# **BigTable**

BigTable — это база данных разработанная компанией Google для собственных нужд. Эта база представляет собой большую таблицу с тремя измерениями: **колонки, строки** и **временные метки**. Такая архитектура позволяет добиться очень высокой производительности, кроме того, она хорошо масштабируется на множество компьютеров. Но это не реляционная база, и она не поддерживает многие возможности реляционных баз. В частности в BigTable нет join-ов, нет сложных запросов и т.д. BigTable предназначена для хранения значений, таких как миллиарды веб-страниц, со многими версиями на странице; более 100 ТБ данных спутниковых изображений; сотни миллионов пользователей; и выполнение тысячи запросов в секунду. BigTable используется более чем в шестидесяти продуктов и проектов Google, включая Google Analytics, Google Finance, Orkut, Writely и Google Earth. Google не распространяет BigTable, поэтому на рынке появилось несколько независимо разработанных клонов этой базы. В частности это такие проекты как: Hadoop, Hypertable и Cassandra.

**Характеристика**

*Map*. *Map* представляет собой ассоциативный массив; структура данных, которая позволяет быстро искать значение для соответствующего ключа. BigTable представляет собой набор пар (ключ, значение), где ключ идентифицирует строку и значение является набор столбцов.

*Persistant.* Данные хранятся на диске постоянно.

*Distributed.* Данные BigTable распределяется среди многих независимых машин. В Google, BigTable построен на вершине GFS (Google File System). В Apache HBase, BigTable построен на вершине HDFS (Hadoop Distributed File System). Таблица разбивается между строками, с группами соседних строк, управляемых сервером. Сама строка никогда не распределяется.

*Sparse.* Таблица разреженная, а это означает, что разные строки в таблице могут использовать различные столбцы, со многими из пустыми столбцами для конкретной строки.

*Sorted.* Большинство ассоциативные массивы не сортируются. BigTable сортирует свои данные с по ключу. Это помогает держать соответствующие данные близко друг к другу, как правило, на той же машине. Например, если доменные имена используются в качестве ключей в BigTable, имеет смысл хранить их в обратном порядке, чтобы гарантировать, что соответствующие домены близко друг к другу. Например:

1. edu.some.cs
2. edu.some.nb
3. edu.some.www

*Multidimensional.* Таблица индексируется по строкам. Каждая строка содержит один или несколько именованных семейств столбцов. *семейство столбцов* определяются, когда таблица создается. В пределах *семейства столбцов*, может быть один или несколько именованных столбцов. Все данные внутри семейства столбцов, как правило, того же типа. Реализация BigTable обычно сжимает все столбцы в *семействе столбцов* вместе. Столбцы внутри *семейства столбцов* могут быть созданы на лету. Например:

edu.rutgers.cs" : { // row  
 "users" : { // column family  
 "watrous": "Donald", // column  
 "hedrick": "Charles", // column  
 "pxk" : "Paul" // column  
 }  
 "sysinfo" : { // another column family  
 "" : "SunOS 5.8" // column (null name)  
 }  
 }

Получить данные можно таким образом, полное имя в виде (семейство столбцов: столбец). К примеру, users:pxk.

*Time-based.* Каждое *семейство столбцов* может хранить несколько версий данных *семейства столбцов*. Если приложение не определяет временную ветку, будет получена самая последняя версия *семейства столбцов*. В качестве альтернативы, можно указать временную ветку и получить самую последнюю версию, которая <= этой временной метки.

# **Столбцы и *семейства столбцов***

# 

Пример таблицы, которая хранит веб-страницы (этот пример от Google, BigTable). Ключ строки это страница URL. К примеру, com.cnn.www. Различные атрибуты страницы хранятся *семействе столбцов.*  Столбец может содержать короткое значение, как в *семействе столбцов “language”.* В BigTable, однако, нет никакого типа, ассоциированного с колонкой. Это - только набор байтов.

# **Строки и разделение**

Таблица логически разделена между рядами на несколько подтаблицам называемых *tablet*. *Tablet* представляет собой набор последовательных строк таблицы и является единицей распределения и балансировки нагрузки в пределах BigTable. Поскольку таблица всегда сортируется по строкам, считывает из коротких диапазонов строк эффективны: один, как правило, взаимодействует с небольшим количеством машин. Следовательно, ключом к обеспечению высокой степени локальности для выбора ключей строк должным образом (как в предыдущем примере использования доменных имен в обратном порядке).

# **Временные ветки**

Каждая ячейка *семьи столбцов* может содержать несколько версий контента. Например, в предыдущем примере, мы можем иметь несколько версий содержимого страницы, связанной с URL. Каждая версия идентифицируется 64-битной временной меткой, которая либо представляется в режиме реального времени или значение, назначенное клиентом.

# **Реализация**

*HBase.* **Описание:** Open Source разработка на основе оригинального дизайна Google Bigtable от Apache. Разработана в рамках проекта Hadoop. Используется самим Facebook в качестве основы сервиса обмена сообщениями.

**Производительность:** на кластере из 7 серверов (16Gb RAM, 8x core CPU, HDD) проводились операции в таблице с 3 миллиардами записей. Было запущено 300 процессов чтения/записи одновременно, измерялось время, затраченное на операцию. В результате, среднее время записи составило 10ms, чтения — 18ms.

*Hypertable.* **Описание:** интересная разработка, похожая на HBase. Имеет чуть большую производительность и значительно более привычный по синтаксису запросов HQL.

**Производительность:** Производительность похожа на HBase*.*

*Cassandra.*  **Описание:** Распределенное хранилище, изначально разрабатываемое в Facebook, впоследствии переданное в Apache. Имеет язык запросов CQL, очень похожий на SQL с некоторыми ограничениями.

**Производительность:** хорошие сравнительные тесты с графиками, по результатам которых, на 8 серверах при соотношении 50/50 операций чтения/записи *Cassandra* совершает около 9.000 операций в секунду. *HBase* делает около 2.500 при тех же условиях.