Indexes

Relational Databases Basics

Index

Read and remember!

Index – a specific kind of physical access path (an implementation construct, intended to improve the speed of access to data as physically stored).

Index for a database is like a map for a human. It helps finding objects of interest quickly and easy.

Indexes: what’s good and bad

Good	Bad
Indexes are rather small (enough to fit the RAM)	Many indexes take a lot of RAM
Index structure is optimized for search operations	Indexes are to be updated after data modification
Indexes may dramatically increase search speed	Index update process may take significant time

When to use indexes

Read operations on a table are performed much more often than modification

Indexed fields are often used in WHERE clause

Experiment shows that the presence of an index improves query performance

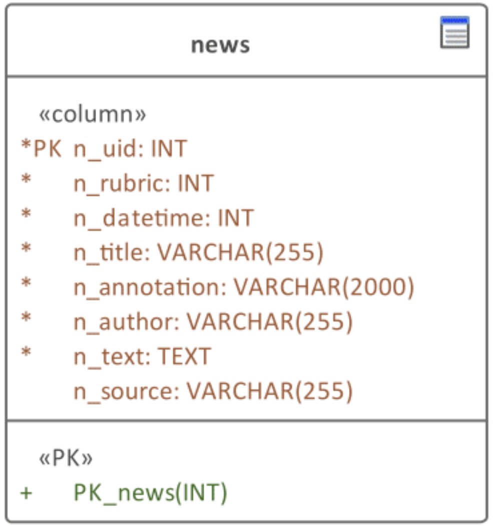
Index is used for field values uniqueness purpose

Indexed filed is a Foreign Key

…

Quick "demo"

Imagine, we have the following table (with no indexes except the Primary Key):



```
news

  «column»
*PK n_uid: INT
*     n_rubric: INT
*     n_datetime: INT
*     n_title: VARCHAR(255)
*     n_annotation: VARCHAR(2000)
*     n_author: VARCHAR(255)
*     n_text: TEXT
      n_source: VARCHAR(255)

  «PK»
+     PK_news(INT)
```

Quick “demo”

First attempt: let’s execute the following queries 1000 times each

Query	Avg time
INSERT {1000 records}	0.027289
SELECT * from \`news\` where \`n_rubric\`='…'	4.035899
SELECT * from \`news\` where \`n_rubric\`='…' AND \`n_dt\`>='…' AND \`n_dt\`<='…'	4.065648
SELECT * from \`news\` where \`n_dt\`>='…' AND \`n_dt\`<='…'	4.508579
SELECT * from \`news\` where \`n_title\`='…' AND \`n_dt\`>='…' AND \`n_dt\`<='…'	4.207702
SELECT * from \`news\` where \`n_title\`='…' AND \`n_author\`='…' AND \`n_dt\`>='…' AND \`n_dt\`<='…'	4.187432
SELECT * from \`news\` where \`n_title\`='…' AND \`n_author\`='…'	4.210264
SELECT * from \`news\` where \`n_title\`='…'	4.173025
SELECT * from \`news\` where \`n_author\`='…'	4.161251

Quick “demo”

Now let’s create the following indexes

n_rubric

n_rubric, n_dt

n_title, n_dt

n_dt, n_author

One of these indexes is redundant. Which is? And why?

Quick “demo”

Second attempt: let’s execute the following queries 1000 times each again

Query	Avg time	Avg time
INSERT {1000 records}	0.027289	0.896445
SELECT * from `news` where `n_rubric`='…'	4.035899	1.200757
SELECT * from `news` where `n_rubric`='…' AND `n_dt`>='…' AND `n_dt`<='…'	4.065648	0.207999
SELECT * from `news` where `n_dt`>='…' AND `n_dt`<='…'	4.508579	1.318613
SELECT * from `news` where `n_title`='…' AND `n_dt`>='…' AND `n_dt`<='…'	4.207702	0.003918
SELECT * from `news` where `n_title`='…' AND `n_author`='…' AND `n_dt`>='…' AND `n_dt`<='…'	4.187432	0.000468
SELECT * from `news` where `n_title`='…' AND `n_author`='…'	4.210264	0.000909
SELECT * from `news` where `n_title`='…'	4.173025	0.000279
SELECT * from `news` where `n_author`='…'	4.161251	4.567766

How to create an index?

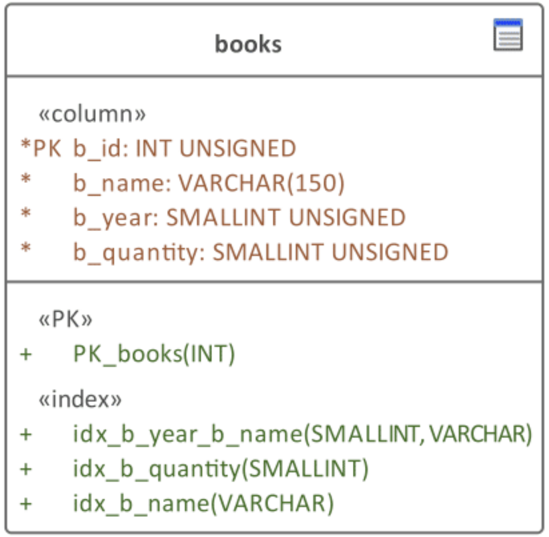
With an SQL query

```
CREATE TABLE `books`
(
 `b_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
 `b_name` VARCHAR(150) NOT NULL,
 `b_year` SMALLINT UNSIGNED NOT NULL,
 `b_quantity` SMALLINT UNSIGNED NOT NULL,
 CONSTRAINT `PK_books` PRIMARY KEY (`b_id`)
);

CREATE INDEX `idx_b_year_b_name`
ON `books` (`b_year`, `b_name`);

CREATE INDEX `idx_b_quantity`
ON `books` (`b_quantity`);

CREATE INDEX `idx_b_name`
ON `books` (`b_name`);
```

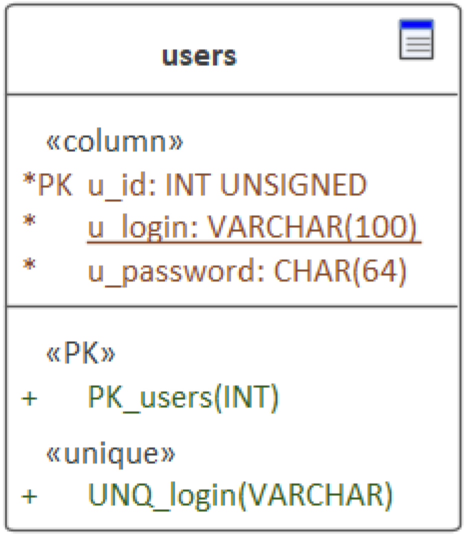

books
«column»
*PK b_id: INT UNSIGNED
* b_name: VARCHAR(150)
* b_year: SMALLINT UNSIGNED
* b_quantity: SMALLINT UNSIGNED
«PK»
+ PK_books(INT)
«index»
+ idx_b_year_b_name(SMALLINT, VARCHAR)
+ idx_b_quantity(SMALLINT)
+ idx_b_name(VARCHAR)

How to create a unique index?

With an SQL query

```
CREATE TABLE `users`
(
 `u_id` INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
 `u_login` VARCHAR(150) NOT NULL,
 `u_password` CHAR(64) NOT NULL,
 CONSTRAINT `PK_users` PRIMARY KEY (`u_id`)
);

ALTER TABLE `users`
 ADD CONSTRAINT `UNQ_login` UNIQUE (`u_login` ASC);
```



How to create an index?

Using some special tool

```
-- Here goes live demo :)
```

Live demo in Sparx Enterprise Architect

How to find out if an index is useful indeed

With an SQL query and/or special tool

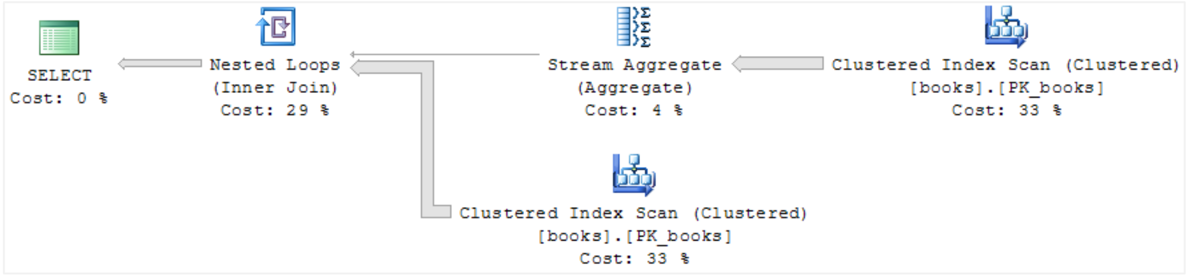
```
-- MySQL
EXPLAIN
SELECT *
FROM   `books`
WHERE  `b_quantity` = (SELECT MAX(`b_quantity`)
                       FROM   `books`);
```

id	select_type	table	type	possible_keys	key	key_len	ref	rows	Extra
1	PRIMARY	books	ALL	NULL	NULL	NULL	NULL	9730420	Using where
2	SUBQUERY	books	ALL	NULL	NULL	NULL	NULL	9730420	NULL

How to find out if an index is useful indeed

With an SQL query and/or special tool

```
-- MS SQL Server
SELECT *
FROM    [books]
WHERE   [b_quantity] = (SELECT MAX([b_quantity])
                        FROM    [books]);
```



How to find out if an index is useful indeed

With an SQL query and/or special tool

```
-- Oracle
EXPLAIN PLAN FOR
SELECT *
FROM   "books"
WHERE  "b_quantity" = (SELECT MAX("b_quantity")
                       FROM   "books");

SELECT PLAN_TABLE_OUTPUT
FROM TABLE(DBMS_XPLAN.DISPLAY());
```

```
-----------------------------------------------------------------------------------------------
| Id  | Operation              | Name                | Rows   | Bytes | Cost  (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT       |                     | 40590 |  1545K| 24367   (1)| 00:04:53 |
|*  1 |  INDEX FAST FULL SCAN  | SYS_IOT_TOP_101259 | 40590 |  1545K| 12193   (1)| 00:02:27 |
|   2 |   SORT AGGREGATE       |                     |     1 |     4 |             |          |
|   3 |    INDEX FAST FULL SCAN| SYS_IOT_TOP_101259 |  8118K|    30M| 12174   (1)| 00:02:27 |
-----------------------------------------------------------------------------------------------
Predicate Information (identified by operation id):
1 - filter(b_quantity= (SELECT MAX(b_quantity) FROM books books))
```

Indexes

Relational Databases Basics