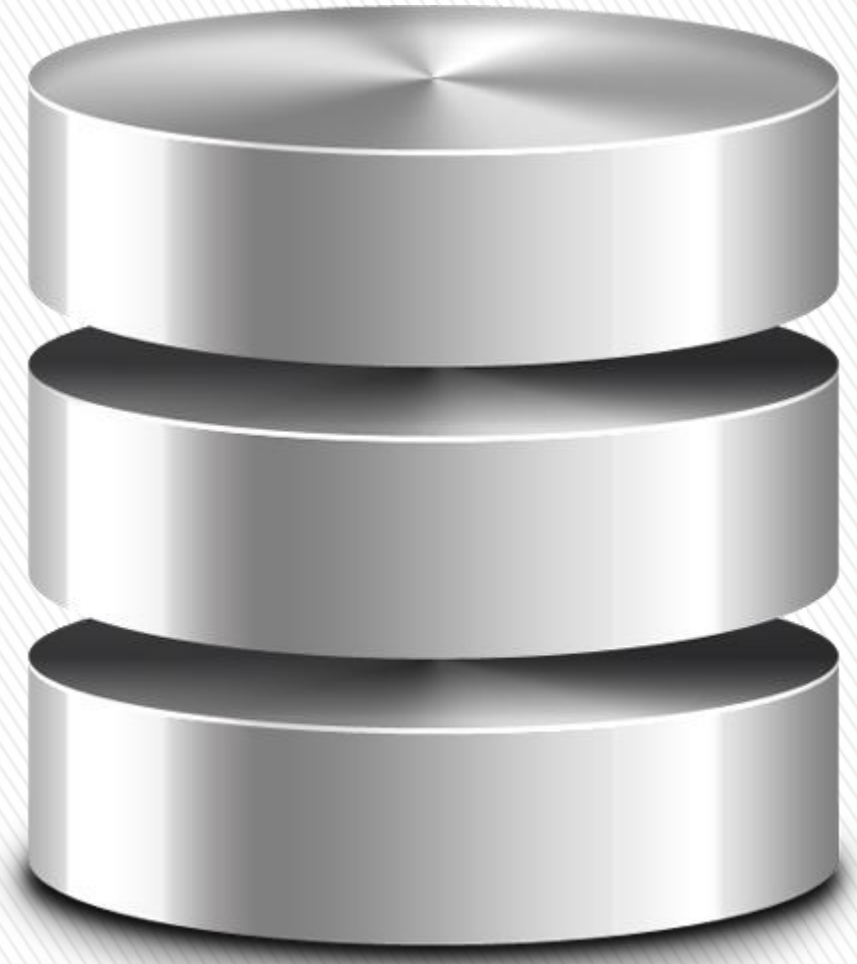


»Гл. ас. д-р Георги Чолаков

»Бази от данни

Индекси

Какво е индекс?

Структура от данни, съдържаща копие на част от данните от таблица.

Обикновено атрибут или комбинация от атрибути, за предпочитане с минимална дължина.

Цел: ускоряване извличането на данни.

За сметка на:

- Допълнителни операции при промяна на данните в таблицата;
- Използване на допълнително място за съхранение на данните.

Какво е индекс?

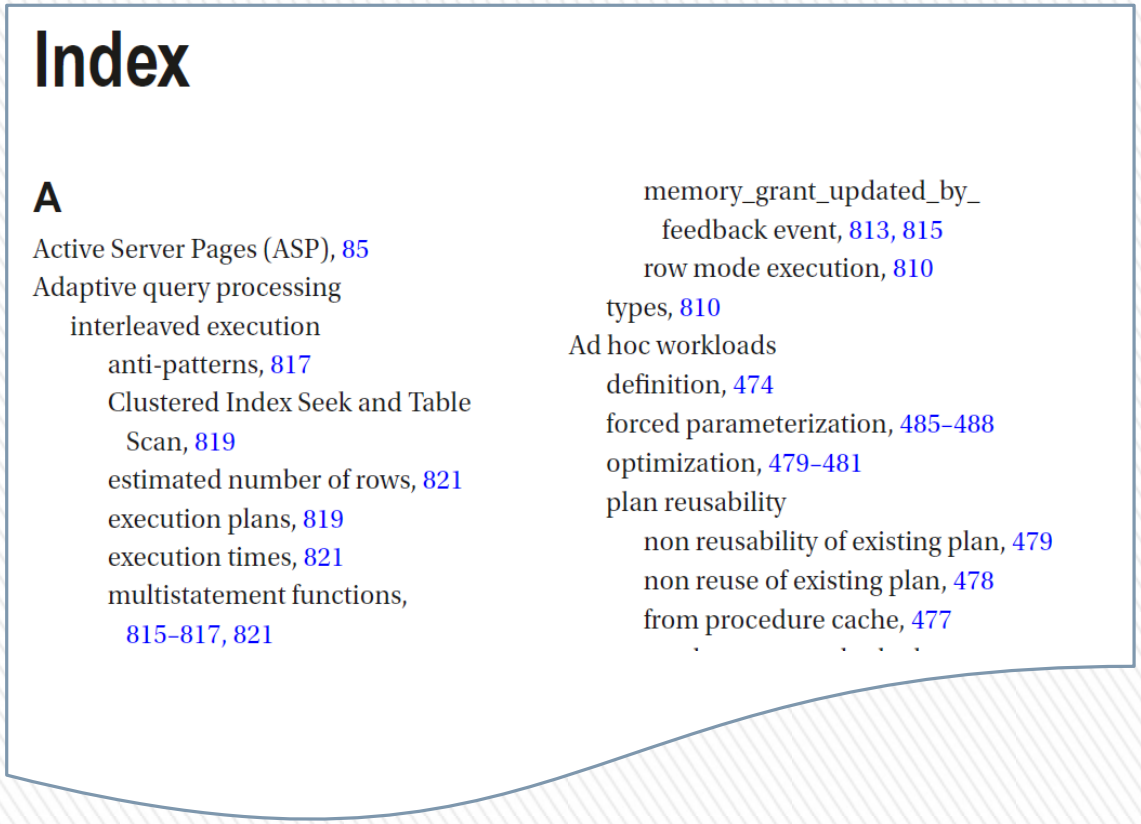
Индексите се използват за бързо намиране на редовете от таблицата без да е нужно обхождане на всеки ред от таблицата винаги, когато се търсят данни в нея.

Създаването на индекс копира атрибутите, които се индексират, в новата структура в отделно физическо пространство (обикновено). Така логически и физически са независими от данните на таблицата, с която са асоциирани – затова могат да бъдат създавани и премахвани без това да повлияе на данните на таблицата.

При достъп до таблицата процес, обикновено наричан *Optimizer*, решава дали да бъде сканирана цялата таблица, претърсвайки всичките й редове, или е по-бързо да бъде сканиран индекс в съчетание със сканиране на малка част от таблицата.

На какво е подобен?

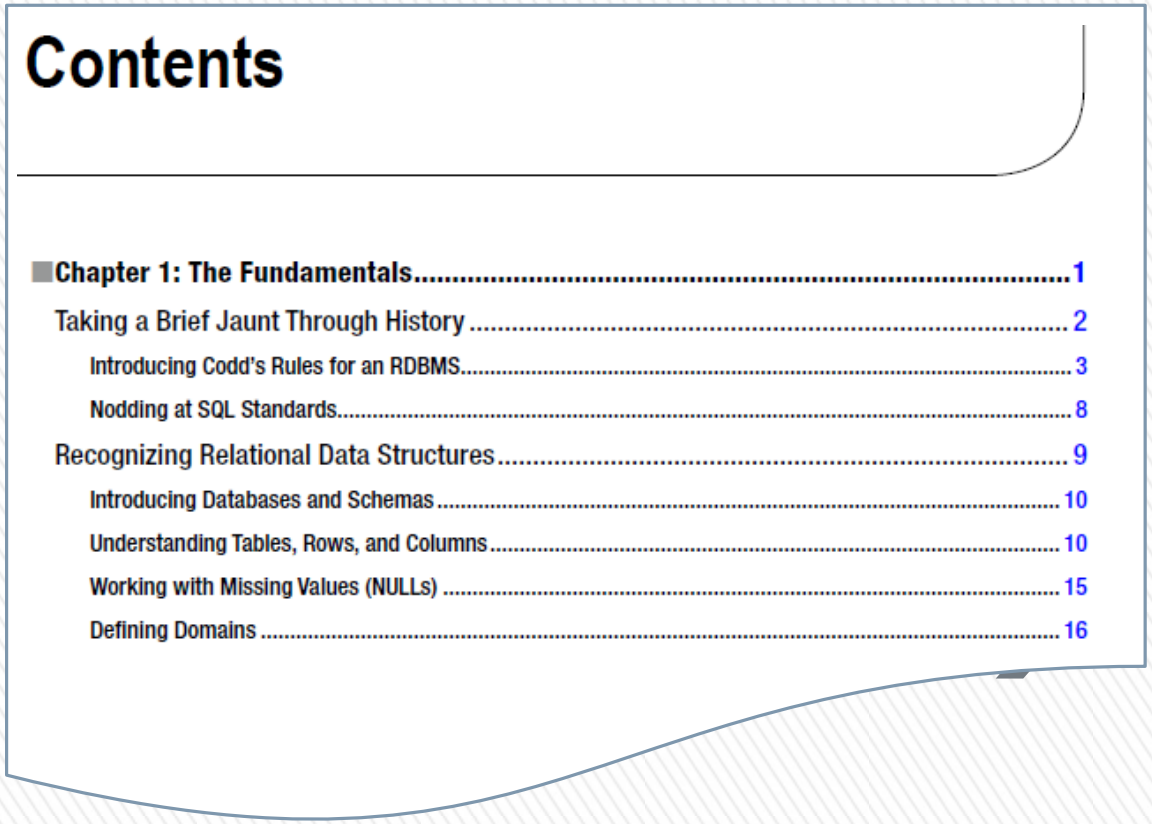
На индекс в края на книга, в който са изброени използвани термини и страниците, на които се използват.



Index

A

На съдържание, в което са изброени темите и страниците, от които започват.



Contents

Блок с данни

Блокът (страница, page) представлява най-малката физическа част от паметта, в която се записват данни.

- ✓ Данните от таблиците и индексите се записват в блокове.
- ✓ Размерът на блока може да се конфигурира и обикновено е 4Kb, 8Kb, 16Kb или 32Kb.
- ✓ Редовете в таблицата обикновено са по-малки от този размер, така че множество редове се вместват в един блок.
- ✓ Когато се търси даден ред се прочита блокът, в който се намира, като останалите редове от този блок се игнорират.
- ✓ Целта е да се намалят излишните прочитания на данни (I/O операции).

Типове индекси

Clustered

- Създаването на такъв индекс преструктурира блоковете с данни в ред, съвпадащ с подредбата в индекса, като в резултат се получава сортиране на редовете с данни в блоковете. Реализират се в дървовидна структура B-Tree, като листата в дървото съдържат самите блокове с данни (редове), подредени възходящо или низходящо;
- Таблица може да има само един клъстериран индекс, защото физическото подреждане на данните може да бъде само по един начин;
- Тъй като данните физически са подредени като индекса не е нужна допълнителна операция за намиране на актуалното физическо разположение на данните в блока, защото те самите са в листата на дървото;
- Могат да увеличат драматично бързодействието при извличане на данни, особено когато се осъществява последователен достъп до тях в същия или обратен ред на индекса;
- Някои производители (Oracle напр.) реализират тази концепция като Index Organized Table (IOT).

Типове индекси

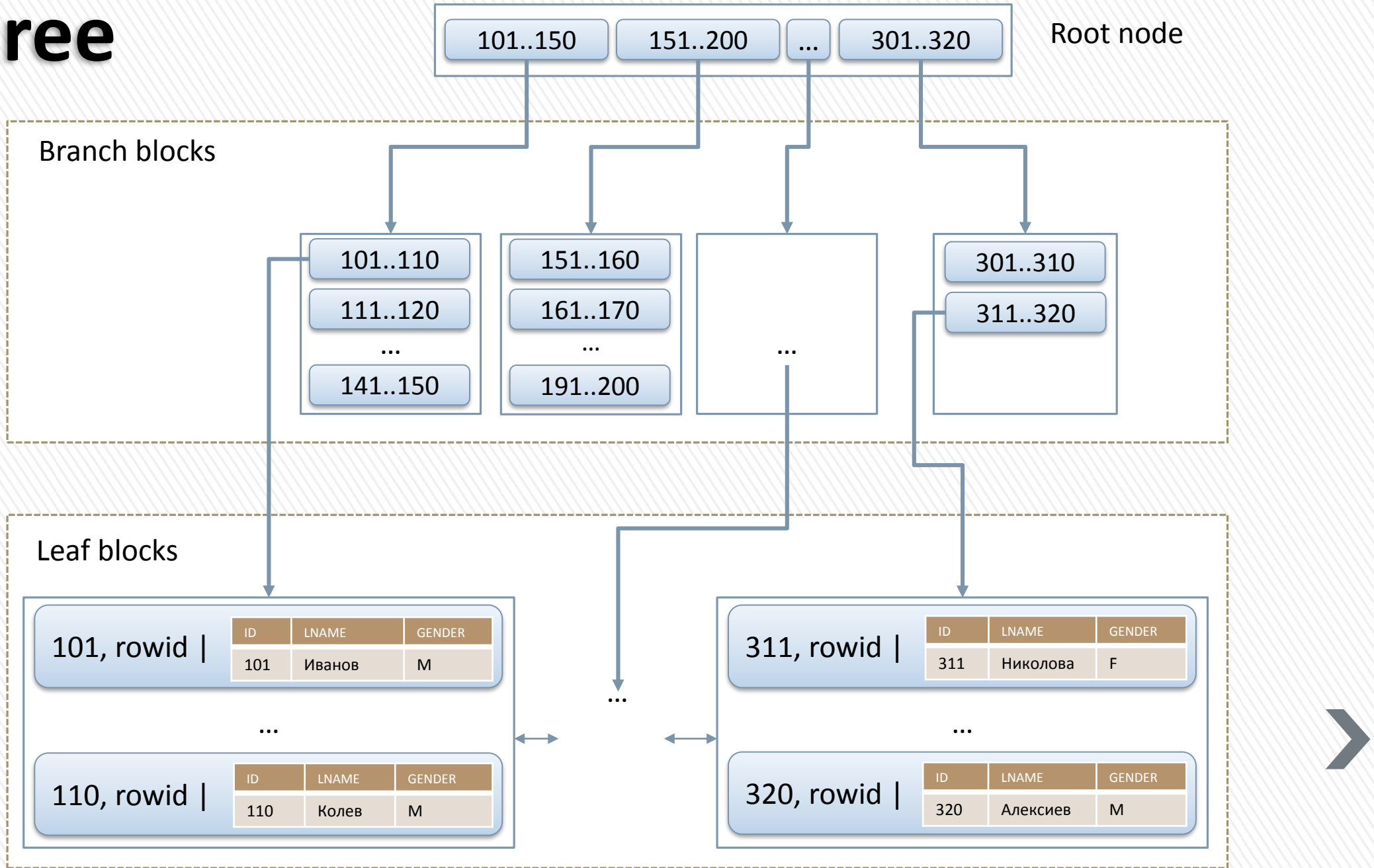
Nonclustered

- Индексът е структуриран по начин, който не кореспондира с актуалното разположение на записите с данни, т.е. той съдържа логически сортирани данни, т.е. индексираните стойности са сортирани;
- По-малко ефективен от клъстерирания, защото съхранява указатели (блок и номер на запис в него) към реалното местоположение на данните и за извличането им ще е нужна една операция повече отколкото при клъстерирания;
- Обикновено се създава за неключови колони в таблицата, използвани в JOIN, WHERE или ORDER BY клаузи;
- Могат да бъдат създавани повече от един за таблица.

Архитектура на clustered & nonclustered

Реализират се като B-tree (Balanced) – балансирано дърво означава автоматично да настройва нивата си така, че всички листа да са на „почти“ еднакво разстояние от корена (идеално балансирано дърво – разстоянието на всички листа до корена не се различава с повече от 1).

B-tree


101..150
151..200
...
301..320
Root node
Branch blocks
101..110
111..120
...
141..150
151..160
161..170
...
191..200
...
301..310
311..320
Leaf blocks
101, rowid |
ID
LNAME
GENDER
101
Иванов
M
...
110, rowid |
110
Колев
M
...
311, rowid |
311
Николова
F
...
320, rowid |
320
Алексиев
M

Пример

Ако търсим по ключ 105 започваме от корена. Ключ 105 се намира в първия клон в блока от 101 до 110. Неговото първо листо съдържа блока с:

- Ключа и указател (rowid, locator) към реда с данните от таблицата (nonclustered) или
- Ключа и самия ред с данните (clustered)

Със същия брой операции ще бъдат намерени данните по кой да е от ключовете (индексираната стойност) в индекса.

Пример – 2 индекса

Index (FACNO)

1601681001	
1601681001	
1601261027	
1601261027	
1601261027	
1601681040	
1601681040	
1601261054	

FACNO	COURSE_ID	YEAR	GRADE
1601261027	DB	2018	6
1601261054	DB	2017	4
1601681001	AI	2017	5
1601681040	DB	2018	4
1601261027	AI	2017	6
1601261027	SE	2018	5
1601681001	SE	2018	4
1601681040	AI	2017	3

Index (COURSE_ID)

	AI
	AI
	AI
	DB
	DB
	DB
	SE
	SE

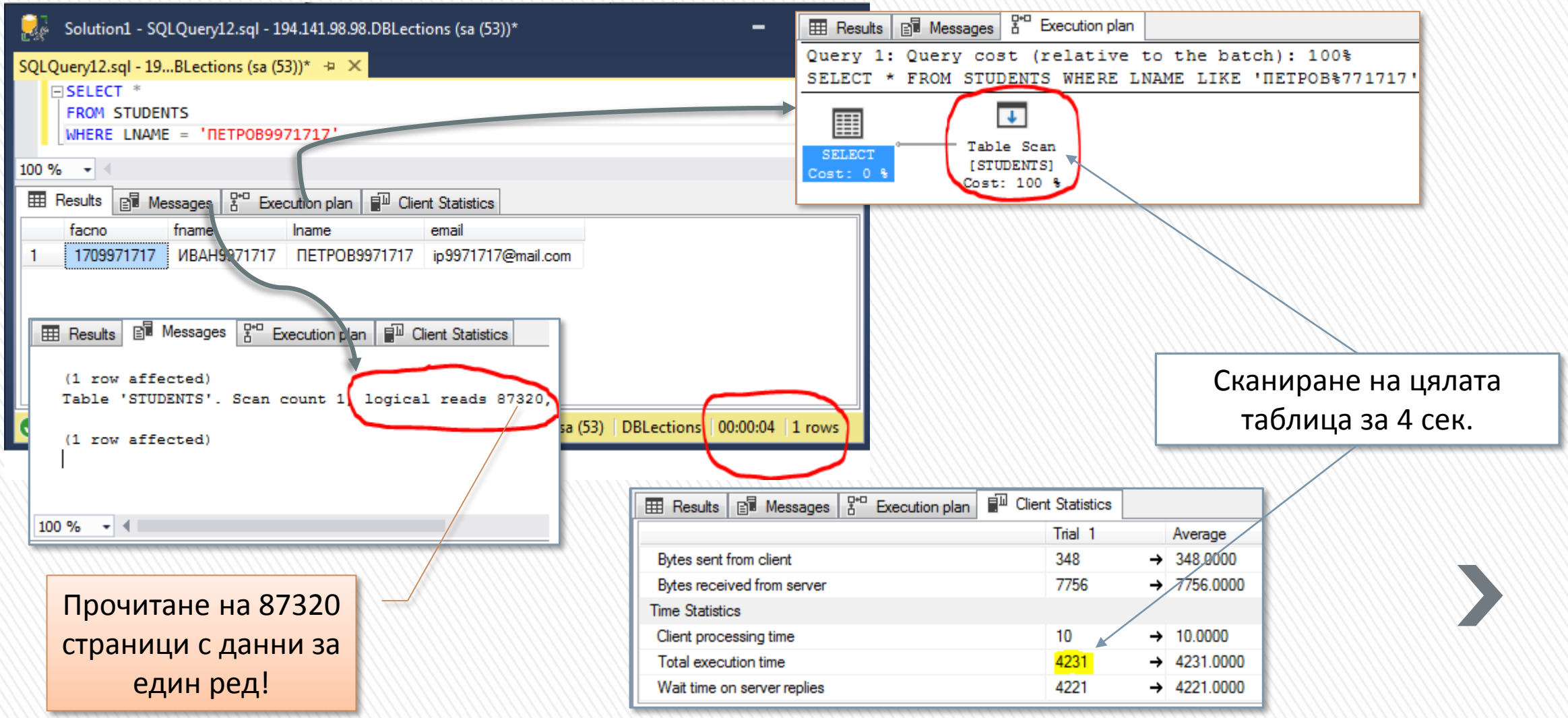
Други типове

- Bitmap index – съдържа двоично представяне на всеки запис, използвайки 0 и 1. Удобни за оптимизатора, защото не се налага да преобразува указателите до двоични стойности. Но могат да затруднят много DML операциите и да бъдат проблем с нарастването на данните;
- ISAM (Indexed Sequential Access Method) – използва проста структура със списък на номерата на редовете. Най-полезни за статични данни, защото вътрешната структура не позволява лесни промени, което ги прави уязвими за препълване;
- IOT (Index Organized Table) – комбинира таблица и индекс в едно. Индексът трябва да е първичният ключ. Листата съдържат целите записи на таблицата. Тъй като таблицата е построена по индексната структура четенето й в различен от индекса ред може да е проблем за бързодействието;
- Функционален – индексът е базиран на стойност от функция и е използваем при търсене в WHERE клаузата със сравнение по тази функция;
- Други…

Описаните типове и архитектури може да не са налични или имплементирани точно по този начин в някои бази от данни или да се наричат с други имена!

Пример – без индекси

Извличане на един запис от таблица с 10 милиона записа без използване на индекс.



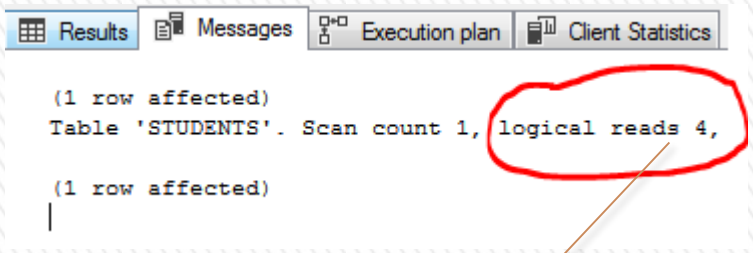
Прочитане на 87320 страници с данни за един ред!

Сканиране на цялата таблица за 4 сек.

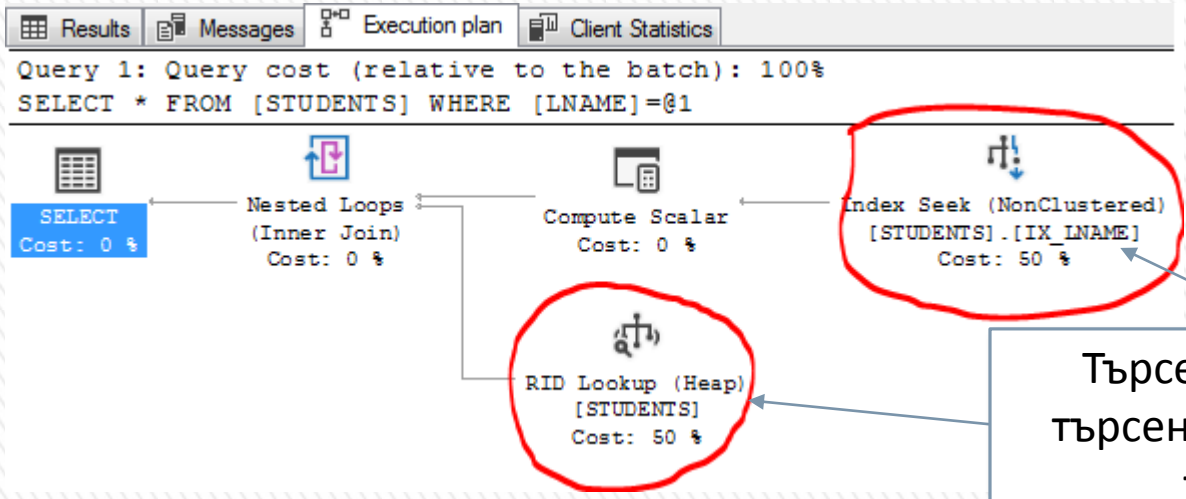
С индекс по предиката

Създаваме индекс по колоната LNAME, използвана в предиката, и изпълняваме отново:

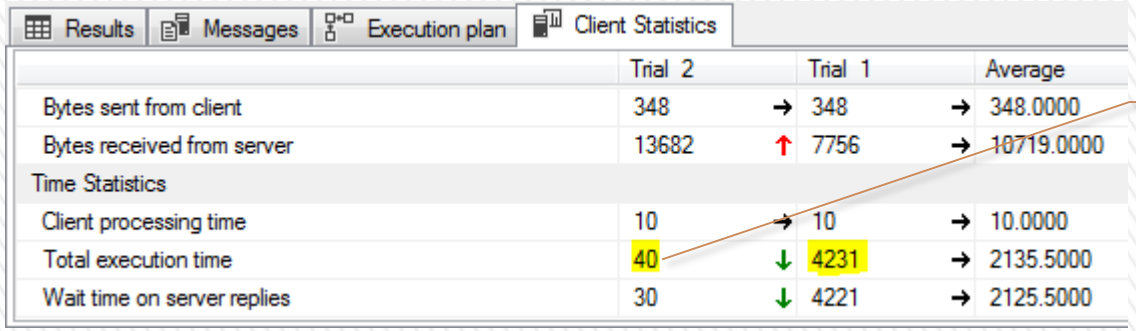
```
CREATE INDEX IX_LNAME ON STUDENTS(LNAME)
```



Прочитане на само 4 страници с данни!



Търсене в индекса и търсене в таблицата на този 1 ред



	Trial 2	Trial 1	Average
Bytes sent from client	348 →	348 →	348.0000
Bytes received from server	13682 ↑	7756 →	10719.0000
Time Statistics			
Client processing time	10 →	10 →	10.0000
Total execution time	40 ↓	4231 →	2135.5000
Wait time on server replies	30 ↓	4221 →	2125.5000

Драматично намаление на времето за извличане

При индексиране трябва да се избягва:

- *Създаването на твърде много индекси* – това може да забави операциите по промените на данните. Това е защото при всяка промяна на данни се променя и всеки индекс, направен за тази таблица;
- *Индексирането на твърде много колони* – не само, че индексът става по-сложен, но и по-голям относно заеманата памет. Индексът трябва да е относително много по-малък от самата таблица и да е създаден върху възможно най-малко колони;
- *Избор на колони за клъстериран индекс, чиито стойности често се променят* - това ще доведе до промяна на всички указатели на неклъстерираните индекси, значително утежнявайки UPDATE операциите. Също така ще има негативен ефект върху конкурентността като блокира всички заявки към същата част от таблицата и неклъстерираните индекси за времето на операцията. Затова създаването на клъстериран индекс върху често обновявани колони трябва да се избягва.