SQL performance tutorial

Roan Kattouw
Berlin Hackathon 2012

Indexes

- Pre-sorted lists of rows
- Used to speed up queries

`page_id PRIMARY KEY`

page_id	**page_len**
1	10775
67	1305
69	548
70	34
74	50085
77	10292
...	...

`INDEX foo (page_len)`

page_id	***page_len***
1320629	15
9609464	15
340226	16
725948	16
940255	16
1065064	16
...	...

```
SELECT * FROM page
ORDER BY page_touched LIMIT 5;
```

page_id	**page_touched**
1	20120522185124
67	20120515122531
69	20120115072601
70	20120123124730
74	20120520174854
...	...

page_id	***page_touched***
10212	20050626044452
13394	20050626044503
13395	20050626044503
28307	20050626044522
31045	20050626044523

```
SELECT * FROM page
ORDER BY page_len LIMIT 5;
```

page_id	***page_len***
1320629	*15*
9609464	*15*
340226	*16*
725948	*16*
940255	*16*
1065064	16
…	…

page_id	***page_len***
1320629	15
9609464	15
340226	16
725948	16
940255	16

```
SELECT * FROM page
WHERE page_is_new=0 LIMIT 5;
```

page_id	**page_is_new**
1	0
67	0
69	0
70	0
74	0
77	0
78	*1*
83	0
...	...

page_id	**page_is_new**
78	1
110	1
221	1
262	1
263	1

```
SELECT * FROM page
WHERE page_id = 94109 LIMIT 1;
```

page_id
1
67
69
...
94106
94109
94112
...

binary search

```
SELECT * FROM page
WHERE page_id >= 94109
ORDER BY page_id LIMIT 3;
```

page_id
1
67
...
94106
94109
94112
94115
94123
...

binary search

```
SELECT * FROM page
WHERE page_id >= 94109
ORDER BY page_len LIMIT 3;
```

page_id	**page_len**
1	10775
67	1305
...	...
94106	465
94109	***688***
94112	***533***
94115	***31***
94123	***287***
...	...

page_id	***page_len***
1320629	15
9609464	15
340226	16

```
SELECT * FROM image
WHERE img_name LIKE 'Flag\_of\_C%'
ORDER BY img_name LIMIT 3;
```

img_name
...
Flag_of_Březany.jpeg
Flag_of_CANU.svg
Flag_of_CARICOM.svg
Flag_of_CEFTA.svg
Flag_of_CELAC.png
...

img_name
Flag_of_CANU.svg
Flag_of_CARICOM.svg
Flag_of_CEFTA.svg

```
SELECT * FROM image
WHERE img_name LIKE 'Flag\_of\_C%'
ORDER BY img_size LIMIT 3;
```

img_name	**img_size**
...	
Flag_of_Březany.jpeg	9264
Flag_of_CANU.svg	*11071*
Flag_of_CARICOM.svg	*3651*
...	*...*
Flag_of_Cúcuta_(Norte_de_Santander).svg	*358694*
Flag_of_DDR(hanging).gif	23913

img_name	***img_size***
Flag_of_Chhatarpur.svg	181
Flag_of_Ceuta_without_coa.svg	192
Flag_of_Cossonay.svg	192

```
SELECT * FROM image
WHERE img_name LIKE '%.jpg'
ORDER BY img_name LIMIT 3;
```

img_name
Fl-90-0020-Hitchhiking.jpg
Fl-map-natfor.gif
Fl-resultaat.pdf
Fl-waldeck-Grab.JPG
Fl.jpg
Fl.png
...

img_name
Fl-90-0020-Hitchhiking.jpg
Fl.jpg
Fl10_001.jpg

Index on two fields

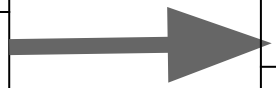
Works just like a phone book

lastname	firstname	phone
Stevens	Daniel	526-3401
Stevenson	Amy	945-8547
Stevenson	John	324-0625
Stevin	Aaron	777-6004

page_namespace	page_title	page_id
0	Main_Page	3401
0	Nanobots	526
1	Main_Page	8547
2	Catrope	9

```
SELECT * FROM image
WHERE img_user_text='Rotatebot'
ORDER BY img_timestamp LIMIT 3;
```

img_user_text	***img_timestamp***
...	
Rostocker	20120305200249
Rotatebot	*20080831180021*
Rotatebot	*20090528000041*
Rotatebot	*20090908180012*
Rotatebot	20091005180217
...	...



img_user_text	***img_timestamp***
Rotatebot	20080831180021
Rotatebot	20090528000041
Rotatebot	20090908180012

```
SELECT * FROM image
WHERE img_user_text LIKE 'Ro%'
ORDER BY img_timestamp LIMIT 3;
```

img_user_text	***img_timestamp***
Rnt20	20050502210737
RoRo	*20061215190509*
RoRo	*20070303165159*
RoRo	*20070303165610*
RoadyTom	*20060906094405*
...	*...*
Royy1984	*20111026191844*
Rp.	20081210184810

img_user_text	***img_timestamp***
Roger Zenner	20050317173844
Romanm	20050730223825
Rosenzweig	20050813151322

```
SELECT * FROM image
WHERE img_user_text LIKE 'Ro%'
ORDER BY img_user_text,
img_timestamp LIMIT 3;
```

img_user_text	***img_timestamp***
Rnt20	20050502210737
RoRo	*20061215190509*
RoRo	*20070303165159*
RoRo	*20070303165610*
RoadyTom	*20060906094405*
...	*...*
Royy1984	*20111026191844*
Rp.	20081210184810

img_user_text	***img_timestamp***
RoRo	20061215190509
RoRo	20070303165159
RoRo	20070303165610

LIMIT yourself

- Limit number of rows returned (LIMIT)
- Limit number of rows *scanned* (harder)

DON'T

- Run unindexed queries
  - WHERE on rarely false conditions usually OK
  - Unindexed ORDER BY (*filesort*) **never OK**
- ORDER BY expression --> filesort == bad
- Page with OFFSET (or LIMIT 50,50)
  - LIMIT 10 OFFSET 5000 scans 5010 rows
  - Use WHERE foo_id >= 12345 instead
- Use COUNT(), SUM(), GROUP BY, etc
  - No limit on rows scanned
  - MAX()/MIN() of indexed field on entire table is OK

DO

- Think about how the DB will run your query
- Add indexes where needed
- Batch queries (when it makes sense)
- In some cases, denormalize for performance
  - Add information duplicated from other tables
  - Summary tables, counter tables, cache tables, etc.

ContributionScores query

```
SELECT user_id, user_name, user_real_name, page_count, rev_count, page_count+SQRT
(rev_count-page_count)*2 AS wiki_rank
FROM `bw_user` u
JOIN (
  (
    SELECT rev_user, COUNT(DISTINCT rev_page) AS page_count, COUNT(rev_id) AS rev_count
    FROM `bw_revision`
    WHERE rev_user NOT IN (SELECT ug_user FROM `bw_user_groups` WHERE ug_group='bot')
    GROUP BY rev_user ORDER BY page_count DESC LIMIT 50
  ) UNION (
    SELECT rev_user, COUNT(DISTINCT rev_page) AS page_count, COUNT(rev_id) AS rev_count
    FROM `bw_revision`
    WHERE rev_user NOT IN (SELECT ug_user FROM `bw_user_groups` WHERE ug_group='bot')
    GROUP BY rev_user ORDER BY rev_count DESC LIMIT 50
  )
) s ON (user_id=rev_user) ORDER BY wiki_rank DESC LIMIT 50;
```