Day 8

*** Index :**

- الـ Data بتتخزن فى الـ Hard disk على صورة Packets (Data Pages)

- بيتم ترتيبها بالـ Primary Key

- لو عملت Table من غير Primary Key دا هيأدى إن الـ Data مش هتكون Sorted على الـ Hard disk

(الـ insert هتبقى أسهل لأنه بيضيف الـ recordالجديد فى الأخر مش هيحتاج ياخد وقت على ما يحطه فى مكانه الفعلى بالترتيب)

(الـ Search هيبقى صعب جداً لأنه هيضطر يعمل حاجة إسمها الـ Table Scan إنه يلف على كل الـ table حتى لو انت عاوز أول قيمة)

(الحالة دى بتبقى شبيه لـ Heap)

- فايدة الـ Primary Key فى Table :

1) بيعمل Sorting لـ Data على الـ hard disk

2) بيعمل Clustered index

(لكل Table بيعمله Tree خاصة بيه)

(الـ search أسهل بكتير لأنه هيعمل Data page scan مش Table)

(الـ Insert هيبقى أبطأ عشان هيحط الـ new record فى مكانه)

(الـ Treeبتبقى عبارة عن 3 Levels :

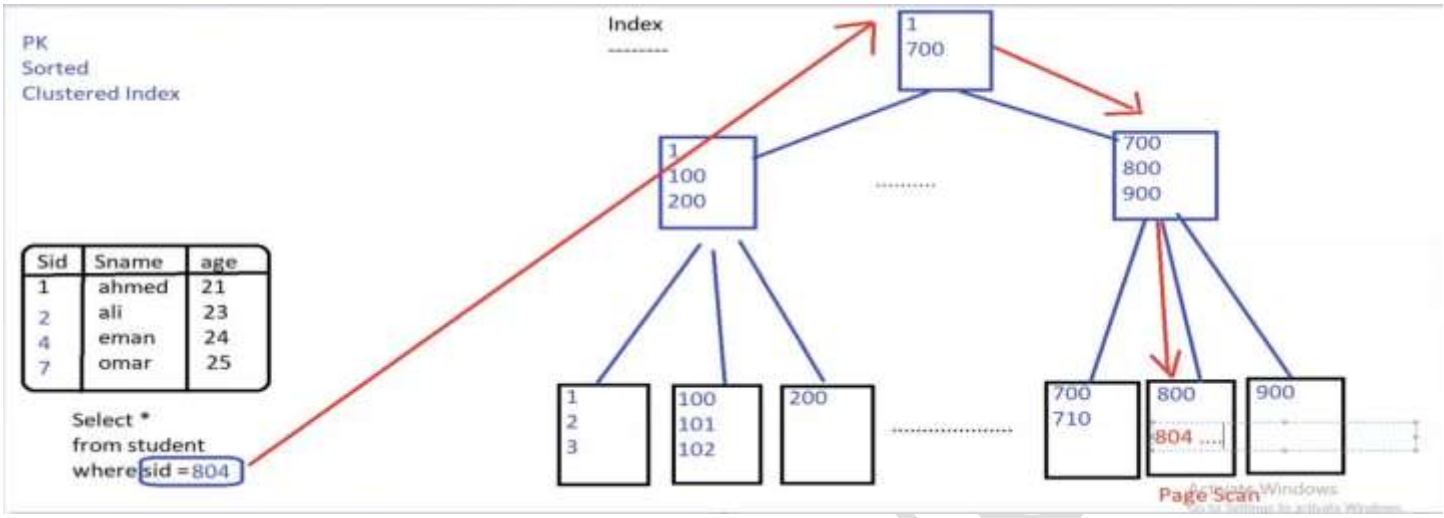
1) Root level

2) Non-Leaf Level

3) Leaf Level)

(بيبنى الـ tree على قيم الـ primary Key بحيث كل Node تحتها مجموعة من الـ data pages وبياخد قيمة أول PK فى الـ data page كـ indicator ليه "عشان يسهل على نفسه ويسرع الوصول للمطلوب")

(الـ server اللى بيعمل الـ tree دى فى الـ memory بتاعت الـ server لكن الـ data pages على الـ hard disk)



*** Non-Clustered Index :**

- فى الـ clustered Index كان بيتم ترتيب الـ Data بالـ Primary key وكان بيتم تلقائى مجرد إنك تعمل column من الـ table عبارة عن Primary Key

- طب لو عاوز أ search بس مش بالـ primary key

(فى الحالة دى إن الـ Index عندك يكون الـ search Column لأنك لو معملتش كدا هيـ search بإستخدام الـ Table Scan ودا بطئ "أنا عاوز أسرع الوصول لـ Data ")

(بستخدم الـ Non-Clustered Index ودا أنا بحدده بقوله إعمل non clustered Index بالـ Column)

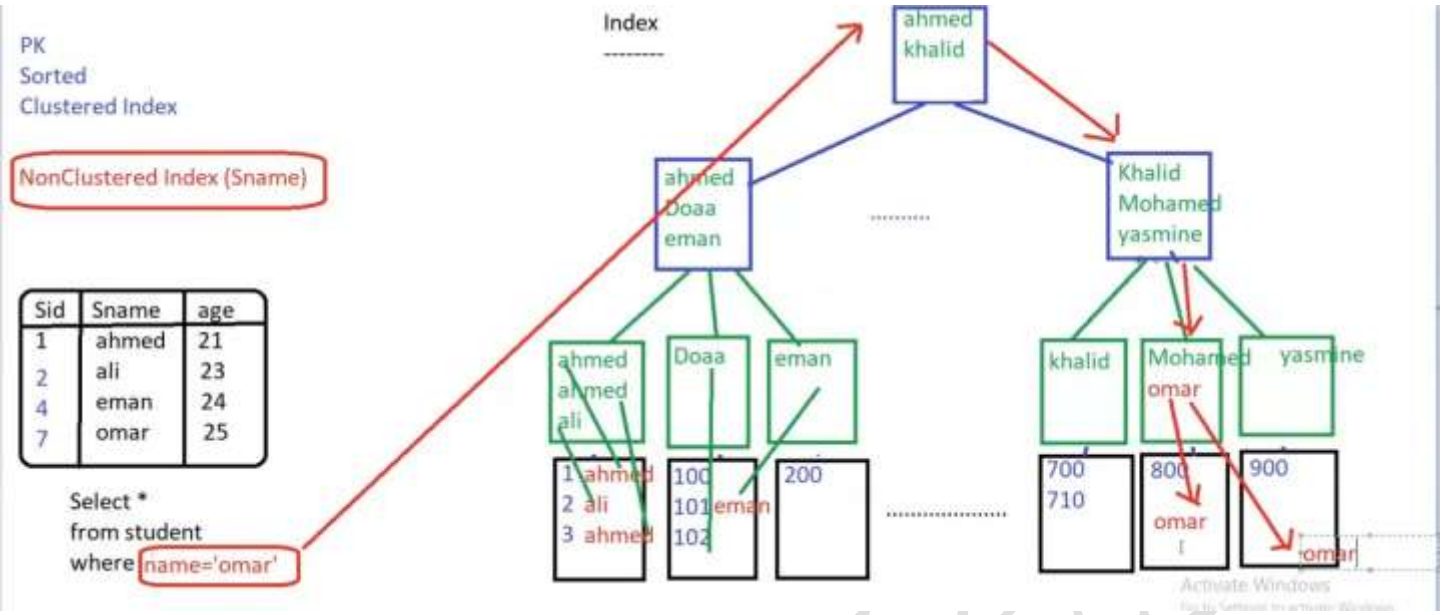
(بيتم عمل Tree برضو زى الـ clustered ولكن بيحط Level قبل الـ data pages وهو عبارة عن Data pages تانية)

(كل New Data Page قصادها Data page قديمة لكن الفرق
إن الـ New فيها الـ Valuesبتاعت الـ Column و Sorted وكمان بتـ Point لـ مكان الـ Data فى الـ Data page الأساسية)

- طبعاً أبطأ من الـ clustered برضو ولكنه أسرع من الـ table Scan

- الـ Non Clustered ممكن أعمله على كذا Column

(لكن الـ clustered بيبقى One Column بس وغالبا بيبقى الـ Primary Key)



*** Syntax of Cluster \ Non-Cluster :**

Create [Clustered | NonClustered] Index Index_Name

On Table_Name(Column_Name)

- الـ PK Constraint بينتج عنه Clustered Index

- الـ Unique Constraint بينتج عنه Non-Clustered

Create Unique Index Index_Name

On Table_Name(Column_Name)

(فى الحالة دى هيطبق الـ Unique Constraint على الـ Column لو الـ data اللى فيه Unique فعلاً هيعمل Non Clustered Index)

(الـ constraint بتتطبق على الـ data القديمة والجديدة\ Rule على الجديد بس "من بعد إنشاؤها")

*** Columns that used as index :**

(متفقين إن الـ Index فايدته إنه يوصل لـ data أسرع)

(طب على أى أساس بختار الـ column)

(بيتم إختيار الـ columns عن طريق أن أحدد مدى إستخدامه فى الـ Select Statement)

(طبعاً إنت مش هتقعد تراقب الـ Queries فى الـ Server عندك)

(لكن فيه طريقة وهيا إنك تستخدم Tools تساعدك فى الـ purpose دا وهما Two Tools
1) الـ SQL Server Profiler

2) الـ SQL Server Tuning Advisor)

(بتفتح الـ SQL Server Profiler وتعمل New Trace File و تختار مكان يتم الـ File Saving

بعد كدا بتبقى تسيبه شغال فى الـ background لمدة زمنية بحيث يبقى تم عمل مجموعة كبيرة من الـ queries)

(بتفتح الـ SQL Server Tuning Advisor وتفتح الـ Trace File وتختار الـ Database اللى انت عاوزه يعمل Analysis عليها وبناء عليه بيتم عمل Recommendation لـ columns اللى المفروض تبقى indexes عشان بتستخدم كتير فى الـ select Queries خلال الفترة الزمنية اللى الـ profiler كان شغال فيها وحددها)

(

*** 4 main Database :**

فى الـ Server هتلاقى 4 Databases دول by default غير الـ Created Database
1) Master DB

2) Model DB

3) ms DB

4) Temp DB

1) الـ Master DB :

- بيبقى فيها كل الـ Server Configuration (Meta data)

(User Names \ Permissions \ Passwords \ DB Names)

- لو مش موجودة مش هنقدر نـ Connect على الـ Engine

---

2) الـ Model DB :

- عبارة عن Template لـ Databases على الـ Server

- بيبقى فيها Built in functions و users وحاجات كتير فى الـ DB

- لما بعمل New Database بيتم أخذ Image من الـ Model DB واضيف الجديد بقا اللى محتاجه

- ممكن أحط UDF او User فى الـ Model DB وبالتالى مع كل New Database هتلاقى الإضافة اللى انت عاملها فى الـ Model DB

- بتتطبق على الـ New DB اللى بعد الإضافة لكن اللى قبل كدا واخدين الـ Image من الـ Model بدون الإضافة

---

3) الـ Ms DB (Management Studio DB) :

- المسئولة عن متابعة الـ Jobs فى الـ Server

- الـ Job هو عبارة عن Query with alert

---

4) الـ Temp DB :

- هى عبارة عن Temporary Database (بتستخدمها فى حالة إنت عاوز تـ Create حاجة مؤقتة)

- بيجى معاها الـ Local Table والـ Global Table

(معروف إن الـ physical Table دا بيبقى موجود على الـ Hard disk)

(طلع فكرة إن أستخدم Table قريب منى فى الـ Queries يبقى فى الـ Memory وأعدل و أعمل كل اللى عاوزه وبعدين أعمله Save على الـ hard disk)

(أسرع طبعا من التعامل مع الـ Hard Disk على طول)

(الـ Local Table بيبدأ بـ # أما الـ Global Table بـ ##)

(الـ Local Table والـ Global Table نفس الإستخدام لكن الفرق إن

الـ Local بيبقى خاص باليوزر اللى عمله Creation بس وطبعا كل User يقدر يعمل نفس الـ table وهيبقوا فى الـ Temp DB مختلفين عشان كل Table ليه ID خاص بيه يدل على الـ User

الـ Global هو shared بين الـ Users وكل User يقدر يـ Edit ويستخدمه)

- حذف الـ Tables :

1) الـ physical Table : عن طريق الـ Drop Statement

2) الـ Local Table : عن طريق الـ Drop + إن لما الـ User يـ Close الـ Session

3) الـ Global Table : عن طريق الـ Drop + إن لما كل الـ Users يـ Close الـ Sessions

- Syntax

```
Create table # \ ## Table_Name
(
[Column_Name] DataType
)
```

- هيبقوا موجودين فى الـ Temp DB

---

* Pivot And Group :

- محتاج تعرف 5 keywords فى الـ Topic وهم

1) الـ Roll up

2) الـ Cube

3) الـ Grouping Sets

4) الـ Pivot

5) الـ Unpivot

- فى العادى لما كنت بحب أعمل Aggregate Function مع Colum

(كنت لازم أستخدم الـ Group By بالـ Column)

```
select Column1_Name ,sum(Column2_Name)
from Table_Name
group by Column1_Name
```

- الـ Previous Query هيقسم الـ table لـ Groups بإستخدام الـ Column1 ويحسب الـ Aggregate Function لكل Group

- لو عاوز أجيب الـ Sum لكل الـ Table مع الـ Sum لكل Group (ناتج الـ Query اللى فات) (Total sum query مع الـ Union هعمل)

```
select Column1_Name,sum(Column2_Name)
from Table_Name
group by Column1_Name
union
select 'Total Values',sum(Column2_Name)
from Table_Name
```

---

1) الـ Rollup :

- بتستخدم مع الـ group by clause

- بتحسب الـ Subtotal لكل Group والـ Grand total لـ Tableكله

(هتقسم الـ Table لـ Groups وكل Group تحسب الـ Aggregate Function + الـ Subtotal لكل Group "لو معمول Group by بـ N Columns بيحسب لأول Column 1 بس" + الـ grand Total لكل الـ Table)

```
SELECT
    Region,
    Country,
    Product,
    SUM(Amount) AS TotalAmount
FROM
    Sales
```

```sql
GROUP BY
    ROLLUP (Region, Country, Product);
```

Result Set :

Region	Country	Product	TotalAmount	
North	USA	WidgetA	1000	
North	USA	WidgetB	1500	
North	Canada	WidgetA	800	
North	Canada	WidgetB	1200	
North	NULL	NULL	3500	-- Subtotal for North
South	Brazil	WidgetA	600	
South	Brazil	WidgetB	900	
South	Argentina	WidgetA	400	
South	Argentina	WidgetB	600	
South	NULL	NULL	2500	-- Subtotal for South
NULL	NULL	NULL	6000	-- Grand Total

---

2) الـ Cube :

- بتستخدم مع الـ group by clause

- بتحسب الـ Subtotal لكل الإحتمالات الممكنة لـ Group والـ Grand total لـ Tableكله

(هتقسم الـ Table لـ Groups وكل Group تحسب الـ Aggregate Function + الـ Subtotal لكل Group "لو معمول Group by بـ N Columns بيحسب لأول Column 1 بس" + الـ grand Total لكل الـ Table)

```sql
SELECT
    Region,
    Product,
    Year,
    SUM(Amount) AS TotalAmount
FROM
    Sales
GROUP BY CUBE (Region, Product, Year);
```

This query will produce a result set with subtotals and grand totals for all possible combinations of Region, Product, and Year. The result set will include:

Subtotals for each unique Region
Subtotals for each unique Product
Subtotals for each unique Year
Subtotals for each combination of Region and Product
Subtotals for each combination of Region and Year
Subtotals for each combination of Product and Year
Grand total for the entire dataset

---

3) الـ Grouping Sets :

- بتستخدم مع الـ group by clause

- بتعمل Multiple Grouping فى Single query

```
SELECT
    Column1,
    Column2,
    SUM(Amount) AS TotalAmount
FROM
    YourTable
GROUP BY
    GROUPING SETS (
        (Column1),
        (Column2),
        (Column1, Column2),
        ()
    );
```

- الـ result :

الـ First Group Set بيجيب الـ Total amount لكل distinct value فى Column 1

الـ Second Group Set بيجيب الـ Total amount لكل distinct value فى Column 2

الـ Third Group Set بيجيب الـ Total amount لكل distinct value فى الـ Combination بين Column 1 و Column 2

الـ Forth Group Set بيجيب الـ Total amount لـ Table كله (Grand Total)

---

* الـ Pivot Table :

- هى طريقة لتحويل الـ Table Format بحيث إنه يظهرلى As a Matrix

(بحيث الـ Rows تبقى Columns والعكس)

```
SELECT *
FROM  YourTable
PIVOT
(
Aggregate_Function(Column_to_be_aggregated)
FOR pivot_Column IN ([Val1],[Val2])
) as Alias_Name
```

- عشان أرجع الـ Formatلأصله بستخدم العملية العكسية الـ Unpivot

```
SELECT *
FROM  YourTable
UNPIVOT
(
Column_to_be_aggregated
FOR unpivot_Column IN ([Val1],[Val2])
) as Alias_Name
```

*** Views :**

- الـ View عبارة عن Container لـ Select Statement

(يعتبر زى الـ Function اللى جواها Select statement الخاص بـ Business Logic)

- لما بكون فى الـ APP وأضغط على Button معين المفروض بيعمل query اللى بينفذ المطلوب

(طب ما كدا الـ Metadata مكشوفة ومفيش أى نوع من الـ Security على الـ Data عندى)

(الـ View بتـ hide metadata عن الـ Application

"إن أجيب الـ Data اللى عاوزها لكن بإستخدام إسم الـ View بدون ما أكتب الـ Meta data ")

(الـ performance زى ما هو مفيش فرق هو بينفذ الـ query برضو)

(اللى إستفدت بيه من الـ View هو الـ Database object Hiding عشان يـ Improve security)

- الـ View :

<u>Only Select Statement</u>

<u>No DML inside View</u>

<u>No Parameters</u>

*** Types of View (Microsoft) :**

1) Standard View :

(Virtual Table \ One server)

2) Partitioned View

(Virtual Table \ Different Servers)

- بيجمع Data من Tables ليها نفس الـ Structure فى One logical View (شبيهه بالـ Union)

3) Indexed View

(View has Data \ Improve Performance)

- بستخدم مع الـ creation الـ Keyword (With Schemabinding)

*** View Syntax :**

```
CREATE VIEW [schema_name.]view_name (column_name [,...n])
AS
    SELECT select_statement
    FROM table_name
    WHERE condition;
```

- ممكن أستخدم أى حاجة ممكنة مع الـ select (Join مثلا)

- ممكن أستخدم Stored procedure تعرفنى معلومات عن الكود اللى إتكتب بيه الـ db Object

وهو الـ Sp_helptext 'View_Name' (ممكن أعرف الـ Metadata من الـ Code طب فين الـ Security ؟؟؟)

(عشان أحقق الـ security أستخدم With Encryption قبل الـ As)

(هيـ Encrypt الـ View Code)

```
CREATE VIEW [schema_name.]view_name (column_name [,...n])
With Encryption
AS
    SELECT select_statement
    FROM table_name
    WHERE condition;
```

*** DML On Views :**

- ممكن أستخدم الـ view فى الـ DML Queries بس بشروط
(بس هيتعامل مع الـ Original Table)

```
Insert into View_Name
Values (Val1,Val2,.....)
```

- الشروط : إن باقى الـ Columns تكون بتـ allow حاجة من الـ 4 حاجات :

1) NULL

2) Default Value

3) Driven

4) Identity

- لو الـ View بيستخدم Multiple Tables فى الحالة دى :

(الـ Delete مش مسموح

الـ insert والـ update مسموحين بس بشرط التأثير فى table واحد فى نفس الـ query)

- ممكن أستخدم الـ view يـ Insert أى value فى الـ Table مدام محقق الشرط

(طب لو عاوز الـ View يـ Insert فى الـ View Range بس هستخدم الـ With Check Option
عشان يشوف الـ Values اللى بتحقق الـ Condition بس)

```
CREATE VIEW [schema_name.]view_name (column_name [,...n])
With Check option
AS
    SELECT select_statement
    FROM table_name
    WHERE condition;
```

*** Merge Statement (DML) :**

- لو عاوز أعمل Compare بين Two tables

(بحيث مثلا فى حالة القيم المتشابهة فى الـ Two tables هاخد Action معين

و القيم اللى موجودة فى Table ومش موجودة فى التانى هاخد Action معين)

- الـ two tables عبارة عن

1) Target Table

2) Source Table

- ممكن الـ source يبقى عبارة عن SubQuery

- ممكن أستخدم كذا Source

- Merge Syntax :

```
MERGE TargetTable AS target
USING SourceTable AS source
ON target.ID = source.ID
WHEN MATCHED THEN
    UPDATE SET
        target.ColumnName1 = source.ColumnName1,
        target.ColumnName2 = source.ColumnName2
WHEN NOT MATCHED BY TARGET THEN
    INSERT (ID, ColumnName1, ColumnName2)
    VALUES (source.ID, source.ColumnName1,
source.ColumnName2)
WHEN NOT MATCHED BY SOURCE THEN
    DELETE;
```

- فى حالة الـ Matched (اللى موجود فى الـ Two tables)

- فى حالة الـ Not Matched by Target (اللى موجود فى الـ source ومش موجود فى الـ Target)

- فى حالة الـ Not Matched by Source (اللى موجود فى الـ Target ومش موجود فى الـ Source)

- أخره Semicolon عشان دا كله عبارة عن one Query