SQL Related Topics

*** How Data internally stored in SQL Server ?**

- محتاج أعرف وأفهم الـ topic دا عشان هو related بالـ SQL Query optimization و الـ Performance Tuning

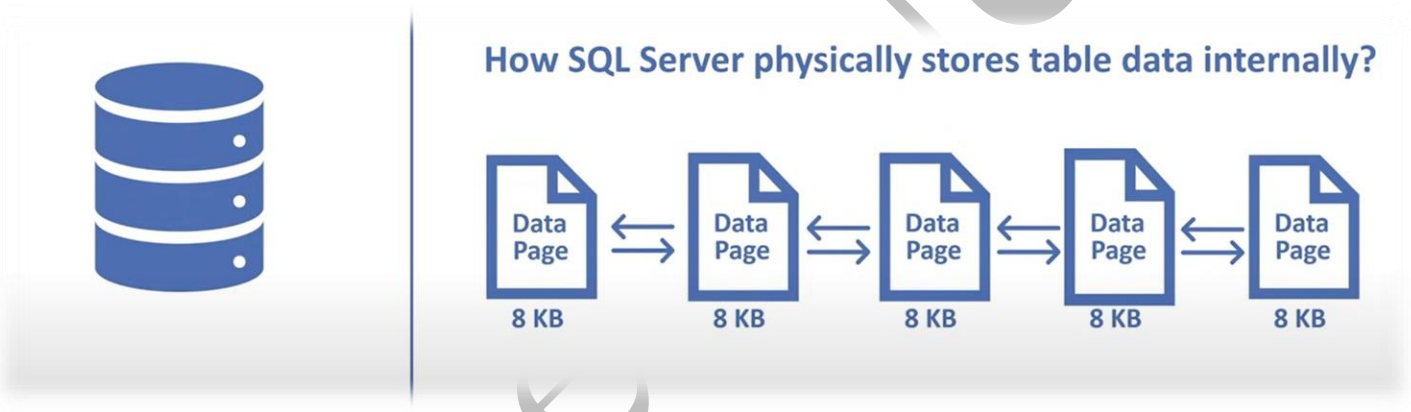
- الـ Data فى الـ Tables بتبقى على Row\Column Format فى الـ Logical Form

- فى الـ physical بتتخزن على شكل Data pages

- الـ Data pages هيا عبارة عن الـ Fundamental Unit فى الـ SQL Server

- حجم الـ Data Page عبارة عن 8 kb

- الـ Data بتتخزن Physically على صورة Series of Data pages



- نفترض عندنا الـ Table الأتى :

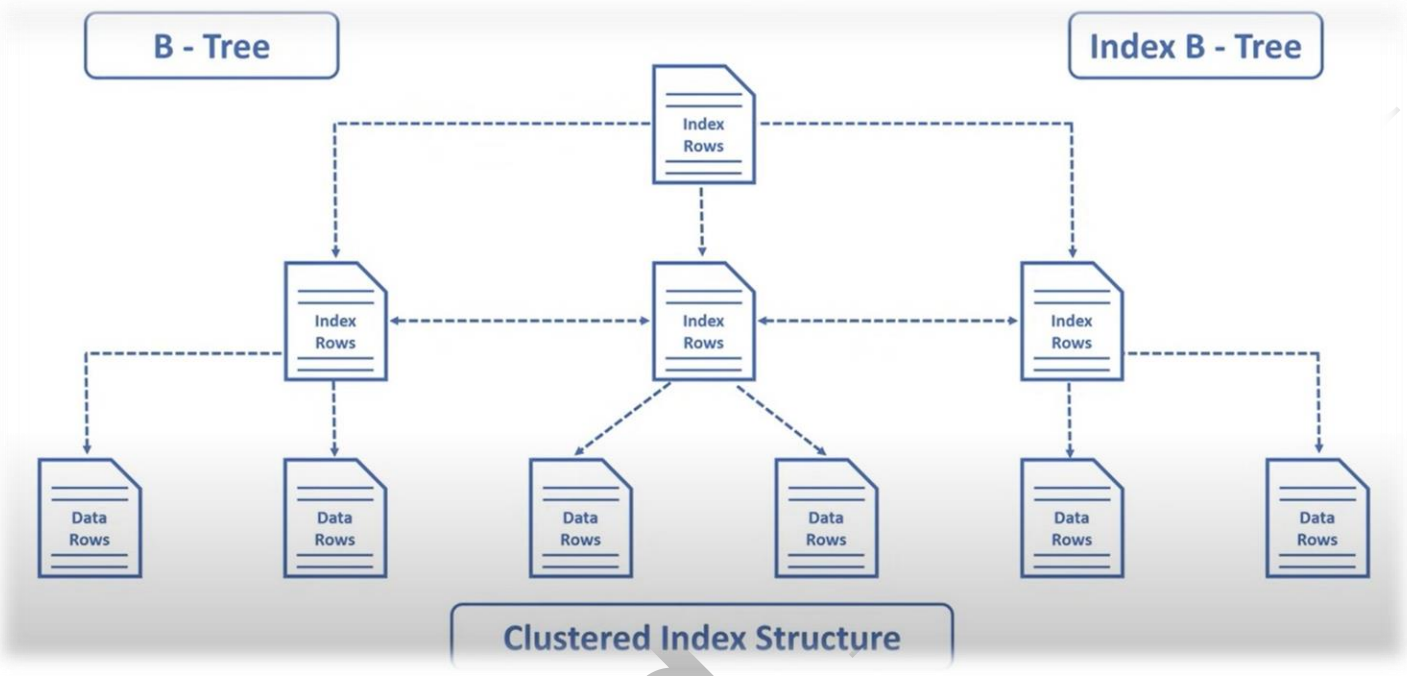
Primary Key

Employees Table

EmployeeId	Name	Email	Department
1	Mark	mark@pragimtech.com	IT
2	John	john@pragimtech.com	HR
3	Sara	sara@pragimtech.com	HR
4	Mary	mary@pragimtech.com	IT
5	Dave	dave@pragimtech.com	IT
...	.....	.....	.....
...	.....	.....	.....
1200	Steve	steve@pragimtech.com	HR

- الـ Employee ID هو الـ primary Key وبالتالى بيتم إنشاء Clustered Index بالـ PK

- يعنى بيـ sort الـ table بالـ PK Column (وبالتالى بقا هو اللى مرتب الـ Data وهستخدمه فى الـ B-Tree / Index B-Tree / Clusterd Index Structure)



- بتتكون من 3 main levels :

1) Root Node :

- عبارة عن الـ Tree Top Node

- بتـ contain Index Rows

2) Intermediate levels Nodes :

- بتكون one or Multiple Levels (بيـ Depend على الـ Clustered Index Column)

- بتـ Contain Index Rows

3) Leaf Nodes :

- بتبقى الـ Tree Bottom Nodes

- بتـ Contain Data Pages

- تقدر تعتبر الـ Nodes هيا عبارة عن Pointers بتوديك للـ row اللى انت محتاجه بسرعة

(بدل ما تلف على الـ Data كلها زى ما بيحصل فى الـ file-based system أو لو مش عامل Index فى الـ database system وتعمل حاجة إسمها table Scan)

(بيفرق معاك فى الـ Query Performance

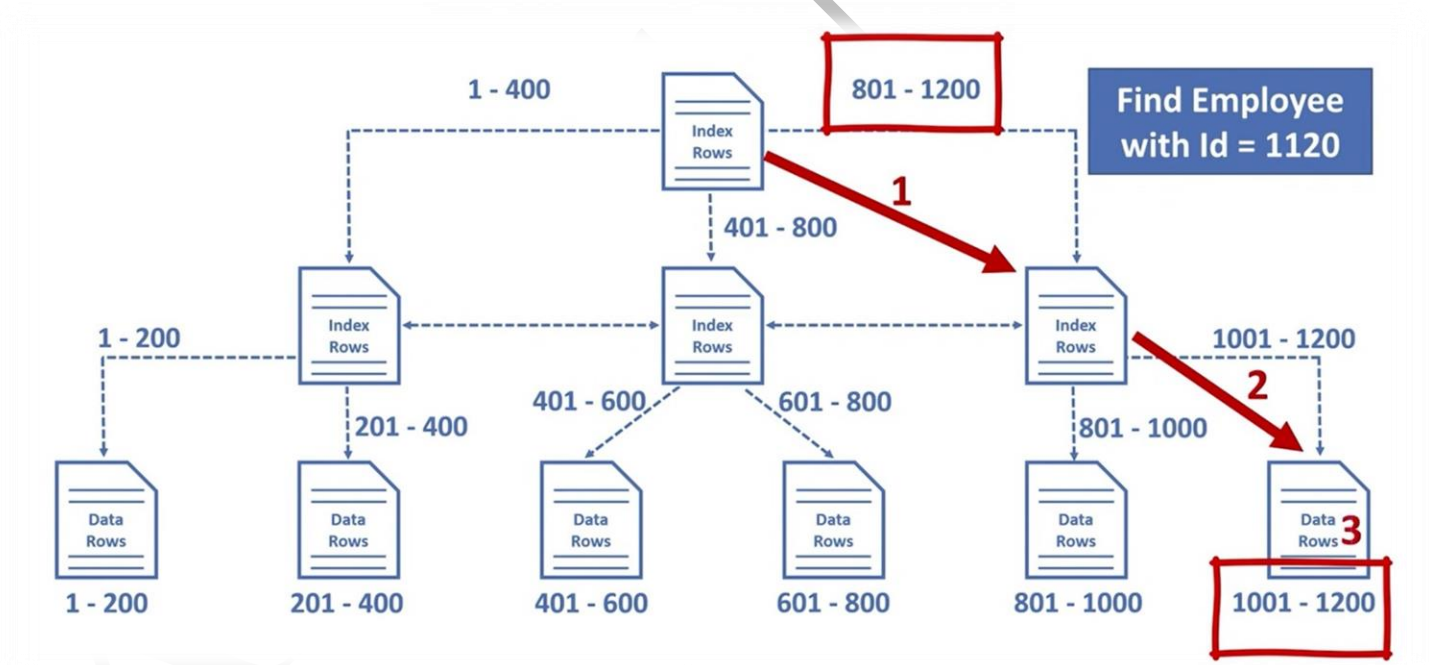
- الـ Select هيبقى أسرع

– لكن الـ Insert أو الـ DML هتبقى أبطأ ودا لأن بيحط الـ New Row فى مكانه "فى حالة مفيش Index بيحط الـ New Row فى أخر الـ table وخلاص ")

- هنفترض بنعمل الـ query الأتى :

```
Select * from Employee where EmployeeID = 1120
```

- الـ Database Engin هيعمل الأتى :



- فى خلال 3 خطوات قدر يوصل للـ record المطلوب ودا لأن الـ table فيه Clustered Index اللى هو الـ Primary Key وأنا بـ search بالـ ID

- الفكرة إنه بيقلل الـ Scope اللى بيـ search فيه
(بدل ما يدول فى الـ 1200 كلهم لا دور من 1001 لـ 1200) ودا عرفه من خلال الـ Clustered Index

- عدد الـ records فى كل data page هو العدد اللى بيقدر يـ fit الـ data page size (8kb)

*** Non-Clustered Index :**

- فى حالة لو أنا عاوز أ Search بـ Column غير الـ Primary Key
- هكتب الـ query

Select * from Employees where name = “ABC 932000 ”

وأفعل خاصية الـ Execution Plan

هلاقى إن Number of Rows Read هو كل الـ Rows برغم إن عاوز Row واحد فقط ودا لإن مفيش Index على الـ column

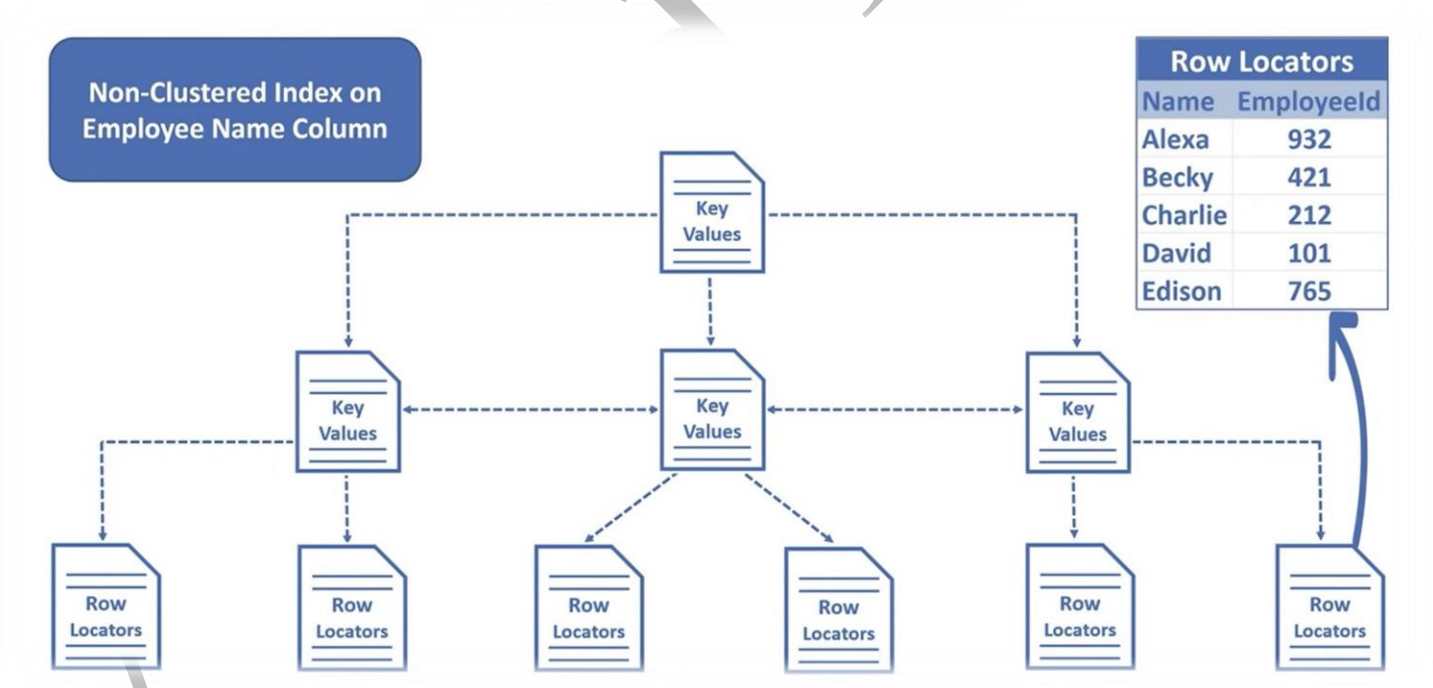
- الحل إن أستخدم الـ Non-Clustered Index على الـ Name Column

Create NonClustered Index Index_Name

On dbo.Employees(name)

- هلاقى بعدها إن الـ Number of Rows Read هو 1 مش كل الـ records

- How Non-Clustered Index Tree is stored in DB ?

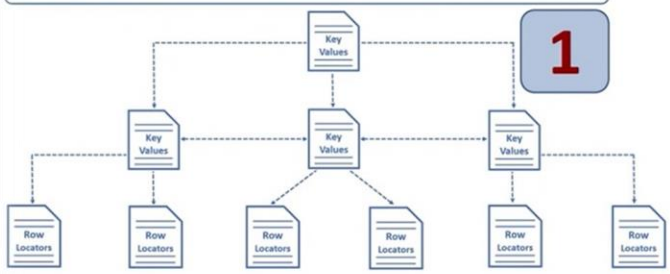


- الـ Key Values هيا عبارة عن الـ Name Values sorted in Alphabetical order

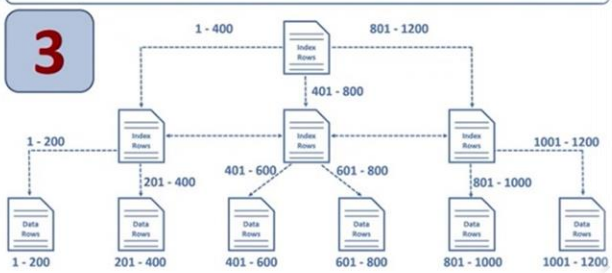
- الـ row Locator عبارة عن الـ Name ومعاه الـ Clustered Index (PK)

- لما بستخدم الـ Non-clustered Index بيستخدم معاه الـ clustered ويربطهم ببعض عشان يـ find الـ Record المطلوب

Non-Clustered Index on Employee Name Column



Clustered Index on Employee ID Column



Row locators contain employee Name & Employee Id

Row Locators	
Name	EmployeeId
Alexa	932
Becky	421
Charlie	212
David	101
Edison	765

2

Data rows in the leaf node are sorted by Employee ID

4

Employees Table			
EmployeeId	Name	Email	Department
1	Mark	mark@pragimtech.com	IT
2	John	john@pragimtech.com	HR
3	Sara	sara@pragimtech.com	HR
4	Mary	mary@pragimtech.com	IT
5	Dave	dave@pragimtech.com	IT
...	.....	.....	.....
...	.....	.....	.....
1200	Steve	steve@pragimtech.com	HR

- فى البداية بيتم البحث عن الـ name فى الـ non clustered Tree

- بـ Extract الـ Clustered Index لـ Name المطلوب

- بـ inner join بين الـ Clustered Index والـ table عشان يجيب الـ record المطلوب

Without index on Name column	
SELECT	
Cached plan size	16 KB
Estimated Operator Cost	0 (0%)
Degree of Parallelism	0
Estimated Subtree Cost	11.0685
Estimated Number of Rows Per Execution	1.00118
Statement	
SELECT * FROM [Employees] WHERE [Name]=@1	

With index on Name column	
SELECT	
Cached plan size	24 KB
Estimated Operator Cost	0 (0%)
Degree of Parallelism	0
Estimated Subtree Cost	0.0065704
Estimated Number of Rows Per Execution	1
Statement	
SELECT * FROM [Employees] WHERE [Name]=@1	

*** Heap Table :**

→ What is it ?

- هو عبارة عن table مفيهوش lustered Index (سواء كان Created بالـ PK أو Explicitly)

- ممكن يبقى فيه one or more Non-Clustered Index

→ in case that Table has not Clustered and Non-Clustered

- الـ SQL server هيستخدم الـ Table Scan (All records Scan)

- هيـ stored كـ DB Table عادى

- الـ Order can't be Predictable

- لو عاوز أضمن الـ Order هستخدم الـ Order By Clause

- مش دايماً الـ Table Scan is bad

-الـ SQL Server بيستخدم فى الـ Tables اللى فيها Few Rows عن الـ Index Seek (Non-Clustered) وبيكون better Performance

- لو عاوز أعمل forcing لـ SQL Server إنه يستخدم الـ Index seek بكتب الـ Query :

```
Select * From Table_Name
With ( Index (Index_name))
Where Column = Value
```

Sp_helpindex Table_Name → determines that there is an index or not

---

→ Create Heap :

1) Table Without Clustered Index

2) if Clustered Index exists , Drop it

→ Remove Heap :

- Create Clustered Index .

→ **When to use :**

1) Stagging Table :

- فى حالة الـ Insertion Operation (Large – Unordered)

- لأن هسرع الـ insertion بحيث إنه ميهتمش بالـ Row Order (يضيف وخلاص بدون ordering)

2) Table is small :

- فى حالة عندى table فيه data قليلة بالتالى أنا مش محتاج أعمل index عشان أعمل search

- فى الحالة دى الـ table scan بيبقى better من الـ Index seed

3) Data is always accessed by Non-Clustered :

- فى حالة إن دايماً بـ Access الـ data بالـ non-Clustered Index

---

→ **RID lookup Vs Key Lookup :**

RID → Row ID

- الـ Row ID Lookup بيستخدم فى حالة عندى table Heap مفيهوش Clustered Index وأنا بستخدم الـ Non clustered Index

- من المعروف إن الـ Non-Clustered Index بيستخدم الـ Clustered Index معاه

(فى الحقيقة هو بيستخدم الـ Key lookup)

- الملخص : فى حالة الـ Non Clustered :

1) لو فيه Clustered Index : بيستخدم الـ Key Lookup

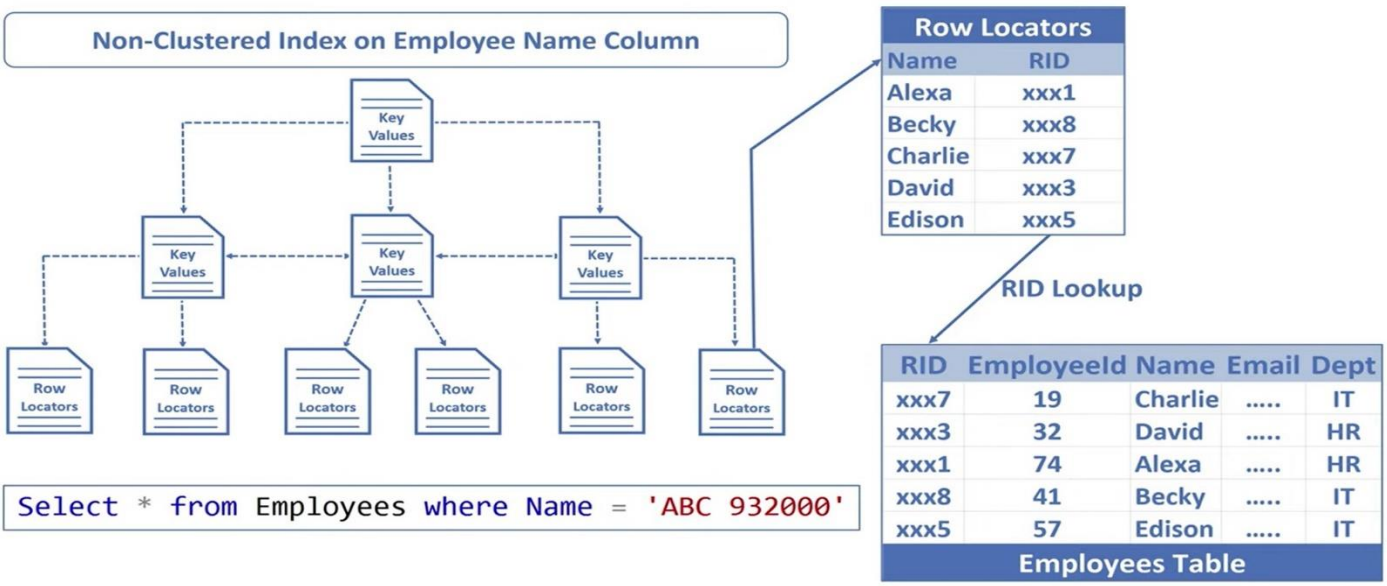
2) لو مفيش Clustered Index: بيستخدم الـ RID Lookup

- الـ SQL Server ممكن يستخدم الـ RID \ Key فى حالتهم وممكن لأ :

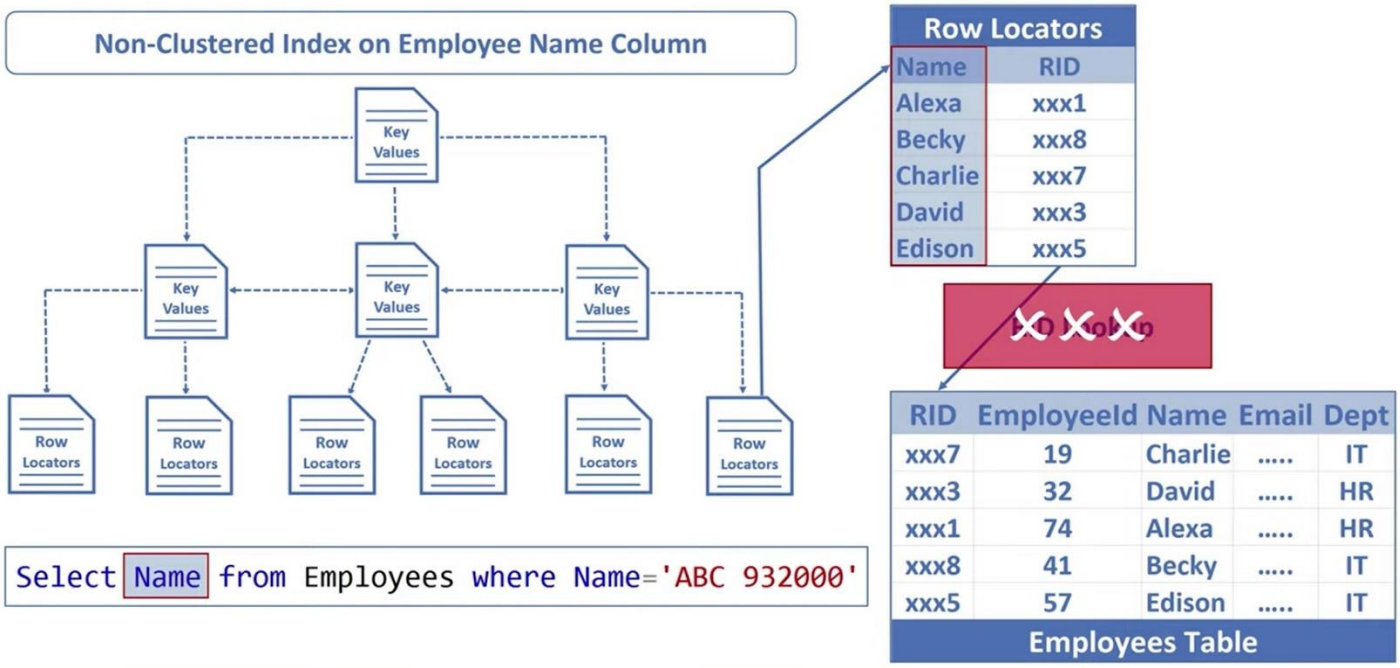
- دا بيـ Depend على الـ Query (إنت عاوز ايه فى الـ select)

* فى حالة الـ RID Lookup :

1) هيستخدم الـ RID Lookup لو عاوز كل الـ Columns أو Column غير اللى عليه الـ Non-Clustered Index



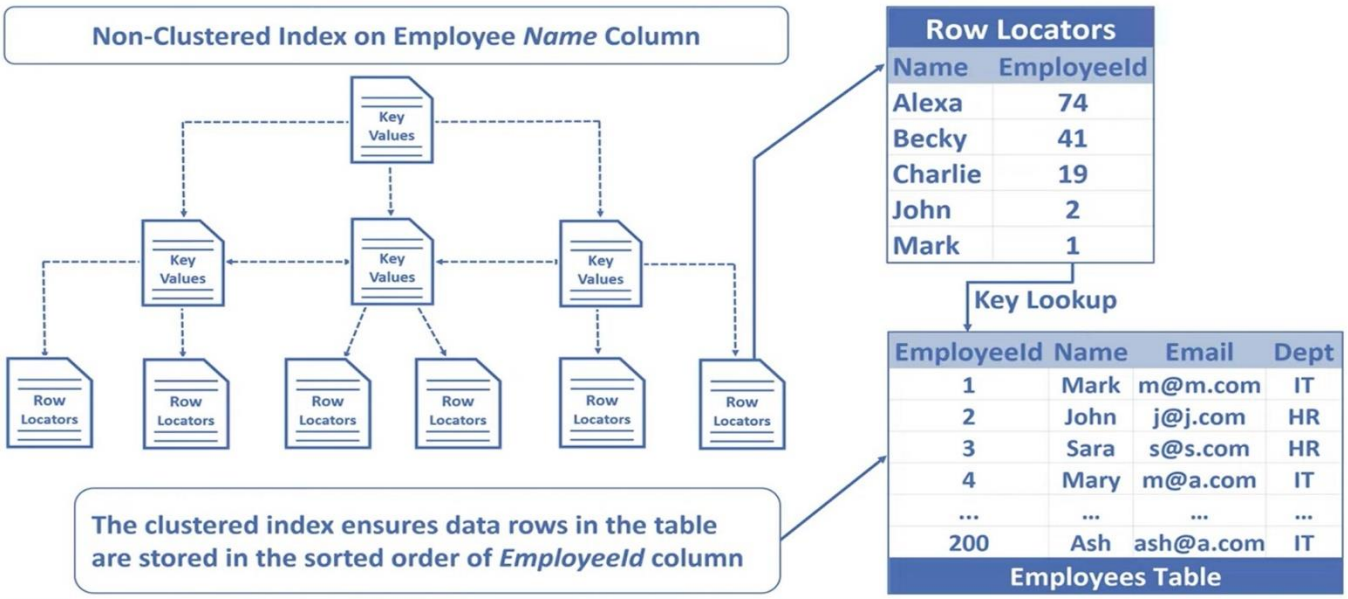
2) مش هيستخدم الـ RID Lookup لو عاوز الـ column اللى عليه الـ Non-Clustered Indexفقط (ودا عشان معلومة الـ name موجودة معاه فى الـ Row Locators)



*** فى حالة الـ Key Lookup :**

1) هيستخدم الـ Key lookup لو عاوز كل الـ Columns أو Column غير اللى عليه الـ Clustered Index والـ Non-Clustered Index

(لأن دول الـ two columns اللى موجودين فى الـ Row locators)



2) مش هيستخدم الـ Key lookup لو عاوز الـ Columns الـ Clustered Index والـ Non-Clustered Index

(لأن دول الـ two columns اللى موجودين فى الـ Row locators)