

Database systems I

Radim Bača

radim.baca@vsb.cz

dbedu.cs.vsb.cz

VSB TECHNICAL
UNIVERSITY
OF OSTRAVA

FACULTY OF ELECTRICAL
ENGINEERING AND COMPUTER
SCIENCE

DEPARTMENT
OF COMPUTER
SCIENCE



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání



MINISTERSTVO ŠKOLSTVÍ,
MLÁDEŽE A TĚLOVÝCHOVY

Content

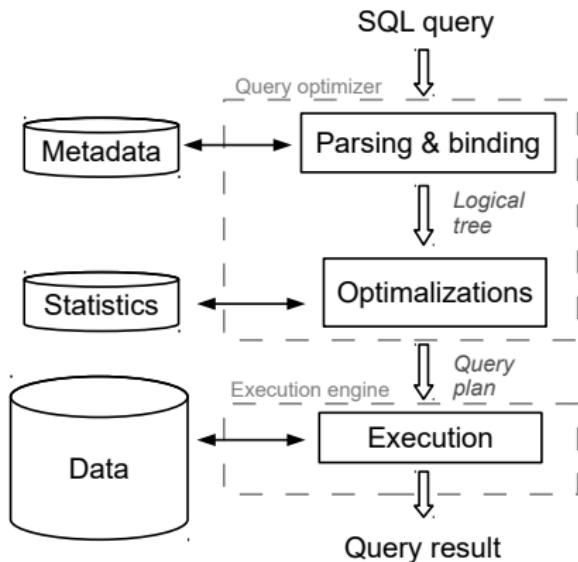
- Introduction
- What is an index?
- What is an appropriate index?

Introduction

- Until now we were mostly working on a conceptual level of a database system:
 - Relational data model
 - SQL
 - Normal forms
- Today we will look closer under the hood of physical representation of the data

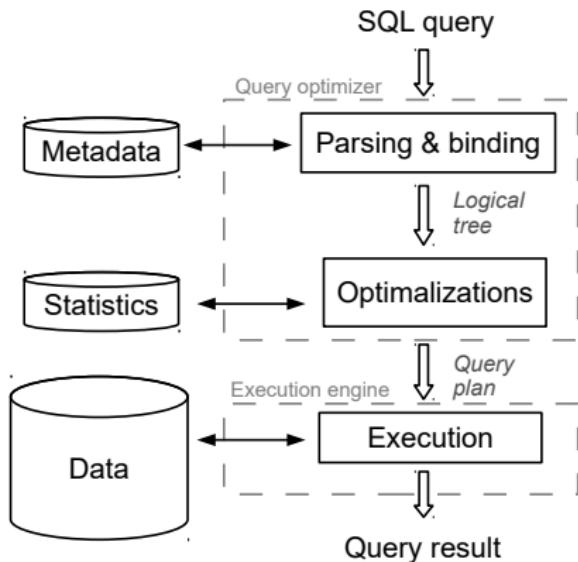


SQL Query Compilation - Remainder



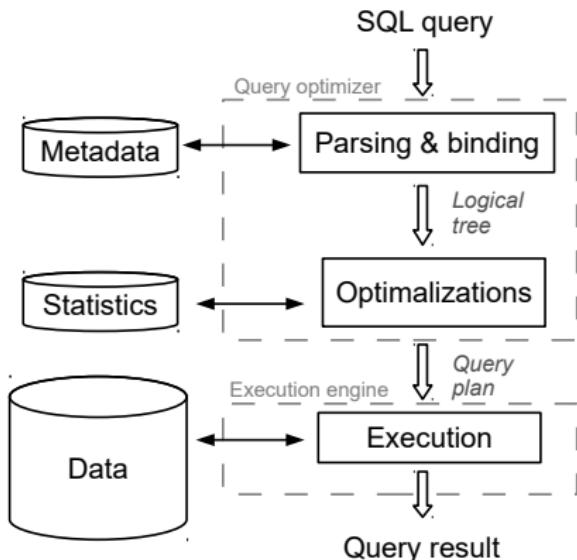
- SQL query optimization is a process which takes a SQL and output an query plan
- The process of plan creation should be deterministic
- If there are appropriate index then the optimizer uses it

SQL Query Compilation - Remainder



- SQL query optimization is a process which takes a SQL and output an query plan
- The process of plan creation should be deterministic
- If there are appropriate index then the optimizer uses it

SQL Query Compilation - Remainder

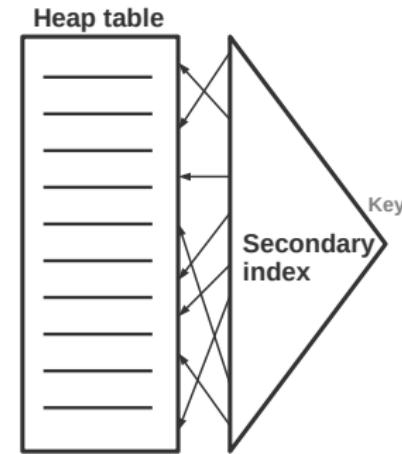


- SQL query optimization is a process which takes a SQL and output an query plan
- The process of plan creation should be deterministic
- If there are appropriate index then the optimizer uses it

But what is an index and what is an appropriate index?

What is an Index?

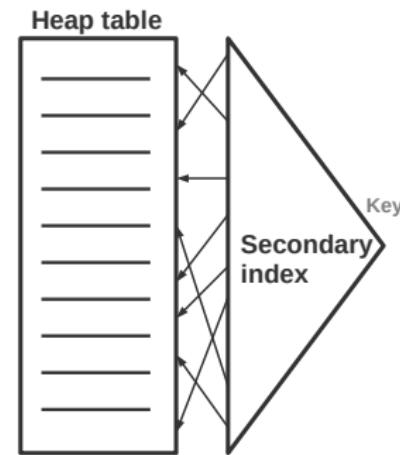
- Index allows fast search of a **value** according to a **key**
- It is data structure in a database system that can be created or dropped (see Lecture 6)
- The most typical data structure used in all database systems is a B+tree



Why B+tree?

It has a following properties

- Search of an value according to a key in $\log(n)$ time
- Reasonably fast of writes into the index (insert/update/delete)
- It support range scan in leaf nodes
- Data are organized into pages

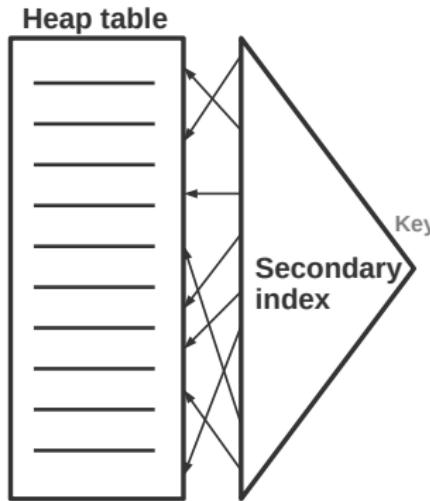


Other Data Structures

- We may use other data structures if it is appropriate:
 - R*tree - fast spatial search
 - Hash table - fast single record lookup
 - Skip list - lock free data structure
- It depends on availability in a database system
- However, `CREATE INDEX` creates a B+tree in most database systems

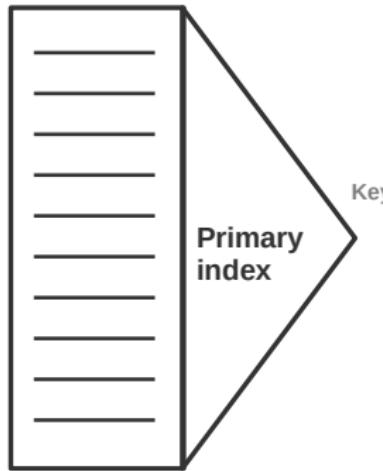
Types of Indexes

- Secondary index - value is a pointer to a table



Types of Indexes

- Primary index - value is a record of a table
- Database can contains just one such index



Primary Index

- It has different names:
 - Clustered index (SQL Server, MySQL)
 - Index organized table (Oracle)
- It is not available in all database systems (PostgreSQL)

When we Need an Index?

- Do we need an index for a table where only insert data and perform a query

SELECT * FROM table?

- Definitely not
- Index helps us to process the predicates of a query more quickly
- The important observation:

The index is always connected to a SQL query (or a set of queries)

When we Need an Index?

- Do we need an index for a table where only insert data and perform a query
SELECT * FROM table?
- Definitely not
- Index helps us to process the predicates of a query more quickly
- The important observation:

The index is always connected to a SQL query (or a set of queries)

How to Create an Index

```
SELECT * FROM table WHERE atribut = hodnota
```



```
CREATE INDEX ix_table_atr ON table(atribut)
```

How to Create an Index

```
SELECT * FROM table WHERE atribut = hodnota
```



```
CREATE INDEX ix_table_atr ON table(atribut)
```

- We create indexes according to the predicate in a query

Example - SQL Server

- `SELECT * FROM Actor
WHERE first_name = 'johnny'`



Search using a sequential scan through the actor table

Example - SQL Server

- `SELECT * FROM Actor
WHERE first_name = 'johnny'`
- Now lets create an index ?
- `CREATE INDEX
ix_actor_firstname
ON Actor(first_name)` →

Example - SQL Server

- `SELECT * FROM Actor
WHERE first_name = 'johnny'`
- Now lets create an index
- `CREATE INDEX
ix_actor_firstname
ON Actor(first_name)`



The index is not used



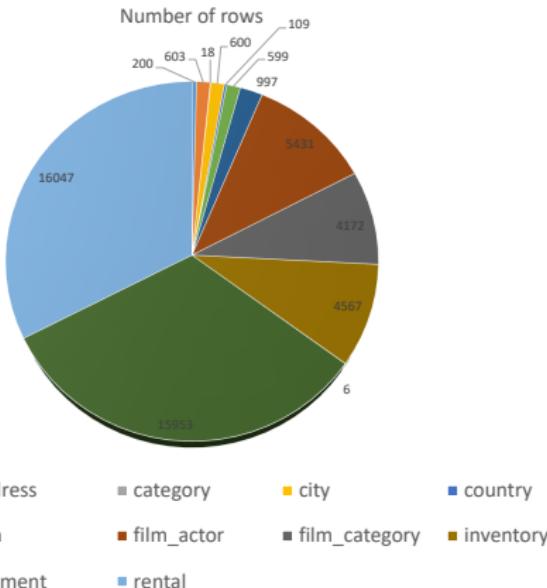
Lesson 1:

Always check the query plan after an index creation to see whether the index is used or not

Index Usage

- We need to inspect the query plan (if possible)
- There are two major types of query plan representations:
 - Graphical
 - Text
- Each database system has its own proprietary way how to display a query plan
- The query plan should contain the newly created index name

Table Statistics of Sakila DB



Lesson 2:

Do not bother with indexing of small tables

SARGable Predicate

- ```
SELECT * FROM rental
WHERE YEAR(rental_date) = 2005
 and MONTH(rendal_date) = 10
```
- ```
CREATE INDEX ix_uq ON
rental(rental_date, inventory_id, customer_id)
```

SARGable Predicate

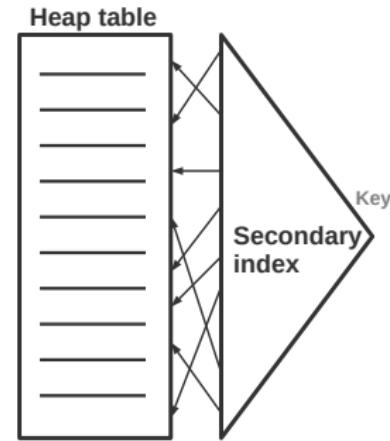
- ```
SELECT * FROM rental
WHERE YEAR(rental_date) = 2005
 and MONTH(rendal_date) = 10
```
- ```
CREATE INDEX ix_uq ON
rental(rental_date, inventory_id, customer_id)
```
- The table is scanned despite the index
- Database systems requires no attribute manipulation in the predicate in order to use an index
- We need to rewrite the query

Lesson 3:

Do not manipulate the attributes in predicates if possible

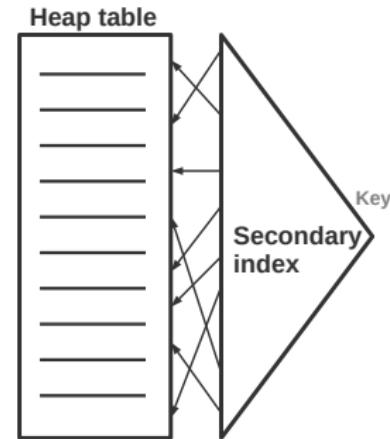
Asterix

- ```
SELECT * FROM rental
WHERE YEAR(rental_date) = 2005
 and MONTH(rendal_date) = 10
```
- We may observe that the query plan consists from two data accesses:
  - Secondary index seek
  - Heap table access
- What if we are interested just in few attributes
- ```
SELECT rental_date, customer_id
FROM ...
```



Asterix

- ```
SELECT * FROM rental
WHERE YEAR(rental_date) = 2005
 and MONTH(rendal_date) = 10
```
- We may observe that the query plan consists from two data accesses:
  - Secondary index seek
  - Heap table access
- What if we are interested just in few attributes
- ```
SELECT rental_date, customer_id
FROM ...
```



Lesson 4:

Do not use an asterix in the production

Covering Indexes

- We avoid table access completely in the previous example
- We use only data in `idx_uq`
- We say that such index *cover* the query
- Having an index with more attributes you are more likely to witness the index to be used during a scan (not during a seek)
- ```
SELECT rental_date, inventory_id, customer_id
FROM rental WHERE inventory_id < 100
```

## Covering Indexes

- We avoid table access completely in the previous example
- We use only data in `idx_uq`
- We say that such index *cover* the query
- Having an index with more attributes you are more likely to witness the index to be used during a scan (not during a seek)
- ```
SELECT rental_date, inventory_id, customer_id
FROM rental WHERE inventory_id < 100
```

Index Scan vs. Index Seek

```
SELECT rental_date, inventory_id, customer_id
FROM rental
WHERE rental_date >= '2005-10-01'
    and rental_date <= '2005-10-31'
```

rental_date	inventory_id	customer_id
2005-01-02	1	1
2005-01-02	1	50
2005-01-02	200	1
2005-10-02	1	1
2005-10-02	200	1
2005-11-05	99	3
2005-11-05	100	2
2005-11-06	1	1

```
SELECT rental_date, inventory_id, customer_id
FROM rental
WHERE inventory_id < 100
```

rental_date	inventory_id	customer_id
2005-01-02	1	1
2005-01-02	1	50
2005-01-02	200	1
2005-10-02	1	1
2005-10-02	200	1
2005-11-05	99	3
2005-11-05	100	2
2005-11-06	1	1

Index Scan vs. Index Seek

```
SELECT rental_date, inventory_id, customer_id
FROM rental
WHERE rental_date >= '2005-10-01'
    and rental_date <= '2005-10-31'
```

rental_date	inventory_id	customer_id
2005-01-02	1	1
2005-01-02	1	50
2005-01-02	200	1
2005-10-02	1	1
2005-10-02	200	1
2005-11-05	99	3
2005-11-05	100	2
2005-11-06	1	1

```
SELECT rental_date, inventory_id, customer_id
FROM rental
WHERE inventory_id < 100
```

rental_date	inventory_id	customer_id
2005-01-02	1	1
2005-01-02	1	50
2005-01-02	200	1
2005-10-02	1	1
2005-10-02	200	1
2005-11-05	99	3
2005-11-05	100	2
2005-11-06	1	1

Insert/Update/Delete and Indexes

- Now lets try an INSERT into rental table:

```
INSERT INTO rental  
VALUES ('2005-05-26 22:50:00', 367, 130,  
       '2005-05-26 22:04:31', 1,  
       '2006-02-15 21:30:50')
```

Insert/Update/Delete and Indexes

- Now lets try an INSERT into rental table:

```
INSERT INTO rental  
VALUES ('2005-05-26 22:50:00', 367, 130,  
      '2005-05-26 22:04:31', 1,  
      '2006-02-15 21:30:50')
```

- The database ensures referential integrity during the write operations
- Due to this we should never miss:
 - Indexes on primary keys
 - Indexes on foreign keys

Lesson 5:

Indexes on PK and FK are important to INSERT, UPDATE and DELETE operations (therefore, not only to SELECT)

Intelligent Physical Design

- Sometimes it may take a time to come up with intelligent physical design that:
 - supports critical queries
 - do not significantly slow down the insert/update/deletes

Falkirk Wheel



Falkirk Wheel



- It lift 550 tons with 22.5 kW

References

- Course home pages <http://dbedu.cs.vsb.cz>