

DB indeksavimas ir SQL įterptinės užklausos

*INDEXING,
INJECTION*

Indeksavimas

Indeksai

Padidina užklausų našumą (select,update,delete);

Indeksai yra kuriami stulpeliams;

Indeksai kuriami pirminiams raktams arba esant unikalumo apribojimams;

Indeksų tipai

Grupiniai indeksai – duomenys yra saugomi surūšiuoti pagal indeksą;

Paprastas indeksas – suformuojamas surūšiuotų nuorodų sąrašas, pagal kuri greitai randami reikiami duomenys;

Unikalūs indeksas;

Full text indeksas - pilnos tekstinės paieškos indeksai;

Erdvinių duomenų indeksai;

XML indeksas – paieškai xml duomenų tipo laukuose.

Indeksų tipai

```
CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX  
index_name [index_type]  
ON tbl_name (index_col_name,...)  
[index_col_name: col_name [(length)] [ASC | DESC]
```

MYSQL db variklis

InnoDB

MyISAM

MEMORY

NDB

Indekso architektūra

BTREE

BTREE

HASH, BTREE

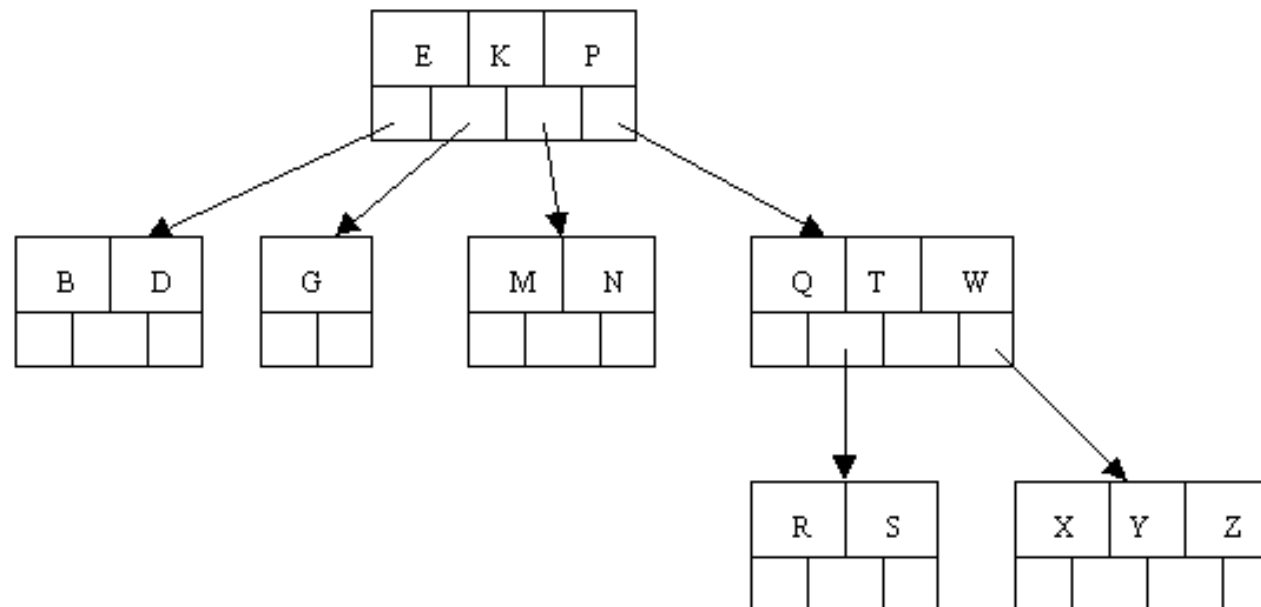
HASH, BTREE

Indeksų tipai MySQL atveju

Storage Engine	Index Type	Index Class	Stores NULL Values	Permits Multiple NULL Values	IS NULL Scan Type	IS NOT NULL Scan Type
InnoDB	BTREE	Primary key	No	No	N/A	N/A
		Unique	Yes	Yes	Index	Index
		Key	Yes	Yes	Index	Index
	Inapplicable	FULLTEXT	Yes	Yes	Table	Table
		SPATIAL	No	No	N/A	N/A
MyISAM	BTREE	Primary key	No	No	N/A	N/A
		Unique	Yes	Yes	Index	Index
		Key	Yes	Yes	Index	Index
	Inapplicable	FULLTEXT	Yes	Yes	Table	Table
	Inapplicable	SPATIAL	No	No	N/A	N/A
MEMORY	HASH	Primary key	No	No	N/A	N/A
		Unique	Yes	Yes	Index	Index
		Key	Yes	Yes	Index	Index
	BTREE	Primary	No	No	N/A	N/A
		Unique	Yes	Yes	Index	Index
		Key	Yes	Yes	Index	Index
NDB	BTREE	Primary key	No	No	Index	Index
		Unique	Yes	Yes	Index	Index
		Key	Yes	Yes	Index	Index
	HASH	Primary	No	No		
		Unique	Yes	Yes		
		Key	Yes	Yes		

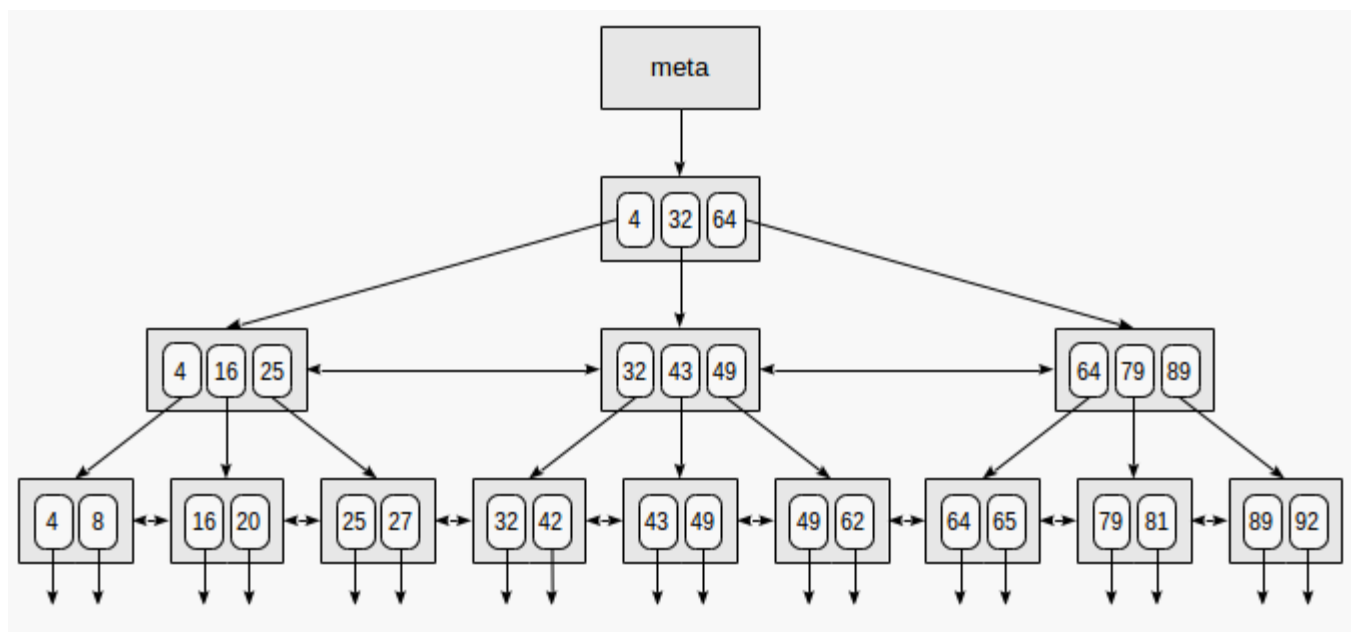
B-tree

Indeksų struktūra dažniausia saugoma, kaip binarinis medis (B-tree);



B-tree

Indeksų struktūra dažniausia saugoma, kaip binarinis medis (B-tree);



B-tree

Kodėl naudoti binarinį medį?

- Binariniai medžiai lengvai subalansuojami, todėl visos reikšmės surandamos per tą patį laiką;
- Binariniai medžiai yra daugiašakiai ir kiekviena šaka gali saugoti šimtus nuorodų į duomenų bazės lentelių įrašus, todėl net ir labai didelių lentelių B-tree struktūra gali būti 4-5 lygių;
- Binarinių medžių lapai tarpusavyje gali turėti abipusius ryšius (pvz. **PostgreSQL**), todėl norint skaityti nuosekliai surūšiuotą sąrašą nereikia keliauti per medžio hierarchiją;

Užklausų optimizatorius

Užklausų optimizatorius (query optimizer) – jis įvertimas visus galimus užklausos vykdymo variantus bandydamas aptikti tinkamiausią.

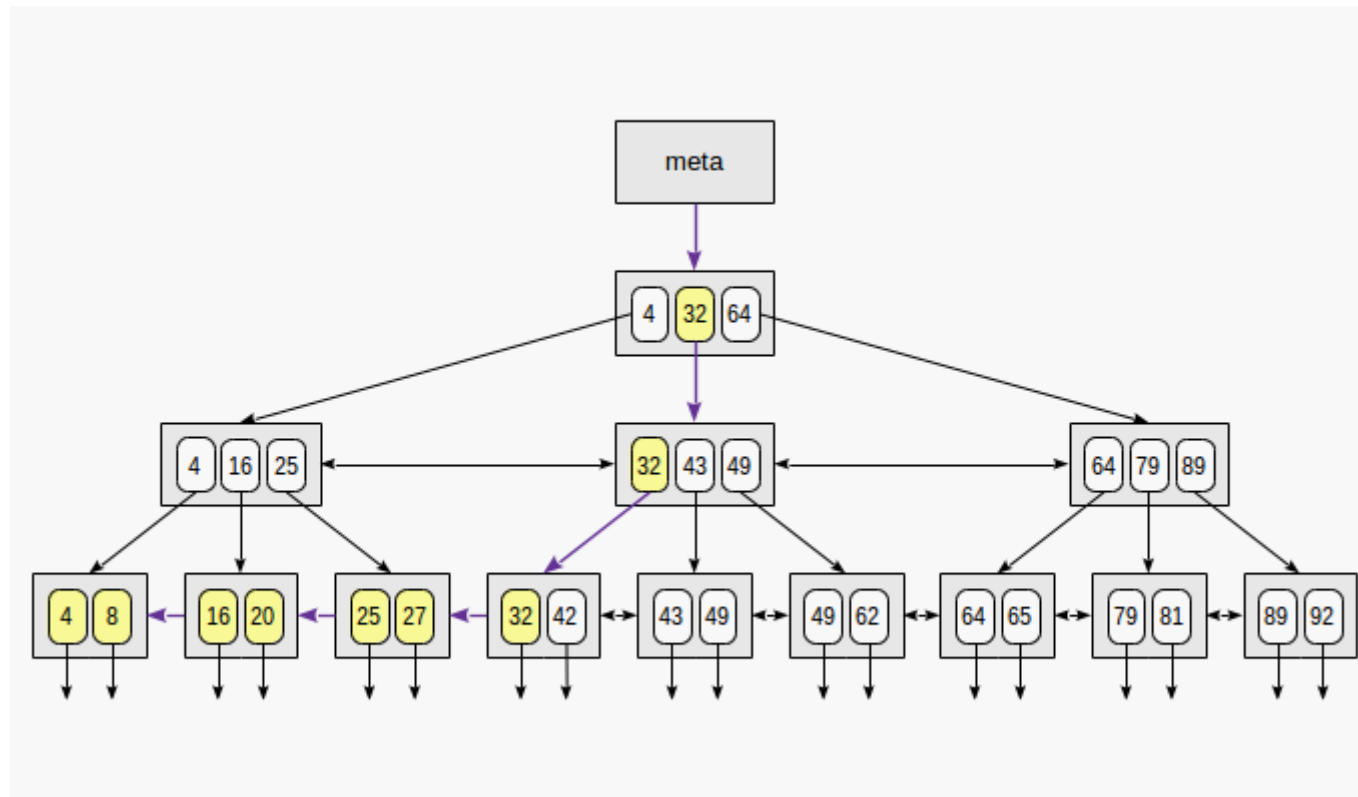
Pagrindinis veikimo principas:

1. Pirmiausia jis įvertina atrenkamų duomenų struktūrą;
2. Analizuoja ar atrenkami duomenys turi indeksus;
3. Įvertina tinkamiausius indeksus ir sudaro užklausos vykdymo planą;
4. Jei nustatoma, kad užklausa gražina ženklią dalį duomenų iš galimo aibės, tai tam tikroms lentelėms užklausų optimizatorius gali ir nepanaudoti indekso.

11

Paieška B-tree medžiuose

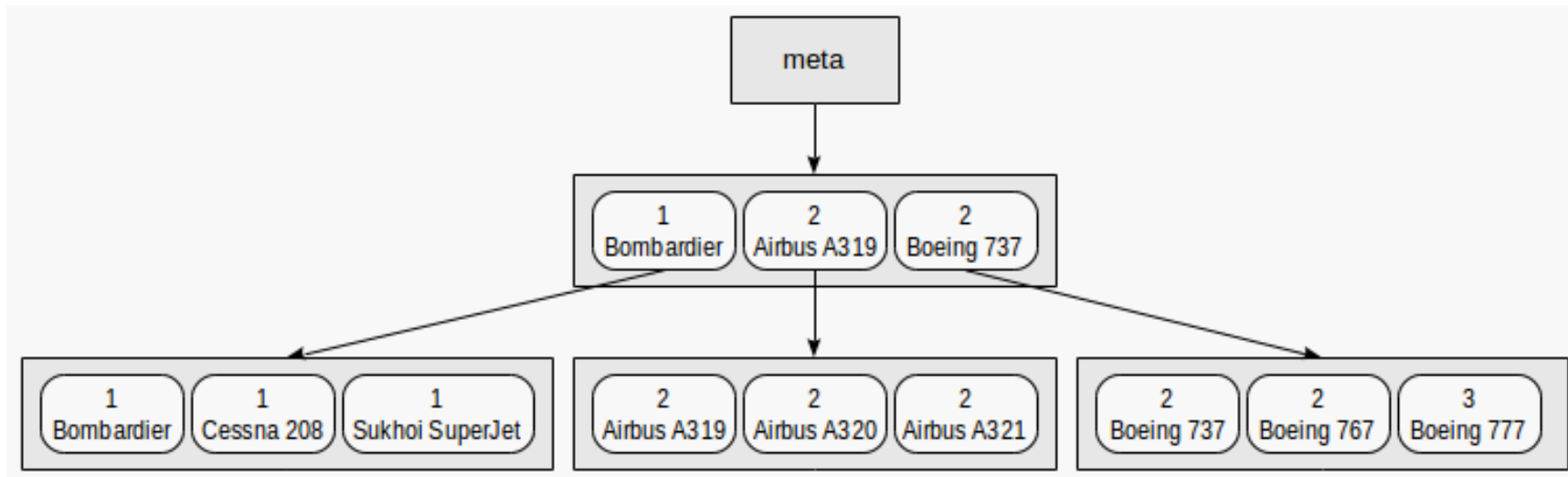
Reikia surasti reikšmę $n \leq 32$



Keleto stulpelių indeksai

Lentelė sauganti informacija apie lėktuvų skrydžio nuotolį pagal klases ir lėktuvo modelį

Binariniame medyje, aktuali pasirinktų laukų seka, nes indeksas bus kuriamas ir rūšiuojamas pagal pirmą lauką pirmiausia, o paskui pagal sekančius laukus;



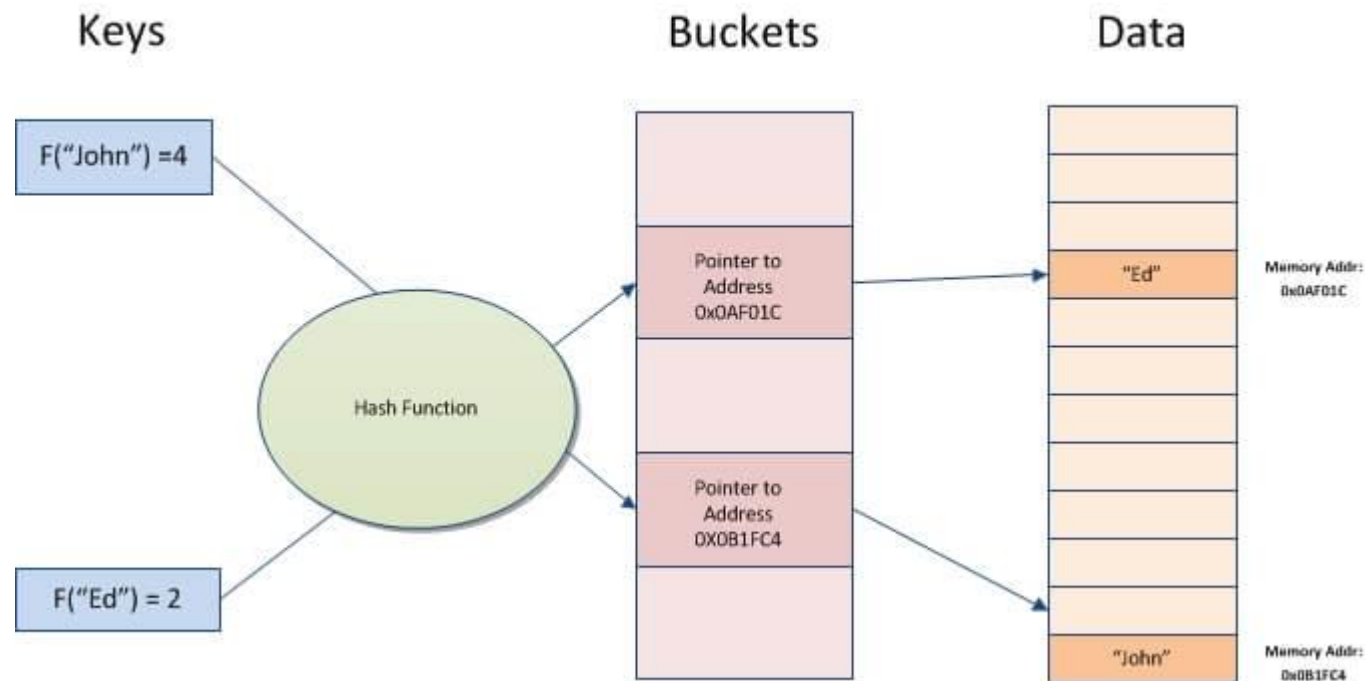
PASTABA: toks indeksas bus efektyvus, jei paiešką atliksite pagal pirmą lauką arba visą aibę, tačiau visiškai nefunkcionalus, jei paieška bus vykdoma tik pagal lėktuvo modelį.

Maišos (angl. HASH) indeksai

HASH indeksas yra KEY/VALUE reikšmių porų lentelė.

Maišos funkcija yra bet koks algoritmas, kuris deterministiniu ir beveik atsitiktiniu būdu susieja kintamo ilgio duomenis su fiksuoto ilgio duomenimis.

PVZ.: teksto eilutės ilgis.



HASH indeksai

Kodėl naudoti HASH indeksą?

- HASH indeksai puikiai veikia, kai reikia atlikti $n =$ arba $n \neq$ operaciją;
- HASH indeksai netinkami intervalų paieškai $n >$, $n \leq$, $n <$ ir t.t.
- HASH indeksai visiškai nepasitarnauja rūšiavimo operacijai (ORDER BY)
- HASH indeksas netinkamas dalinės duomenų eilutės paieškai naudojant simbolių pakaitas (angl. wildcard), nes HASH kodas sugeneruotas visai eilutei.

```
SELECT * FROM studentai WHERE vardas LIKE 'Pat%k%';
```


Grupiniai ir paprasti indeksai

Paprasti indeksai naudojami:

- kai užklauso gražina nedideles duomenų aibes;
- kai naudojamos tikslios užklaustos (where dalyje nurodomos konkrečios duomenų reikšmės);
- kai indeksas apima keletą ir daugiau stulpelių, pagal kuriuos atliekamas duomenų filtravimas;
- abu indeksų tipai tinkamai, kai stulpelių duomenys pasižymi dideliu unikalumu;
- paprastų indeksų naudojimas patartinas, išorinių raktų stulpeliams, kai yra santykis *vienas:vienu*;

Abu indeksų tipai vienodai gerai veikia ir reikalingi *group by* dalies stulpeliams;

Grupiniai (angl. Clustered) indeksai naudojami:

- kai užklauso gražina dideles duomenų aibes;
- kai duomenų filtravimo kriterijai užduodami diapazonais, aibėmis, apytikslėmis reikšmėmis operatorius (>,<, like ir .t.t);
- grupinių indeksų naudoti rekomenduojama, išorinių raktų stulpeliams, kai yra santykis *vienas:daug*;
- indeksas pasitarnauja, jei naudojama užklausoje *order by*;

Bendrosios rekomendacijos

Nepersistenkite su indeksais – nes duomenų įterpimo metu indeksai turi būtų pertvarkomi;

Indeksuokite laisvai lenteles, kuriuose retai atliekamas duomenų modifikavimas (insert, update, delete);

Jei vyksta didelis ir nuolatinis duomenų įterpimo ir modifikavimo srautas, tokiu atveju **indeksus reikia kurti tik būtiniams atvejams**, o norint atlikti nuolatinę ir intensyvią duomenų paiešką ir gavybą, gal reiktų pasirengti net atskirą DB egzempliorių ir joje sukurti, jau reikiamus indeksus.

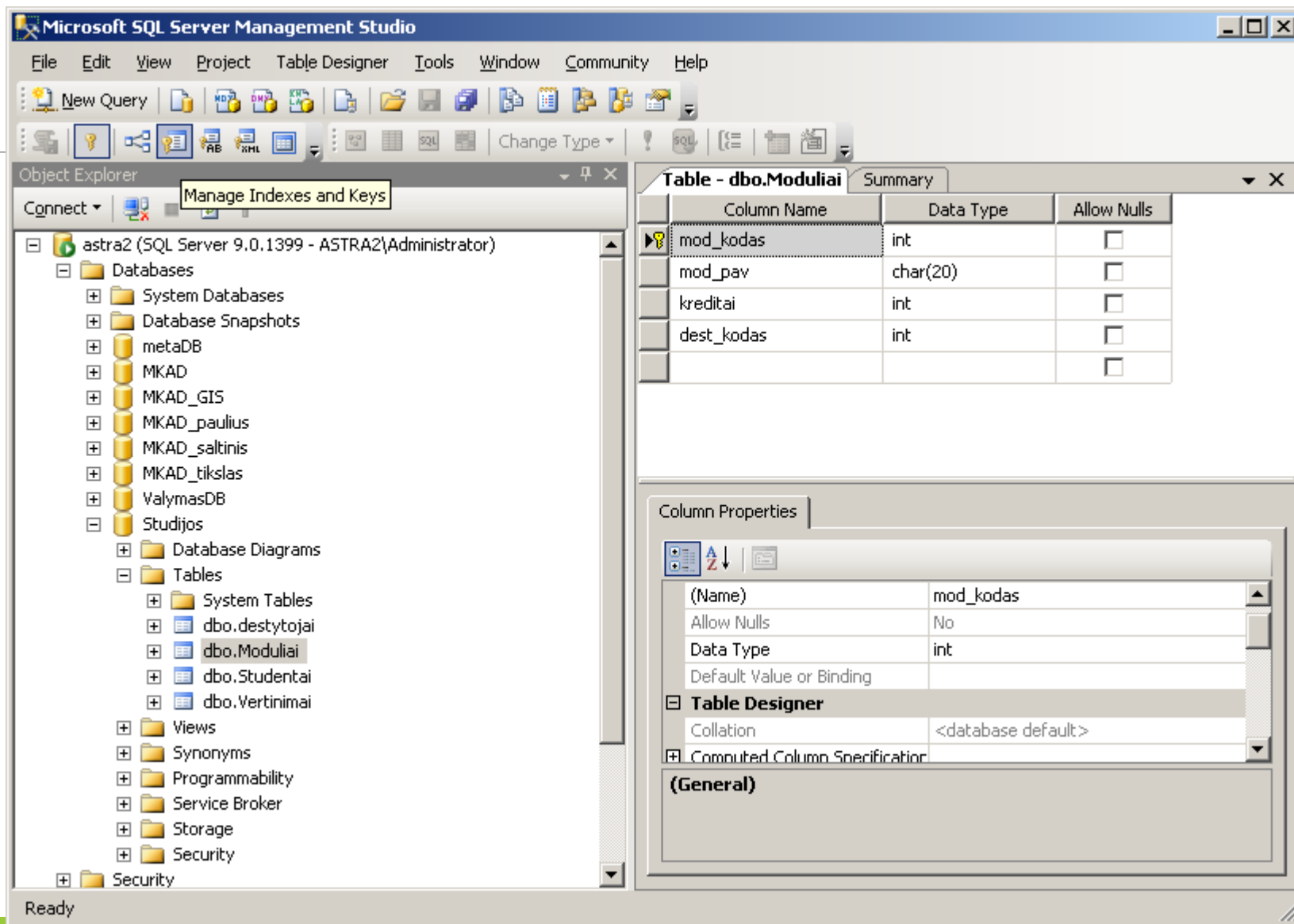
Neindeksuokite mažų lentelių, kai duomenų kiekiai yra labai maži ir lentelės duomenų sudėtis yra pakankamai statiška.

Indekso kūrimas

```
CREATE [ UNIQUE ] [ CLUSTERED |  
NONCLUSTERED ] INDEX index_name ON  
<object> ( column [ ASC | DESC ] [ ,...n ] )
```

```
DROP INDEX index_name
```

Indeksų kūrimas



Microsoft SQL Server Management Studio

File Edit View Project Table Designer Tools Window Community Help

Object Explorer

Connect Manage Indexes and Keys

astras2 (SQL Server 9.0.1399 - ASTRA2\Administrator)

- Databases
 - System Databases
 - Database Snapshots
 - metaDB
 - MKAD
 - MKAD_GIS
 - MKAD_paulius
 - MKAD_saltinis
 - MKAD_tikslas
 - ValymasDB
- Studijos
 - Database Diagrams
 - Tables
 - System Tables
 - dbo.destytojai
 - dbo.Moduliai
 - dbo.Studentai
 - dbo.Vertinimai
 - Views
 - Synonyms
 - Programmability
 - Service Broker
 - Storage
 - Security

Table - dbo.Moduliai

Column Name	Data Type	Allow Nulls
mod_kodas	int	<input type="checkbox"/>
mod_pav	char(20)	<input type="checkbox"/>
kreditai	int	<input type="checkbox"/>
dest_kodas	int	<input type="checkbox"/>

Column Properties

(Name)	mod_kodas
Allow Nulls	No
Data Type	int
Default Value or Binding	

Table Designer

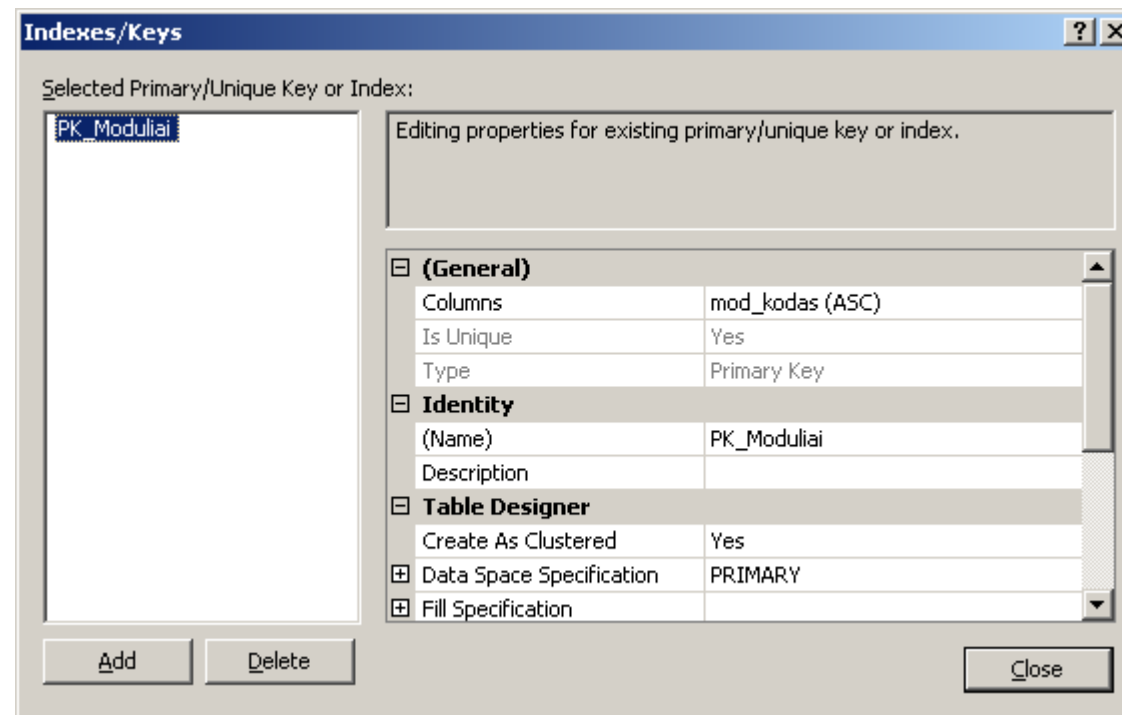
Collation	<database default>
-----------	--------------------

Computed Column Specification

(General)

Ready

Indeksų kūrimas



Indeksų kūrimas

The screenshot shows the Microsoft SQL Server Management Studio interface. The Object Explorer on the left displays the database structure for 'astra2 (SQL Server 9.0.1399 - ASTRA2\Administrator)'. The 'Tables' folder is expanded, showing 'dbo.Moduliai'. The 'Indexes' folder is also expanded, showing 'PK_Moduliai (Clustered)'. The 'Table - dbo.Moduliai' window is open, displaying the table's columns: 'mod_kodas' (int), 'mod_pav' (char(20)), 'kreditai' (int), and 'dest_kodas' (int). The 'Column Properties' window is open for the 'mod_kodas' column, showing its properties: (Name) mod_kodas, Allow Nulls No, Data Type int, and Default Value or Binding. The 'Table Designer' tab is also visible, showing the table's structure.

Table - dbo.Moduliai

Column Name	Data Type	Allow Nulls
mod_kodas	int	<input type="checkbox"/>
mod_pav	char(20)	<input type="checkbox"/>
kreditai	int	<input type="checkbox"/>
dest_kodas	int	<input type="checkbox"/>

Column Properties

(Name)	mod_kodas
Allow Nulls	No
Data Type	int
Default Value or Binding	

Table Designer

Collation	<database default>
Computed Column Specification	

Index Properties - PK_Moduliai

Select a page

- General
- Options
- Included Columns
- Storage
- Fragmentation
- Extended Properties

Script Help

Table name: Moduliai

Index name: PK_Moduliai

Index type: Clustered

☒ Unique

Index key columns:

Name	Sort Order	Data Type	Size
mod_kodas	Ascending	int	4

Add... Remove Move Up Move Down

Connection

Server: astra2

Connection: ASTRA2\Administrator

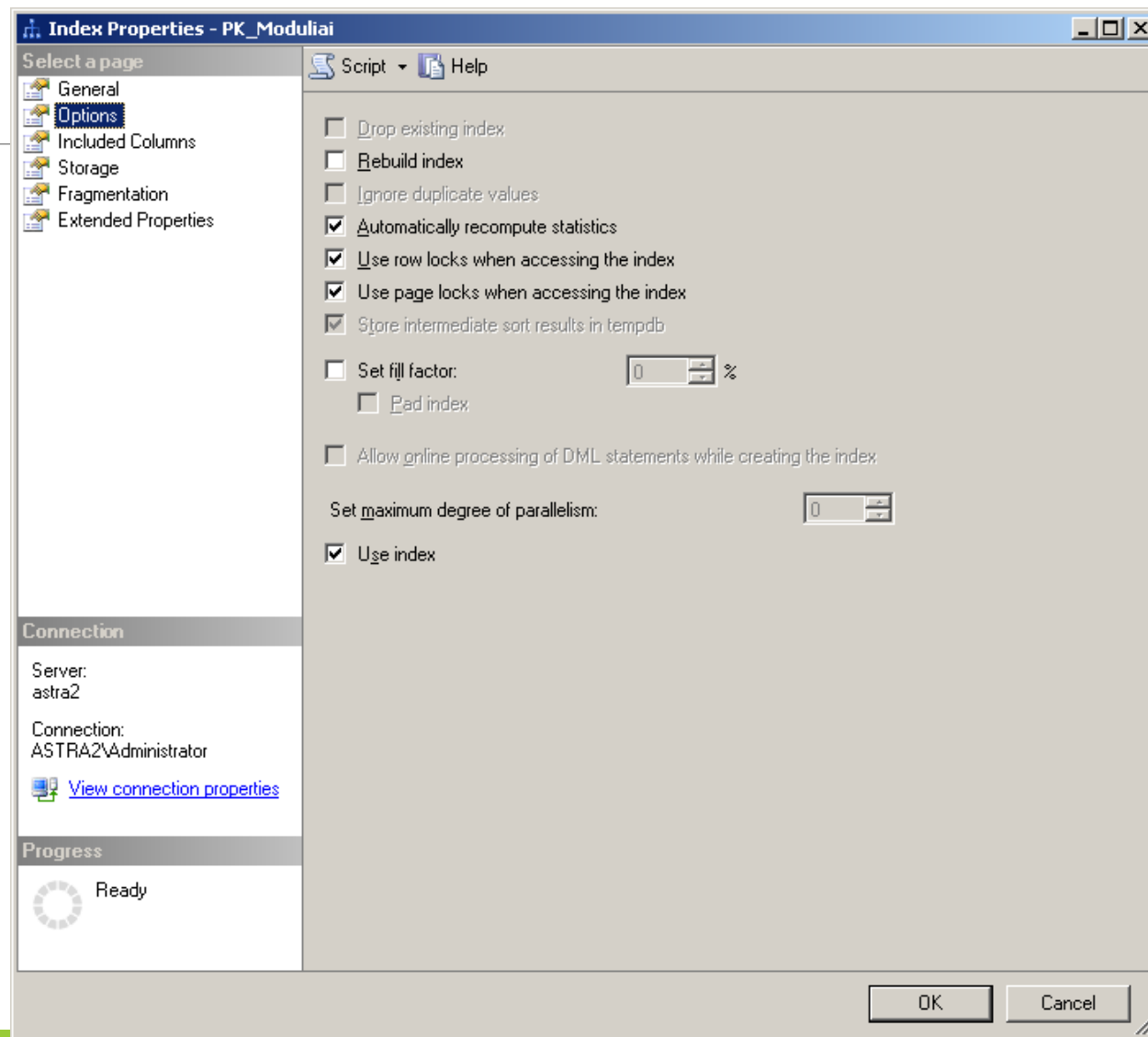
[View connection properties](#)

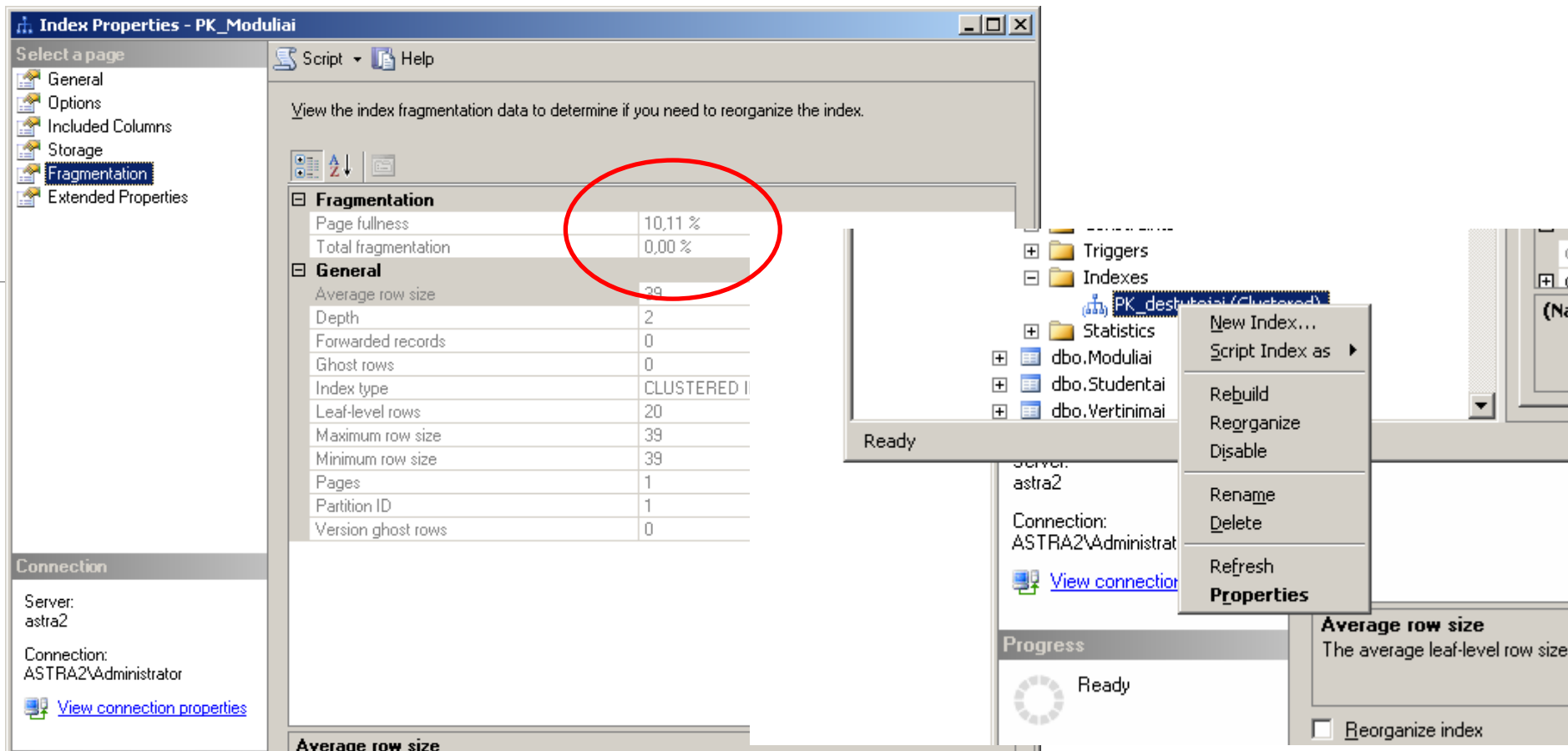
Progress

Ready

OK Cancel

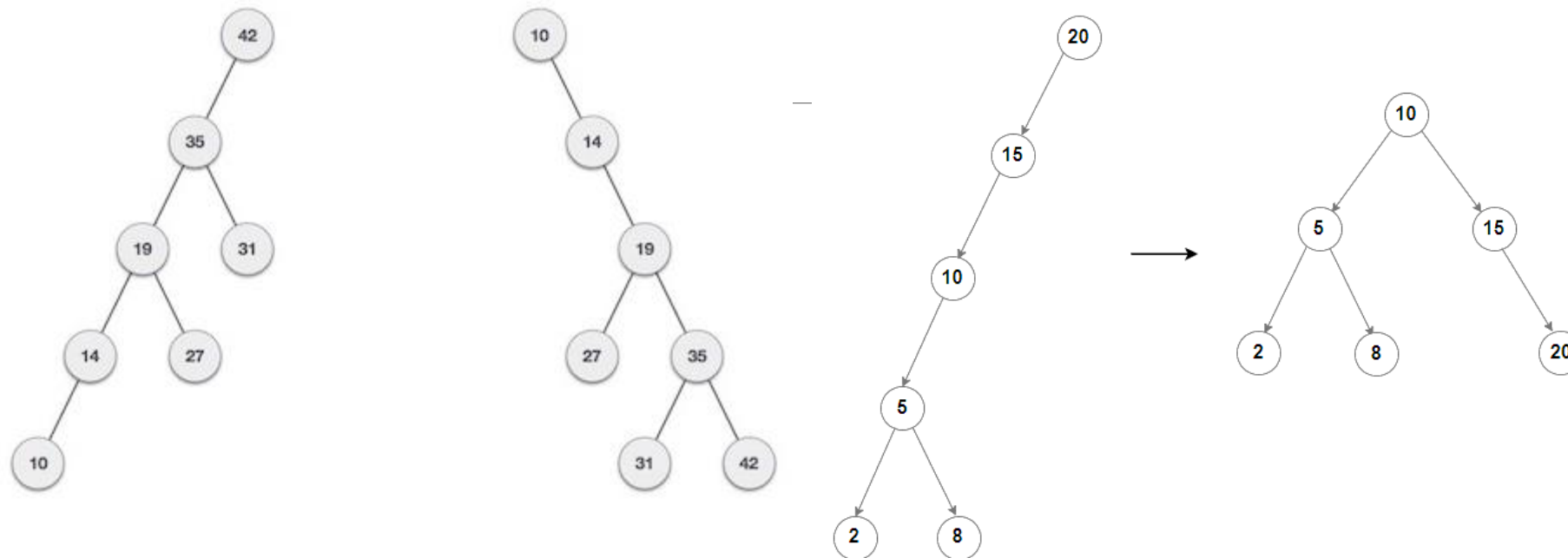
Indeksų kūrimas





Total fragmentation	Kaip pertvarkyti indeksą
> 5% and < = 30%	ALTER INDEX REORGANIZE
> 30%	ALTER INDEX REBUILD WITH (ONLINE = ON)*

Nesubalansuoti b-tree medžiai



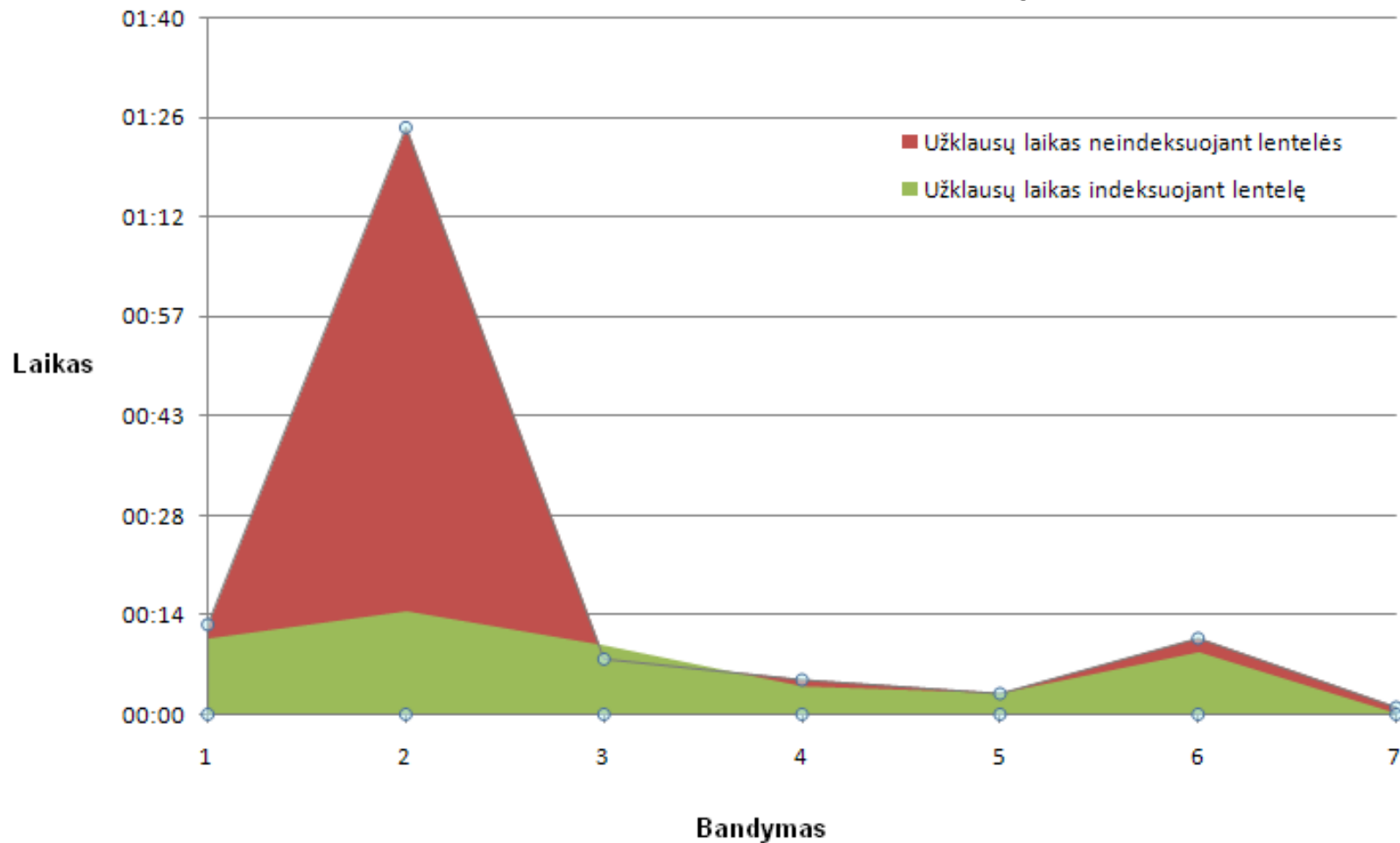
PASTABA: tokiuose medžiuose paieška gali prilygti nuosekliam duomenų skaitymui iš DB lentelės.

TOP 5 kinijos vardai

- 1.王 Wáng
- 2.李 Lǐ
- 3.张 Zhāng
- 4.刘 Liú
- 5.陈 Chén

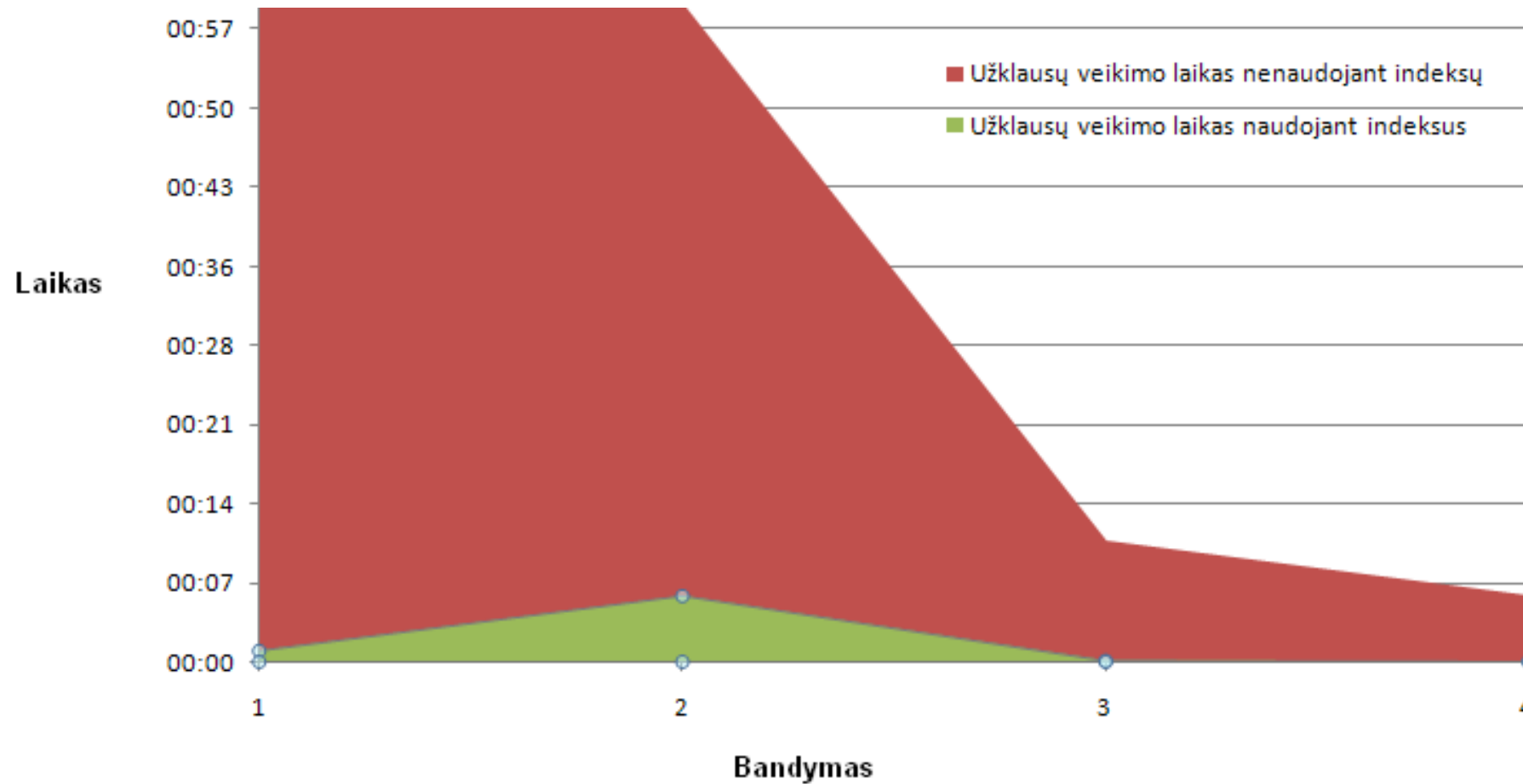
Bandymų rezultatai

Vienos lentelės indeksavimo tyrimas

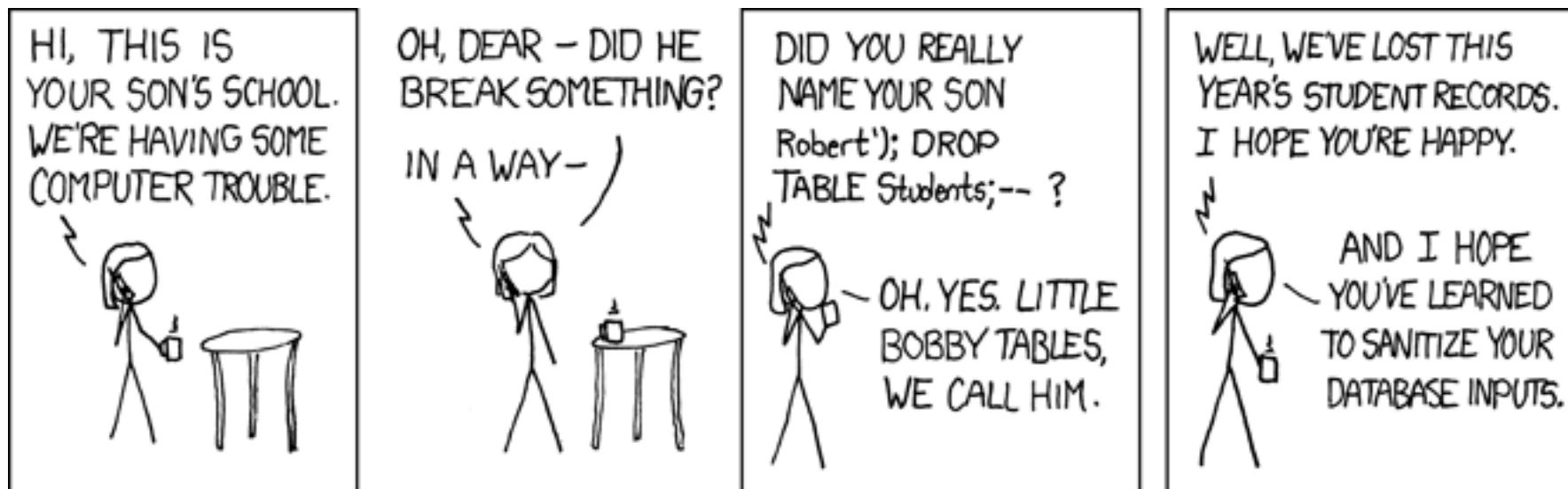


Bandymų rezultatai

Dviejų lentelių indeksavimo tyrimas



SQL injekcijos (įterptinės užklauskos)



ĮTERPTINĖS UŽKLAUSOS

- SQL-Injekcija yra viena iš daugelio web skript'ų atakų. Tai vienas paprasčiausių ir populiariausių web atakų metodų. Šis pažeidžiamumas dažniausiai sutinkamas PHP ir kituose web skriptuose.
- Pagrindinė problema, jei web aplikacija parametrus perduoda iš nepatikimų šaltinių (pavyzdžiui formos) deda tiesiai į SQL skriptą, nepatikrinus SQL raktinių žodžių, simbolių, backslashai arba kabutės.
- Tokio modifikuoto skripto pagalba „blogasis lankytojas“ gali 'patobulinti' jūsų SQL užklausą ir sukurti duomenis su neteisingais duomenimis, arba perimti iš duomenų bazės slaptus duomenis arba sugadinti jūsų duomenis.

INTERPTINĖS UŽKLAUSOS

users.php:

```
<?php
```

```
// Prisijungimas prie db.
```

```
$id = $_GET['id'];
```

```
$query = "SELECT * FROM users WHERE UserId = '$id'";
```

```
$rez = mysql_query($query);
```

```
// galutinis rezultatas
```

```
?>
```

users.php?id=1 , tai rezultate pamatysime vartotojo su id=1 duomenis.

O tarkim users.php?id=1' or 1=1";– rezultate to pamatysime visų vartotojų duomenis.

INJEKCIJA 1=1 pagrindu

```
SELECT * FROM Users WHERE UserId = 105 or 1=1
```

Sąlyga WHERE UserId = 105 or 1=1 visada bus teisinga

```
SELECT UserId, Name, Password FROM Users WHERE UserId = 105 or 1=1
```


INTERPTINĖS UŽKLAUSOS



INJEKCIJA „“=„“ pagrindu

User Name:

Password:

```
uName = getQueryString("UserName");
```

```
uPass = getQueryString("UserPass");
```

```
sql = "SELECT * FROM Users WHERE Name ='" + uName + "' AND Pass ='" + uPass + "'"
```

INJEKCIJA „“=„“ pagrindu

```
uName = getQueryString("UserName");  
uPass = getQueryString("UserPass");
```

```
sql = "SELECT * FROM Users WHERE Name ='" + uName + "' AND Pass ='" + uPass + "'"
```

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

Sąlyga WHERE ""="" visada bus teisinga

INJEKCIJA keletu užklausų vykdymo pagrindu (angl. **Batched Statements**)

```
SELECT * FROM Users;  
DROP TABLE Suppliers;
```

SERVERIO KODAS

```
txtUserId = getRequestString("UserId");  
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

```
SELECT * FROM Users WHERE UserId = 105; DROP TABLE  
Suppliers
```

PADUODAMAS PARAMETRAS „105; DROP TABLE Suppliers“

Nužudys jūsų sistemą, jei naudotojas turės tokias teises ir

INTERPTINĖS UŽKLAUSOS



Paruoštos užklausos (angl. prepared statements)

C# ADO.NET[edit]

This example uses C# and ADO.NET:

```
using (SqlCommand command = connection.CreateCommand())
{
    command.CommandText = "SELECT * FROM users WHERE USERNAME = @username
AND ROOM = @room";

    command.Parameters.AddWithValue("@username", username);
    command.Parameters.AddWithValue("@room", room);

    using (SqlDataReader dataReader = command.ExecuteReader())
    {
        // ...
    }
}
```

Paruoštos užklausos (angl. prepared statements)

PHP PDO

This example uses PHP and PHP Data Objects (PDO):

```
$stmt = $dbh->prepare("SELECT * FROM users WHERE USERNAME = ? AND  
PASSWORD = ?");
```

```
$stmt->execute(array($username, $pass));
```

Alternately:

```
$stmt = $dbh->prepare("SELECT * FROM users WHERE USERNAME=:username  
AND PASSWORD=:pass");
```

```
$stmt->execute(array('username' => $username, 'pass' => $pass));
```

Paruoštos užklausos (angl. prepared statements)

Naudojant paruoštas užklausas

```
$stmt = $mysqli->prepare("SELECT pavarde FROM users WHERE vardas=?") $stmt->bind_param("s",$vardas);
```

Nereikia pakartotinai rengti ir padavinėti užklausos

```
$vardas = "vardenis"; $stmt->execute();
```

Kitas žmogus:

```
$vardas = "jonas"; $stmt->execute();
```


SQL injekcijos(angl. SQL injection)

Naudojant paruoštas užklausas ir MySQLi, nereikia naudoti `mysqli_real_escape_string` ir t.t. Už mus viską tai padaro PHP.

Object oriented style

```
string mysqli::real_escape_string ( string $escapestr )
```

Procedural style

```
string mysqli_real_escape_string ( mysqli $link , string $escapestr )
```

Ačiū už dėmesį
