Reading Club

Zhiyi Xiao

Ch3. Storage and Retrieval

- storage engines
  - log-structured storage engines,
  - page-oriented storage engines

```bash
#!/bin/bash

db_set () {
    echo "$1,$2" >> database
}

db_get () {
    grep "^$1," database | sed -e "s/^$1,//" | tail -n 1
}
```

```
$ db_set 123456 '{"name":"London","attractions":["Big Ben","London Eye"]}'

$ db_set 42 '{"name":"San Francisco","attractions":["Golden Gate Bridge"]}'

$ db_get 42
{"name":"San Francisco","attractions":["Golden Gate Bridge"]}
```

Hash Indexs

key
byte offset
123456
0
42
64
In-memory hash map
Log-structured file on disk
(each box is one byte)
1 2 3 4 5 6 , { " n a m e " : " L o n d o n " , " a t t r a
0
10
20
c t i o n s " : [" B i g B e n " , " L o n d o n E y e
30
40
50
"] } \n 4 2 , { " n a m e " : " S a n F r a n c i s c o "
60
70
80
, " a t t r a c t i o n s " : [" G o l d e n G a t e B
90
100
110
r i d g e "] } \n
120

Compaction

Data file segment

mew: 1078	purr: 2103	purr: 2104	mew: 1079	mew: 1080	mew: 1081
purr: 2105	purr: 2106	purr: 2107	yawn: 511	purr: 2108	mew: 1082

Compaction process

Compacted segment

yawn: 511	mew: 1082	purr: 2108

SSTable

handbag: 8786
handful: 40308
handicap: 65995
handkerchief: 16324
handlebars: 3869
handprinted: 11150
Segment 1
handcuffs: 2729
handful: 42307
handicap: 67884
handiwork: 16912
handkerchief: 20952
handprinted: 15725
Segment 2
handful: 44662
handicap: 70836
handiwork: 45521
handlebars: 3869
handoff: 5741
handprinted: 33632
Segment 3
+
Compaction and merging process
handbag: 8786
handcuffs: 2729
handful: 44662
handicap: 70836
handiwork: 45521
handkerchief: 20952
handlebars: 3869
handoff: 5741
handprinted: 33632
Merged 1, 2, 3

Sparse index
in memory
key
byte offset
…
…
hammock
100491
handbag
102134
handsome
104667
hangout
106812
…
…
Sorted segment file (SSTable) on disk
……
hand: 91541
handbag: 8786
handcuffs: 2729
handful: 44662
handicap: 70836
handiwork: 45521
handkerchief: 20952
handlebars: 3869
handoff: 5741
handprinted: 33632
handsome: 86478
handwaving: 44005
handwriting: 22846
……
compressible block

Constructing and maintaining SSTables

- When a write comes in, add it to an **in-memory balanced tree data structure** (for example, a red-black tree).
- When the memtable gets bigger than some **threshold**—typically a few megabytes —write it out to disk as an SSTable file.
- In order to serve a read request, **first try to find the key in the memtable**, then in the most recent on-disk segment, then in the next-older segment, etc.
- From time to time, **run a merging and compaction process in the background** to combine segment files and to discard overwritten or deleted values.

B-Tree

- **branching factor**
  - **page references**
  - **range boundaries**

Disk I/O Cost

DISK I/O COST

Draveness

Queuing	3ms
Seek	4ms
Half Rotation	2ms
Transfer	1ms
Total	10ms

B+ Tree

MYSQL INNODB B+ TREE
Draveness
6
3
5
7
8
id
1
name
draven
id
4
name
draveness
id
2
name
github
id
6
name
draven
id
7
name
draven

After adding key 334:

B-Trees vs LSM-Trees

- LSM-trees are typically faster for writes, whereas B-trees are thought to be faster for reads.
- LSM-Trees
  -

Transaction Processing or Analytics?

- An application typically looks up a small number of records by some key, using an index.
- Records are inserted or updated based on the user's input.

- What was the total revenue of each of our stores in January?
- How many more bananas than usual did we sell during our latest promotion?
- Which brand of baby food is most often purchased together with brand X diapers?

transaction processing versus analytic systems

Property	Transaction processing systems (OLTP)	Analytic systems (OLAP)
Main read pattern	Small number of records per query, fetched by key	Aggregate over large number of records
Main write pattern	Random-access, low-latency writes from user input	Bulk import (ETL) or event stream
Primarily used by	End user/customer, via web application	Internal analyst, for decision support
What data represents	Latest state of data (current point in time)	History of events that happened over time
Dataset size	Gigabytes to terabytes	Terabytes to petabytes

Data Warehouse

dim_product table

product_sk	sku	description	brand	category
30	OK4012	Bananas	Freshmax	Fresh fruit
31	KA9511	Fish food	Aquatech	Pet supplies
32	AB1234	Croissant	Dealicious	Bakery

dim_store table

store_sk	state	city
1	WA	Seattle
2	CA	San Francisco
3	CA	Palo Alto

fact_sales table

date_key	product_sk	store_sk	promotion_sk	customer_sk	quantity	net_price	discount_price
140102	31	3	NULL	NULL	1	2.49	2.49
140102	69	5	19	NULL	3	14.99	9.99
140102	74	3	23	191	1	4.49	3.89
140102	33	8	NULL	235	4	0.99	0.99

dim_date table

date_key	year	month	day	weekday	is_holiday
140101	2014	jan	1	wed	yes
140102	2014	jan	2	thu	no
140103	2014	jan	3	fri	no

dim_customer table

customer_sk	name	date_of_birth
190	Alice	1979-03-29
191	Bob	1961-09-02
192	Cecil	1991-12-13

dim_promotion table

promotion_sk	name	ad_type	coupon_type
18	New Year sale	Poster	NULL
19	Aquarium deal	Direct mail	Leaflet
20	Coffee & cake bundle	In-store sign	NULL

Column-Oriented Storage

fact_sales table

date_key	product_sk	store_sk	promotion_sk	customer_sk	quantity	net_price	discount_price
140102	31	3	NULL	NULL	1	2.49	2.49
140102	69	5	19	NULL	3	14.99	9.99
140102	74	3	23	191	1	4.49	3.89
140102	33	8	NULL	235	4	0.99	0.99

```sql
SELECT
  dim_date.weekday, dim_product.category,
  SUM(fact_sales.quantity) AS quantity_sold
FROM fact_sales
  JOIN dim_date    ON fact_sales.date_key    = dim_date.date_key
  JOIN dim_product ON fact_sales.product_sk = dim_product.product_sk
WHERE
  dim_date.year = 2013 AND
  dim_product.category IN ('Fresh fruit', 'Candy')
GROUP BY
  dim_date.weekday, dim_product.category;
```

fact_sales table

date_key	product_sk	store_sk	promotion_sk	customer_sk	quantity	net_price	discount_price
140102	69	4	NULL	NULL	1	13.99	13.99
140102	69	5	19	NULL	3	14.99	9.99
140102	69	5	NULL	191	1	14.99	14.99
140102	74	3	23	202	5	0.99	0.89
140103	31	2	NULL	NULL	1	2.49	2.49
140103	31	3	NULL	NULL	3	14.99	9.99
140103	31	3	21	123	1	49.99	39.99
140103	31	8	NULL	233	1	0.99	0.99

Columnar storage layout:

date_key file contents: 140102, 140102, 140102, 140102, 140103, 140103, 140103, 140103
product_sk file contents: 69, 69, 69, 74, 31, 31, 31, 31
store_sk file contents: 4, 5, 5, 3, 2, 3, 3, 8
promotion_sk file contents: NULL, 19, NULL, 23, NULL, NULL, 21, NULL
customer_sk file contents: NULL, NULL, 191, 202, NULL, NULL, 123, 233
quantity file contents: 1, 3, 1, 5, 1, 3, 1, 1
net_price file contents: 13.99, 14.99, 14.99, 0.99, 2.49, 14.99, 49.99, 0.99
discount_price file contents: 13.99, 9.99, 14.99, 0.89, 2.49, 9.99, 39.99, 0.99

Column Compression

Column values:

product_sk:	69	69	69	69	74	31	31	31	31	29	30	30	31	31	31	68	69	69

Bitmap for each possible value:

product_sk = 29:	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
product_sk = 30:	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0
product_sk = 31:	0	0	0	0	0	1	1	1	1	0	0	0	1	1	1	0	0	0
product_sk = 68:	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
product_sk = 69:	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
product_sk = 74:	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0

Run-length encoding:

product_sk = 29:	9, 1	(9 zeros, 1 one, rest zeros)
product_sk = 30:	10, 2	(10 zeros, 2 ones, rest zeros)
product_sk = 31:	5, 4, 3, 3	(5 zeros, 4 ones, 3 zeros, 3 ones, rest zeros)
product_sk = 68:	15, 1	(15 zeros, 1 one, rest zeros)
product_sk = 69:	0, 4, 12, 2	(0 zeros, 4 ones, 12 zeros, 2 ones)
product_sk = 74:	4, 1	(4 zeros, 1 one, rest zeros)

Aggregation: Data Cubes and Materialized Views

```
SELECT SUM(net_price)
FROM fact_sales
WHERE date_key = 140101
AND product_sk = 32
```

```
SELECT SUM(net_price)
FROM fact_sales
WHERE date_key = 140101
```

product_sk

date_key	32	33	34	35	……	total
140101	149.60	31.01	84.58	28.18	……	40710.53
140102	132.18	19.78	82.91	10.96	……	73091.28
140103	196.75	0.00	12.52	64.67	……	54688.10
140104	178.36	9.98	88.75	56.16	……	95121.09
……	……	……	……	……	……	……
total	14967.09	5910.43	7328.85	6885.39	……	lots

```
SELECT SUM(net_price)
FROM fact_sales
WHERE product_sk = 32
```