Efficient Pagination Using MySQL

Surat Singh Bhati (surat@yahoo-inc.com)
Rick James (rjames@yahoo-inc.com)

Yahoo Inc

Percona Performance Conference 2009

Outline

1. Overview
   - Common pagination UI pattern
   - Sample table and typical solution using OFFSET
   - Techniques to avoid large OFFSET
   - Performance comparison
   - Concerns

Common Patterns

41 to 80 Newest | < Newer | Older >

1 to 20 of about 374314 Newest | < Newer | Older >

Showing 25 - 48 of 4,593 Results « Previous | **Page:** **1** 2 3 ... | Next

Page 2 of 2128 < 1 **2** 3 4 12 52 102 502 1002 > Last »

« Recent posts Earlier posts »

Basics

First step toward having efficient pagination over large data set

- Use index to filter rows (resolve WHERE)
- Use same index to return rows in sorted order (resolve ORDER)

Step zero

- http://dev.mysql.com/doc/refman/5.1/en/mysql-indexes.html
- http://dev.mysql.com/doc/refman/5.1/en/order-by-optimization.html
- http://dev.mysql.com/doc/refman/5.1/en/limit-optimization.html

Using Index

KEY a_b_c (a, b, c)

ORDER may get resolved using Index

- `ORDER BY a`
- `ORDER BY a,b`
- `ORDER BY a, b, c`
- `ORDER BY a DESC, b DESC, c DESC`

WHERE and ORDER both resolved using index:

- `WHERE a = const ORDER BY b, c`
- `WHERE a = const AND b = const ORDER BY c`
- `WHERE a = const  ORDER BY b, c`
- `WHERE a = const AND b > const ORDER BY b, c`

ORDER will not get resolved uisng index (file sort)

- `ORDER BY a ASC, b DESC, c DESC /* mixed sort direction */`
- `WHERE g = const ORDER BY b, c  /* a prefix is missing */`
- `WHERE a = const ORDER BY c     /* b is missing */`
- `WHERE a = const ORDER BY a, d  /* d is not part of index */`

Sample Schema

```
CREATE TABLE `message` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `title` varchar(255) COLLATE utf8_unicode_ci NOT NULL,
  `user_id` int(11) NOT NULL,
  `content` text COLLATE utf8_unicode_ci NOT NULL,
  `create_time` int(11) NOT NULL,
  `thumbs_up` int(11) NOT NULL DEFAULT '0', /* Vote Count */
  PRIMARY KEY (`id`),
  KEY `thumbs_up_key` (`thumbs_up`,`id`)
) ENGINE=InnoDB

 mysql> show table status like 'message' \G
         Engine: InnoDB
        Version: 10
     Row_format: Compact
           Rows: 50000040    /* 50 Million */
 Avg_row_length: 565
    Data_length: 28273803264 /* 26 GB */
   Index_length: 789577728   /* 753 MB */
      Data_free: 6291456
    Create_time: 2009-04-20 13:30:45
```

Two use case:

- Paginate by time, recent message one page one
- Paginate by thumps_up, largest value on page one

Typical Query

1. Get the total records
   ```
   SELECT count(*) FROM message
   ```

2. Get current page
   ```
   SELECT * FROM message
   ORDER BY id DESC LIMIT 0, 20
   ```

- http://domain.com/message?page=1
  - `ORDER BY id DESC LIMIT 0,  20`
- http://domain.com/message?page=2
  - `ORDER BY id DESC LIMIT 20,  20`
- http://domain.com/message?page=3
  - `ORDER BY id DESC LIMIT 40,  20`

Note: id is auto_increment, same as create_time order, no need to create index on create_time, save space

Explain

```
mysql> explain  SELECT * FROM message
       ORDER BY id  DESC
       LIMIT 10000, 20\G
*************** 1. row **************
           id: 1
  select_type: SIMPLE
        table: message
         type: index
possible_keys: NULL
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 10020
        Extra:
1 row in set (0.00 sec)
```

- it can read rows using index scan and execution will stop as soon as it finds required rows.
- LIMIT 10000, 20 means it has to read 10020 and throw away 10000 rows, then return next 20 rows.

Performance Implications

- Larger OFFSET is going to increase active data set, MySQL has to bring data in memory that is never returned to caller.
- Performance issue is more visible when your have database that can't fit in main memory.
- Small percentage of request with large OFFSET would be able to hit disk I/O Disk I/O bottleneck
- In order to display “21 to 40 of 1000,000” , some one has to count 1000,000 rows.

Simple Solution

- Do not display total records, does user really care?

- Do not let user go to deep pages, redirect him http://en.wikipedia.org/wiki/Internet_addiction_disorder after certain number of pages

Avoid Count(*)

1. Never display total messages, let user see more message by clicking 'next'

   41 to 80 Newest | < Newer | Older >

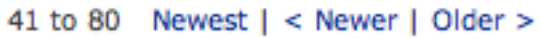
2. Do not count on every request, cache it, display stale count, user do not care about 324533 v/s 324633

3. Display 41 to 80 of **Thousands**

4. Use pre calculated count, increment/decrement value as insert/delete happens.

Solution to avoid offset

1. Change User Interface
   - No direct jumps to Nth page



2. LIMIT N is fine, Do not use LIMIT M,N
   - Provide extra **clue** about from where to start given page
   - Find the desired records using more restricted WHERE using given **clue** and ORDER BY and LIMIT N without OFFSET)

Find the clue

150
111
102
101
100

Page One

Next

98
97
96
95
94

Prev

Page Two

Next

93
92
91
90
89

Prev

Page Three

Next

Solution using clue

Next Page:

`http://domain.com/forum?page=2&`**`last_seen=100`**`&dir=next`

```
WHERE id < 100 /* last_seen *
ORDER BY id DESC LIMIT $page_size /* No OFFSET*/
```

Prev Page:

`http://domain.com/forum?page=1&`**`last_seen=98`**`&dir=prev`

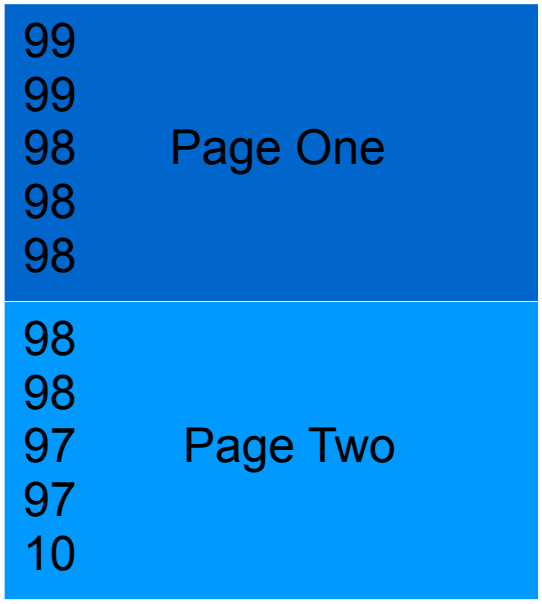
```
WHERE id > 98 /* last_seen *
ORDER BY id ASC LIMIT $page_size /* No OFFSET*/
```

Reverse given 10 rows before sending to user

Explain

```
mysql> explain
       SELECT * FROM message
       WHERE id < '49999961'
       ORDER BY id DESC LIMIT 20 \G
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: message
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         Rows: 25000020  /* ignore this */
        Extra: Using where
1 row in set (0.00 sec)
```

What about order by non unique values?




```
We can't do:
          WHERE thumbs_up < 98
          ORDER BY thumbs_up DESC /* It will return few seen rows */

Can we say this:
          WHERE  thumbs_up <= 98
          AND <extra_con>
          ORDER BY thumbs_up DESC
```

Add more condition

- Consider thumbs_up as major number
  - if we have additional minor number, we can use combination of major & minor as extra condition

- Find additional column (minor number)
  - we can use id primary key as minor number

Solution

First Page

```
SELECT thumbs_up, id
FROM message
ORDER BY thumbs_up DESC, id DESC
LIMIT $page_size
```

```
+-----------+----+
| thumbs_up | id |
+-----------+----+
|        99 | 14 |
|        99 |  2 |
|        98 | 18 |
|        98 | 15 |
|        98 | 13 |
+-----------+----+
```

Next Page

```
SELECT thumbs_up, id
FROM message
WHERE thumbs_up <= 98 AND (id < 13 OR thumbs_up < 98)
ORDER BY thumbs_up DESC, id DESC
LIMIT $page_size
```

```
+-----------+----+
| thumbs_up | id |
+-----------+----+
|        98 | 10 |
|        98 |  6 |
|        97 | 17 |
```

Make it better..

Query:

```
SELECT * FROM message
WHERE thumbs_up <= 98
      AND (id < 13 OR thumbs_up < 98)
ORDER BY thumbs_up DESC, id DESC
LIMIT 20
```

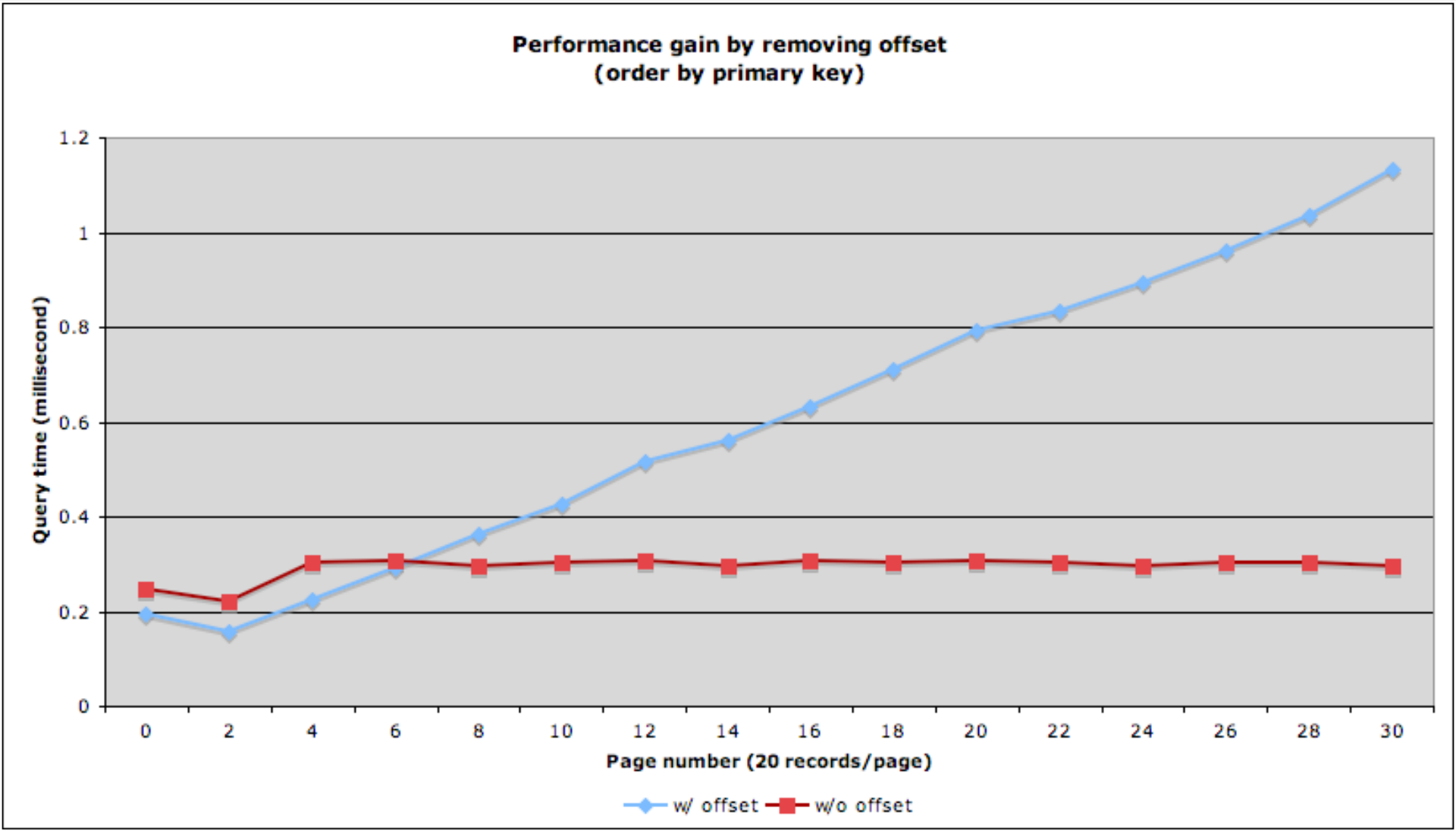
Can be written as:

```
SELECT m2.* FROM message m1, message m2
WHERE m1.id = m2.id
      AND m1.thumbs_up <= 98
      AND (m1.id < 13 OR m1.thumbs_up < 98)
ORDER BY m1.thumbs_up DESC, m1.id DESC
LIMIT 20;
```

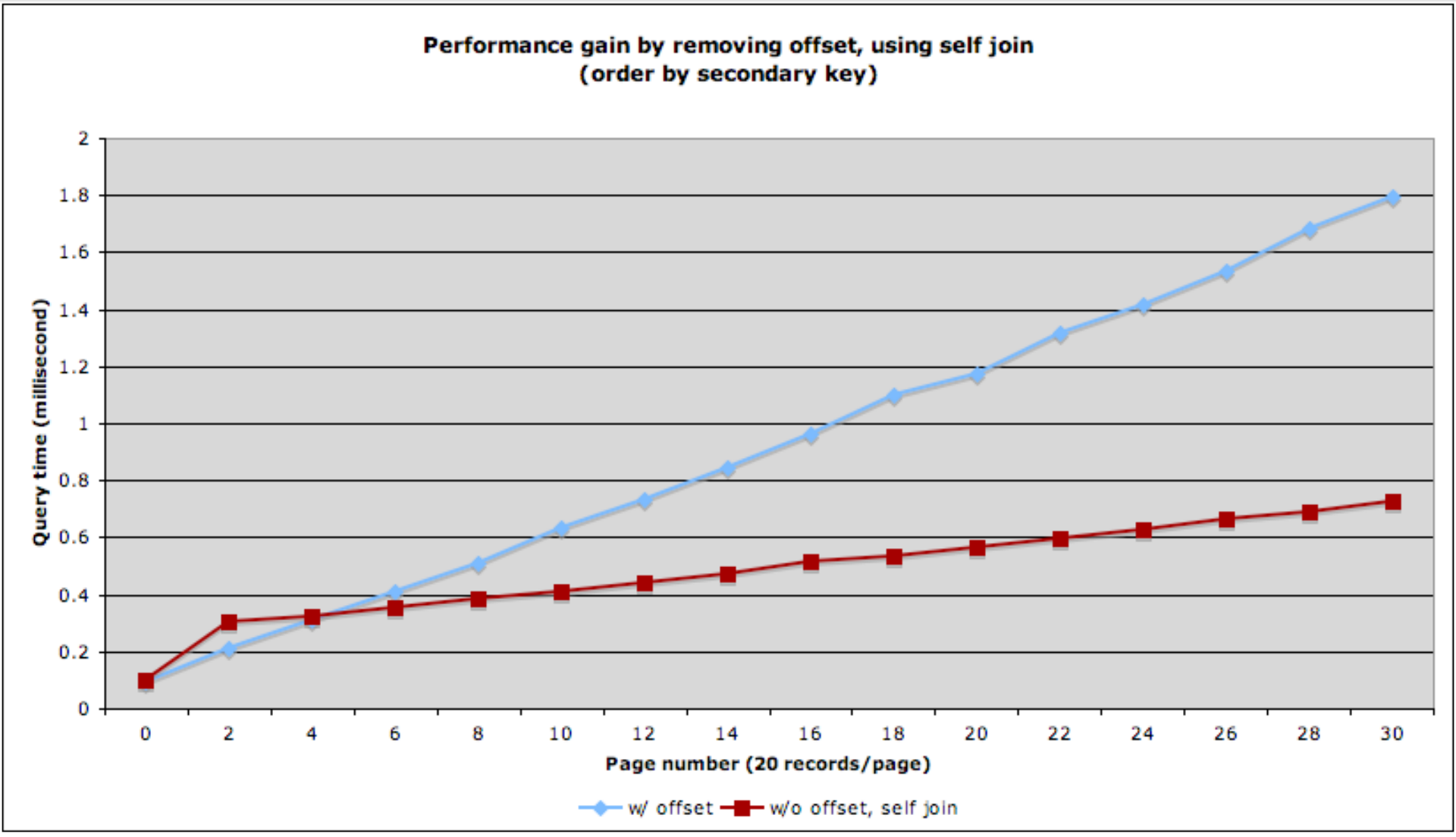
Explain

```
*************************** 1. row ***************************
           id: 1
  select_type: SIMPLE
        table: m1
         type: range
possible_keys: PRIMARY,thumbs_up_key
          key: thumbs_up_key  /* (thumbs_up,id) */
      key_len: 4
          ref: NULL
         Rows: 25000020 /*ignore this, we will read just 20 rows*/
        Extra: Using where; Using index /* Cover */
*************************** 2. row ***************************
           id: 1
  select_type: SIMPLE
        table: m2
         type: eq_ref
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: forum.m1.id
         rows: 1
        Extra:
```

Performance Gain (Primary Key Order)



Performance Gain (Secondary Key Order)



Throughput Gain

- Throughput Gain while hitting first 30 pages:
  - Using LIMIT OFFSET, N
    - 600 query/sec
  - Using LIMIT N (no OFFSET)
    - 3.7k query/sec

Bonus Point

Product issue with LIMIT M, N

User is reading a page, in the mean time some records may be added to previous page.

Due to insert/delete pages records are going to move forward/backward as rolling window:

- User is reading messages on 4th page
- While he was reading, one new message posted (it would be there on page one), all pages are going to move one message to next page.
- User Clicks on Page 5
- One message from page got pushed forward on page 5, user has to read it again

No such issue with news approach

Drawback

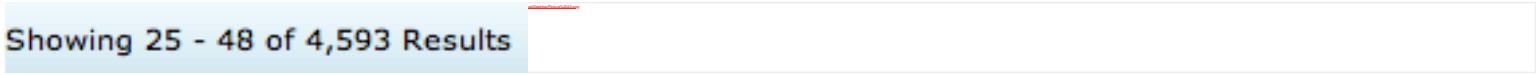
Search Engine Optimization Expert says:

Let bot reach all you pages with fewer number of deep dive

Two Solutions:

- Read extra rows
  - Read extra rows in advance and construct links for few previous & next pages

- Use small offset
  - Do not read extra rows in advance, just add links for few past & next pages with required offset & last_seen_id on current page
  - Do query using new approach with small offset to display desired page



Additional concern: Dynamic urls, last_seen is not constant over time.

Thanks