагмент b_1c_0, агрегация которого вычисляется путем обработки фрагментов 5-8.
- Продолжая подобным образом, мы можем вычислить весь кубоид *BC*.
- Для вычисления всех фрагментов *BC* одномоментно необходимо присутствие только *одного* фрагмента в буфере фрагментов.



Multiway array aggregation

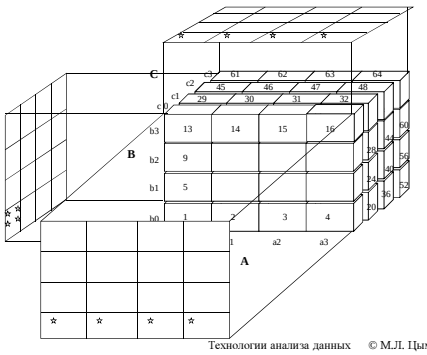
57

- При вычислении кубоида *BC* были сканированы все 64 фрагмента.
  - Вопрос: Существует ли способ избежать повторного сканирования этих фрагментов при вычислении кубоидов *AC* и *AB*?
  - Ответ: Да!
- Когда просканирован фрагмент 1, все 2-D фрагменты, относящиеся к $a_0b_0c_0$, должны быть одновременно вычислены: b_0c_0, a_0c_0, a_0b_0 на 2-D агрегатных плоскостях *BC*, *AC*, *AB*.
- Общее правило: одновременная агрегация проводится по каждой из 2-D плоскостей, пока 3-D фрагмент находится в памяти.



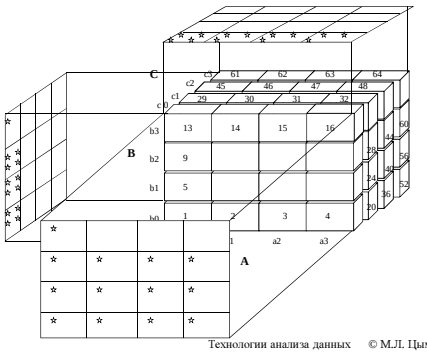
Multiway array aggregation

58



Multiway array aggregation

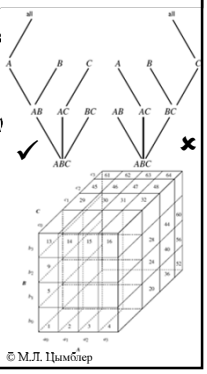
59



Multiway array aggregation

60

- Как порядок сканирования фрагментов влияет на эффективность вычисления куба?
  - Пусть *A*, *B*, *C* имеют размеры 40, 400, 400
    - *AB*=16 000
    - *AC*=16 000
    - *BC*=160 000
- Порядок сканирования 1, 2, 3, …
  - 156 000 блоков памяти
- Порядок сканирования 1, 17, 33, 49, …
  - 1 641 000 блоков памяти



Multiway array aggregation

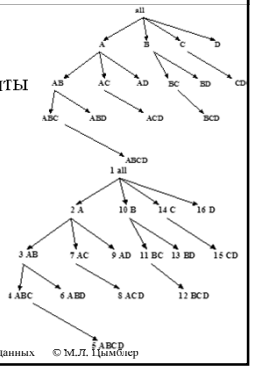
61

- Метод: плоскости должны быть отсортированы и вычислены в соответствии с уменьшением их размера.
- Идея: хранить наименьшую плоскость в памяти, брать и вычислять только один фрагмент одномоментно для наибольшей плоскости.
- Ограничение метода: хорошо работает только для небольших измерений.
- Если имеется большое количество измерений, полезно рассмотреть методы "сверху-вниз" и /или методы вычисления айсберг-кубов.

BUC (Bottom-Up Computation)

62

- Вычисление айсберг-кубов от вершинного кубоида к базовому
- Разбиение измерений на фрагменты для обеспечения отбрасывания нижней части айсберга
- Если фрагмент не превосходит минимальную поддержку, его потомки могут быть отброшены
- Если *min_sup* =1, вычисляем полный куб!
- Нет одновременной агрегации.



BUC

63

Algorithm: BUC. Algorithm for the computation of sparse and iceberg cubes.

Input:

- *input*: the relation to aggregate;
- *dim*: the starting dimension for this iteration.

Globals:

- constant *numDims*: the total number of dimensions;
- constant *cardinality*[*numDims*]: the cardinality of each dimension;
- constant *min_sup*: the minimum number of tuples in a partition in order for it to be output;
- *outputRec*: the current output record;
- *dataCount*[*numDims*]: stores the size of each partition. *dataCount*[*i*] is a list of integers of size *cardinality*[*i*].

Output: Recursively output the iceberg cube cells satisfying the minimum support.

Method:

```
(1)  Aggregate(input); // Scan input to compute measure, e.g., count. Place result in outputRec.
(2)  if input.count() == 1 then // Optimization
         WriteAncestors(input[0], dim); return;
     endif
(3)  write outputRec;
(4)  for (d = dim; d < numDims; d++) do //Partition each dimension
(5)      C = cardinality[d];
(6)      Partition(input, d, C, dataCount[d]); //create C partitions of data for dimension d
(7)      k = 0;
(8)      for (i = 0; i < C; i++) do // for each partition (each value of dimension d)
(9)          c = dataCount[d][i];
(10)         if c >= min_sup then // test the iceberg condition
(11)             outputRec.dim[d] = input[k].dim[d];
(12)             BUC(input[k...k+c], d+1); // aggregate on next dimension
(13)         endif
(14)         k += c;
(15)     endfor
(16)     outputRec.dim[d] = all;
(17) endfor
```

BUC

64

- select A, B, C, D, count(*)
  from K
  cube by A, B, C, D
  having count(*)>3