SQL SERVER 2012 FULL REVISION

By: asem asker

2016

محاضـرة رقم 1

اي System بعمله بيحتوي علي اكثــر من 75% بيكون Data Based

لما يطلب مني عمل System معين :

- **اول حاجة بعملها هي الـ System Analyst وفي الخطوه دي بحاول اجمع معلومات عن الـ System المطلوب وشكلة وبعمل في الاخـر حاجة اسمها Requirement Document.**
- **ثم بعد ذلك بحول الـ Requirement Document الي ERD وده بيوصف العلاقات بين المكونات بتاعتي.**
- **ومن خلال الـ ERD بعمل Mapping فبطلع حاجة اسمها DB Schema.**
- **بعد كده بنرجع للـ System Analyst ونخليه يشوف الـ Tables اللي احنا هنعملها عشان نشوف شغلنا ماشي صح ام لا.**
- **وفي الاخر بنعمل Implementation للـ Data مثل RDBMS(Relational DB Management System) وهنستخدم هنا الــ SQL (Structured Query Language)**

ملحوظة: الـ Ansi SQL (American National Standards Institute)هو الـ Standard او الـ Basic اللي بنكتب بيه الـ DB Code ولكن:

- **Microsoft اخذت الـ Ansi SQL وطلعت منه Transaction SQL (T-SQL).**
- **Oracle اخذت الـ Ansi SQL وطلعت منه الـ Procedural Language (PL-SQL).**
- **IBM اخذت الـ Ansi SQL وطلعت منه الـ (IBM-Procedure Language).**

هنتكلم طبعاً عن الـ T-SQL ودي بيكون لها 5 انواع مختلفين:

1- **الــ DDL: هي اختصار لـ (Data Definition Language) ودي فيها بعمل Create Table او Create Trigger.**
2- **الــ DML: هي اختصار لـ (Data Manipulation Language) عايز اعمل Edit للـData بتاعتي واتعامل معاها عن طريق Insert, Update, Delete.**
3- **الــ DCL: هي اختصار لـ (Data Control Language) ودي بتكون خاصة بالـ Administrator وبيكون فيها**
   a. **Grant: الـ Admin يعطي صلاحية معينة لـ User معين.**
   b. **Revoke: الـ Admin يمنع صلاحية من شخص وفيها معرفش ايه بالضبط اللي مش مسموح له.**
   c. **Deny: يمنع عنه تماما الـ Select مثلا حتي لو كان واقع في Group مسموح له بصلاحية معينه.**
4- **الــ DQL: هي اختصار لـ (Data Query Language) هي كل جمل الـ Select لانه بترجعلي Data.**
5- **الــ TCL: هي اختصار لـ (Transaction Control Language) هي Commands اللي بستخدمها عشان اتحكم في الـ Transactions اللي بتتم علي الـ DB ودي بيكون فيها الـ Commit والـ Rollback.**

–

Entity Relationship Diagram (ERD)

- **الـ Entity: اي حاجة موجوده في الـ System محتاج اخزن عنها شوية معلومات وانواعها هي:**
  1. **Strong Entity**
  2. **Weak Entity وهي Entity وجودها بيعتمد علي وجود Entity اخري.**
- **كل Entity بيكون لها مجموعه من الـ Attributes**
- **انواع الـ Attributes:**
  1. **Simple: بيمثل عمود واحد في الـ DB.**
  2. **Composite: لو الـ Attribute بتاعي مقسم الي اكثر من حاجة (مثلا: الـ Phone وال، ZIP).**
  3. **Multi Valued: الـ Phone بيكون له اكثر من قيمه بحيث اني هخزن اكثر من قيمه (Value).**
  4. **Derived: يعني Attribute هيتحسب من خلال مجموعه من الـ Attributes الأخري.**
  5. **Complex: بيكون نوعين فمثلا لو عندي Multi Valued Attribute نوعه Composite ده يعتبر Complex.**
- **عندنا نوعين من الـ Keys مهمين جـــداً:**
  1. **الـ Primary Key:**
     - **بيكون Not NULL.**
     - **ولازم يكون Unique.**
  2. **الـ Foreign Key:**
     - **هو Primary key ولكن في Table اخر.**
     - **ممكن يكون NULL.**
     - **مش شرط يكون Unique يعني ممكن يتكرر.**
     - **لازم الـ Value بتاع الـ F.K بتكون بتعمل Map مع الـ Value في الـ P.K**
- **انواع الـ Relations في الـ DB:**
  1. **Unary Relation: هي علاقة بين الجدول ونفسه.**
  2. **Binary Relation: هي علاقة بين 2 Entities.**
  3. **Ternary Relation: هي علاقة بين 3 Entities.**
- **الـ Participation بيكون لها نوعين وكل منهم بيعتمد تحديده علي حسب الـ System بتاعي :**
  1. **يعني اقول (Must) Mandatory يعني لازم ولّا لا.**
  2. **او اقول (May) Optional يعني اختياري (مش لازم).**
- **الـ Cardinality: هي الـ Uniqueness (التميّز) لقيم الـ Data اللي بتكون موجوده في ال Columns واحيانا بتـ Refer الي الـ Relationship بين الـ Tables وبعضها ولها 3 انواع هي:**
  1. **One to One: مثلا الشخص وبطاقته.**
  2. **One to Many: مثلا الـ Dept بيكون فيه اكثر من Instructor والـ Instructor بيكون موجود في قسم واحد فقط.**
  3. **Many to Many: الـ Student بيكون له اكثـــر من Course والـ Course ده بياخذه اكثـر من Student.**

- **الخطوات اللي بفكر فيها وانا بعمل الـ Mapping:**
  1. لو عندي Multi Valued Attribute هيكون عندي Table جديد.
  2. لو عندي Complex Attribute هيتحول الي جدول جديد.
  3. كل Entity عندي هحولها الي Table.
  4. بعد كده اشوف الـ Cardinality:
     - Many-to-Many: هنعمل Table جديد هنضع فيه الـ P.K بتاع الجدول الاول والجدول التاني.
     - One-to-Many: هناخذ الـ P.K بتاع الـ One وهنضعه في الـ Many.
     - One-to-One: بنشوف الـ Tables اللي فيه Records كتير ناخذ الـ P.K بتاعه ونضعه في الـ Table اللي الـ Records بتاعته قليلة.

ملحوظة:

- **الـ Space Parsing:** هي feature عملتها Microsoft مؤخرا لمنع وجود الـ Nulls واخذ مساحة.

SQL SERVER 2012

- هنلاحظ ان الـ SQL اصبح احسن حاليا في اني استخدمه بدلا من الـ Oracle.
- الـ SQL Diagram (او الـ DB Diagram): هو Diagram موجود في الـ SQL ودائماً علاقته بتكون One-to-Many يعني في النهاية هو عبارة مجموعة من الـ Relations كلها One-to-Many.
- **اهم Topics كانت موجوده في الـ SQL server 2005:**
  1. اهم حاجة كانت موجوده هي الـ Integration Service ودي بتساعدني اني اعمل Integration لمجموعة من الـ Files واحولها الي شكل واحد.
  2. Microsoft جعلت الـ SQL نفسه يطلع الـ Data في شكل XML تشتغل علي اي حاجة.
  3. ايضا الـ CLR ممكن يكون عندي Function صعبة مثلا الـ Function دي بتعمل Encryption لـ Data فبذلك هعمل الـ Function في الـ C# وهعملها Deploy علي الـ SQL. فهو Component بينزل مع الـ Visual Studio.
- **اهم Topics موجوده في الـ SQL server 2012:**
  1. الـ Always on: يوجد حاجة اسمها الـ High Availability موجوده في الـ DB فبدور علي حاجة تعمل Maintain للـ High Availability يعني اجعل الـ Website بتاعي شغال دائما فالـ Always on واحده من الحاجات اللي بتعمل كده.
  2. الـ Profile: هي tool منفصله من الـ SQL بعمل لها Run فطول ماهي Run فهو بيخزن كل حاجة بعملها علي الـ SQL ويحولها الي Code بيخزنها عنده ثم يرسلها للـ Tuning Advisor.
  3. الـ Tuning Advisor: بتعطي Recommendation علي الـ SQL عشان ازود الـ Performance بتاعها اي بعطي مجموعة Indexes باخذها واعمل لها Run علي الـ SQL عشان ازود الـ Performance.

- **عندنا 4 tools بستخدمهم عشان اعمل Run للـ Queries:**
  1. **SQL Server Management Studio ودي تعتبر الـ Interface اللي بستخدمه عشان اشتغل عليه.**
  2. **XML**
  3. **SQL CMD**
  4. **PowerShell وفيها بروح لشاشة الـ cmd.**
- **مفهوم الـ DB Object: تُطلق علي كل الـ Creations اللي بعملها في الـ DB وبستطيع اعمل Manage ليهم كلهم فعلي سبيل المثال**
  1. **Create Table**
  2. **بعمل Constrain مثلا علي Column معين**
  3. **Create View**

ملحوظة: لما باجي اعمل Create لـــ Login بيكون علي مستوي الـ Server اما لما اعمل Create لـ Table بيكون علي مستوي الـ DB.

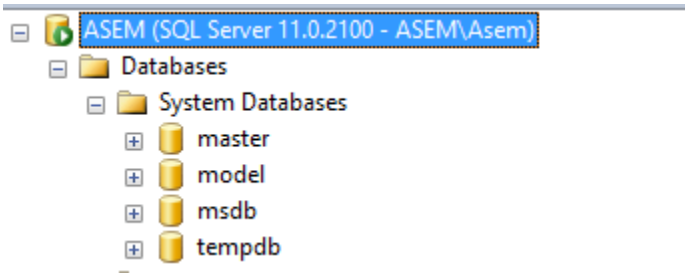
- **مفهوم الـــ Authentication في الـ SQL عندي يعني الشخص له Username وPassword وعندنا نوعين:**
  1. **Windows Authentication: يعني الشخص اللي قدر يدخل علي الـ Windows يقدر يدخل علي الـ SQL Server وهيكون له نفس الـ Username وPassword بتوع الـ Windows.**
  2. **SQL Authentication: بيكون موجود مجموعة من الـ Username وPassword داخل الـ SQL نفسه.**
- **مفهوم الـ Authorization في الـSQL يعني الشخص ده يقدر يعمل ايه علي الـ DB ايه الـ Limits اللي عليه وبتُجيب علي سؤال "What to do".**
- **الـ File Stream: في عملية التنصيب للبرنامج الـ SQL Server بنعملها Enable ودي بتمكني اني لو عندي فيديو عايز استخدمه في الـ DB.**
- **الـ Backup DB: بستخدمه لو عايز استرجع الـ Data بتاعتي.**
- **الـ SQL Server فيها:**
  1. **الـ Server Type: بحدد الـ Service اللي هعمل Connect عليها وهستخدمها.**
  2. **الـ Server Name: سواء كان Local او اسم الجهاز او الـ IP هقدر اعمل Connect من خلالهم.**
  3. **الـ Authentication Mode: بحدد الـ Mode اللي هستخدمه سواء كان Windows Authentication او SQL Server Authentication.**
- **ملحوظة: لو عايز اغير من الـ Windows Authentication Mode الي الـ SQL Server Authentication Mode**
  - ***SQL Server → Properties → Security → choose the Mode***

انواع الـ Database

من انواع الـ DB اللي موجوده عندى :

- الـ System DB
- الـ Snapshot DB
- الـ Report DB
- الـ Sample DB
- الـ User Defined DB

الـ System DB: ودي بتنزل معايا علي الجهاز وهي 4 انواع كالاتي:



1. **الـ Master: فيها كل الـ Configuration اللي بتساعد الـ SQL Server انه يشتغل وكذلك الـ Users.**
2. **الـ Model: هي الـ DB الفاضيه اللي كل حد هيعمل DB جديدة بياخذ Image منها.**
3. **الـ msdb: هي الـ DB المسئوله انها تعمل Run للـ Jobs الموجوده علي الـ SQL Server.**
4. **الـ tempdb: بتعمل حاجات معينه في الـ Runtime فهي بتساعد الـ Users لما يعملوا حاجة معينه في الـDB ومحتاج الغيها بعد فتره معينه.**

ملحوظة:

- **الـMaster لو مش موجوده مش هيكون فيه DB اصلا.**
- **الـ Job: هي Query بعملها Save في الـ SQL واحدد له Runtime معين بعمل Run للـ Job في الوقت اللي انا محدده.**

الـ Snapshot DB: دي مش بيحصل عليها insert, update, delete هي بيحصل عليها select فقط مثلا بكون عايز شكل الـ DB بتاعتي في وقت معين فعمل Snapshot من الـ DB في الوقت اللي انا عايزه.

الـ Report DB: دي الـ DB اللي بخزن فيها الـ Reports والـ Queries بتاع الـ DB.

الـ Sample DB: دي DB تعتبر حاجة Microsoft عملتها كـ Sample نجرب شغلنا عليها.

الـ User Defined DB: دي اللي انا كـ User بعملها.

لما احتاج اعمل Queries:

- احدد الـ DB اللي هشتغل عليها باستخدام Use ITI يعني بستخدم Use قبل اسم الـ DB.
- عشان اعمل comment لـ One Line هنستخدم --
- عشان اعمل Comment لاكثـــر من Line هنستخدم /* */

الـ DDL

لما اجي اعمل Create لـ Table شكل الكتابه هيكون كالاتي:

مع ملاحظة ان اي حاجة فيها create بيكون فيها Alter, Drop.

1. الـ Alter والـ Drop بيغيروا في الـ Structure بتاع الـ Table مش في الـ Data.

```
create table Employee
(
id int identity(1,1) primary key,
name nvarchar(50),
sal int default 1000
)
```

2. انا لو عايز اعمل Drop او Alter لـ Column مثلاً لازم نكتب سطر الـ Alter الاول كما هو موضح في الامثلة:

```
alter table Employee
drop column name

alter table Employee
add name nvarchar(50)

alter table Employee
ALTER COLUMN name NVARCHAR(60)

drop table Employee
```

الـ DML

عندنا اكثــر من شكل في الـ DML:(Insert, Update, Delete)

اولا: الـ Insert

- Insert
- Simple Insert
- Insert Constructor
- Insert based on Select
- Insert based on execute

1- الـ Simple Insert:

```
Insert into Student(St_Id,St_Fname)
Values(130,'Asem')
```

```
Insert into Department
values(100,'DBA','database Administrator','cairo',107)
```

ملحوظة: لو فيه Column مش هحطه في الـ Insert لازم يكون فيه 5 حاجات كالاتي:

- NULL Value
- Default Value
- Calculated
- Identity
- Sequence

```
--using null and default in insert

Insert into Department
values(100,'DBA',null,'cairo',default)
```

2- الـ Insert Constructor: يعني ممكن اضع اكثــر من Value في نفس الســطر.

```
--Insert data using the new row constructors
INSERT INTO PayRate
VALUES (1, 40.00, 5),
       (2, 45.50, 4),
       (3, 39.50, 6);
```

3- الـ Insert based on Select:

```
--Insert based on select

insert into test
select st_id,st_fname from Student
where St_Address='cairo'
```

4- الـ Insert based on Execute: دي بتعتمد علي وجود Stored Procedure

```
Insert into Student
execute sp1
```

ثانيا: الـ Update

```
--Update Statment
update Department
set Dept_Location = 'alex'
where Dept_Name='erp'
```

```
--using null and default with update
update Department
set Dept_Location = DEFAULT
where Dept_Name='erp'
```

```
--update with joins
update stud_course
set grade +=10
from stud_course sc,course c
where sc.crs_id=c.crs_id
      and top_id=(select top_id
                  from topic
                  where top_name='programming')
                  Update Course
```

ثالثا: الـ Delete

```
--Delete Statment
Delete from Student
where St_Id=10
```

رابعا: الـ Truncate

```
--Truncate Statment
truncate table test
```

ملحوظة: الـ Shrink معناه اني بمسح كل الـ Spaces الناتجة عن الحذف.

ايه الفرق بين الـ Delete والـ Truncate والـ Drop؟

- **الـ Drop: بتمسح الـ Data والـ Structure بتاع الـ Table.**
- **الـDelete: بتمسح الـ Data اللي انا محددها فقط يعني جزء من الـ Data وبتترك الـ Structure وايضا الـ Delete مش بتمسح الـ Spaces يعني مش بتطبق مغهوم الـ Shrink.**
- **الـ Truncate: بتمسح كل الـ Data وايضا الـ Truncate بتمسح الـ Spaces يعني بتطبق مغهوم الـ Shrink والسبب ان الـ Truncate مش بتتخزن في الـ log file.**

ملحوظة: هنلاحظ ان الـTruncate اسرع في الـ Performance من الـ Delete لان الـ Delete بتتخزن في الـ Log File ولكن الـ Truncate مش بتتخزن في الـ Log File.

ملحوظة: مش اقدر اعمل Truncate لـ Table بيكون Referenced بـ Foreign Key Constraint.

الـ DQL

اللي بيعبر عن الـ DQL هي جملة الـ Select :

```
Select St_fname,st_lname from Student
Select * from student

--alias name and dealing with space in the column names
Select St_fname+' '+st_lname as [full name] from Student
Select * from student
```

```sql
--is null and IS NOT NULL with where clause
select * from Course
where Crs_Duration = null    --null is not a value

select * from Course
where Crs_Duration is not null
```

ISNULL (), COALESC (), NULLIF ()

ملحوظة: عندنا 3 من الـ Functions الجاهزة بتحل مشكلة القيم الـ Null وهم:

- **الـ ISNULL():** هنلاحظ في المثال القادم - هقوله ان اي قيمه بـ Null في الـ Column اللي اسمه st_age هخليه بـ 20.

```sql
select ISNULL(st_age,20) from Student
```

- **الـ COALESC():** يعني بشوف الـ Column الاول لو فيه Value هتتعرض ولو مش فيه Value هيدخل علي الـ Column التاني لو فيه Value هتتعرض ولو مش فيه هيدخل علي الـ Column الثالث ولو مفيش قيمة في الـ Column التالت علي اساس انه اخر واحد في الـ Expression بتاع المثال القادم ممكن نضع قيمه Default مثلا بـ Zero.
  - Queries where NULL values may exist and you wish to substitute one of several possibilities into a column of the result set.

```sql
SELECT COALESCE(hour_rate * 40,salary,Bonus )  AS 'Total Salary'
FROM instructor
```

- **الـ NULLIF():**
  - Queries that you want to offer a more meaningful value in place of the NULL keyword being displayed in the result.

```sql
select nullif(st_age,dept_id)
from student
where st_id=14
```

Operators >, <, (), or, and, not

```
Select St_fname,st_lname from Student
where St_Age=25 or St_age=30

--In Statment
Select St_fname,st_lname from Student
where St_Age in(25,26,27)

--select into
select st_fname,st_age
into test2
from Student

select st_fname,st_age
into test3
from Student
where 1=2
```

Like

```
--like
Select St_fname,st_lname from Student
where st_fname like 'A%'

Select St_fname,st_lname from Student
where st_fname like '_A%'

Select St_fname,st_lname from Student
where st_fname like '[_]A%'

Select St_fname,st_lname from Student
where st_fname like '___'

select st_fname
    from student
    where st_fname like '[a-h]%'
```

ملاحظات علي الـ Like:

'a%' ودي تعني اي كلمه تبدا بحرف a

'%a%' يعني اي حاجة تحتوي علي حرف الـ a

'a__' يعني اي اسم 3 حروف ويبدا بحرف الـ a

'a_%' كل الاسماء اللي علي الأقل حرفين وتبدا بحرف الـa

%[_] يعني الاسمـاء اللي تبدا بحرف الـ a

%[a-h] اي حاجة تبدا بحرف من الحروف من الـ a-h

%[^a-h] اي حاجة تبدا بحرف غير الحروف من الـ a-h

%124[124] اسم يبدا بـ 124 والرقم الرابع بيبدا باي حاجة اما 1 او 2 او 4

Joins

1- الـ Cross Join: هو نفس الـ Cartesian Product يعني حاصل ضرب الجدولين في بعض.

```
--Cross join or Cartsian product
select st_fname,dept_name
from student s cross join department d

--A Cartesian Product
select st_fname,dept_name
from student s , department d
```

2- الـ Inner Join : هو نفس الـ Equi Join

```
--Inner join
select st_fname,dept_name
from student s inner join department d
on s.dept_id=d.dept_id

--equi join
select st_fname,dept_name
from student s,department d
where s.dept_id=d.dept_id
```

```sql
--inner with 3 tables
select st_fname,dept_name,ins.ins_name
from student s inner join department d
on s.dept_id=d.dept_id inner join instructor ins
on ins.dept_id=d.dept_id
order by ins_name
```

3- الـ Outer Join: عندي مثلا جدولين جدول الـ Student وجدول الـ Departments

a. الـ Left Outer Join: هتطلع كل اسماء الطلبه الاول وبعدين تنفذ الـ Join.

b. الـ Right Outer Join: هتطلع كل الـ Departments الاول وبعدين تنفذ الـ Join.

c. الـ Full Outer Join: بيجيب كل الـ Data بتاع الجدولين.

```sql
--Outer join ===> left, right and full
select st_fname,dept_name
from student s left outer join department d
on s.dept_id=d.dept_id

select st_fname,dept_name
from student s right outer join department d
on s.dept_id=d.dept_id

select st_fname,dept_name
from student s full outer join department d
on s.dept_id=d.dept_id
```

4- الـ Self-Join: عايز اكتب Query تطلع اسم الـ Student واسم الـ Supervisor.

```sql
--Self join
select stud.st_fname,super.st_fname as "supervisor Name"
from student super,student stud
where super.st_id=stud.st_super
```

Aggregate Functions

ملاحظات هامه :

- **الـ Count(*) :** بيحسب عدد الـ Records كلها بالـ Nulls اللي عندي.
- **الـ Count(fname) :** بيعد الـ Values ولو فيه Null مش هيحسبها.
- **الـ Max():** لو عايز اكبر قيمه في الـ Column.
- **الـ AVG():** لو عايز احسب متوسط القيم لـ Column معين.

```
--count avg sum max min
select MAX(crs_duration) from Course

select COUNT(st_fname) from student

select COUNT(*) from student

SELECT AVG(ST_AGE) FROM Student

SELECT AVG(ISNULL(ST_AGE,0)) FROM Student
```

- **الـ Group by:** يعني زي المثال اشوف كل قسم عندي فيه كام طالب.
- **الـ Having:** هي بتتنفذ علي Aggregate Function يعني انا هشوف كل قسم فيه كام طالب بحيث ان يظهرلي فقط اعداد الطلبه في القسم ده اللي اكبر من 2.

ملحوظة: مثلا هنلاحظ ان الـ Dept_Id جاء بعد الـ Group by ولازم يكون مع الـ Select.

```
--group by
select COUNT(st_id),Dept_Id from Student
group by Dept_Id


--having
select COUNT(st_id),Dept_Id from Student
group by Dept_Id
having COUNT(st_id)>2
```

ملحوظة: ممكن الـ Having تاتي من غير Group by ولكن لازم يكون فيه Aggregate Function مع الـ Select.

```
--having without group by
--You should use aggregate in select clause
select COUNT(*) from student
having Count(*)<25
```

- **الـ Distinct: بستخدمها عشان ارجع القيم المختلفه.**

```
--Distinct
select distinct st_fname
from Student

select distinct st_fname,st_lname
from Student
```

- **الـ Order by: بستخدمها لو عايز ارتب.**

```
Select *
from Student
order by St_Fname
```

- **الـ Top: لو عايز اجيب اول 4 item من الجدول، وفي المثال التاني هيعرض 10% من الـ Rows.**

```
--Top
select Top 4 *
from student

select top(10) percent*
from student
```

- **الـ Top with ties: بترتب الـ Table بشكل معين وبعد كده بتشوف اخر قيمه موجوده عندي لو متكرره هيجيب كل المتكرر.**

```
--Top with ties
select Top 4 witH ties *
from student
order by st_age
```

- **الـ Top Randomized: هنا احنا بنرتب بالـ NEWID() يعني بيعطي لكل Row هنا ID في كل مره بعمل له Run فيها ولذلك كل مره هيجيب اول 5 وهنلاحظ في كل مره مختلفين.**

```
--Top randomized
SELECT TOP(5) *
FROM student
ORDER BY NEWID();
```

- الـUnion والـ Union All:
- الـ Intersect:
- الـ Except:

```
--Union [all] AND Rules
select St_Fname from Student
union all
select Ins_Name from Instructor
```

Sub Queries

- بنحاول نبعد عن استخدامها ولكن بلجأ لها في بعض الحالات.
- المشكلة وتم حلها باستخدام بالـ Sub Query كالاتي:

```
--sub queries
--the problem
Select St_fname,st_lname from Student
where St_Age> AVG(st_age)

--solusion
Select St_fname,st_lname from Student
where St_Age> (select AVG(st_age) from Student)
```

امثلة اخري باستخدام الـ in والـ exists :

```
Select ins_name
from instructor
where Ins_Id in (select dept_manager from Department where Dept_Manager is not null)

--Exists
select Ins_Name from Instructor
where exists
(select dept_manager from Department where Dept_Manager is not null)
```

ملاحظات:

- الـ Inner Query بترجعلي اما بـ True او False.
- طبعا مش بترجعلي Data.
- مش اقدر استخدم الـ Order by مع الـ Sub Query.

مقارنة بين الـ Sub Query وبين الـ Joins:

- احنا بنستخدم الـ Join والـ Subquery عشان اجيب Data من Tables مختلفة.
- الـ Subquery فيها نوعين :
  1. Scalar Subquery: بترجعلي Single Row من الـ Data.
  2. Tabular Subquery: بترجعلي Multiple Row من الـ Data.
- الـ Joins بتتنفذ بصورة اسرع من الـ Subquery في الـ SQL Server.
- احيانا الـ Joins بتعطيني Performance افضل من الـ Subquery في بعض الاحيان.

ترتيب تنفيذ امر الـ Query:

```
--The order of executing commands:
------  From
------  On
------  Join
------  Where
------  Group By
------  With Cube and With rollup
------  Having
------  Select
------  Distinct
------  Order by
------  Top
```

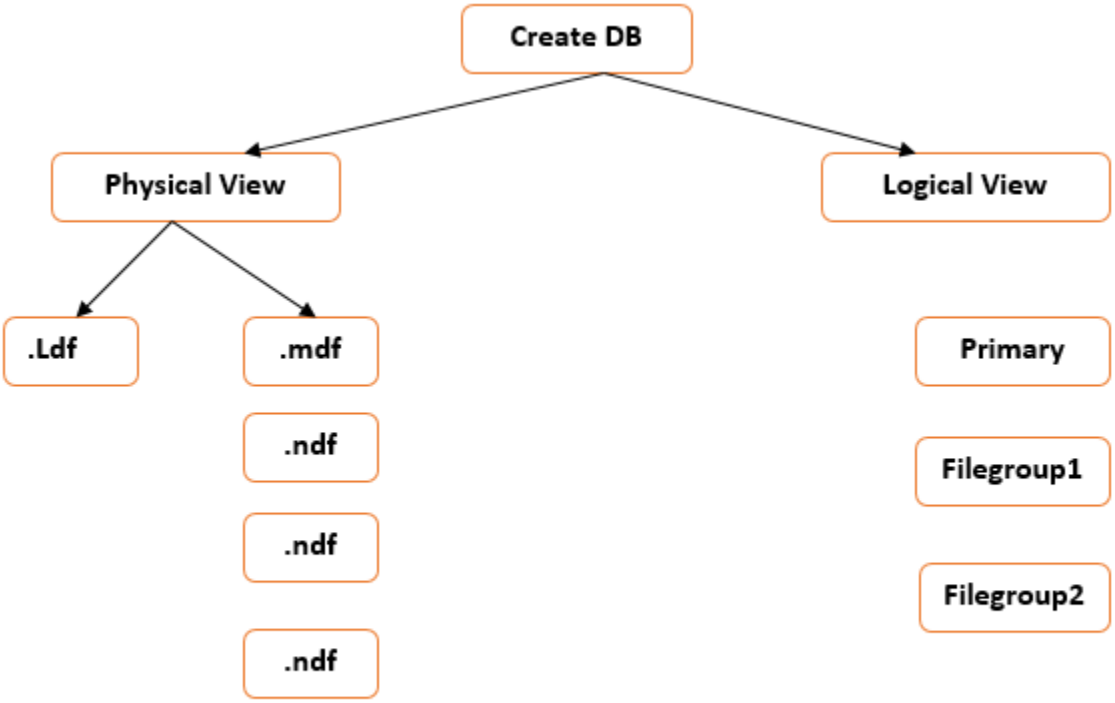
محاضـرة رقم 2

- عندنا كل RDBMS له طريقته الخاصه في الـ Manage بتاع الـ Structure الخاص بيه.
- وانا بعمل Create Database بيكون عندي للت DB الواحدة:
  1. Physical View
  2. الـ Logical View
- ايضا وانا بعمل Create Database بيكون عندي 2 Files:
  1. .ldf بيوضع فيه الأوامـر Insert, Delete وبيتاكد انه الاوامر دي مكتوبه بشكل صحيح.
  2. .mdf ده بيقوم بتنفيذ الأوامـــر.

ملحوظة:

- ممكن يكون عندي اكثر من Secondary File (.NDF)
- لما اجي اعمل Table جديد هيكون موجود في mdf.
- وممكن الـ Tables توجد في اكثر من File عن طريق الـ File Groups.
- الـ File Groups: عبارة عن Folder بيحتوي علي اكثر من File واعمل الـ Table ينتمي الي Group معين.

3. احنا بنعمل Create لـ File Group اسمه Primary وبضع جواه mdf File. والت Table ممكن اضعه في الـ Primary اللي هو فيه mdf File. لانه مش ينفع اضعه مباشرةً في الـ mdf File.
4. ممكن اعمل DB علي اكثر من File (مثلا 5 Files) واحد mdf واخر ldf والثلاثة الباقيين Secondary Files يعني NDF هقسم الت Data كالاتي:



5. الـ SQL بينزل علي Server وبيكون فيه HD كتير وممكن اقسم شغلي علي اكثر من HD فمثلا لو وضعت الـ mdf file. علي HD ووضعت الـ ldf. علي HD هيكون اسـرع في الـ Performance.
6. لو كان عندي الـ Tables كلها علي HD واحد هينتظر كل Table لحد مايتنقل بالكامل في الـ Memory عشان اعمل Join. علي العكس لو كان اكثر من HD الـ Table هتتنقل Parallel ويكون الـ Performance اسرع.

- **الخطوات في الـ DB كالاتي:**
  1. بعمل Create DB هيطلع لي 2 Files (.mdf, .ldf).
  2. هضيف انا 3 files من عندي.
  3. هروح لـ Groups واضيف 2 groups (fg1, fg2).
  4. هروح لكل File واحدد له الـ Group اللي تبعه.
  5. لما اعمل Create Table اروح احدد في الـ Properties هو تبع اي Group فيكون موجود علي HD1 وTable تاني موجود علي HD2.

بذلك لما اجي اعمل Join بين 2 Tables هيكون اسرع لانه هينقل الـ Tables لـ Memory في نفس الوقت لان انا عندي 2 HD.

- **عندنا بعض الـ Properties الخاصه بالـ File:**
  1. **Initial Size**: بحدد الـ Size الابتدائي بتاعه مثلا 4M.
  2. **Auto growth**: بستخدمها في حالة لو الـ initial File اتملي. ولذلك بعمل لها enable عشان لما يوصل لـ 4M مش يقف فبعمل growth بحوالي 50%. وممكن احدد له هيزيد لحد امتي ممكن اختار لما يوصل لحجم معين يقف او اعمله Unlimited يعني مش ليه Size محدد.
  3. **Path**: بحدد الـ Path بتاع الـ File.

ملحوظة:

- لو عندي 2 Files علي نفس الـ Group وواحد 4M والتاني حجمة 2M فهو هيملي الـ file Group الاول وبعد مايخلصه هيروح يملي الـ File التاني قبل مايروح لـ Auto generated.
- الـ Table الواحد ممكن يتقسم علي اكثـر من File.

لما اجي اعمل Create لـ Table في الـ DB

- هضيف فيه بعض الـ Columns واحدد لهم الـ Data Type بتاعتهم.
- احدد الـ Primary Key في الـ Table.

الـ Properties بتاعة الـ Column: (هـــــام جـــــداً)

1. **Allow Nulls**
2. **Length**
3. **Computed Column**: يعني لو الـ Column هيتحسب عن طريق Columns اخري بكتب فيه الدالة اللي بتحسب العملية.
4. **Identity**: هحدد هيبدا بكام وهيزيد بكام عن طريق الـ Identity Set.
5. **Is sparse**: لو عندي Column هياخذ Null Values كتير وانا مش عايزها تتحجز علي الت HD هعمل Enable لـ ISSPARSE بـ True.
6. **Default Value**: لو عايز اعمل قيمة Default هستخدمها.

الـ Relation مابين الـ 2 Tables:

- اضغط Click يمين علي الـ Relation واختار New Relation واحدد العلاقة بتاعتي مثلا One-to-Many
- توجد للـ Relation مجموعة من الـ Properties منها:

1- Insert and Update

a. بحدد منه الـ no action فمش هينفع احذف P.K له Child.
b. او تكون Cascade يعني لما يعمل delete للـ Parent يروح يعمل delete للـ Childs اللي عندي وبعدين يحذف الـ Parent
c. او Set Default لازم تكون قيمه من القيم اللي موجوده في الـ P.K فبيروح للـ Childs ويضع في الـ F.K الـ Default Value ولازم اضمن انها مش تتمسح ابداً.
d. او الـ Set Null يروح للـ Childs الاول ويضع Null Values وبعدين نحذف الـ Parent.

DB Schema

- وُجدت مشاكل في الـ SQL من خلالها ظهر الـ Schema والمشاكل هي:
  1. مستحيل اننا نعمل Create لـ 2 tables بنفس الاسـم في الـ DB.
  2. لو كان عندي User وعايز اعطي له Permission معين فهروح الاول اعمل Create للـ User وبعدين اعطي له Permission فلو كان مثلا الـ Permission نفسها علي 10 Tables هروح اربطه بالـ Table ده 10 مرات ولو عايز اعمل Delete للـ User هروح اعمل Disconnect الاول للـ User وبعدين احذفه.
  3. لو عندي DB فيها 200 جدول ومحتاج ادور علي Tables معينه موجودين جوه الـ 200 table ومش موجود تقسيم للـ DB.
- الحـل هو استخدام الـ DB Schema : لما اعمل Create لـ Table هحدد معاه بعض الـ Folders واعطي كل واحد اسم ولما اعمل Create للـ Table هحدد هو تبع اي Folder وتم حل المشاكل الثلاثة السابقة كالأتـي:
  1. هنا ينفع يكون عندي 2 Objects بنفس الاسم ولكن في Schema مختلفه.
  2. لو كان عندي User فهربطه مره واحده بالـ Schema واعمل عليها Permission بدلا من اربطه بالـ Table.
  3. هنا عمل Logical Meaning للـ Tables بتاعتي فهقدر اعمل Search علي الـ Files بتاعتي بطريقه اسهل.
- ملحوظة: كل Object بيكون له Scope بتتعمل له:
  1. Login Object: بيتعملها Create علي مستوي الـ Server.
  2. Table: بيتعملها Create علي مستوي الـ DB.
  3. Schema: بيتعملها Create علي مستوي الـ DB.
- ازاي بنعمل الـ Schema ؟

طريقة الـ Wizard:

هروح للـ DB واختار الـ Security واختار منها الـ Schemas واعمل Click يميـن واختار New Schema.

او اعملهـا بالكـــود كالأتـي :

- هاعمل Create لـ Schema اسمها x كالأتـــي:

```
create schema x
```

- لو عايز اعمل Transfer لـ Table مثلا يعني جدول الـ Student هيكون تبع الـ Schema اللي اسمها x مش الـ dbo:

```
alter schema x transfer student
```

- ولو هاعمل Select لـ Table الـ Student لازم اكتب الـ Schemaاللي هوّ تبعها:

```
select * from x.student
```

- الـــ Synonym: يعني لو عندي Object اسمه كبير في الـ DB وهستخدمه كثير فهلجا الي استخدام الـ Synonym يعني عبارة عن Shortcut للحاجة وهو في المثال كالاتـــي:

```
create synonym mySyn
for HumanResources.Employee
                -----------------------
select * from mySyn
                -----------------------
use AdventureWorks
--to create an alias for the table name
create synonym p
for Production.ProductListPriceHistory
```

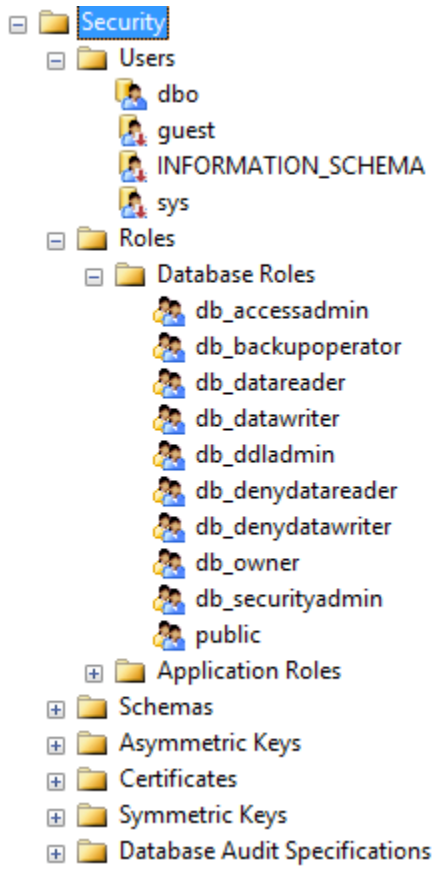
ازاي اعمل Create لـ User وأعطي له Permission علي الـ Schema :

- الـ Login: هو الحاجة اللي بتعمل Create علي مستوي الـ Server فهو الشخص اللي من حقة يدخل علي الـ Server.
- الـ User: لازم احدده بعد مابدخل علي الـ Server فاحدد الـ User علي الـ DB.

Security → New → User

- الـ Server Roles: موجود فيه Group اسمه Public اي الـ Login من حقة فقط يدخل علي الـ Server فقط.

- **الـ User Mapping: بعمل Map علي الـ DB اللي بحددها واللي هيقدر يشوفها ويتعامل معاها بنفس الاسم بتاع الـ Login.**



هحدد للـ User ازاى هيشوف الـ Schema:

- **هروح مثلا للـ ITI DB واختار منها الـ security واختار منها الـ Users واختار الـ User بتاعي واعمل Search واختار الـ Schema اللي هعمل access للـ User عليها.**
- **"فمثلا هروح اعمل Grant لـ Select وDeny لـ Delete والباقي هيكون Revoke يعني هروح للت Parent بتاعه ويشوف لو كان يقدر يشوفه هيشوف زيه (With Grant) لو كان من حق الـ Parent يعمل Delete فهو هيعمل."**
- **لو الـ User كان في Group معين هو هياخذ الـ Permission بتاعة الت Group بجانب الـ Permission بتاعته.**

```
--Create user
USE [ITI]
GO
CREATE USER [Test_iti_db] FOR LOGIN [test]
GO
```

DB Integrity

- يعني الـ Data اللي انا مدخلها في الـ DB تكون صحيحة والـData تكون بتحقق خصائص الـ F.K
- ازاي اقدر احقق الـ DB Integrity: لازم 3 حاجات يتحققوا
  1. Domain
  2. Entity
  3. Referential
- اولا الـ Domain: يعني Range Of Values وعشان اقدر احققها لازم احقق الأتـــي
  1. الـ Constraint:
     - هعمل Check Constraint اللي هو بيحقق الـ Domain.
     - الـ F.K نفسه بيحقق الـ Domain.
     - الـ Null, Not Null تحقق الـ Domain.
     - الـ Default Value بتكون واحده من الـ Ranges اللي موجوده وبتحقق الـ Domain.
  2. الـ DB Objects : اللي هقدر استخدمه عشان احقق الـ Range of Values.
     - الـ Triggers and Rules: هستخدم الـ Roles في الـ DB عشان احقق الـ Range of Values.
- ثانيا الـ Entity: فيه عايز احقق واحافظ علي الـ Uniqueness بتاعة الـ Entity وده بيتحقق عن طريق:
  1. الـ Constraint:
     - الـ Primary Key يكون not null وايضا Unique.
     - الـ Unique Key هو Unique ولكن عنده قيمه واحده Null (مع ملاحظة: اي P.K لازم يكون Unique ولكن Unique مش معناه انه يكون P.K).
  2. الـ DB Objects : اللي هقدر استخدمه عشان احقق الـ Entity وهستخدم فيه الـ Triggers and Rules.
- ثالثا الـ Referential: وده بيتحقق كذلك عن طريق:
  1. الـ Constraint: الـ F.K بيحافظ علي الـ Relation.
  2. الـ DB Objects : الـ Triggers.

الامثلة كالاتي:

```
create table dept
(
dept_id int primary key,
name text
)
```

```sql
create table emp
(
emp_id int  not null,
emp_name nvarchar(50),
emp_sal int,
constraint c1
check (emp_sal>100)
,
emp_address nvarchar(50) default 'cairo'
,
constraint c2
check (emp_address in ('cairo','alex'))
,
overtime int,
constraint c4
check (overtime between 100 and 1000)
, constraint c3
primary key(emp_id)
,
dept_id int,

constraint c5
foreign key(dept_id)
references dept(dept_id)
)


alter table emp
add constraint c55
unique (emp_name)

alter table emp
drop constraint c55
```

عشان اشوف معلومات عن الـ Constraint فقط :

```sql
EXEC sp_helpconstraint emp
```

لو عايز اعمل Create لحاجة Default على مستوي الـ DB:

```
create default dcity as 'tanta';
```

```
sp_bindefault dcity,'student.st_address'
```

لو عايز الغي الـ Default كالأتـــي:

```
sp_unbindefault 'student.st_address'
drop default dcity
```

ملحوظة:

- **ممكن اعمل Datatype معين له Default Value ولكن هعمله Create علي مستوي الـ DB كلها. الـ Datatype هيكون Constraint علي مستوي الـ DB كلها عشان كل الـ Tables تقدر تشوفه بذلك هعمل Create Rule كالاتي:**

```
create rule rage as @age>10
```

هربط الـ Rule دي بعمود معين يعني هعمل ليها Bind:

```
sp_bindrule rage,'student.st_age'
```

ولو عايز الغي الـ Rule:

```
sp_unbindrule "student.st_age"
  drop rule rage
```

ملحوظة:

- **الـ Constraint بتنطبق علي الـ DB القديمة والجديدة.**
- **الـ Rule بتنطبق علي الـ DB الجديدة فقط.**
- **لو كان عندي Constraint وRule علي نفس الـ Column عكس بعض هياخذ الـ Constraint الاقرب للـ Table.**

لو عايز اعمل Create لـــ Datatype:

```
sp_addtype new_dtype,'nvarchar(50)'

--change sname datatype to be
-- see table modification wizerd

alter table student
alter column st_fname new_dtype
```

لو عايز الغي الـ Datatype:

```
sp_droptype new_dtype
```

ممكن استفيد من اني اعطي للـ Datatype (new_dtype) كل الـ Constraint :

```
--binding rule+default+datatype
sp_bindefault dcity,new_dtype
```

Ranking

- لو عايز اعمل Query تحسب الـ Salary لأكبر salary 2 وكان عندي شخصين بياخذوا نفس الراتب 2000 جنيه مثلا واثنين بياخذوا 1000 جنيه. باللي احنا اخذناه الي الان لو طبقته هيطلع ليَّا 2000 لشخصين فقط.
  لذلك ظهر عندنا في الـ SQL مفهوم الـ Ranking: واهم 3 Functions في الـRanking وبنستخدمهم مع الـ Order by:
  - ROW_NUMBER()
  - RANK()
  - DENSE_RANK()

اولاً الـ ROW_NUMBER(): بنعطي لكل Row رقم من غير تكرار.

```
select st_fname,st_age,ROW_NUMBER() over(order by st_age) as "order by age"
from student
```

```
select st_fname,st_age,ROW_NUMBER() over(order by st_age) as "order by age"
from student
```

100 %

Results | Messages

	st_fname	st_age	order by age
1	Amr	NULL	1
2	Heba	NULL	2
3	Mohamed	NULL	3
4	Asem	NULL	4
5	Khalid	18	5
6	Noha	20	6
7	Said	23	7
8	ASKER	25	8
9	ASKER	28	9
10	ali	30	10
11	Ahmed	30	11
12	Mona	30	12
13	ASKER	30	13
14	NULL	30	14
15	ASKER	35	15

ثانياً الـ RANK() بتعطي لكل Row رقم نفس الـ ROW_NUMBER ولكن الناس المتساوية مع بعضها في الـ Order by هتاخذ نفس الـ RANK.

```
select st_fname,st_age,rank() over(order by st_age) as "rank"
from student
```

100 %

Results | Messages

	st_fname	st_age	rank
1	Amr	NULL	1
2	Heba	NULL	1
3	Mohamed	NULL	1
4	Asem	NULL	1
5	Khalid	18	5
6	Noha	20	6
7	Said	23	7
8	ASKER	25	8
9	ASKER	28	9
10	ali	30	10
11	Ahmed	30	10
12	Mona	30	10
13	ASKER	30	10
14	NULL	30	10
15	ASKER	35	15

ثالثاً الـ DENSE_RANK(): هي نفس الـ RANK ولكن هوّ بيرقم Sequential.

```
select st_fname,st_age,DENSE_RANK() over(order by st_age) as "Dense ranking"
from student
```

100 %

Results | Messages

	st_fname	st_age	Dense ranking
1	Amr	NULL	1
2	Heba	NULL	1
3	Mohamed	NULL	1
4	Asem	NULL	1
5	Khalid	18	2
6	Noha	20	3
7	Said	23	4
8	ASKER	25	5
9	ASKER	28	6
10	ali	30	7
11	Ahmed	30	7
12	Mona	30	7
13	ASKER	30	7
14	NULL	30	7
15	ASKER	35	8

ودي لو كانوا مع بعض في Query واحدة:

```
select st_fname,ST_AGE
       ,ROW_NUMBER() over(order by st_age) as "order by age"
       ,RANK() over(order by st_age) as "rank"
       ,DENSE_RANK() over(order by st_age) as "exact ranking"
   from student
```

100 %

Results | Messages

	st_fname	ST_AGE	order by age	rank	exact ranking
1	Amr	NULL	1	1	1
2	Heba	NULL	2	1	1
3	Mohamed	NULL	3	1	1
4	Asem	NULL	4	1	1
5	Khalid	18	5	5	2
6	Noha	20	6	6	3
7	Said	23	7	7	4
8	ASKER	25	8	8	5
9	ASKER	28	9	9	6
10	ali	30	10	10	7
11	Ahmed	30	11	10	7
12	Mona	30	12	10	7
13	ASKER	30	13	10	7
14	NULL	30	14	10	7
15	ASKER	35	15	15	8

ودي مجموعة تطبيقات علي الـ Ranking:

```
select st_fname,st_address,ST_AGE
       ,ROW_NUMBER() over(ORDER BY st_age DESC) as "order by age"
FROM Student

select st_fname,st_address,ST_AGE
       ,ROW_NUMBER() over(ORDER BY st_address,st_age DESC) as "order by age"
FROM Student

select st_fname,st_address,ST_AGE
       ,ROW_NUMBER() over(PARTITION BY st_address ORDER BY st_age DESC) as "order by age"
FROM Student

 SELECT *, ROW_NUMBER() OVER(PARTITION BY dept_id ORDER BY st_age DESC) AS rk
      FROM student
      where st_age is not null
```

ملحوظة:

- **الـ Partition by: يعني هيقسمهم الاول Departments وبعدين يعمل لكل Department الـ Order by بتاعته يعني هيرتب الـ Age لكل Department.**

مثال اخر:

```
SELECT *
FROM (
      SELECT *, ROW_NUMBER() OVER(PARTITION BY dept_id ORDER BY st_age desc) AS rk
      FROM student
      where  dept_id is not null
      )
      AS L
WHERE rk<=2;
```

100 %

Results | Messages

	St_Id	St_Fname	St_Lname	St_Address	St_Age	Dept_Id	St_super	grade	rk
1	3	Mona	Saleh	Mansoura	30	10	1	NULL	1
2	4	ASKER	Mohamed	Mansoura	30	10	1	NULL	2
3	7	NULL	Hussien	NULL	30	20	6	NULL	1
4	8	Mohamed	Fars	Mansoura	NULL	20	6	NULL	2
5	9	ASKER	Ahmed	Mansoura	35	30	NULL	NULL	1
6	11	ASKER	Ahmed	Mansoura	28	30	9	NULL	2
7	13	Said	NULL	NULL	23	40	12	NULL	1
8	12	Noha	Omar	Mansoura	20	40	NULL	NULL	2
9	22	ali	NULL	NULL	30	50	NULL	NULL	1

محاضـرة رقم 3

Variables

- الـ Variables بنستخدمها عشان تسهلنا اعادة استخدام الـ Constant Values وايضا عشان اقدر اباصي information من والي الـ Stored Procedure وكذلك عشان اتجنب استخدام الـ Subquery.
- الـ Variables عندنا نوعين :

1. الـ Local Variables: بتكون Local علي مستوي الحاجة اللي هي متعرفه فيها زي مثلا الـ Stored Procedure او الـ Trigger او علي مستوي الـ Batch. ولازم تبدا بـ @ ولازم يكون ليها Data Type والـ Initial Value بتكون بـ NULL.

ملحوظة: الـ Batch ده مجموعة من الـ Statements بنفذها مع بعضها ولكن مش معتمدين علي بعض.

- اولا احنا بنعرف الـ Variable

```
--Declare variable:
    Declare @x int
```

- ثانيا احنا بنـ Assign قيمه للـ Variable ده وبيكون له اربع طرق علي حسب الاستخدام

```
--Assign value      4ways
    Set @x=10
    Select @x=20
    Select @x=id from emp where Id=3
    Update emp set name='ahmed' , @x=id where salary=1000
```

- ثالثا ازاي بستخدم الـ Variable ده

```
--Using variable
    Select * from emp where id=@x
```

- رابعا ازاي اعرض قيمة الـ Variable ده

```
--View variable value
    Select @x
```

ملاحظات هامه جدا على الـ Local Variables:

- لو الـ Select Statement مش رجعت حاجة الـ Variable هيكون جواه اخـر قيمه كانت فيه.
- لو الـ Select Statement رجعت اكثر من Value الـ Variable وقتها هيحتفظ باخر Value جايه في الـ Select.
- ممكن Variable من نوع table عشان استقبل فيه اكثر من قيمه.

```sql
--table variable
declare @t table(id int)

insert into @t
select st_id from student

select @t
```

كذلك ممكن استخدم الـ Local Variables مع الـ Top.

```sql
--compound assignment "short hand operators"
DECLARE @MyNumber int = 2
SET @MyNumber += @myNumber

--top with variables
declare @top int
set @top=2
select Top(@top)* from Student
```

2- الـ Global Variables: تعتبر مجموعة من الـ Variables بتكون Predefined داخل الـ Server نفسه والـ Server هوّ اللي بيعرفها وبيعمل Assign لقيم جوّاها.

```sql
select @@servername
```

الـ @@servername: بترجع اسم الـ Server وفي حالتنا بترجع اسم الجهاز.

```sql
select  @@rowcount
```

الـ @@rowcount: بترجع عدد الـ Records اللي حصل عليها Affect من اخر Statement سواء كانت Update, Select, Delete.

```sql
select  @@error
```

الـ @@error: كل الـ Errors اللي بتطلع في الـ SQL موجوده في الـ SQL ومرقمه بارقام فالـ @@error بتشيل رقم الـ error message اللي Affected من اخر Statement ولو مش رجعت error هيكون في @@error = 0.

```
select  @@identity
```

الـ @@identity: لو كان table فيه Identity Column وقتها الـ Server هوّ اللي بيعمل Assign لقيم الـ @@identity فتحتفظ بقيم اخر Identity دخلت في الـ Table (ولو كان مش فيه قيم هيكون @@identity بـ Null)

```
select  @@version
```

الـ @@version: بترجع لي بالنسخة اللي انا شغال عليها.

ملاحظة:

- **عشان اعرض القيمة اللي داخل الـ Global Variable هعمل :**

```
select  @@rowcount
```

Control of Flow Statement

IF | IF Exists | Case | Begin | End

Continue | Break | While

-1 الـ IF Statement:

```
--      1)IF
declare @num_rows  int
    delete from student where st_id=190
    select @num_rows = @@rowcount
    if @num_rows = 0
            select  'No rows were deleted.'
    Else if @num_rows = 1
            select  'One row was deleted.'
    Else
            select ' Multiple rows were deleted.'
```

2- **الـ IF Exists:** ودي كل اللي يهمها الـ Inner Query بترجع Result ولَّا لا لانها لو مش رجعت Result مش هتنفذ الـ Outer Query. عايز اعمل Query تعمل الاتي: تعمل select لكل الـ Instructor بشرط ان يكون فيه واحد Manager كما هو موضح في المثال القادم.

```
Select * from Instructor
where exists (select Dept_Manager from Department where Dept_Manager is not null)
```

مثال اخـــر: هنلاحظ في المثال القادم ان هو هينفذ الـ Inner Query الاول وبعدين بناءا علي الـ Result هتنفذ الـ Outer Query. وطبعا الـ IF Exists اسرع في الـ Performance.

```
--3) If exists
if exists(select 1 from sys.tables where name='studen')
select * from studen
```

ملحوظة: في المثال السابق وجدنا sys في الـ Inner Query وهي اساسا من الـ Views وبتكون Built in موجوده في الـ model ودي بستخدمها في حالة لما اجي اعمل Run لـ Query مش يطلع لي error مثلا عايز اشوف table معين وانا بنفذ الـ Query هـوَّ موجود ام لا.

3- **الـ Case Statement:** عندنا نوعين من الـ Case Statement وهم:
   a. Simple Case ودي لما يكون فيها Values.
   b. Searched Case ودي لما يكون فيها Condition.

في المثال القادم هنوضح استخدام الـ Case مع الـ Update.

```
update dbo.Instructor
        set Ins_Salary =
        case
            when Ins_salary<1000 then Ins_salary*0.1
            when Ins_salary=2000 then Ins_salary*0.2
            else Ins_salary*0.3
        end
```

في المثال القادم هنوضح استخدام الـ Case مع الـ Expression.

```
------based on expression

SELECT   Ins_Name, Gender =
      CASE Gender
         WHEN 'M' THEN 'Male'
         WHEN 'F' THEN 'Female'
      END
from Instructor
```

في المثال القادم هقول لو كان الـ Salary اقل من 5000 هكتب Low Salary ولو كان الـ Salary اكبـر من 5000 هكتب High Salary ولذلك هستخدم الـ Case. هنجد ان ميزة الـ Case اني بعمل Remove لـ Column واضع Column جديد خالص.

```
--case statment with select
select Ins_Name,Salary =
        case
            when salary<5000 then 'low salary'
            when salary>=5000 then 'high salary'
            else 'has no salary'
        end
from Instructor
```

4- الـ IIF: هي عبارة عن Case Statement ولكن هي IF – else فقط. تعتبر if – else في سطر واحد بستخدمها في حالة لما يكون عندي case واحده.

```
-- IIF
Select Ins_Name,IIF(Salary>5000,'high','low')
from Instructor
```

5- الـ While Statement:

```
while (select avg(price) from titles) > $20
        begin
           update titles
            set price = price / 2
           if (select max(price) from titles) < $40
            break
        end
```

Script, Batch and Transaction

1- الـ Batch: مجموعة من الـ Statements بنفذها مع بعضها يعني في نفس الوقت ولكن مش معتمدين علي بعض.
2- الـ Script: مجموعة من الـ Batches مفصول بينهم بـ go لان بعض الـ Queries لازم تكون في Batch لوحدها. علي سبيل المثال : مش ينفع اعمل جملة Create وجملة Drop في نفس الوقت. وخطوات تنفيذه كالاتي: اروح علي الـ DB واختار Tasks واختار منها Generate Scripts هيطلع لي Wizard اختار منها الـ Tables اللي انا عايز استخدمها وبعدين اختار اني اعرضه في new Query.
3- الـ Transaction: لو عايز انفذ 3 جمل مع بعض كـ One Unit مثلاً (Insert, Update, Delete) فلو نفذ جملة الـ Insert وبعدين الـ Update والـ Delete كان فيها Error فهيعمل Remove للـ Statements اللي قبل كده كمان بيحافظ علي الـ Data في حالة الـ Power Failure. ومن انواع الـ Transactions:
   a. Implicit Transaction: مثلا جملة Insert يطلق عليها Implicit.

```
insert into Student
values(100,'asem','asker','Cairo',23,10,null,null)
```

   b. Explicit Transaction: بستخدمه لما احتاج انفذ اكثر من جمله مع بعض. هنا لازم احدد اما Commit او Rollback.
      i. الـ Rollback: بكتبها لما يكون فيه error.
      ii. الـ Commit: بكتبها في حالة عدم وجود error هينفذ الـ Transaction دي.

```
begin tran t1
declare @r int
update student set St_Fname='ahmed' where St_Id=5     --there is no errors
insert into student(St_Id,St_Fname) values(66,'ali')        --error so rollback
set @r=@@error   --look to the last statment only
if @r = 0
    begin
    commit tran
    select 'true'
    end
else
    begin
    rollback tran
    select 'false'
    end
```

ملاحظات:

- في حالة الـ Power Failure يروح ينفذ الـ Rollback لما يكون مش مكتوب Commit او Rollback.
- @@error: هي اللي هتساعدني اني اعرف ان فيه error ام لا ولكن المشكلة ان IF @@error! = 0 في هذه الحالة @@error هتنفذ علي اخر واحده فيها. ولذلك عملوا الـ try – catch
  1. الـ try: بكتب بداخلها الـ Transaction الجمل بتاعة الـ Insert.
  2. الـ Catch: بكتب فيها الـ Rollback لما يحصل error عندي.
- توجد بعض الـ Functions المهمه:

1. **Error number**: دي بتطلع لي رقم الـ error.
2. **Error message**: بتطلع لي رسالة بتاع الـerror اللي حصل.
3. **Line**: السطر اللي حصل فيه الـ error.

الخلاصة فى الـ Transaction:

- في حالة ان الجمل اللي موجوده في الـ Trans كانت كلها صحيحة توجد commit هتنفذ كل اللي في الـ try.
- وفي حالة ان احدي الجمل فيها error بذلك هيدهل في الـ Catch ويدخل ينفذ اللي جواها اللي هو الـ Rollback يعني هيعمل remove للـ Statements اللي كانت موجوده.

بعض الامثله الهامه فى الـ Transaction:

```
--try and catch

create table Parent
(
pid int primary key not null
)

create table Child
(
cid int references parent(pid)
)

insert into parent values(1)
insert into parent values(2)
insert into parent values(3)
insert into parent values(4)

insert into child values(1)
insert into child values(5)
insert into child values(2)
```

```
begin tran
insert into child values(1)
insert into child values(3)
insert into child values(2)
commit tran

begin tran
insert into child values(1)
insert into child values(5)
insert into child values(2)
rollback tran --rollback all statments if there is error
```

```
begin tran
insert into child values(1)
insert into child values(5)
insert into child values(2)
if @@error!=0
    rollback tran
else
    commit

select * from parent
select * from child

delete from child
```

```
begin try
    begin tran
        insert into child values(1)
        insert into child values(5)
        insert into child values(2)
    commit tran
    print 'transaction commited'
end try
begin catch
    rollback
    print 'transation rolled back'
    select error_number() as "number",
    error_message() as "message",
    error_line() as "line"
end catch
```

```
--save point

delete from child

declare @test int
begin try
    begin tran mytran
        insert into child values(1)
        insert into child values(2)
        insert into child values(3)
        set @test=1
    save tran save1
        insert into child values(2)
        insert into child values(3)
        insert into child values(4)
        set @test=2
    save tran save2
        insert into child values(3)
        insert into child values(4)
        insert into child values(5)

    commit tran
    print 'transaction commited'
end try
begin catch
    if @test is null
        rollback tran mytran
    else
    if @test =1
        rollback tran save1
    else
    if @test =2
    rollback tran save2
end catch

select * from child
delete from child
```

```
--nested transaction
--Using try/catch
begin try
    select 1/0
end try
begin catch
    print 'Error'
    select ERROR_MESSAGE() 'Error Message'
    ,ERROR_NUMBER() 'Error Number'
    ,ERROR_LINE () 'Error Line Number'
    ,ERROR_SEVERITY () 'Error Severity Level'
    ,ERROR_PROCEDURE() 'Error Procedure'
    ,ERROR_STATE () 'Error State'
end catch
```

ملاحظة:

- دائما بيكون الـ System DB مع الـ Analyst فممكن اعمل الكود بالـ C# او بالـ Transact SQL ولكن هنلاحظ ان الـ Transact SQL اسرع في الـ Performance.

Functions

Built in Functions

الـ Built in functions: بطلق عليها Scalar Functions يعني Function بتعمل Return لقيمه واحده فقط.

- **الـ NULL:**

1. ISNULL():

```
select ISNULL(st_age,20) from Student
```

2. COALESC():

```
SELECT COALESCE(hour_rate * 40,salary,Bonus )  AS 'Total Salary'
FROM instructor
```

- الـ CONVERT:

1. CONVERT()

```
select convert(nvarchar(20),hiredate,3)
from instructor --1:5

select t_id from instructor
where convert(nvarchar(50),hiredate,3) like '01%'
```

2. CAST()

```
select * from Department
where cast([Manager_hiredate] as varchar(40)) like '%1'
```

- الـ DATE:

1. Getdate() والـ FORMAT()

```
--5)date
select getdate()

select datename(dd,hiredate) from Department

select datename(mm,hiredate) from Department

select datename(yy,hiredate) from Department

select datename(mm,getdate())

select EOMONTH(GETDATE(),0)
```

الـ EOMONTH(): دي FUNC بستخدمها عشان اعرف الشهر اللي انا محدده 30 او 31

- الـ String :

1. UPPER() والـ Lower()

```
select lower(st_fname) from student
select lower(st_lname) from student
select substring(st_fname,1,5) from student  --start--length
```

- الـ Aggregate:
  1. COUNT():

```
Select COUNT(St_Fname) from Student
```

- الـ Math:
  1. SIN() والـ COS()

```
select sin(100)
select power(100,2)
```

- الـ System:
  1. db_name() والـ suser_name()

```
--3)system function
select db_name()
select suser_name()
```

User Defined Functions

الـ User اللي هو الـ Developer يقدر يعرف 3 انواع من الـ Functions:

1- الـ Scalar Function: بكتب كود بتاع Function بتعمل Return لقيمه واحده فقط.
2- الـ Table Valued: بكتب كود لـ Function بترجع Table وهستخدمه لو كان الـ Body بتاعه زي الـ View او عبارة عن جملة Select واحده بس هستخدمها.
3- الـ Multi-statement: لو كان الـ Body بتاعي فيه اكواد كثيرة هستخدمه ويقال عليه "Insert Based on Select".

الـ Scalar Function: بنعمل لها Create علي مستوي الـ DB.

```
Create function getSname(@sid int)
returns nvarchar(20)
begin
declare @name nvarchar(20)
    if @sid > 0
        select @name=st_fname from student
        where st_id=@sid
    else
        set @name='sid must be positive'
return @name
end
```

```
select dbo.getSname(-1) as "name"

print dbo.getSname(-1)
```

ملحوظة: في الـ User Defined لما اجي اعمل Run لـ Function بستخدم اسم الـ Schema عشان افرق بينها وبين الـ Built in Functions.

الـ Table Valued:

```
create function highage()
returns table
as
return
(
select st_fname,st_age from student where st_age>=20
)

select * from dbo.highage()
```

ملحوظة:

- في الـ Run استخدمت * لانها بترجع Table.
- في الـ Table Valued مش محتاج وانا بعمل ليها Run استخدم الـ Schema لان مفيش Inner Function بترجع Table فهكتبها علي طول.

الـ Multi Statements: هبعت كلمه ما وحسب الكلمة هيرجع حاجه من ضمن اكتر من حاجة فهعملها Multi.

- **@t عبارة عن Variable من نوع table عملته عشان افرق بين الـ Table valued والـ Multi Statements.**

```
create function student_names(@format nvarchar(50))
returns @t table
        (
         student_id int primary key,
         student_name nvarchar(50)
        )
as
begin
    if @format='fullname'
        insert @t
        select st_id,st_fname+' '+st_lname
        from student
    else
    if @format='firstname'
        insert  into @t
        select st_id,st_fname
        from student
return
end

select * from student_names('fullname')

select * from student_names('firstname')
```

مثال اخـر:

```
create function myday(@x datetime)
returns nvarchar(20)
as
begin
declare @d nvarchar(20)
select @d=datename(mm,@x)
return @d
end

select dbo.myday(02/02/2000)
```

ملحوظة:

- **Alter: عشان اعدل في الـ Function.**
- **Drop: عشان احذف الـ Function.**

ملاحظات هامه جدا :

- **الـ Identity: وجودها تسبب انه يكون موجود حاجة في الـ Server اسمها gap Identity يعني مثلا فيه gabs في الـ ID وقتها الـ Developer بيعمل هو الـ Insert وبعدين نشغلها تاني فتشتغل علي اخر قيمه كانت موجوده.**
- **لو عايز اجيب الـ Identity Column كالاتي:**

```
--SHOW IDENTITY COLUMN
select AddressID from Person.Address
select Address.$IDENTITY from person.Address
```

- **الـ Identity Insert: يعني اقدر اعمل insert للـ gabs اللي موجوده في الـ Identity Column وهي بتكون ON (هقدر وقتها اضيف قيم) أو OFF (ودي بتكون الـ Default ودي بستخدمها في حالة لو عايز اوقف الـ Insert).**

```
CREATE TABLE dbo.T1 ( column_1 int, column_2 VARCHAR(30),
                      column_3 int IDENTITY primary key);
 GO

SELECT * FROM T1

 INSERT T1 VALUES (1,'Row #1');
 INSERT T1 (column_2) VALUES ('Row #2');
 GO
SET IDENTITY_INSERT T1 ON;
 SET IDENTITY_Insert T1 off;
 GO
INSERT INTO T1 (column_3,column_1,column_2)  VALUES
 (34,1, 'Explicit identity value');
 GO
SELECT column_1, column_2,column_3
 FROM T1;

 drop table T1
```

- الـ Case Statement: ممكن تستخدم مع الـ Update زي مابتشتغل مع الـ Select.
- في SQL Server 2008 ممكن استخدم TOP مع الـ DML (select, insert, update, delete).

```
--Top with DML
--after using it with select
--insert with top
INSERT INTO test
    SELECT TOP (3) St_Id,St_Fname
    FROM Student

INSERT top(3) INTO test
    SELECT St_Id,St_Fname
    FROM Student

--update with top
update top(2) Department
set Dept_Location = 'alex'

--Top with delete
--Delete with top
delete Top(2) from test

--delete 2.5% from the data
DELETE TOP (2.5) PERCENT
FROM Production.ProductInventory;
```

- الـ Rollup: بيمسك القيم كلها ويعمل ليها Sum او بمعني اخـر هي group by وبعض الـ Aggregates علي اول Column. واليكم المثال كاملا.

```
--rollup and cube

use test

create table sales
(
ProductID int,
SalesmanName varchar(10),
Quantity int
)
```

```
insert into sales
values  (1,'ahmed',10),
        (1,'khalid',20),
        (1,'ali',45),
        (2,'ahmed',15),
        (2,'khalid',30),
        (2,'ali',20),
        (3,'ahmed',30),
        (4,'ali',80),
        (1,'ahmed',25),
        (1,'khalid',10),
        (1,'ali',100),
        (2,'ahmed',55),
        (2,'khalid',40),
        (2,'ali',70),
        (3,'ahmed',30),
        (4,'ali',90),
        (3,'khalid',30),
        (4,'khalid',90)
```

```
select ProductID,SalesmanName,quantity
from sales

select ProductID,SalesmanName,sum(quantity) as "Quantities"
from sales
group by ProductID,SalesmanName


select ProductID,sum(quantity) as "Quantities"
from sales
group by rollup(ProductID)
--order by ProductID,SalesmanName

select ProductID,SalesmanName,sum(quantity) as "Quantities"
from sales
group by rollup(ProductID,SalesmanName)
--order by ProductID,SalesmanName
```

- ولو استخدمت مع الـ **Rollup** عمودين دائما بيستخدم اول عمود.

- ولو عايزه يعمل عن طريق الاثنين هنستخدم cube والـ Cube هي rollup ولكن بتعمل علي اكثر من column.

```
select ProductID,SalesmanName,sum(quantity) as "Quantities"
from sales
group by cube(ProductID,SalesmanName)
--order by ProductID,SalesmanName
```

- الـ Grouping Sets: بتطرح الـ Result بتاعة الـ Cube الكبير من الـ Group by يعني بيطلع الـ Rollup علي كل Column.

```
--grouping sets
select ProductID,SalesmanName,sum(quantity) as "Quantities"
from sales
group by grouping sets(ProductID,SalesmanName)
order by SalesmanName
```

- الـ Pivot: لو عايز اعمل اسماء الـ Columns هي اسماء الـ Salesman. او بمعني اصح عايز اقلب الـ Rows لـ Columns مع الـ Aggregate.
هنلاحظ ان [Ahmed, Khalid, Ali] هم اسماء الـColumns الجديدة.

```
----Pivot and Unpivot OLAP
--if u have the result of the previouse query
SELECT * into pivoting FROM sales
PIVOT (SUM(Quantity) FOR SalesmanName IN ([Ahmed],[Khalid],[Ali])) PVT

select * from pivoting
```

- الـ Unpivot: بقلب الـ Columns لـ Rows من غير Aggregate يعني برجعه لـ group by.

```
--how to get the table
SELECT * FROM pivoting
UNPIVOT (Quantity FOR SalesmanName IN ([Ahmed],[Khalid],[Ali])) UNPVT
```

محاضـرة رقم 4

- الـ Hierarchy ID: الـ Cell الواحده فيه عبارة عن Tree فهنا مش هينفع اعمل Assign لقيم فيها مباشرة ولكن بقدر اجيب الـ Parent والتسلسل بتاعي عن طريق مجموعة من الـ Functions.
- ولذلك معظم اللي هنتكلم عليه في محاضرة اليوم حاجات بتسرع الـ Performance بتاع الـ Engine بتاعي.

- اي Record بعمل له Insert في Table معين بيتحط في الاخر ولذلك هنجد ان الـ Insert والـ Update سريعة ولكن هنجد ان الـ Select بطيئه.

```
select *
from student
where st_id = 2
```

فمثلا جملة الـ Select السابقة دي بتتنفذ ازاي ؟

- هوَّ بيبدأ من اول الـ Table ويعمل حاجه اسمها Table scan فبيعدي علي كل الـ Records الموجوده في الـ Table وذلك بفرض عدم وجود Primary Key.

Indexes

- الـ mdf. بيكون ظاهر انه File واحد علي الـ HD ولكنه في الحقيقة الـ Data File الواحد بيكون مقسم الي مجموعة من الـ Data Pages كل واحد بيكون حجمه 8KB.
- لو مش عندي اي Index او P.K فهحتاج اعمل Table Scan وبنرسم الـ Data Pages كـ Tree وهتكون مجموعة الـ Pages دي عبارة عن Nodes بتاعة الـ Tree.
- يوجد نوعين من الـ Indexes وهم:
  1. Clustered Index: لما اجي اعمل Table وبيكون له P.K هوَّ من نفسه بيعمل Create لـ Clustered Index معناه ان الـ Data اللي جايَّه علي الـ HD مُرَتَّبَه بحيث لو دخل لي 5 هتدخل في مكانها بذلك الـ Insert هتكون بطيئة وبيعمل ايضا كل مجموعة Data Pages كـ Tree ولكن الـ Select بتكون هي الاسرع.
     - ملحوظة: لما اجي اعمل Search مع وجود P.K في الـ Table فهوَّ بيبني Tree ويدخل جواها ويعمل Table Scan ولكن في Page واحده فهكون وفرت وقت.
     - ملحوظة اخري: اخر Level في الـ Clustered Index هوَّ الـ Actual Data Pages.

```
create clustered index i1
on student(st_id)
```

2. **Non-clustered Index:** لما اقدر اعمل غير الـ P.K في الـ Table وبيطلع كل الـ Unique Values ويضعها في الـ Data Pages ويعمل Pointers لاماكن الحاجة اللي بدور عليها في الـ Data Pages علي الـ H.D وبعد كده بيعمل generate Key من نفسه ويرتب الـ Data.
   - ملحوظة: الـNon-Clustered بيتعمل علي الـ Column اللي مش مترتب ويطلق علي الـ Pages في الوقت ده الـ Heap يعني اي حاجة مش مرتبه.

```
create nonclustered index i2
on student(st_age)
```

- **ملاحظات هامه:**

1. لما اعمل Table من الاول وفيه P.K بياخذ علي طول Clustered ومش ينفع اعمل Clustered تاني.
2. وفي حالة اني عملت Table مش فيه P.K فهيكون Non-Clustered وبعدها ممكن اعمل P.K في الجدول وهيكون عندي Clustered Column.
3. مسموح ليَّا ان يكون عندي في الـجدول One Clustered Column ومن (1 الي 999) من الـ Non-Clustered Column.
4. لو عندي P.K وعندي Name وعملته Non-Clustered Column وببحث بيه هيحصل الأتـــي:
   - هيدخل لـ Data Pages اللي فيها الـ Name اللي بعمل عليه Search فلما يوصل في الـ Non-Clustered علي الـ Name هيكون اخـــر Level هو الـ Pointer اللي بيشاور علي الأماكن فوَقتها الـ Pointer هيشاور علي الـ Clustered حيث ان الـ Actual Data موجود في اخر Level في الـ Clustered.
5. اذا انا لو عايز اسرع الـ Search علي اي Column اروح اعمل عليه Index لانه هيروح يعمل Search في الـ Actual Data الموجوده في الـ Data Pages.

Partitioning

- ينفع الـ Table الواحد يكون موجود في اكثر من File Group فهحتاج اعلم Partitioning.
- عشان اعلم Partition لازم اعمل 3 حاجات :
  1. اعمل Partition Function.
  2. ثم اعمل Partition Schema.
  3. ثم بعد كده اعمل Create Table with Partition.
- مع ملاحظة ان الـ DB بتاعتي لازم يكون فيها اكثـر من File Group.

اولا الـ Partition function: محتاج احدد هقسم الـ Table بتاعي عن طريق اي column والـ Data Type والـ Values اللي هاعمل Partition علي اساسها.

```
create partition function pfn(int)
as range left
for values (10,20,30)
```

ثانيا الـ Partition Schema: هنا بنربط الـ Partition Function بالـ File Group.

```
create partition scheme pschema
as partition pfn
to (fg1,fg2,fg3,fg4)
```

رابعا الـ Create Table With Partition:

```
create table t
(
fullName nvarchar(30),
age int
)on pschema(age)
```

هعمل insert في الـ Table t كالاتي:

```
insert into t values
('Farida',12)
,('Adham',9)
,('Mohab',21)
,('Darin',33)
,('Logy',25)

select * ,$partition.pfn(age)
from t
```

الـ Merge:

- **عندي System لـ Bank وفيه 2 tables واحد بيحتفظ باخر transaction عملها الـ Client والجدول التاني فيه Daily Transaction يحتوي علي كل عميل والمبلغ اللي سحبه كل يوم. بحتاج اعمل كل يوم Update لجدول الـ Last_trans باخر trans عملها العميل. ولذلك هنعمل Merge Statement(هتمسك كل Row في الجدول التاني اللي هو الـ Daily_trans ويمشي علي كل Record ويقارنه بالجدول الاول Last_trans ولو موجود يعمل له Update بالقيمه الجديدة ولو مش موجود يعمل له assign في الجدول) والمثال كالاتي:**

ASEM.ITI - dbo.daily_trans | ASEM.ITI - dbo.last_trans

	acc_id	acc_name	value
	1	Ahmed	1000
	2	Eman	2000
	3	Mazen	3000

ASEM.ITI - dbo.daily_trans | ASEM.ITI - dbo.last_trans

	id	name	Value
	1	Ahmed	5000
	2	Eman	3000
	3	Ali	4000

```
Merge into last_trans as T
using daily_trans as S
on T.acc_id=S.id
when matched then
update set T.value=S.Value
when not matched then
insert (acc_id,acc_name,value)
values(S.id,S.name,S.Value);
```

- **مثال اخر علي الـ Products:**

```
create table product_Alex
(
ProductID int,
Pname varchar(50),

)

create table product_Cairo
(
ProductID int,
Pname varchar(50),
Price int
)
```

```
Merge into [dbo].[product_Cairo] as T
using [dbo].[product_Alex] as S
On T.[ProductID]=S.[ProductID]

When matched then
update set T.=S.price

When not matched by target Then
insert(ProductID,Pname,Price)
values(S.ProductID,S.Pname,S.Price)

When not matched by Source
Then delete
```

Views

- **الـ View:** عبارة عن Select Statement او بيكون عبارة عن Virtual Table بيكون ليها مجموعة من المميزات
  1. بحدد الـ User اللي هيستخدم الـ View.
  2. بقدر اعمل Limit Access للـ DB Objects.
  3. اقدر اعمل Hide للـ DB Object عشان مفيش حد يقدر يوصل للـ Names بتاع الـ Tables والـ Data بتاعتي.
  4. بتزود الـ Performance بتاعي.
- الـ Selecting من الـ View اسرع من الـ Table في الـ Performance.
- ممكن استخدم الـ View عشان اعمل Insert, Update ولكن بـ Rules معينه.
- لو كنت عامل View علي Table وعملت Drop للـ Table فالـ View مش هيحصل ليها Drop لوكن لو رحت انفذ الـ View مش هيلاقي Table فهيطلع error.
- **انواع الـ View:**
  1. **الـ Standard View:** النوع ده مس بيحتوي علي Data.
  2. **الـ Indexed View:** ده بيكون فيه Data ووقتها هقدر اعمل Create للـ Index وهيضعه علي جزء من الـ Data فقط مش الـ Table كله.
  3. **الـ Partition View:** هو View جايب Data من Partition Server.

ملحوظة: ايه الفرق بين الـ View والـ Function؟

- الـ View مش بتاخذ Parameters ولكن الـ Function بتاخذ Parameter.

```
create view v2(studept_name,dept_name)
as
    select st_fname,dept_name
    from student s,department d
    where   s.dept_id=d.dept_id
    and s.st_address='cairo'

select * from v2
select studept_name from v2
```

```
create view v3
as
    select st_fname from student
    union
    select dept_name from department
```

ودي مجموعة اوامر بستخدمها مع الـ View:

```
sp_depends v2
sp_help v2
sp_helptext v2
sp_rename v2,v10
select * from v10
```

ولو عايز استخدم الـ Encrypted View:

```
--ENCRYPTED VIEW
CREATE VIEW V1
WITH ENCRYPTION
AS
SELECT * FROM Student

SP_HELPTEXT V1
```

الـ SP_HELPTEXT: من خلالها بقدر اجيب الـ Structure بتاع اي حاجة (مثلا الـ View).

الـ Indexed View: اي View موجود فيه Schema Binding وقتها بيكون Indexed view ووقتها هنعمل Select باستخدام الـ Default Schema (.dbo)

```
--INDEXED VIEWS
--U CAN CREATE INDEX ON VIEW AS ON TABLES

--An indexed view has been computed and stored.
--You index a view by creating a unique clustered index on it
alter VIEW VP
WITH schemabinding
AS
SELECT st_fname,st_age
FROM dbo.Student
where st_age>10


select * from VP

create unique clustered index Myindex on VP(st_age)

drop index VP.myindex
```

استخدام الـ Partition View: في حالة لو الـ Data معمول لها Partition علي اكثـر من Server.

```
--U CAN CREATE PARTITIONED VIEWS
--PARTITIONED VIEWS COMBINE DATA FROM DIFFERENT SOURCES
--PARTITIONED VIEWS lock changing in table design
CREATE VIEW vSales
AS
SELECT *  FROM SQLServerNorth.Sales.Sale
UNION ALL
SELECT *  FROM SQLServerSouth.Sales.Sale
```

استخدام الـ With Check Option مع الـ View: ينفع اعمل Update, Insert ولكن في الـ Range بتاع الـ Table فقط.

```
--with check option
----working with inset update only
alter view v13
as
    select st_id,st_fname,st_age
    from student
    where st_age=20
    with check option

select * from v13

insert into v13
values(111,'ali',33)

update v13
set st_age=30
where st_age=20
```

عندنا اربع انواع من الـ Tables في الـ SQL:

1- **الـ Standard Table: هو اي table بتعمل له Create علي الـ User Defined DB وهيتحذف عن طريق الـ Drop.**
2- **الـ Temp Table: هعمل create لـ Table جوه الـ temp DB وهيتشال لما اعمله Drop وهيكون الـ Lifetime بتاعه هو الـ Lifetime بتاع الـ Server وهينتهي معاه. ولذلك لما اجي اعمل Restart للـ Server الـ Table هيتشال لان الـ Lifetime بتاعة الـ Table هي الـ Lifetime بتاعة الـ Server.**

```
--1) shareable tables
----shared by all users per server
use tempdb
create table share_table
(
sid int
)
--droping
drop table share_table
Restart server
```

3- الـ **Local Table:** هو Table هيتعمل له Create في الـ Runtime وهيعمل الـ Lifetime بتاعته هو الـ Lifetime بتاع الـ Connection فبيكون temp table لكل User داخل (Session based Table). يعني الـ Table ده بيكون Local للـ User اللي عليه فقط ولوفيه User تاني مش هيقدر يشوفه او يتعامل معاه فكل واحد شايل نسخه لواحده اول مالـ Session بتاعة الـ User تتقفل الـ Table هيتشال.

```
--3)session based (local)
--until session ends
--use any database
create table #session
(
sid int
)
insert into #session values(1)
select * from #session
--droping
drop table #session
--Disconnecting server "end session"
```

4- الـ **Global Table:** هو Table هعمل له Create هيكون Session Based table ولكنه Shared علي مستوي عدد من الـ Users. والـ Lifetime بتاعة الـ Table هي الـ Session او الـ Lifetime بتاعة الـ Admin اللي عمل الـ Table.

```
--2)global temp tables
------shared by all users per server

create table ##share_table
(
sid int
)
--droping
drop table ##share_table
```

لو عندي جملة Query زي دي :

```
select *,(select COUNT(*) from Student) from  student
```

ازاي الـ Execution ده بيتم ؟

- هيجيب اول Row من الـ Student وبعدين هيلاقي Subquery هيروح يجيب الـ Data بتاعة الـ Student ويحسب الـ Count بتاعها وبعدين يعمل كده في الـ Row التاني والتالت وهكذا. ودي تعتبر مشكلة لان لو عندي

مليون Row فبذلك الكلام السابق ده هيتنفذ مليون مره ولذلك الحل اني ادور علي Variable مثلا استخدمه ولذلك هنستخدم الـ CTE(Common Type Expression).

- الـ CTE: هي View ولكنها في الـ Memory بعد ما بنخلص العملية بتاعة الـ Select الـ View بيتشال.

```
--CTE
with cte (sid)as
(
select st_id
from Student

)
select *,(select COUNT(*) from cte) from  cte
```

الـ Offset والـ Fetch: (او يطلق عليها الـ Paging او الـ Page Data): يعني لو عايز اعمل Select لـ 5 Records اي مابعد الـ Record الخامس. هنستخدم الـ Offset يعني هيروح يقف علي الـ Record الخامس وبعدين اعمل Select.

```
--offset and fetch is called server side pageing
--Fetch and Offset or Page Data
--like data paging in data grid
SELECT *
FROM student
ORDER BY st_age
OFFSET 5 ROWS
FETCH NEXT 5 ROWS ONLY;

--in SQL 2008
SELECT *
FROM (
SELECT *,ROW_NUMBER() OVER(ORDER BY st_age) AS age
FROM student) AS TempTable
WHERE age > 5 and age <= 10
```

محاضـرة رقم 5

Stored Procedure

لو كنت بعمل Run لجملة Select جوه الـ DB، جملة الـ Query دي لازم تمر ببعض المراحل عشان تطلع الـ Query Set:

1- الـ Parsing: الـ Engine نفسه بيتاكد ان كل الـ Statement اللي استخدمها صحيحة.
2- الـ Optimization: هو Parsing لكل الـ Keywords اللي موجوده فبيتاكد ان الـ Objects اللي المستخدمه جوه الـ DB.
3- الـ Query Tree: كل Query له اكثـر من طريقه في التنفيذ وكل طريقه يطلق عليها Query Tree وبنختار افضل Query Tree فمثلا
   a. ممكن ننفذ الـ Subquery الاول وبعدين الـ Join وبعدين الـ Select.
   b. وممكن ننفذ الـ Select ثم الـ Join وبعدين ينفذ الـ Subquery.
4- الـ Query Plan: بيدخل الـ Memory ويعمل حاجة اسمها Query Plan وبعدين هترجع لنا Result.

ولو عملت Run لـ Select تاني هيدخل في Cycle ويعمل نفس الخطوات السابقه بالترتيب.

الـ Stored Procedure: حاجة زي الـ Function بضع فيها Select, Insert Statement. او عبارة عن جملة Query مخزنه علي الـ Server.

مميزات الـ SP:

- الـ Performance بتاع الـ Stored افضل من اني اكتب Query حيث اني وفرت علي نفسي الـ Cycle اللي اي Query بتمر بيها وهي (*Parsing → Optimization → Query Tree → Query Plan*).
  1. ازاي ده بيتم ؟
     - في الـ Stored الخطوات دي بتتنفذ اول مره وبتتخزن علي الـ Server ولذلك تاني مره بيتنفذ علي طول لان حصل ليها Execution قبل كده ومرت بالاربع مراحل.
- بستخدم الـ Stored في الـ Catch Errors قبل ماتحصل في الـ DB فهعمل ليها Validate علي الـ Business Rule ولو صحيحة هتدخل الـDB فمثلا لو وضعت Insert في الـ Stored فوقتها هقدر اتاكد من كل القيم بتاعتي فممكن اضع شرط لو اتحقق يدخل يعمل Insert ولو مش اتحقق يطلّع لي Message مثلا.
- الـ SP بيحقق الـ Security من خلال انه في حالة الـ Run بنادي عليه علي طول.
- بيمنع الـ SQL Injection وهي طريقه بيحصل بيها Hacking علي الـSQL.
- بيقلل الـ Network Traffic يعني عملية ارساله علي الـ Network اسرع لاني ببعت اسم الـ Stored دي فقط وبيروح يتنفذ علي الـ Server وده بيقلل الـ Traffic.

انواع الـ Stored Procedure:

- الـ Built in SP: اي حاجة بتبدا بـ SP ودي بيتم استخدامها علي طول.
- الـ User Defined SP: يعني هكتب SP بنفسي.

ملحوظة: الـ SP بيحصل ليها Create علي مستوي الـ DB وبتوجد في الـ Programability.

```sql
create proc p1
as
select ins_name,salary
from instructor where ins_id=10
return

[exec] p1
```

```sql
create proc p4
as
insert into Student(st_id) values(99)

exec p4
```

يوجد نوعين من الـ Passing Parameters جوه الـ SP:

- **Passing by Position: يعني بعمل passing للـ Parameters بالترتيب فالـ Value الاولي اللي بكتبها بتروح للـ Parameter الاول وهكذا.**
- **Passing by Name: لازم اكون عارف اسماء الـ Parameters اللي عندي ولذلك مش محتاج يكون عندي ترتيب.**

انواع الـ Parameters اللي بتُستخدم جوه الـ SP:

- **Input Parameter: هو Parameter لازم يتبعت مع الـ User فلازم ابعت القيمه بتاعته.**

```sql
--input parameters

create proc dd @x int,@y int=10
as
select @x,@y
```

```sql
create proc deptname
(@fname varchar(40), @lname varchar(20)=NULL)
    as
      select d_name
        from student s,department d
        where st_fname = @fname
        and st_lname = @lname
        and s.d_id=d.d_id
    return

deptname 'ahmed','kariem'
```

- **Output Parameter:**

```
--output parameter
alter PROC P10 @id INT,@age int output
as
select @age=St_Age from Student where St_Id=@id
return 100

declare @age int
declare @r int
exec @r=p10 1,@age output
select @age,@r
```

هنا هاستقبل النتيجة اللي جايه من الـ P10 في الـ @age والـ @r.

- **Return Parameter: تستخدم للـ Validation يعني الـ Return عامله زي الـ Function ولكن مش بستخدمها دايما لان لو كان عندي اكثر من قيمه عايز اعمل لها return فبذلك هحتاج ارجع كل قيمه لوحدها.**
  1. **اي قيمه موجوده جنب الـ Return لاتستخدم عشان اعمل return value فهي بتمثل error معين او حاجة حصلت فهي بتستخدم عشان تعمل indicate للـ Developer ايه هو اللي حصل جوه الـ SP.**
  2. **الـ Return Parameters بتكون Standard للـ Developers يعني مثلا Return 1 فرقم 1 ده او اي رقم كل الـ Developers متفقين علي انه بيعبر عن حاجة معينه.**

```
Return 1
```

الـ Recompile: بعمل Run للـ Stored Procedure تاني عن طريق الـ Recompile. فالـ Recompile بتخلق الـ Query من اول وجديد (كاني بعمل Refresh للـ Query) فالـ Run بتاع الـ Recompile زي الـ Run بتاع الـ Query ولكن الـ SP مازالت محتفظة بمميزاتها. وممكن اكتب في الـ SP نفسه With Recompile لو كانت الداتا بتاعتي بتتغير بشكل سريع فهحتاج اعمل Recompile علي طول كل شوية ولكن استخداماتها بتكون قليلة.

```
exec sp_recompile dd

Create proc p10
with recompile
as
select @x+@y
```

وايضا ممكن استخدم With Encryption مع الـ SP.

ملحوظة: بنكتب الـ Execute في حالة لو عايز اعمل Run لـــ 2 Statement في نفس الوقت.

ودي مجموعة امثلة علي الـ SP:

```
------------Example(1)---------------
create proc p22       -- procedure that don't take parameter
as
declare @x nvarchar(20)
set @x='test'
print @x             --without return

p22  --calling or excuting it
GO
```

```
-----------Example(2)-------------------
create  proc p23(@x nvarchar(20) output) -- proc takes output parameter
as
select @x='test'         --without return

declare @y nvarchar(20)
exec p23 @y output
select @y
```

```
------------Example(3)-----------------------------------------
GO
create proc p24
as
begin
    exec('create proc vv as select * from student')
end
/*
1st time command exec succf.
2nd time
Msg 2714, Level 16, State 3, Procedure vv, Line 4
There is already an object named 'vv' in the database.
*/
p24 -- calling or excuting
```

```
----------Example(4)-------------------------------------
Go
create proc pp(@id int)--proc that take input parameter
as
select @id

pp @id=8--work correctly

pp @id=8
select @id=@id+3
select @id
/*Msg 137, Level 15, State 2, Line 2
Must declare the scalar variable "@id".
*/
```

Trigger

- هو حالة خاصة من الـ SP.
- وبيحصل له Firing والـ Firing بيكون According للـ Action اللي بيتعمل للـ Trigger عملية Create عليه.
- لو عملت (delete, insert) وكاتب Trigger بيتعمل له Firing بعد ماانفذ الـ Insert Statement او انفذ حاجة معينه لما اعمل Delete.
- ملحوظة: لما اعمل Create لـ Trigger علي مستوي الـ DB بيمنع اني اعمل Create لاي Object بنفس الاسم مثلا (view - table).
- هنعمل Create لـ Table اسمه Action:

```
-Triggers

create table actions
(
username nvarchar(50),
actiondate datetime
)
```

انواع الـ Triggers:

- **الـ After Trigger :** وبنحددها باستخدام الـ Keyword اللي اسمها after او الـ For.

```
create trigger t1
on student
for insert
as
insert into actions values(suser_name(),getdate())

insert into student(st_id,st_fname) values (400,'rami')

SELECT suser_name()
--delete from student where st_id=44

select * from actions
-----------------------------------------------------


create trigger t2
    on student
    for insert
    as
      if datename (dw,getdate()) = 'sunday'
        begin
          select 'error'
          rollback tran
end

begin tran t1
insert into student(st_id,st_fname)
values (450,'rami')
commit

-----------------------------------------------------

drop trigger t2

alter table student
[disable,enable] trigger t1
```

ممكن اعمل Disable للـ Trigger بدلا من اني اعمل drop للـ Trigger ولما احتاجه اعمل له enable.

```
alter trigger t4
on student
for delete,insert,update
    as
    select * from deleted
    select * from inserted

insert into student(st_id,st_fname) values (89,'rami')
delete from student where st_id=89

update student set st_fname='ahmed' where st_id=89
```

:Instead of Trigger الـ

```
create trigger t5
on student
instead of delete
    as
    select 'not allowed'

insert into student(st_id,st_fname) values (89,'rami')
delete from student where st_id=89

update student set st_fname='ahmed' where st_id=89
```

What is the difference between after and instead of trigger?

- **AFTER triggers fire after the triggering action occurs.**
- **INSTEAD OF triggers are executed instead of the corresponding triggering action.**
- **AFTER triggers can be created only on tables**
- **INSTEAD OF triggers can be created on both tables and views**

Task:
المطلوب اني امنع اي حد يعمل (Insert, Update) يوم الجمعة مثلا على جدول الـ Student:

```
Create Trigger
on Student
after insert, update, delete
as
if(datename(mm,getdate())='friday')
rollback
```

ملحوظة: كل Trigger بيحصل له Fire بيتعمل Create جوَّاه لـ Table اسمه inserted وTable اسمه deleted. وبيكونوا بنفس الـ Structure بتاع الـ Trigger. واسماء الـ Columns اللي موجوده فيهم هي نفس اللي موجوده في الـStructure.

1. لو بعمل Delete Statement وقتها الـ Deleted Table هيكون فيه Data والـ Inseted فاضي.
2. لو بعمل Insert Statement وقتها الـ Inserted Table هيكون فيه data والـ Deleted فاضي.

Task:
المطلوب اعمل Table واعمل Trigger عليه وعايز لما حد يعمل Delete يرجع لي الـ id, Name بتاعه وايضا عايزه يرجع لي الـ Date:

```
Create table auditing
(
    id int,
    sname nvarchar(50),
    Tdate Date
)

create triger t8
on Topic
instead of Delete
as
declare @id int
select @id=top_id from deleted
insert into auditing
values(@id,suder_name(),getdate())

delete from topic
where top_id=20
```

مشكلة الكود ده اني لو بعمل Delete اكثر من Row فهيحتفظ باخر Row فقط والحل بتاعها Insert based on select.

Sequence

هو احد الـ Queries اللي موجوده في SQL Server 2012 وبيكون علي مستوي الـ DB مش علي مستوي الـ Table فممكن Two tables هيمشوا علي نفس الـ Sequence وهيحصل لهم Run مع بعض ولذلك هو مثل الـ Identity بس علي مستوي الـ DB.

```
--Create Sequence Object
Create SEQUENCE MySequence
START WITH 1
INCREMENT BY 1
MinValue 1
MaxValue 5
CYCLE; --default

alter SEQUENCE MySequence
ReSTART WITH 1 --changed from start to restart
INCREMENT BY 1
CYCLE; --default

drop SEQUENCE MySequence
```

في المثال السابق هيعد من 1 الي 5 وبعدين يكرر تاني العد وهنلاحظ انه بيعمل Cycle وبيبدا من الـ MinValue اللي هي 1 ولو عملت NoCYCLE هيقف لحد 5 ومش هيتكرر.

عشان انفذ الـ Sequence:

- هروح اعمل Create لـ Tables 2 وادخل فيهم Data، كل Insert بيروح يدخل في الـ Table اللي بحدده وبيصنع Sequence بالترتيب، يعني لو عملت Insert في الـ Table الاول مرتين هيكون الـ ID فيه 1و2 وبعدين هعمل Insert في الـ Table التاني الـ ID هيكون بـ 3 وهعمل Insert في الـ Table الاول الـ ID هيكون بـ 4 وهكذا.

استخدام الـ Sequence:

- لو عندي tables 2 وعايز اعرف هو الـ Record دخل قبل اي Record التاني ولا لا.

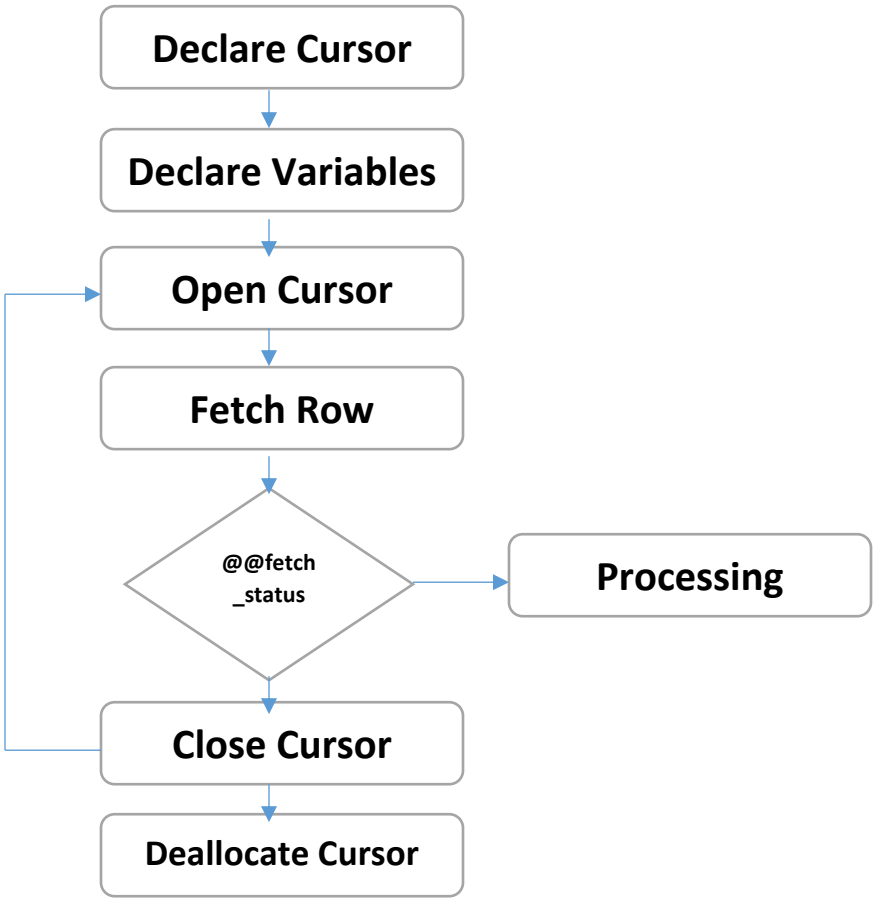
مثال هام:

```
create TABLE person1
( ID int,
 FullName nvarchar(100) NOT NULL );

create TABLE person2
( ID int,
 FullName nvarchar(100) NOT NULL );

drop table person1
drop table person2

truncate table person1
truncate table person2
```

```sql
-- Insert Some Data
INSERT into person1
VALUES (NEXT VALUE FOR MySequence, 'ahmed')

INSERT into person2
VALUES (NEXT VALUE FOR MySequence, 'ahmed1')


select * from person1
select * from person2


select name,minimum_value,maximum_value,current_value,is_cycling
from sys.sequences
where name='Mysequence'

update person1
set id=NEXT VALUE FOR MySequence
where id=2
```

Cursor

عبارة عن Select Statement فبترجع لي الـ Rsult Set فالـ Cursor بيمسك الناتج بتاع الـ Select وبيكون row by row قبل مايرجع للـ Client. وبعدين هعرف الـ Variables علي قد الـ Columns اللي عامل ليها Select عشان اضع فيها القيم اللي هترجع للـ Client.



الـ Fetch Row: يعني هيمسك كل Row وينفذ.

الـ Processing: الـ Cursor بياخذ Memory كبيـرة اوي.

الـ Deallocate Cursor: بتعمل Remove لكل الـ Spaces اللي حجزها الـ Cursor جوه المكان.

مثال1:

```
declare s_cur cursor
    for select st_id,st_fname from student
    for read only  --update


declare @id int
declare @name nvarchar(50)
open s_cur
fetch s_cur into @id,@name
begin
    While @@fetch_status=0  --returns 0 success -- 1 failed  --2 no more rows to fetch
    begin
        select @id,@name
        fetch s_cur into @id,@name
    end
end
close s_cur
deallocate s_cur
```

مثال2:

```
declare t_cur cursor
    for select t_id,t_name,sal from teacher
    for update

declare @id int
declare @name nvarchar(50)
declare @s int
open t_cur
fetch t_cur into @id,@name,@s
begin
    While @@fetch_status=0
    begin
        if @s<1000
            begin
                update teacher  set sal= @s*1.10
                where current of t_cur
        fetch t_cur into @id,@name,@s
            end
    end
end
close t_cur
deallocate t_cur
```

ملحوظة:

- "*Where current of t_cur*" دي عشان يروح يعمل Update للـ Record اللي انا واقف عليه.
- "*Fetch t_cur into @id, @name, @s*" لازم اكتبها عشان يطلع من الـ Loop.
- الـ Fetch زي الـ Counter.

Backup & Restore

ممكن الـ DB يحصل ليها Crash في اي وقت من الاوقات ولذلك لازم اعمل Backup ويوجد 4 انواع:

1- Full Backup: لو عندي DB عملت ليها create في 2012/1/1 وجيت في 2012/12/1 وقررت اعمل Full Backup فهيجيب كل الـ Data من الـ Data File فكل مره هيعمل كده هياخذ وقت كبير اوي.
2- Differential Backup: بيشوف اخر Full Backup
3- Transaction log Backup: بيشوف اخر Backup كان موجود سواء Full او Differential وبعدين بيعمل Backup للـ Queries اللي في الـ Log File فبيروح للـLog File وياخذ Queries ويعمل ليهم Backup ويفضي الـ Log File. فالنوع ده سريع ولو قررت اني ارجع الـ DB في يوم معين فهو بياخذ كل Query والـ time بتاعها فبيدخل جوه الـ Log File وبيعمل Map علي الـ Time بتاع الـ Query.
4- File Group: لو كانت الـ DB بتاعتي كبيره فهعمل File Group Backup فهعمل Backup لكل File Group لوحده عشان مش ياخذ وقت كبير.

XML

هقدر احول الـ DB بتاعتي الي XML او العكس.

عندنا 4 Modes للـ For XML Clause:

- **الـ For XML Row:** دي بتحوللي كل Row في الـ Result Set الي XML Element.

```
select * from Student
for xml raw

select * from Student
for xml raw('Student')

select * from Student
for xml raw('Student'),ELEMENTS


select * from Student
for xml raw('Student'),ELEMENTS,ROOT


select * from Student
for xml raw('Student'),ELEMENTS,ROOT('STUDENTS')

--how to show null values in xml
select * from Student
for xml raw('Student'),ELEMENTS xsinil,ROOT('STUDENTS')
```

- **الـ For XML Auto:**

```
select Topic.Top_Id,Top_Name,Crs_Id,Crs_Name
from Topic ,Course
where topic.Top_Id=Course.Top_Id
for xml raw,elements

select *
from Student as st
for xml auto

select *
from Student
for xml auto,elements

select *
from Student
for xml auto,elements,root('ALLstudents')
```

- الـ For XML Explicit:

```
select 1 as tag,NULL as parent,
       Top_id as [Topic!1!TopicID],
       Top_name as [Topic!1!Name!element]
from Topic
for xml explicit

-------------------------------------------------------------------------
--  Tag  --  Parent  -- Topic!1!TopicID  --  Topic!1!Name!element
--   1   --   NULL   --        1         --           Ahmed
--   1   --   NULL   --        2         --           Ali
-------------------------------------------------------------------------
```

- الـ For XML Path:

```
select st_id "@StudentID",
       St_Fname "StudentName/FirstName",
       St_Lname "StudentName/LastName",
       St_Address "Address"
from Student
for xml path

select st_id "@StudentID",
       St_Fname "StudentName/@FirstName",
       St_Lname "StudentName/LastName",
       St_Address "Address"
from Student
for xml path('Student'),root('Students')

select Topic.Top_Id "@TopicID" ,Top_Name "Name",
       (select Crs_Id "CourseID",Crs_Name "CourseName"
        from Course
        where topic.Top_Id=Course.Top_Id
        for xml pATH('Course'),TYPE,root('Courses')
       )
from Topic
for xml path('Topic'),root('Topic_Courses')
```
