TP2 SQL Server

LAHMAM SalahEddine

Lebbar Hamza

Ils sont créés par l'utilisateur et utilisés par le système. Leur rôle est d'améliorer les performances d'accès en se basant sur un arbre physique. Généralement, on crée un index unique sur les clés primaires, et un index dupliqué sur les clés étrangères.

1) Les champs des trois tables qui méritent des index sont les clés primaires et les clés étrangères.

Ainsi, nous créons des index pour les tables suivantes :

- **Etudiants : Code**
- **Matières : Code**
- **Examens : Code, Code_Etud, Code_Mat**

Dans la fenêtre « Explorer » :

dbo.Table => Indexes => New Index => Add "case unique" in Index Key Columns

2) "Execution Plan"

Select * From Examens

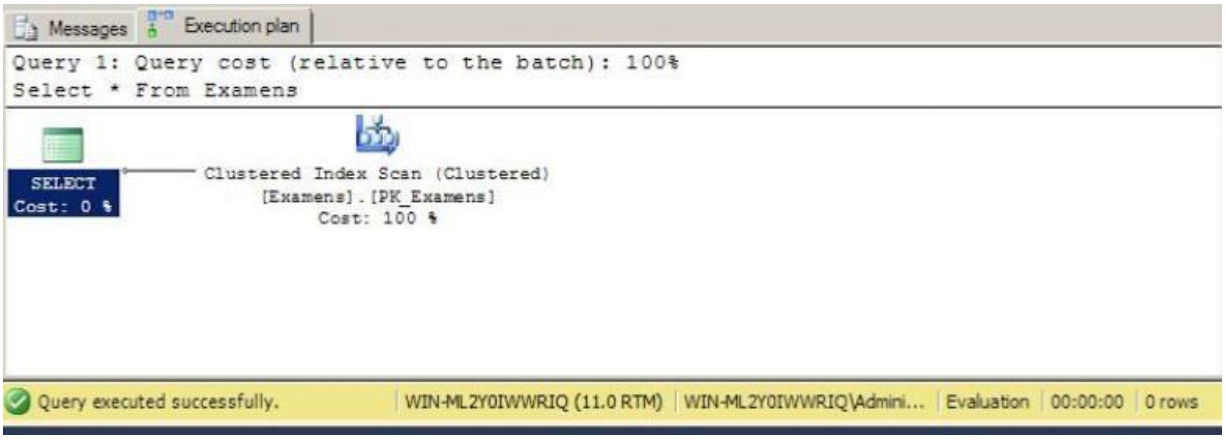

Figure 1 : Plan d'exécution de la requête « Select * From Examens »

Select * From Examens Where Code < 10

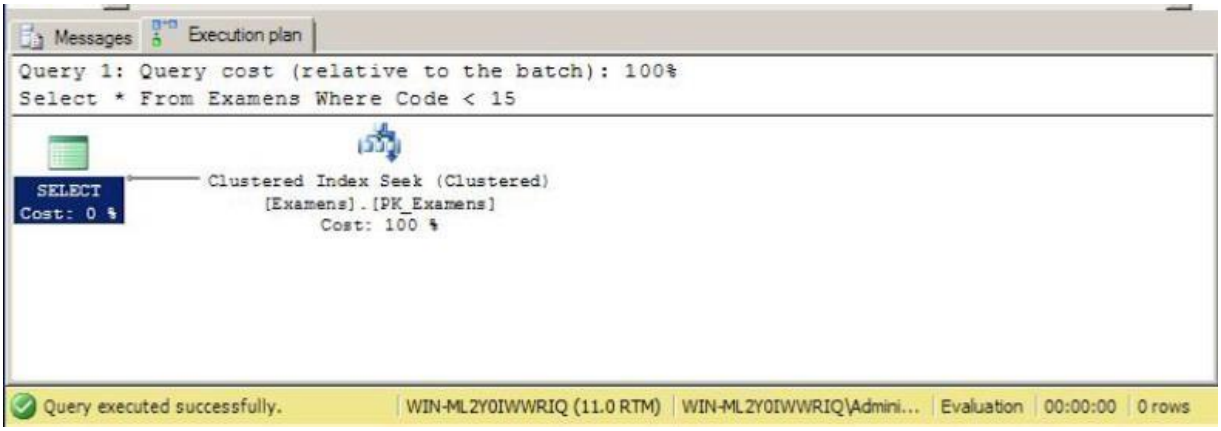

Figure 2 : Plan d'exécution de la requête « Select * From Examens Where Code < 10 »

Select Et.Code, Ex.Code, Nom From Etudiants Et, Examens Ex Where Et.Code = Code_Etud

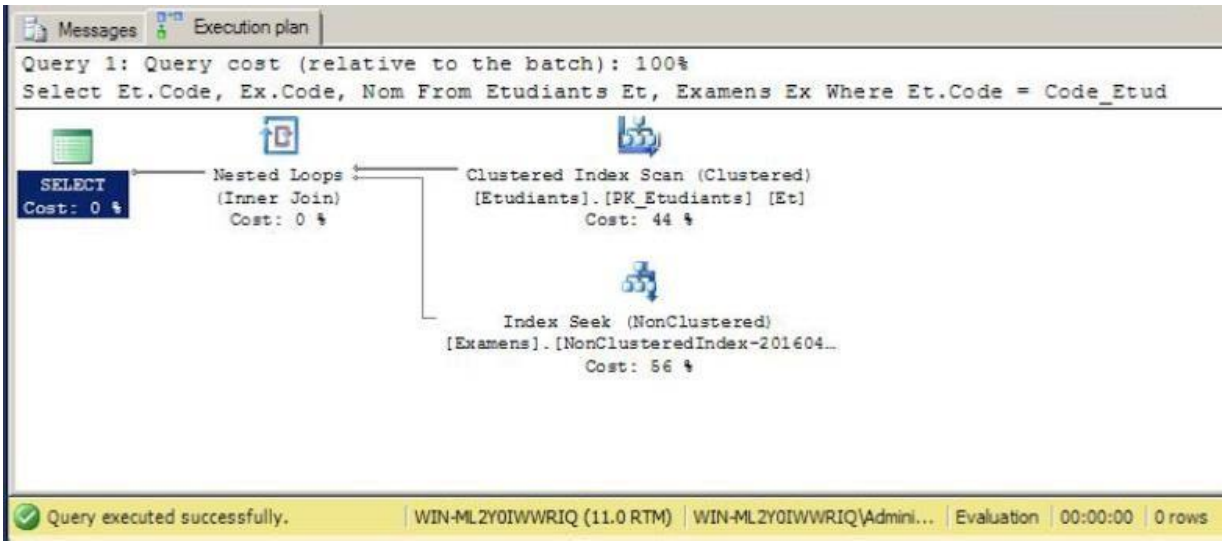

Figure 3 : Plan d'exécution de la requête « Select Et.Code, Ex.Code, Nom From Etudiants Et, Examens Ex Where Et.Code = Code_Etud »

3)

Select Ex.Code_Etud as Code_Etud ,sum(Note) as Somme, sum(Note*Coef) as Somme

From Matières Mat, Examens Ex Where Ex.Code_Mat = Mat.Code Group By Ex.Code_Etud;

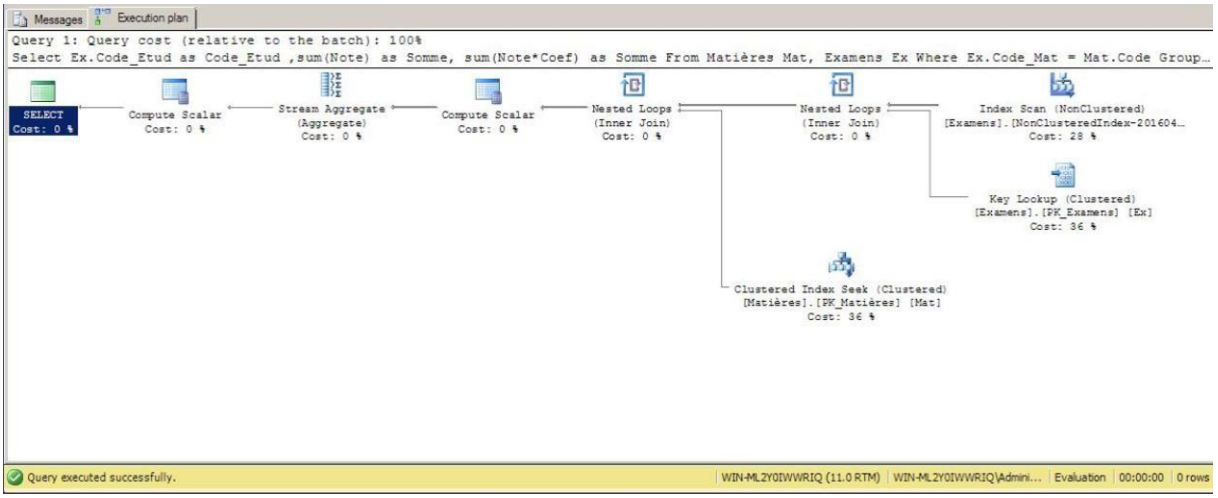

Figure 4 : Plan d'exécution d'un exemple de requête complexe.

Les vues permettent la perception de données d'une ou plusieurs tables. Son intérêt est, principalement, la définition des droits d'accès.

4)

Les requêtes

Create View V1 as Select * From Examens Where Note < 12

Select nom From V1, Etudiants Et Where Et.Code = V1.Code_Etud and nom like 'B%' and Note < 12 Order By Nom, Note desc

Insert Into V1(Code_Etud,Code_Mat, Date_Ex, Note) Values(1,'M2.1',getdate()-1,10) => Inserted

Insert Into V1(Code_Etud,Code_Mat, Date_Ex, Note) Values(1,'M2.1',getdate()-1,14) => Inserted

Commentaire :

La dernière insertion a été effectuée au niveau de la table "Examens" mais pas au niveau de la vue

5)

Alter View V1 as Select * From Examens Where Note < 12 With Check Option

Insert Into V1(Code_Etud,Code_Mat, Date_Ex, Note) Values(1,'M2.1',getdate()-1,8) => Inserted

Insert Into V1(Code_Etud,Code_Mat, Date_Ex, Note) Values(1,'M2.1',getdate()-1,19) => Non inserted

6)

Create View V2 as Select Et.Code, Et.Nom as EtNom, Mat.Nom as MatNom, Coef, Date_Ex, Note From Etudiants Et, Matières Mat, Examens Ex Where Ex.Code_Etud = Et.Code and Ex.Code_Mat = Mat.Code

Select * From V2

Non, parce que c'est une vue multi-tables.

Remarque : Les champs non vus par la vue doivent avoir une valeur (par défaut ou null), sinon on ne peut pas faire une insertion.

Alter View V2 With Encryption as Select Et.Code, Et.Nom as EtNom, Mat.Nom as MatNom, Coef, Date_Ex, Note From Etudiants Et, Matières Mat, Examens Ex Where Ex.Code_Etud = Et.Code and Ex.Code_Mat = Mat.Code

sp_helptext V2 => "The text for object 'V2' is encrypted."

Alter View V2 as Select Et.Code, Et.Nom as EtNom, Mat.Nom as MatNom, Coef, Date_Ex, Note From Etudiants Et, Matières Mat, Examens Ex Where Ex.Code_Etud = Et.Code and Ex.Code_Mat = Mat.Code

sp_helptext V2 => Affichage de la requête de création non cryptée.

PS1 à créer par le générateur : Programmability => Stored Procedures => New Stored Procedure, sans paramètres

PS2 :

Create Procedure PS2

-> paramètres

AS

-> déclaration des variables

Begin

-> select, ...

end

Exécution => Création de la procédure

Pour exécuter une procédure : EXEC nomProc

8)

```
CREATE PROCEDURE PS1
AS
BEGIN
    SET NOCOUNT ON;
    SELECT Et.Code, count(Ex.Code) as NbrExamens, min(Note) as Min, max(Note) as Max, sum(Note)/count(Ex.Code) as Moyenne From Etudiants Et, Examens Ex Where Et.Code = Ex.Code_Etud Group By Et.Code;
END
GO
```

```
Exec PS1
```

9)

```
CREATE PROCEDURE PS2
AS
BEGIN
    SET NOCOUNT ON;
    SELECT Et.Code, count(Ex.Code) as NbrExamens, min(Note) as Min,
max(Note) as Max, sum(Note)/count(Ex.Code) as Moyenne From Etudiants Et, Examens
Ex Where Et.Code = Ex.Code_Etud Group By Et.Code Having min(Note) >= 10;
END
GO
```

10)

```
ALTER PROCEDURE PS2
@notePar decimal(4,2)
AS
BEGIN
    SET NOCOUNT ON;
    SELECT Et.Code, count(Ex.Code) as NbrExamens, min(Note) as Min,
max(Note) as Max, sum(Note)/count(Ex.Code) as Moyenne From Etudiants Et, Examens
Ex Where Et.Code = Ex.Code_Etud Group By Et.Code Having min(Note) >=
@notePar;
END
GO
EXEC PS2 16
```

11)

```
ALTER PROCEDURE PS2
@notePar decimal(4,2), @codePar int
AS
BEGIN
    SET NOCOUNT ON;
    SELECT Et.Code, count(Ex.Code) as NbrExamens, min(Note) as Min,
max(Note) as Max, sum(Note)/count(Ex.Code) as Moyenne From Etudiants Et, Examens
Ex Where Et.Code = Ex.Code_Etud and Et.Code = @codePar Group By Et.Code
Having min(Note) >= @notePar;
END GO
EXEC PS2 16, 101
```

Ils se basent sur le principe de l'Action/Réaction

12)

On va rendre le triplet (Code,Code_Etud, Code_Mat) une clé primaire

13)

Dans la fenêtre « Explorer » :

Databases -> Evaluation -> Tables -> New Table

Une nouvelle fenêtre apparaît.

On saisit les noms des colonnes avec leurs types. Au moment de sauvegarde, on nomme la table.

Ensuite :

Databases -> Evaluation -> Tables -> Absence -> Code_Etud -> Modify

Une nouvelle fenêtre apparaît. Dans la barre en haut à gauche, on clique sur la cellule correspondant à notre clé (Les trois champs)

Dans la fenêtre « Explorer » :

Databases -> Evaluation -> Tables -> Abscence -> Keys -> New Foreign Key

Une nouvelle fenêtre apparaît. Dans le volet :

- **« Général », on clique sur « Tables And Columns Specification » et une nouvelle fenêtre s'ouvre dans laquelle on va définir la clé étrangère et la clé primaire puis on clique sur « OK ».**
- **« Identity », on entre le nom « FK_Absence_Etudiant »**
- **« Table Designer », on clique sur « INSERT and UPDATE Specification ». Ensuite, Update Rule -> Cascade et Delete Rule -> Cascade.**

On reprend les mêmes étapes de la création de la clé étrangère pour Code_Mat.

14)

ALTER TABLE Examens ADD NoteAbs decimal(4,2)

15)

```
CREATE TRIGGER T1
  ON Examen AFTER INSERT

AS DECLARE
@CodeEx int, @NoteEx decimal(4,2) BEGINSET NOCOUNT ON;

 select @CodeEx= Code from
 inserted; select @NoteEx= Note
 from inserted;

 update Examen set NoteAbs=@NoteEx where Code=@CodeEx;

END GO
```

16)

```
CREATE TRIGGER T2
  ON Absence AFTER INSERT
AS DECLARE

@CodeE
td int,
@Code
Mat
char(5)
BEGIN
SET NOCOUNT ON;

 select @CodeEtd= Code_Etud from
 inserted; select @CodeMat= Code_Mat
 from inserted; update Examen set
 noteabs=noteabs*0.9;
 END GO
```