Формализация информации и Big Data

«02.03.03 - Математическое обеспечение и администрирование информационных систем направленность разработка и администрирование информационных систем»

http//vikchas.ru

Лекция 12 «СУБД Greenplum для Big Data цифровой экономике»

Часовских Виктор Петрович
д-р техн. наук, профессор кафедры ШИиКМ

ФГБОУ ВО «Уральский государственный экономический университет»

Екатеринбург 2024

СУБД Greenplum - ПАРТИЦИОНИРОВАНИЕ

Партиционирование, или секционирование, представляет собой логическое разделение таблицы на части по определенному критерию.

Применение партиционирования на больших таблицах значительно улучшает производительность запросов, а также упрощает сопровождение базы данных благодаря возможности разделять перемещать и компактно хранить данные в зависимости от их востребованности.

Партиционированные таблицы, также, как и любые таблицы в СУБД Greenplum, имеют **ключ дистрибьюции**.

Важно различать понятия: **дистрибьюция** — это распределение данных по сегментам с целью равномерно распределить нагрузку по всем узлам кластер; **партиционирование** - это метод оптимизации, при котором на каждом сегменте таблица делится на части с целью увеличения производительности запросов за счет выборочного сканирования. Различия дистрибьюции и партиционирования:

Партиционирование распространяется на все сегменты и выполняется на каждом сегменте одинаково.

Например, если таблица при создании поделена на 6 партиций, то на каждом сегменте кластера будет по 6 партиций.

Когда таблица партиционирована, запрос выполняется гораздо быстрее за счет того, что на каждом сегменте сканируется только часть таблицы.

Дистрибьюция в СУБД Greenplum обязательно для всех таблиц, то есть все таблицы в СУБД Greenplum являются распределенными и имеют ключ дистрибьюции.

В партиционирование — это лишь опция, которую можно применять к таблице в целях оптимизации.

В СУБД Greenplum поддерживаются следующие виды позиционирования:

PARTITION BY RANGE - по диапазону значений;

PARTITION BY LIST – по списку значений;

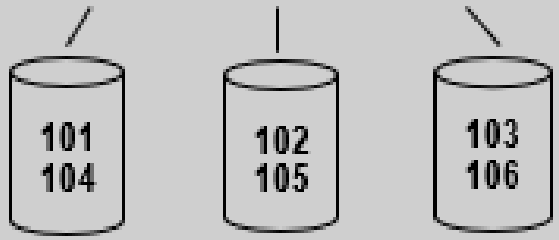
PARTITION BY ... SUBPARTITION BY – многоуровневое.

Рассмотрим таблицу, содержащую записи о клиентах учебного заведения, например, студентах университета

ВИДЫ ПАРТИЦИОНИРОВАНИЯ

id	group	name	birthday
101	A1	Student1	20.08.1995
102	A1	Student2	19.01.1996
103	A1	Student3	22.04.1997
104	A1	Student4	12.11.1998
105	B2	Student5	25.04.1999
106	C3	Student6	01.06.2000

```
CREATE TABLE client
(id numeric,
group text,
name text,
birthday date)
DISTRIBUTED BY id;
```



Согласно заданной политике дистрибьюции, таблица равномерно распределена по сегментам на основе идентификаторов студентов.

Допустим, что для построения отчетности периодически требуется анализировать информацию о всех учащихся того или иного возраста. Вместе с тем мы хотим, чтобы система выдавала результаты запросов для разных возрастов как можно быстрее. Эту задачу поможет решить Range. Партиционирование по году рождения

ПАРТИЦИОНИРОВАНИЕ ПО ДИАПАЗОНУ ЗНАЧЕНИЙ (RANGE)

id	group	name	birthday
101	A1	Student1	20.08.1995
102	A1	Student2	19.01.1996
103	A1	Student3	22.04.1997
104	A1	Student4	12.11.1998
105	B2	Student5	25.04.1999
106	C3	Student6	01.06.2000

```
CREATE TABLE client
(id numeric, group text, name text, birthday date)
DISTRIBUTED BY id
PARTITION BY RANGE (birthday)
    (START (birthday '1995-01-01') INCLUSUV
    END (birthday '2001-01-01') EXCLUSIVE
    EVERY (INTERVAL '1 year'));
```

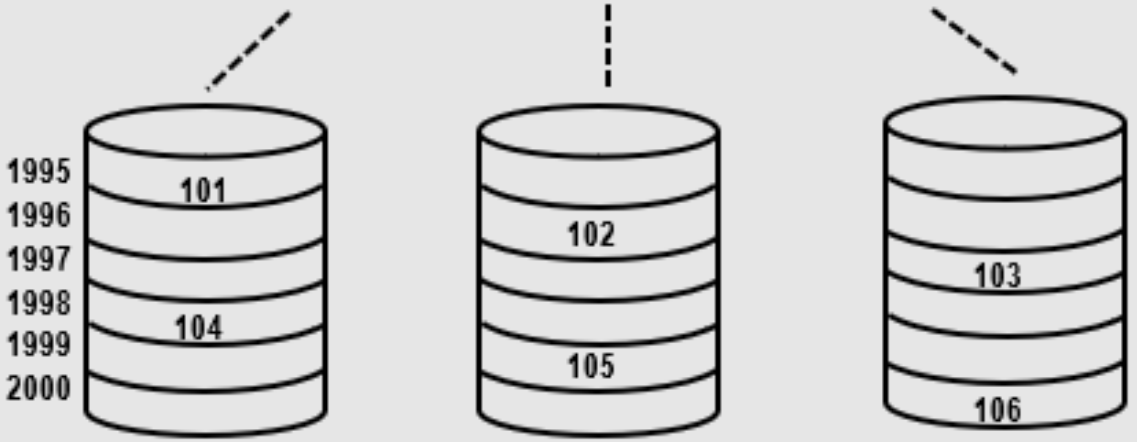
Вид партиционирования задается при создании таблицы с помощью выражения **Partition By Range**. Диапазон значений задан выражением **start** и **end**, а шаг каждой партийцы — выражением **every** интервал. После создания таблицы на каждом сегменте будут созданы по 6 партийцы и хранящих записей по году.

Если в запросе к таблице есть условия фильтрации записи и погоду, то база данных в рамках каждого сегмента сразу обращается к той партийцы, которая соответствует нужному году, в связи с чем запрос выполняется намного быстрее

ПАРТИЦИОНИРОВАНИЕ ПО ДИАПАЗОНУ ЗНАЧЕНИЙ (RANGE)

id	group	name	birthday
101	A1	Student1	20.08.1995
102	A1	Student2	19.01.1996
103	A1	Student3	22.04.1997
104	A1	Student4	12.11.1998
105	B2	Student5	25.04.1999
106	C3	Student6	01.06.2000

```
SELECT id FROM client
WHERE birthday >= to date ('01.01.1996', 'DD.MM.YYYY')
      and birthday < to date ('01.01.1997', 'DD.MM.YYYY');
```



В случае партиционирования типа List разделения данных выполняется на основе списка значений.

Например, мы можем применить к той же таблице партиционирование по полю «группа студентов»

ПАРТИЦИОНИРОВАНИЕ ПО СПИСКУ ЗНАЧЕНИЙ (LIST)

id	group	name	birthday
101	A1	Student1	20.08.1995
102	A1	Student2	19.01.1996
103	A1	Student3	22.04.1997
104	A1	Student4	12.11.1998
105	B2	Student5	25.04.1999
106	C3	Student6	01.06.2000

```
CREATE TABLE client
(id numeric, group text, name text, birthday date)
DISTRIBUTED BY id
PARTITION BY LIST (group)
  (PARTITION A1 VALUES ('A1'),
  PARTITION B2 VALUES ('B2'),
  PARTITION C3 VALUES ('C3'));
```

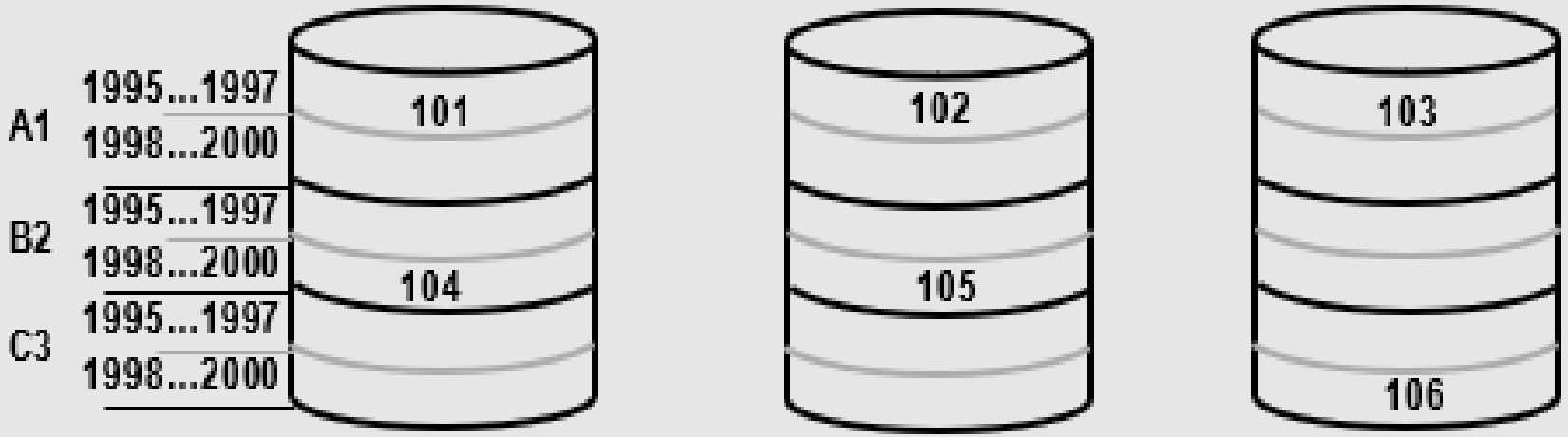
Разделение партийцы и по значениям выполняется с помощью выражения **Partition Values**.

На каждом сегменте создается по 3 партийцы. Для групп A1, B2 и C3. Внутри каждой партийцы есть студенты с разными годами рождения. Например, запись о студенте 4 хранится в сегменте 1. «» в партийцы A1. Этот способ партиционирования будет удобен, если требуется часто строить запросы с фильтрацией по группе студентов, то, кроме того, в одной таблице можно реализовать многоуровневое партиционирование, при котором будут использоваться комбинации обоих типов. Проводится по группе студентов, а внутри группы — по году рождения

МНОГОУРОВНЕВОЕ ПАРТИЦИОНИРОВАНИЕ

```
CREATE TABLE client DISTRIBUTED BY id
PARTITION BY LIST (group)
  (PARTITION A1 VALUES ('A1'),
  PARTITION B2 VALUES ('B2'),
  PARTITION C3 VALUES ('C3'))
    SUBPARTITION BY RANGE (birthday)
      (START (birthday '1995-01-01') INCLUSIV
      END (birthday '2001-01-01') EXCLUSIV
      EVERY (INTERNAL '3 year'));
```

id	group	name	birthday
101	A1	Student1	20.08.1995
102	A1	Student2	19.01.1996
103	A1	Student3	22.04.1997
104	A1	Student4	12.11.1998
105	B2	Student5	25.04.1999
106	C3	Student6	01.06.2000



Следует иметь в виду, что использование многоуровневого партиционирования может привести к созданию избыточного количества партийцев, что, в свою очередь, может замедлить работу всей базы данных.

В при создании партиционированной таблицы можно указывать

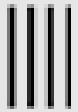
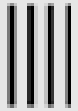
- Отличающиеся имена;
- Отличающиеся диапазоны;
- Разные опции хранения (ориентация, компрессия);
- Можно создать партицию по умолчанию (DEFAULT) для значений, не удовлетворяющих условиям других партиций.

Нельзя:

- Указать разные ключи дистрибьюции для партиций одной таблицы;
- Поменять тип партиционирования без пересоздания таблицы

Рассмотрим пример стратегии партиционирования с применением разных опций хранения данных.

ПРИМЕР СТРАТЕГИИ ПАРТИЦИОНИРОВАНИЯ И ХРАНЕНИЯ ДАННЫХ

Глубина	Ориентация	Партиции	Способ хранения	Компрессия
Сегодня		1 день	HEAP	-
- 3 мес.		1 мес.	AO	Низкая (×2,5) Zstd
-12 мес.		1 год	AO	Средняя (×7) Zstd
-24 мес.		1 год	AO	Высокая (×15) Zstd
-120 мес.		1 год	Внешняя таблица	-

Наиболее свежие данные хранятся без компрессии для того, чтобы можно было быстро выполнять операции обновления.

Старые данные хранятся с более сильными алгоритмами сжатия. Например, для данных трехмесячной давности применяется низкая степень сжатия, что обеспечивает хорошую производительность, так как выполняется мало операций ввода-вывода.

Для данных годичной давности применяется колоночная ориентация с более высокой степенью компрессии, что повышает производительность при выборке по подмножеству колонок.

Для данных глубиной 2 года используется колоночная ориентация и максимальная компрессия. При этом производительность ниже, так как при выполнении запросов строки приходится разжимать.

Архивные данные хранятся за пределами базы данных в виде файлов.

Обращаем внимание, что цель партиционирования состоит в исключении чтения лишних партиций, то есть минимизации количества сканируемых партиций при выполнении запроса.

Следовательно, наиболее эффективным будет такое позиционирование, которое максимально приближено к логике выполнения запроса.

Последовательность операции при выполнении запроса отражается в плане запроса, который выводится с помощью команды Explan.

Рассмотрим план выполнения запросов в случае таблицы с партиционированием или без него.

ПРИМЕР ХОРОШЕГО КЛЮЧА ПАРТИЦИОНИРОВАНИЯ

План запроса

```
EXPLAIN
SELECT *
FROM sales
WHERE
  date = '01-07-21'
AND region = 'usa'
```

QUERY PLAN — **БЕЗ партиций**

```
Gather Motion 32:1 (slice1; segment: 32) (cost=0.00..6834.88 rows=44906244 width=49
-> Seq Scan on sales (cost=0.00..1679.05 rows=1403321 width=49)
      Filter: ((date='2020-04-18'::date) AND (region='usa'::text))
Optimizer: Pivotal Optimizer (GPORCA)
```

QUERY PLAN — **С партициями**

```
Gather Motion 32:1 (slice1; segment: 32) (cost=0.00..6048.26 rows=4689939 width=49
-> Sequence (cost=0.00..663.58 rows=1465611 width=49)
   -> Partition Selector for tloan_prt (dynamic scan id: 1) (cost=10.00..100.00 rows=4 width=4)
      Partition selected: 1 (out of 60)
   -> Dynamic Seq Scan on sales_prt (dynamic scan id: 1) (cost=0.00..663.58 rows=1465611
      width=49)
      Filter: ((date='2020-04-18'::date) AND (region='usa'::text))
Optimizer: Pivotal Optimizer (GPORCA)
```

В плане запроса для таблицы без партиционирования сканируется вся таблица, поэтому стоимость выполнения запроса получается высокой. Для таблицы с партиционированием мы видим, что сканируется только одна нужная партиция, в связи с чем стоимость запроса низкая, и, как следствие, запрос выполняется намного быстрее.

Рассмотрим синтаксис запросов - для разных партиций можно указывать разные интервалы и опции хранения.

На рисунке для партиций с интервалом год или 3 месяца задана колоночная ориентация хранения и тип компрессии задан с разными коэффициентами сжатия.

СИНТАКСИС. ИНТЕРВАЛЫ ХРАНЕНИЯ ДЛЯ ПАРТИЦИЙ

```
CREATE TABLE orders
(order_id BIGINT,
order_date TIMESTAMP WHIT ZONE,
order_mode VARCHAR(8),
customer_id INTEGER)
DISTRIBUTED BY (customer_id)
PARTITION BY RANGE (order_date)
    (start (date '2008-01-01') end (date '2011-01-01') every (interval '1 year')
        with (appendonly=true, orientation=column, compresstype=zlib, compresslevel=3),
    start (date '2011-01-01') end (date '2013-01-01') every (interval '3 months')
        with (appendonly=true, orientation=column, compresstype=zlib, compresslevel=1),
    start (date '2013-01-01') end (date '2014-01-01') every (interval '1 months')
        with (appendonly=true, orientation=row, compresstype=zstd),
    start (date '2014-01-01') end (date '2014-02-01') every (interval '1 day'));
```

Для партиций с интервалом один месяц применяется ориентация по строкам и другой тип компрессии без явного указания коэффициента сжатия. Для партиций с интервалом один день используется способ хранения по умолчанию.

Рассмотрим синтаксис изменение партиции

Добавление партиции

```
ALTER TABLE card_list ADD PARTITUON n_a values (‘NA’);
```

```
ALTER TABLE card_list ADD PARTITUON new
    START (‘2016-01-01’::date) END (‘2017-01-01’::date)
    WITH (appendonly=’true’, compresstype=zlib, compresslevel=’9’);
```

```
ALTER TABLE card_list ADD DEFAULT PARTITUON other;
```

Для изменения партийцы используется команда **Alter table** с дополнительными опциями.

ALTER TABLE sales RENAME PARTITION FOR ('2016-01-01') TO jan16;

Очистка партиции.

ALTER TABLE tabLe.prt.mixed TRUNCATE PARTITION year 2019;

ALTER TABLE tabLe.prt.mixed TRUNCATE PARTITION FOR ('2019-04-28'::date);

Для очистки партийцы и применяется команда Trancate Partition. В примере показано, как очистить партицию с названием year 2019.0 или партицию, содержащую дату 28.0 апреля 2019.0 года.

Удаление партиции.

ALTER TABLE cardUst DROP PARTITION n.a;

ALTER TABLE card.list DROP DEFAULT PARTITION;

Удаление партиции.

ALTER TABLE cardUst DROP PARTITION n.a IF EXIST;

ALTER TABLE card.list DROP DEFAULT PARTITION IF EXIST;

Для удаления партиции используется команда DROP PARTITION.

В примере показано удаление обычной и дефолтной партии. При желании в тексте команды можно указать опцию IF EXIST, которая поможет избежать сообщения об ошибке при выполнении команды в случае отсутствия удаляемой партиции. Для изменения опции хранения в отдельной партии можно создать таблицу с нужной опцией хранения, вставить в нее данные из этой партиции и затем заменить всю прежнюю партицию на только что созданную таблицу с помощью команды ALTER TABLE.

Разделение партиции.

ALTER TABLE sales SPLIT PARTITION FOR ('2021-01-01') AT ('2021-01-16')

INTO (PARTITION jan21.1to15, PARTITION jan21.16to31);

ALTER TABLE sales SPLIT DEFAULT PARTITION START ('2021-01-01') INCLUSIVE

END ('2021-02-01') EXCLUSIVE

INTO (PARTITION jan21, default partition);

CREATE TABLE sales (trans.id int, date date, amount decimal (9,2), region text) DISTRIBUTED BY (trans.id) PARTITION BY RANGE (date) SUBPARTITION BY LIST (region) SUBPARTITION TEMPLATE

(SUBPARTITION usa VALUES ('usa'), SUBPARTITION asia VALUES ('asia'), SUBPARTITION europe VALUES ('europe'), DEFAULT SUBPARTITION other.regions) (START (date '2014-01-01') INCLUSIVE END (date '2014-04-01') EXCLUSIVE EVERY (INTERVAL '1 month'));

Для разделения одной партиции на две части используется команда ALTER TABLE … SPLIT PARTITION. В начале примера выполняется разделение партийцы, в которой хранятся данные за январь, на две партии. В первой будут храниться данные с первого по 15 января, а во второй — с шестнадцатого по 31, если в вашей таблице есть дефолт-партийцы, то добавить новую партицию в такой таблице можно только путем разделения дефолт партийцы, как это показано во втором части. В этом случае в предложении INTO в качестве второй позиции необходимо указать дефолтпартицию.

ПРИМЕРЫ DDL(Data Definition Language)-ОПЕРАЦИЙ

Создание и изменение шаблонов субпартиций.

```
ALTER TABLE sales ADD PARTITION "4"
START ('2014-04-01') INCLUSIVE
END ('2014-05-01') EXCLUSIVE ;
```

При создании таблицы возможно определение шаблона, по которому будут автоматически создаваться наборы субпартиций при добавлении новых партиций.

Создание и изменение шаблонов субпартиций.

```
ALTER TABLE sales ADD PARTITION "4"
START ('2014-04-01') INCLUSIVE
END ('2014-05-01') EXCLUSIVE ;
ALTER TABLE sales SET SUBPARTITION TEMPLATE ( SUBPARTITION usa VALUES ('usa'), SUBPARTITION asia VALUES ('asia'), SUBPARTITION europe VALUES ('europe'), SUBPARTITION africa VALUES ('africa'), DEFAULT SUBPARTITION regions);
```

В приведенном скрипте создания шаблон определяется с помощью предложения SUBPARTITION TEMPLATE, в котором мы определяем 4 субпартиции для разных регионов. Теперь, если мы добавляем в таблицу партийцу и следующим скриптом, то эта партиция по шаблону автоматически разделится на 4 субпартиции и для всех регионов.

Просмотр информации о партициях.

Информация о ключах партиционирования:

SELECT * FROM **pg_catalog.pg_partitions** WHERE tablename = 'sales';

С помощью представления о pg Partitions можно получить подробную информацию о партиционированных таблицах и их иерархии. Представление pg Partition Colance отображает информацию о колонках, которые используются для партиционирования.

Рекомендации по партиционированию данных.

Когда использовать.

Большая таблица.

Фильтры запросов (WHERE) исключают часть партиций.

Нужно "скользящее окно" данных при периодической загрузке архивировании.

Рекомендации по партиционированию данных

Как использовать.

Нужно:

1. Предпочитать RANGE, а не LIST.
2. Партиционировать по часто используемому столбцу.
3. Убедиться в исключении ненужных партиций в плане запроса (EXPLAIN).
4. Выбирать наилучший способ физического хранения для разных партиций (например, по строкам/колонкам).

Не рекомендуется:

1. Создавать большое количество партиций.
2. Использовать дефолтную партицию.
3. Использовать многоуровневое партиционирование.
4. Секционировать по столбцу ключа дистрибьюции.
5. Создавать много партиций при колоночной ориентации (количество физических файлов = сегменты * столбцы *

Обобщение

Традиционно, администрирование ИС взаимосвязано с администрирование БД и её СУБД.

Сквозные технологии цифровой экономики РФ Big Date и Blockchain изменили структуру и функции ИС, существенно изменили функции администрирования данных и программ их обработки.

Появился новый тип СУБД для Big Data. СУБД Greenplum – open-source продукт, массивно-параллельная реляционная СУБД для хранилищ данных с гибкой горизонтальной масштабируемостью и столбцовым хранением данных на основе PostgreSQL.

Благодаря своим архитектурным особенностям и мощному оптимизатору запросов, СУБД Greenplum отличается особой надежностью и высокой скоростью обработки SQL-запросов над большими объемами данных, поэтому эта MPP-СУБД широко применяется для аналитики Big Data в промышленных масштабах.

Основная идея организации хранения и обработки Big Data распараллеливание хранения и обработки данных.

Эти особенности становятся определяющими в администрирование подобных ИС.

Показано преимущество для решения проблемы масштабируемости в ИС аналитической обработки больших данных архитектуры MPP (massive parallel processing), или архитектуры массивно-параллельной обработки данных. В такой архитектуре система состоит из нескольких независимых узлов, соединенных по сети. При этом в каждом вычислительном узле процессор обладает своими собственными физическими ресурсами, такими как память и диски, которые не разделяются с другими узлами, что должно являться основой в администрирование.

СУБД Greenplum лучше всего подходит для построения корпоративных хранилищ данных, решение аналитических задач и задач машинного обучения и искусственного интеллекта.

СУБД Greenplum представляет из себя множество модифицированных экземпляров дисковых баз данных PostgreSQL, работающих совместно как одна связанная система управления базами данных. СУБД Greenplum предоставляет широкий выбор инструментов для решения аналитических задач.

Основным средством управления данными является встроенный процедурный языке SQL, но можно применять и другие популярных языках программирования, которые будут компилироваться и выполняться внутри базы данных.

Есть возможность разрабатывать процедуры, которые будут работать в контейнерах. Такой подход позволяет сделать разработку безопасной путем изолирования исполняемого кода от операционной системы, на которой установлена СУБД.

Дистрибьюция или распределении данных, является одним из важнейших факторов быстродействия ИС.

Партиционирование, или секционирование как логическое разделение таблицы на части по определенному критерию.

Применение партиционирования на больших таблицах значительно улучшает производительность запросов, а также упрощает сопровождение базы данных благодаря возможности разделять, перемещать и компактно хранить данные в зависимости от их востребованности.

Благодарю за внимание!