Contents

SQL: Day1 5
- Database 5
- Database Life Cycle 5
- File Based_System VS DB_System 5
- ER Diagram 6
- Entities 6
- Attributes 6
- Relationships 7
  - Degree of Relationships 7
  - Cardinality of Relationships 7
  - Participation of Relationships 7
- Keys (PK, FK.....) 8
- Database VS Database Management System VS Database System 8

SQL: Day 2 10
- ERD to Relational Mapping Steps 10
- Step1: Mapping of Regular Entity Types 11
- Step2: Mapping of Weak Entity Types 12
- Step3: Mapping of Binary one-one Relation Types 12
- Step4: Mapping of Binary one-many Relation Types 13
- Step5: Mapping of Binary many-many Relation Types 14
- Step6: Mapping of Ternary Relationship Types 14
- Step7: Mapping of Unary Relationship 15
- ANSI SQL (DDL, DML, DCL, DQL, TCL) 15

SQL: Day 3 16
- Join ..... ON 16
- Self Join 16
- Database Diagram(Video 10:40) 16
- IS NULL / Coalesce 16
- Update .... Set 17
- Rename Table 17
- Rename Column 17
- Trim Functions 17
- Replace +Trim Function 17
- Like 17
- Order by 17
- Normalization 18
  - Functional Dependency 18
- Steps in Normalization 19

Agile Development 20
- Software Development Life Cycle 20
- What is The Project? 20
- What is The Project Management? 20
- Predictive VS Adaptive 20
- Project Management Triangle 20
- Waterfall Vs Agile 21
- What is Agile? 21
- Agile Values 21
- Agile Scrum Framework 22
- Backlog Grooming 22
- Agile Important Tools 22
- Agile Scrum Steps 23

SQL: Day 4 24
- Aggregate Functions 24
- Group by 24
- Having 25
- Subqueries 25
- Union Family 26
  - Union all 26
  - Union 26
  - Intersect 26
  - Except 26
- EERD (Enhanced ERD) 27
- Super Entities / Sub Entities Relationships 27
- Constraints in Super Entity 28
  - Completeness Constraints 28
  - Disjointness Constraints 28
- Mapping of Super Entities / Sub Entities Relationships 29
- Built in Functions(substring(),Year(),Month(),Database name,user name) 29

View Template Explorer (Video 02:56:10).....29
SQL: Day 5 30
SQL Server Engine Presentation(Video 02:32) 30
Setup Instance Vs Services(Video 15:00).....30
Windows Authorization (Video 35:00) 30
Windows Authentication (Video 35:00) 30
Execution Orders Sequence.....30
Selects Top () 30
Select…., with ties 30
Select…., newid () 30
The Object Path 31
Insert Data Ways.....31
Create [new_table] from [old_Table].....31
1)Extract the Script of the old table with constrains (Video 01:24:30). 31
2)(Select …..Into[new_table]From [old_table]) 31
(Copy & Past)Tables with data in the same database 31
or to anther database 31
(Copy & Past)Tables Structure without data in the same database.....31
Or to anther database 31
Insert based on Select.....32
Using Having without Group by(Video 1:28:10)..32
Data Types 32
Ranking Functions.....33
Row_Number () / Dense_Rank ().....33
N tile () 33
Denis_Rank () Vs Rank().....33
Case When 35
IIF() 35
Formate() 35
SQL: Day 6 36
Database Properties (mdf, ldf, File Groups) (Video 00:2:00).....36
Database Properties (mdf, ldf, File Groups) Implementation(Video 00:16:00) 36
Column Properties(Identity, Default Value, Formula Computed Column, Is Sparse)(Video 00:23:20).....36
Relationship Properties(Delete &Update Rule)(Set Null & Cascade)(Video 00:37:10).....36
Create Schema 36
User/log in Permissions(Grant, with Grant, Deny) (Video 1:02:00) 36
Create Synonym.....36
Drop Vs Delete Vs Truncate 36
Database Integrity 37
Domain Integrity (Columns).....37
Entity Integrity (Raws) 37
Referential Integrity (Relationship) 37
Constraints.....37
Create Constraint(Video 02:12:00).....37
Add Constraint to table.....38
Drop Constraint.....38
Create Rule 38
Create Default.....38
Create Data Type 39
SQL: Day 7 40
Variables 40
Local Variables 40
Execute Vs Select 41
Global Variables 41
Control Of Flow Statements 41
If 41
If Exists 42
If Not Exists 42
Try…. Catch 42
While 42
Batch 43
Script 43
Transaction (Query → Ldf → mdf).....43
Functions.....44
Bult in Functions (upper,Lean…..) 44
User Defined Functions.....46

SQL: Day 8 48
Index 48
Clustered index 48
Non-Clustered index 48
SQL Server Profiler (Video 39:00) 49
SQL Server Tuning Advisor (Video 39:00) 49
Types of System Databases (master, model, msdb, tempdb) 49
Physical & Local & Global Tables 49
Rollup 50
Cube 50
Grouping sets 50
Pivot 50
View 51
Standard View 51
Partitioned View 51
Sp_helptext () 52
Encryption 52
DML on the View 52
DML View of one Table 52
DML View of Multi-Tables 53
Indexed View 53
Merge into 54
SQL: Day9 55
Stored Procedures 55
The Cycle of the query in the Engine 55
Type of Stored-procedures 56
Bult in Stored-procedure 56
User Defined Stored-procedure 56
Types of Parameters in Stored Procedures 58
Trigger 59
Inserted Table / Deleted Table 60
Output 62
XML: Day 1 63
SGML & HTML 63
XML for What? 63
XML is not for What? 63
How to Write Well Formed XML? 63
How to Validate XML? 64
XML DTD Validation 64
XML: Day 2 67
DTD Issues 67
XML Schema Validation 67
XML Schema Types 68
Simple Type 68
Complex Type 68
XML: Day 3 74
XML Presentation 74
CSS Issues with XML Presentation 74
XSL VS XSLT 74
XPath 75
XSL 76
Day 9 Part 2 80
XML in SQL 80
For XML (Table to XML) 80
Open XML (XML to Table) 81
SQL: Day 10 82
Cursor 82
Backup / Restore (Video 1:13:00) 85
Types Of Backups 85
The Right Way to Do Backup 85
The Three Ways to Create Backup 86
Identity Insert (ON / OFF) 86
How to Reset Identity 86
Bulk Insert 87
Snapshot 87
BI: Day 1 88
BI ETL Life Cycle 88
Operation DB (OLTP) VS Data Warehouse (OLAP) 89
SSIS 89
SSIS Packages 89
BI: Day 2 90
BI ELT Life Cycle 90

DWH Definition.....90
Why we use OLAP not OLTP? 91
Data Mart Definition.....91
Types of data mart.....91
Dependent Data Mart.....91
In-dependent Data Mart.....91
Cube Definition.....92
OLAP Definition.....92
Schema 92
The Level Of Granularity.....93
The Cube.....93
The Roll-up.....93
The Drill-down 93
Cube Types.....94
OLAP & ROLAP & HOLAP 94
The KPI.....94
BI: Day 3.....95
The Reports.....95
Tabular Report.....95
Matrix Report.....95
Chart Report 95
Freeform Report 95
The Cycle of Reporting (Video 2:23:00).....95
Top SQL Server Interview Questions for 2023.....96

.....

ITI_Data Base

Day 1

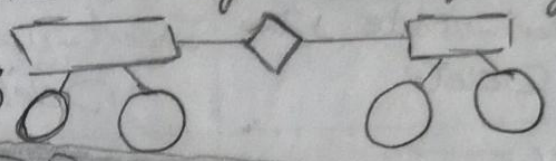
→ Data Base Life cycle

① Analysis → System analyst (Scope [Req Doc])

② Data_Base Designe → DB_Designer (ERD)

Entity Relationship Diagram ← هو طريقة لتحويل الـ Req doc أو الكلام الموصل لرسمة

من خلالها تقدر الـ Components الموجودة في الـ System والعلاقات ما بين الـ Components وبعضها وتكون من ① الـ Entities ② الـ Attributes ③ الـ Relationships



③ Data_Base Mapping → DB_Designer

* Set of Rules
* Acual Schema
* Tables

④ Data_Base Implementation → DB_Developer

* Tools → SQL Server
  MySQL
  Oracle

⊛ Shared Data_Base

|DB server| → |SQL Queries|

⑤ Application → App_Programmer

GUI (Graphical user interface)

* Mobile app
* Web_site
* Desktop

→ |Application Server| ← |access Stored Data|

⑥ EndUser → Client

* browser → |URL|

SQL: Day1 Database

⇒ File Based_System

Delimited File

1, Ahmed, 22, 10
2, Ali, 24, 70
3, emam, 25
⋮

student.txt

Fixedwidth File

10 java 2050
30 admin

Dept.txt

⇒ |Disadvantages|

① Difficult search
② Low Performance
③ Separated Copies
④ No Relationship
⑤ No DB Integrity
⑥ DB Duplication
⑦ Long Development Time
⑧ securites & permissions
⑨ Constrains & Rules
⑩ No Data quality
⑪ Manual Backup & Restore
⑫ No Standard
⑬ Difficult integrations

DB_System

Tables & Relationship

Sid	Sname	
1	Ahmed	10
2	Ali	70
⋮	⋮	⋮

foreigne key

Did	Dname
10	SD
20	IS
⋮	⋮

⇒ |advantages|

① one Standard
② Meta Data + Data
③ Column (Data type)
④ primary Key (unique_Not Null)
⑤ Centeralized DB
⑥ No_Redundancy

Disadvantages

* It needs expertise touse
* It's expensive
* It may be incompatible with other available DBMS

⇒ The ER Model

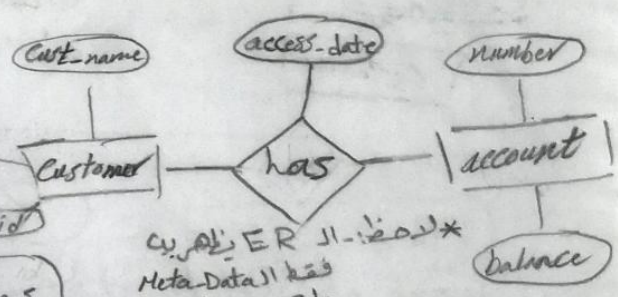
① Entities → Person, place, Object, event.
② Attributes → property or characteristics of an entity.
③ Relationships → Link between entities.

أي Component هي في الـ System مخزنة أو Data Store.

بواسطة الـ Primary Key
والـ Foreign Key

⇒ ER Diagram

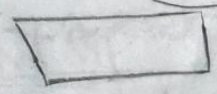
① Rectangles → The entity sets (noun)
② Diamonds → The Relationship sets (verb)
③ Ellipses → The attributes (noun)



* لاحظ: الـ ER بيظهر بس فقط الـ Meta-Data وليس الـ Data

The Entities

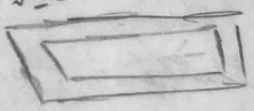
Strong Entity



← وجوده لا يعتمد على وجود Entity آخر

→ Primary Key :-

Weak Entity



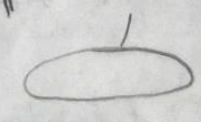
← وجوده يعتمد على وجود Entity آخر
زي الـ account و transactions عند حذف الأكونت يتم مسح وحذف التراكشنز إلي عليه
← لاحظ إن شرط الـ Weak Entity يبقى Weak في جميع الـ Systems بيختلف من System لآخر

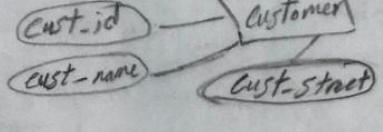
→ Partial Key :-

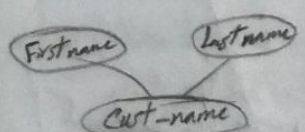
ER Diagram
Entities
Attributes

Types of Attributes

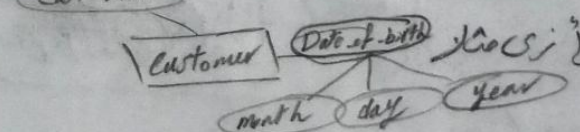
① Simple Attribute :- هو الـ Attribute الذي لا يجزأ ولا يتكرر ولا يحسب في الـ runtime لنفس الشخص مثلاً.



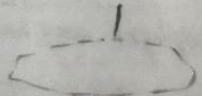


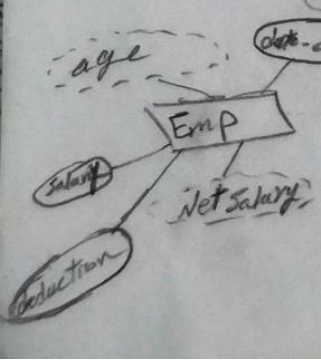


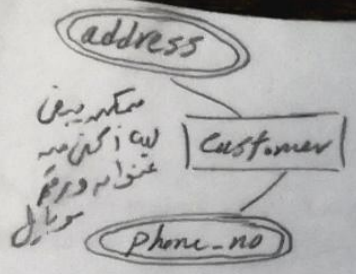
② Composite Attribute :- هو الـ Attribute إلي ينفع يتجزأ زي مثلاً



③ Derived Attribute :- هو الـ Attribute إلي ينفع يتحسب في الـ runtime

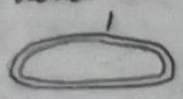


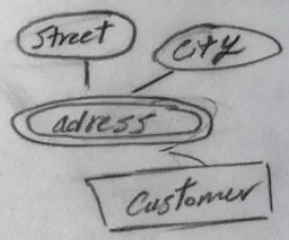




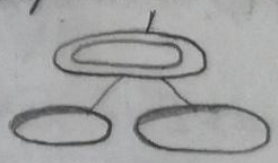
ممكن يبقى ليه أكتر من عنوان ورقم موبايل

④ Multi_Valued Attribute :- هو الحاجة اللي بتكون لنفس الشخص





⑤ Complex Attribute :- هو عبارة عن Multi-Valued + Composite متجمعين مع بعض زي مثلا الـ Customer له كذا عنوان فيبقى Multi-Valued وكل عنوان مجزأ يبقى كدا Composite .



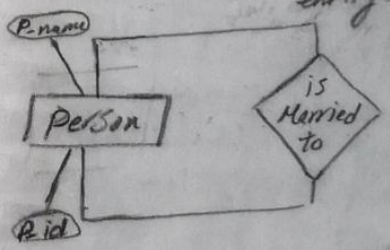
Relationship

Relationships
Degree of Relationships
Cardinality of Relationships
Participation of Relationships

⇒ Degree of Relationships :-

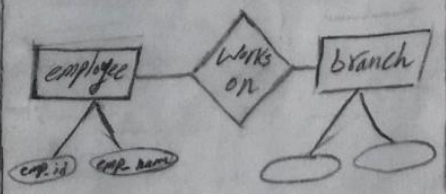
[1] Unary

طرفين الـ Relationship من نفس الـ entity



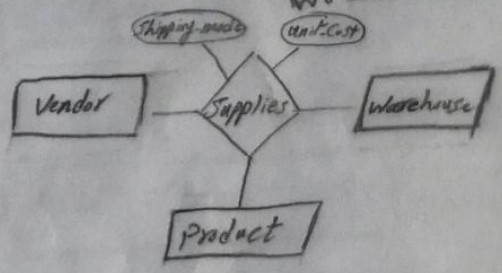
[2] Binary

طرفين الـ Relationship عبارة عن الـ Two entity مختلفين



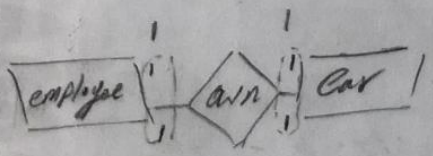
[3] Ternary

هنا إنت بتوصل الـ Diamond بأكتر من Two entities

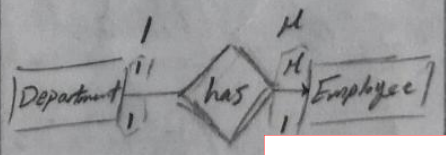


⇒ Cardinality of Relationships :-

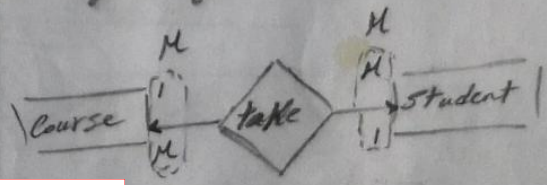
[1] one_one Relationship

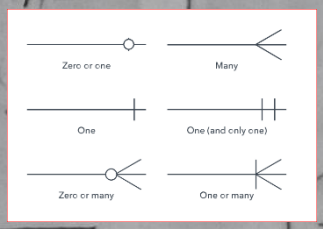


[2] one_Many Relationship

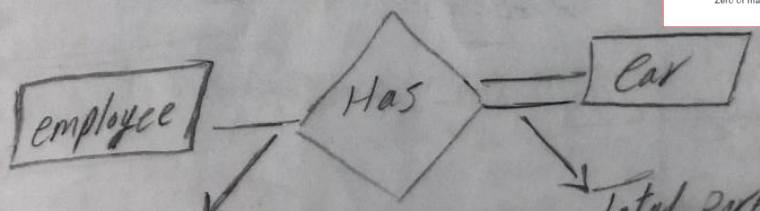


[3] Many_Many Relationship





⇒ Participation of Relationships :-



أي Weak entity لازم تبقى Total Participation
كده معناه شرط كل الـ Total تبقى Weak

Partial Participation

* An employee may have a Car.

Total participation

* A Car must be assigned to employee

Zero or more
optional
may

← Key Words →

one or more
Mandatory
must

The Keys

1. Candidate Key
2. Primary Key → with any entity يوضع قيمة خاصة للتمييز
3. Forergne Key
4. Composite Key
5. Partial Key → with weak entity
6. Alternate Key
7. Super Key

Keys (PK, FK.....)
Database VS Database Management System VS Database System

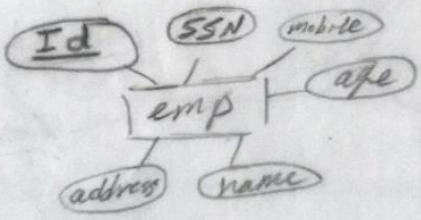
The Candidate Key

* هو مرحلة تسبق إختيار الـ Primary بحيث ممكن ان يكون هناك أكثر من Candidate key موجود ويتم اختيار واحد منهم فقط يكون Primary كما يلي

ID
SSN
Mobile
} Candidate keys

↓ so

ID
or
SSN
or
Mobile
} Primary key



The Composite Key

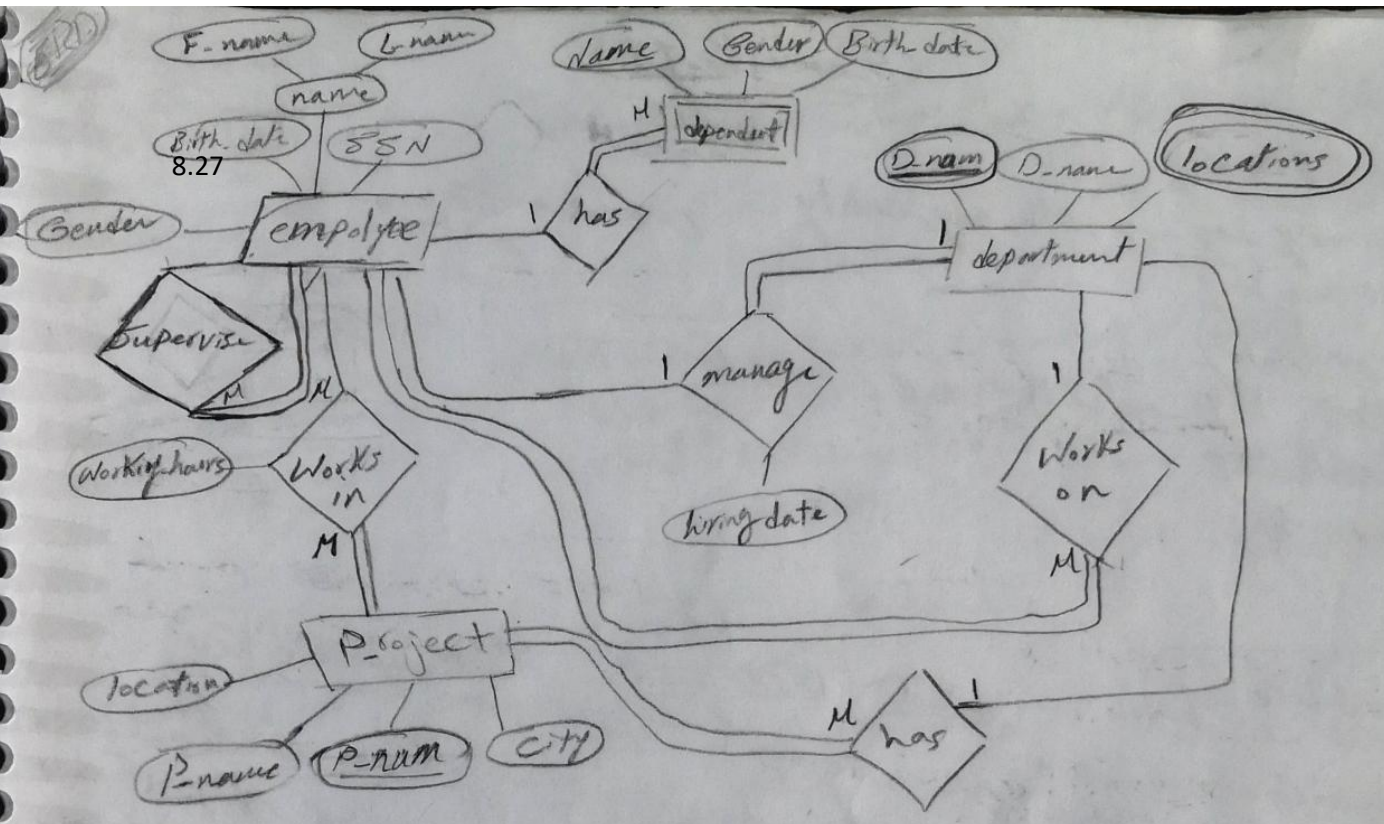
* إذا لم يوجد attribute ينفع يبقى Primary key بحيث يكون unique و notnull مأخوذ من 2 attributes مع بعض ونخليهم new attribute ويبقى هو الـ Primary

Mobile, SSN, ID
name, address
كأن المثال باعتبار فيش موجود الـ primary key هيكون

Database → A collection of Related data ← البيانات الموجودة عندنا

Data Management System (DBMS) → A Software Package ← الـ Tool اللي بتستخدمها في إنشاء الـ Database زي مثلا SQL server - My SQL

Database system → Software + Database ← عبارة عن الـ App اللي بيعرض الداتا مع الـ Database System



8.27

The Mapping

Employee_Table

SSN	BD	gender	F_name	L_name	D_num (FK)	SUP_Id (F.K)

Department_Table

D_Num	D_name	Mng SSN (FK)	Hire_Date

Dep_Locations_Table

DNum	Loc.

Project Table

P_num	P_name	City	loc	D_num (FK)

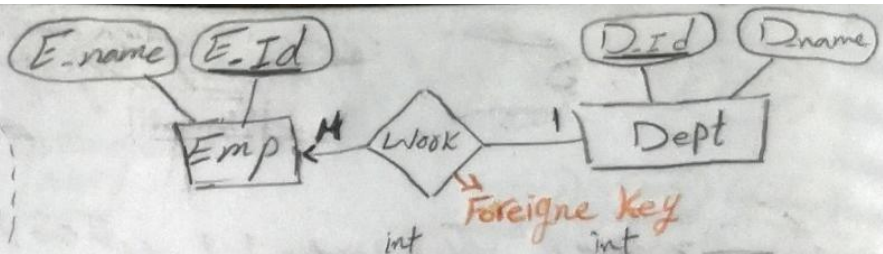
Dependent_Table

SSN	D_name	BD	gender

Emp_Project_Table (Work Table)

SSN	P_num	Working hours

Day 2



M ←	1
Child ←	Parent

int

EId	Ename	DId	
1	ahmed	10	ok
2	Ali	300	XXX
3	eman	10	ok
		null	

int

D_Id	Dname	
10	SD	Del.. XXX
20	IS	Del.. ok
30	CS	

⊛ كل Entity بيتحول لـ Table والـ Columns الخاصة بيه هي الـ attributes

⊛ كل relationship بتتحول إلى Foreigne Key وهو عبارة عن Primary key في Table أخرى.

⊛ لا يشترط الـ FK أن يكون بنفس اسم الـ PK ولكن لابد وأن يكون بنفس النوع Int → Int

⊛ ممكن أن يحتوي الـ FK على Null أو قيم متكررة أي أنه لا يشترط أن يكون unique و notnull كما في الـ PK. لكنه جزء من الـ PK في الـ Parent Table

لا يسمح أو Delet. أو Edit حاجة من الـ Parent Table لـ Child Table إلا إذا كانت القيمة الموجودة في الـ Parent غير مستخدمة في الـ Child.

ERD-to-Relational Mapping Steps

Step1:- Mapping of Regular (Strong) Entity types.

Step2:- Mapping of Weak Entity types.

Step3:- Mapping of Binary 1-1 Relation Types.

Step4:- Mapping of Binary 1-M Relationship Types.

Step5:- Mapping of Binary M-M Relationship Types.

Step6:- Mapping of Many Relationship Types.

Step7:- Mapping of Unary Relationship.

Step1: Mapping of Regular Entity Types

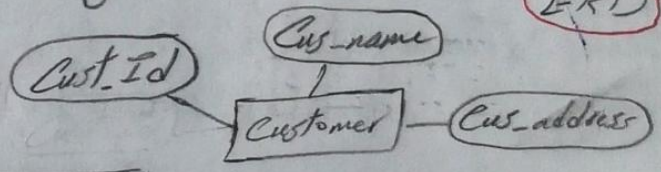
Step 1: Mapping of Regular Entity types

story

* قبل ما نبدأ لازم تبصه لو فيه أي علاقة تكون 1-1 و Total Participating للطرفين لازم تحلها الاول وبعد كدا تكمل.

ⓐ Simple Attribute.

ERD

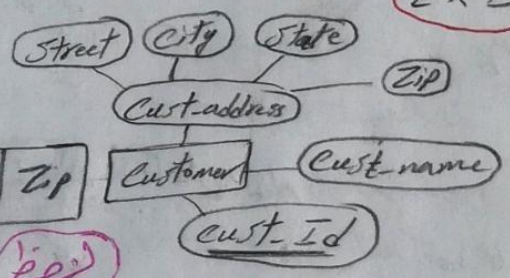


Customer Table

Cust_Id	Cust_name	Cust_address

ⓑ Composite Attribute.

ERD



Customer Table

Cust_Id	Cust_name	Street	city	State	Zip

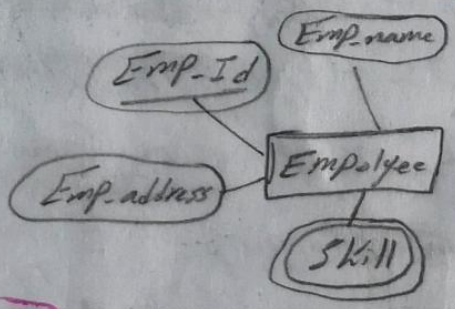
لاحظ

تم حذف الـ Composite Attrib وتم أخذ مكوناته ووضعها كـ Coulmns داخل الـ Table وذلك لأنه معندناش داخل الـ DB حاجة Composite. وتم كتابة الـ Composite داخل الـ ConstrainFile إلى بيتسلم مع الـ Schmas File على هيئة

Cust add = Str + City + State + Zip

ⓒ Multivalued Attribute.

ERD



Empolyee Table

Emp_Id	Emp_name	Emp_address

Forigne Key

Empolyee_Skill Table

Emp_Id	Skill

Composit Primary Key

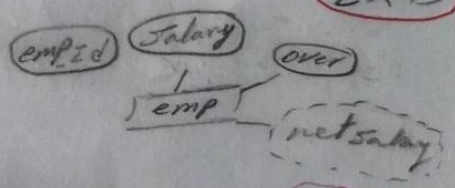
لاحظ

تم عمل Table للقيمة الـ Multi value وذلك لعدم تكرار بيانات الـ employee في الـ Table الرئيسي وعندئذ يكون الـ PK في الـ New Table هو عبارة عن Composite Key

ⓓ Derivied Attribute.

(Computed)

ERD



emp_Table

emp_Id	Salay	over

netsalay = Salay + over

لاحظ

* لم تم وضع الـ netsalary لأنه Computed Attribute يحسب في الـ runtime ولا يوضع في الـ SchmasFile ولا يوضع في الـ ConstrainFile.

* يوضع في حالات خاصة في حالة إننا هنبني على الـ Performance في المستقبل.

ⓔ Complex Attribute.

Emp_Table

SSN	Name	Salary	over

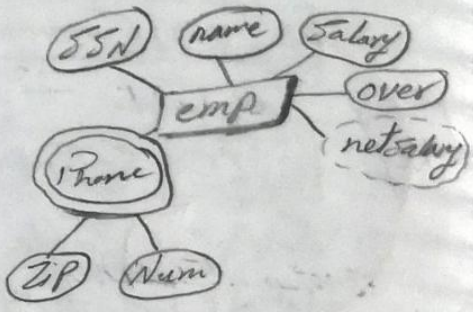
Forigne Key

Emp_Phone Table

SSN	ZIP	Num

Composite Primary Key

EIRD

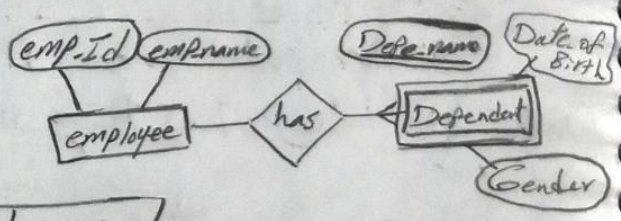


Step2: Mapping of Weak Entity Types

Step3: Mapping of Binary one-one

Step 2: Mapping of Weak Entity Types

EIRD



Employee Table

emp_Id	emp_name

Forigne Key

Dependent Table

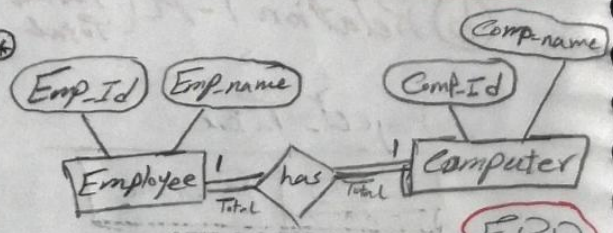
Composite Primary Key

emp_Id	Depen_name	Date_of_Birth	Gender

لاحظ: بيكون الـ Composit Primary Key من [الـ Primary Key من الـ Strong Entity
الـ Partial Key من الـ Weak Entity]
داخل الـ Table الـ Weak Entity

Step 3: Mapping of Binary 1-1 Relation types

EIRD



ⓐ Relation 1-1 (Total – Total) Participation:

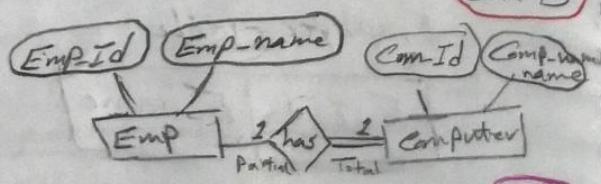
* ونختار أي PK من الـ 2 (Emp_Id, Com_Id) ليكون PK الـ Table

Employee Table

Emp_Id	Emp_name	Comp_Id	Comp_name

* لاحظ تم عمل Table واحد وليس 2 لأنها Total الطرفين 1-1

EIRD



ⓑ Relation 1-1 (Total – Partial) participation:

Employee Table

Emp_Id	Ename

Forigne Key

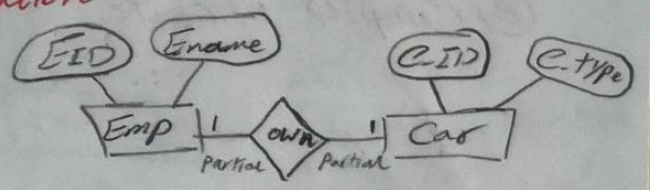
Computer Table

Primary Key

Emp_Id	Comp_Id	Comp_name

لاحظ: - كل ما عدد الـ Partial يزيد كل ما عدد الـ Table يزيد.
- يتم أخذ الـ PK من ناحية الـ Partial ويتم وضعه في الناحية الـ Total كـ Forigne Key.

ⓒ Relation 1-1 (Partial - Partial) Participation:



Employee Table

E_ID	E_name

Car Table

C_ID	C_Type

Emp_Car Table

EId	C_Id FK

Primary Key

Foreign Key

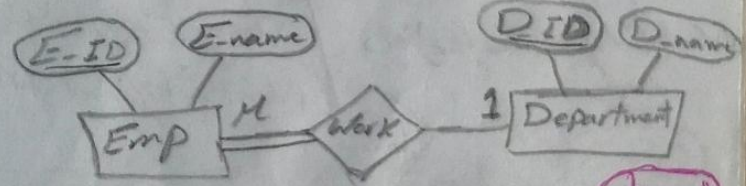
لاحظ الـ Table يحتوي على الإثنين Primary Key من الجدولين الآخرين كـ Foreign keys ويتم اختيار واحد من الإثنين كـ PK للـ Table

Step4: Mapping of Binary one-many Relation Types

Step 4: Mapping of Binary 1-M Relationship Types

لاحظ هنا في حالة الـ [1-M] لا يتم النظر على طرف الـ one ويتم النظر على طرف الـ Many فقط ومن إذا كان Total or Partial

ⓐ Relation (1-M) (Partial - Total) Participation:



Department_Table

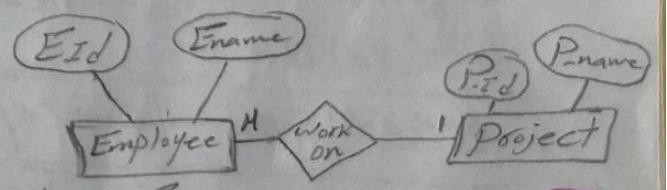
D_ID	D_name

Employee_Table

E_Id	E_name	D_ID FK

لاحظ هنا نأخذ الـ PK بتاع الـ one نضعه في الـ Many كـ FK.

ⓑ Relation 1-M (Partial / Total - Partial) Participation:



Project_Table

P_ID	P_name

Employee_Table

E_ID	E_name

Proj-Emp_Table

EID	P_ID FK

لاحظ في حالة الـ M-1 عندما تكون الـ M Partial نقوم بعمل New Table للربط ما بين الـ Two Table ويتكون من الـ Two Primary Keys من كل Table ونضع أكثر ناحية الـ Many كـ primary key والآخر Foreign Key.

Step 5: Mapping of Binary M-M Relationship Step

Many — Many

Total or Partial — Total or Partial

Student-Table

S-ID	Sname

Course-Table

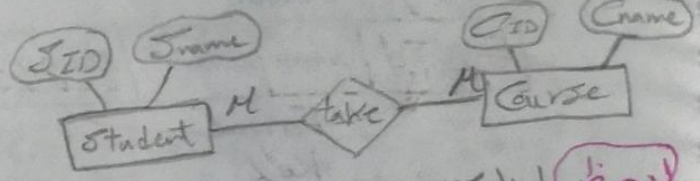
C-ID	Cname

Student-Course-Table

S-ID	C-ID

Composite Primary Key

ERD



لاحظ: إذا كانت العلاقة M-M لا ينطبق العلاقة من حيث الـ Participation ونقوم بعمل New Table ونضع به الـ Two primary Keys من الـ other Two Tables ويتم معاملتهم كـ Composite Key داخل الـ New Table.

ملحوظة مهمة: في حالة وجود Attribute على الـ relation نقوم بوضع الـ Attribute في الجدول الذي يحتوي على الـ Two Keys سواء كان Foreign أو Primary

Step 6: Mapping of Ternary Relationship Types

Doctor-Table

Doct-ID	Doct-name

Patient-Table

Patient-ID	Patient-name

Treatment-Table

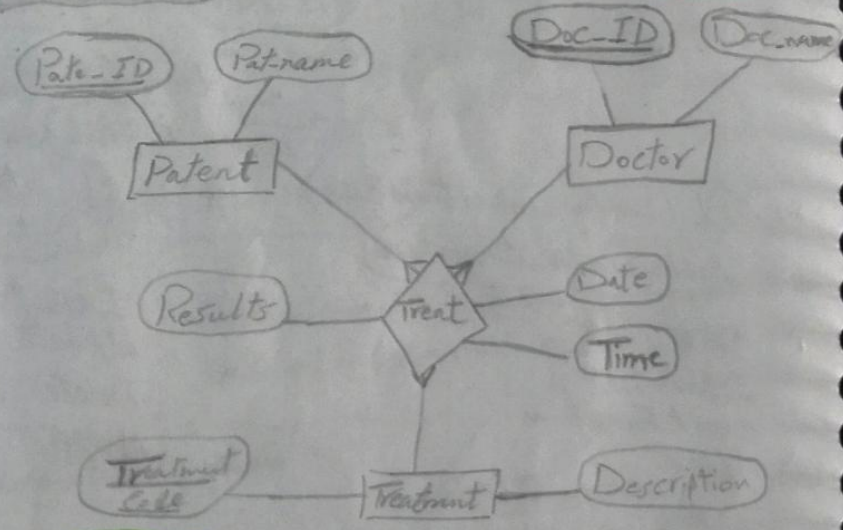
Treatment code	Description

Patient-treatment-Table

Patient-ID	Doctor ID	Treatment code	Date	Time	Results

Composite Primary Key

ERD



لاحظ: في جدول الـ relation لم يتم أخذ الـ Forigne Keys وجعلهم Composite primary Key وذلك بسبب احتمالية وجود تكرار ولذلك تم أخذ الـ C.P.K كـ Combination بين الـ Date والـ Time والـ Doc-ID لتجنب التكرار وإذا لم نجد Combination مناسب يتم وضع Primary key من عندنا.

Step5: Mapping of Binary many-many Relation Types

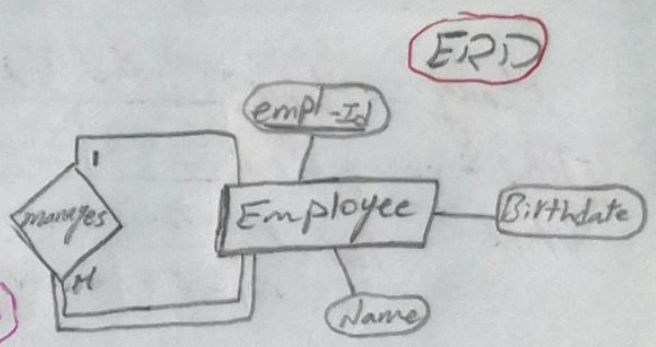
Step6: Mapping of Ternary Relationship Types

Step 7: Mapping Unary Relationship

Empolyee_Table

Emp_Id (P-K)	Name	Birthdate	Manger_ID

Forigne Key



لاحظ

في حالة الـ Unary Relation لم يتم عمل Two Tables [managers, employes] وذلك بسبب تطابق البيانات بين الجدولين ولذلك تم إدغام الـ Two Table في Table واحد يتكون من الـ forigne Key من الـ Table المحتوي بالإضافة إلى البيانات الموجودة في الـ Table المتبقي.

Step7: Mapping of Unary Relationship
ANSI SQL (DDL, DML, DCL, DQL, TCL)

ANSI SQL

- Microsoft — Transact SQL
- Oracle — PL-SQL
- IBM — IBM-PL-SQL
- Open Source — My-SQL

Microsoft
Transact SQL

*DDL (Data Definition Lang)
Meta Data_Structure

- Creat table
- Creat view
- Creat Function
- Alter
- Drop
- Select info

*DML (Data Manipulation Lang)
Data

- Insert
- Update
- Delete
- Merge

*DCL (Data Control Lang)
Security - Permissions

- Grant
- Deny
- Revoke

*DQL (Data Query Lang)
Display

- Select
- + Agg Fun
- Grouping
- Union
- Joins
- Subqueries

*TCL (Transactional Control Lang)
Excution

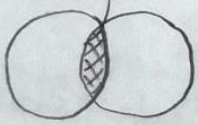
- begin Transaction
- Commit
- Rollback

SQL: Day 3
Join ..... ON
Self Join
Database Diagram(Video 10:40)
IS NULL / Coalesce

Join ... on

Cross Join :- بيضرب الصفوف في بعضها يعني لو عندي Table فيه ٣ صفوف والتاني ١٠ صفوف بيطلعلي ٣٠ نتيجة

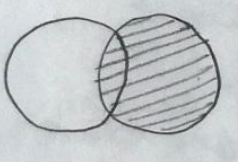
Inner Join :- دا بيطلعلي التقاطع ما بين الجدولين



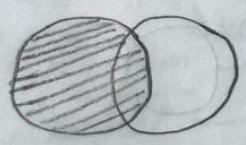
⊛ باختصار الصفوف مش هيظهر في النتيجة إلا اللي ليها Matching

```
Select First_name, Last_name, email
From   Sales.Customer c join Sales.order o
on     c.Customer_Id = o.Customer_id ;
```

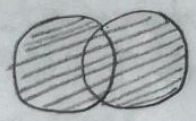
Right outer Join :- إظهر لي كل إلي موجود في الـ Table على اليمين حتى لو ملوش أي Matching مع الـ Table الأخر حتى لو الـ output هيبقى فيه null



Left outer Join :- هي عكس الـ Right outer Join



Full outer Join :- بتعرض الـ Two Tables في الأخر في الاعتبار عدم تكرار القيم إلي بينها Matching في الجدولين



Isnull - Coalesce

```
Select isnull(St_Fname, ' ')
from Student
```

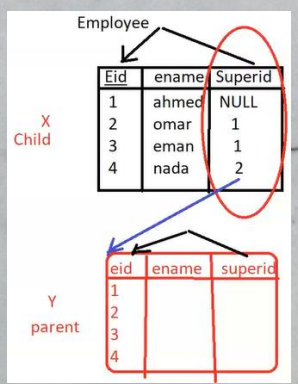
← هنا بيتم عرض الـ St_Fname من جدول Student ولو موجود حاجة null رح يتم إستبدالها بـ ' ' ← مسافة

```
Select isnull(St_Fname, St_Lname) as NewName
From Student
```

← هنا بيتم عرض الـ St_Fname باسم NewName من جدول الـ Student ولو موجود null بشيلها وأحط مكانها St_Lname

```
Select Coalesce(St_Fname, St_lname, St_Address, 'No Data')
From Student
```

← هنا بيتم عرض الـ St_Fname من جدول الـ Studen ولو وجد null هيحط مكانها St_lname ولو لقاها null هيحط مكانها St_Address ولو لقاها null هيحط مكانها 'No Data'




```
Select X.Ename as EmpName , Y.Ename as SuperName
From Employee X , Employee Y
where Y.eid = X.superid
```

```
ALTER TABLE current_table_name
RENAME TO new_table_name;
```

Update....Set

```
UPDATE olist_orders_dataset
SET order_id = REPLACE(TRIM('"'FROM order_id ), '"','');
```

```
update students
set Grad = Grade+5
where St_Id=(5,10,3);
```

← هنا سوف تعديل البيانات داخل جدول الـ students
حيث سيتم زيادة الـ Grad بمقدار 5 درجات
للطلاب ذوات الـ Id (5,10,3).

Concat

```
select Concat (St_Fname,' ',St_age) as S_Data
From student;
```

← Concat تعمل وظيفتين
1 تحول كل ما هو داخل الأقواس إلى String
2 تقوم بإزالة أي null وتضع مكانها ("") فارغة String

S_Data
Ali 25
Jon 30
Esraa 40

← كما بالمثال :- لا يمكن إضافة الـ age إلى الـ St_name وذلك
لاختلاف الـ Data type ولذلك تم استخدام
Concat لتوحيد الـ Date type إلى String.

```
SELECT TRIM('   Hello World   ') AS TrimmedString;
-- Output: 'Hello World'

SELECT TRIM(LEADING '0' FROM '00012345') AS TrimmedNumber;
-- Output: '12345'

SELECT TRIM(TRAILING '.' FROM '10.20.30.') AS TrimmedIP;
-- Output: '10.20.30'

SELECT TRIM(BOTH '-' FROM '-OpenAI-') AS TrimmedText;
-- Output: 'OpenAI'
```

Like

_ → one char
% → Zero or More char
^ → Not Equal

```
Select *
From Student
Where St_name Like 'a%'
```

← عرض جميع الأعمدة الموجودة بداخل الـ Student_table
بشرط أن يكون الـ St_name 'a%'
أي يبدأ بحرف الـ a

'a%h' → string أولها (a) وآخرها (h)
'%a_' → string الحرف قبل الأخير حرف (a)
'ahm%' → " يبدأ بـ "ahm"
'[ahm]%' → " ممكن بـ (a) أو (h) أو (m)
'[^ahm]%' → " لا تبدأ (a) أو (h) أو (m)
'[a-h]%' → يبدأ بالحروف ما بين الـ (a)(h) " '[^a-h]' → (h)(a) string لا يبدأ بالحروف ما بين
'[346]%' → string ممكن أن يبدأ بـ (3) أو (4) أو (6)
'%[%]' → string تنتهي بـ علامة %
'%[-]%' → "Ahmed_Ali" (-) string من منتصفها
'[-]%[-]' → _Ali_ (-) string تبدأ وتنتهي بـ (-)

```
EXEC sp_rename
    @objname = 'your_table.old_column',
    @newname = 'new_column',
    @objtype = 'COLUMN';
```

Update .... Set
Alter Table
Rename Table
Rename Column
Trim Functions
Replace +Trim Function

```
alter table emp add sal int

alter table emp alter column sal bigint

alter table emp drop column sal
```

Order By

```
Select *
From Student
Order by St_Id, St_name
```

St_Id → ← قوم بترتيب الـ output حسب St_Id ثم St_name
2 → ← قوم بترتيب الـ output حسب أول Column موجود في الـ output وإذا كان الـ output كله من 13 أعمدة ولا نستطيع أن نكتب بالكود order by 4

DB Design

ERD → Mapping ← Normalization

Normalization

Normalization
Functional Dependency

Normalization ← هو عبارة عن طريقة لإعادة هيكلة الـ Tables الموجودة وذلك عن طريق تكسير الـ Tables لتكون أسهل للوصول في النهاية لـ Database سليمة مع الحفاظ على الـ Relations بين الـ Tables وتجنب الـ Duplicates والـ Redumancy

← بنعمل الـ Normalization لما يبقى عندنا System أو Database قديمة وكفاءتها سيئة ومحتاجين نعمل System جديد فبنطبق خطوات أو Steps of Normalization.

Fuctional Dependancy ← هي عبارة عن الـ Relations بين الـ Coulmns وليس كالطبيعي تكون العلاقات بين الـ Tables أو الـ Rows

Types of Fuctional Dependancy

1) Full Fuctional Dependancy ← لو كل الـ Coulmns الموجودة بالـ Table بتعتمد على الـ Composite P.K بالكامل وليس جزء منه ومقدرش أجيب قيمة غير لما أعرف الـ C.P.K بالكامل

SSN, Proj.Num → Hours

2) Partial Fuctional Dependancy ← لو كانت القيمة إلى محتاج أجيبها بتعتمد على جزء من الـ C.P.K ومقدرش أعرف الـ C.P.K بالكامل عشان أجيبها

Sid, Subject → Grade

Sid - - - - → Sname

Subject → Teacher

3) Transitive Fuctional Dependancy ← لو كانت القيمة إلى محتاج أجيبها مش أجيبها بسبب الـ P.K أو هنا أدخل الـ C.P.K (Subject و Sid)

Zip-Code → City

Steps In Normalization

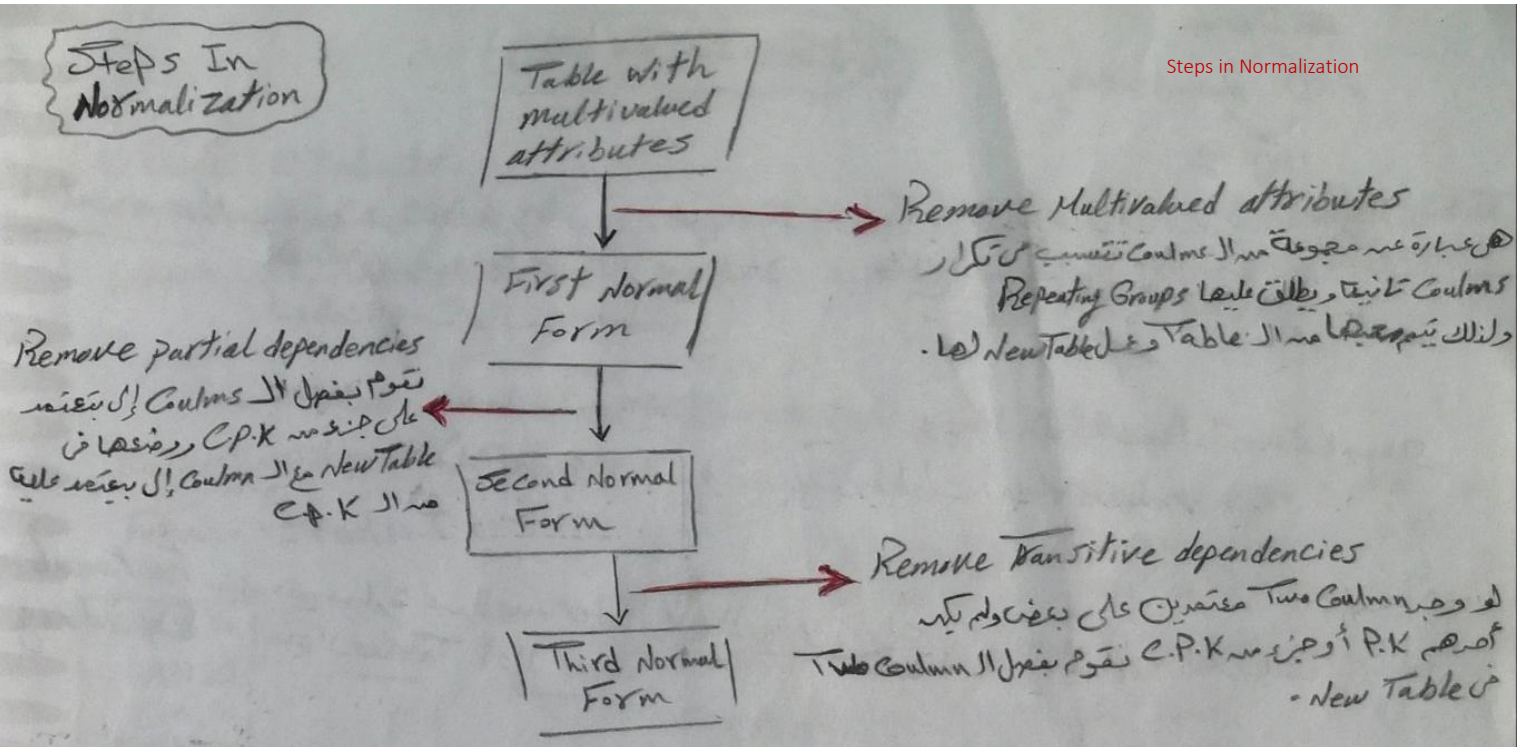


Table (E_Id, E_name, E_address, Pro_Id, Pro_name, Proj_Loc_Id, Proj_loc_City, hours, Bouns)

1st

Employee (E_Id, E_name, E_address)

Emp_proj (E_Id, Pro_Id, Proj_name, Proj_loc_Id, Proj_loc_City, hours, Bouns)

2nd

Employee (E_Id, E_name, E_address)

Project (Proj_Id, Proj_name, Proj_loc_Id, Proj_loc_city)

Emp_proj (E_Id, Proj_Id, hours, Bouns)

3rd

Employee (E_Id, E_name, E_address)

Project (Proj_Id, proj_name, Proj_loc_Id)

Locations (Proj_loc_Id, Proj_loc_city

Emp_proj (E_Id, Proj_Id, hours, Bouns)

20/3/2013

Agile Development

Agile Development
Software Development Life Cycle
What is The Project?
What is The Project Management?
Predictive VS Adaptive
Project Management Triangle

Software Development Life:-

[1] Planning [2] Analysis [3] Design [4] implementation
[5] Testing & Integration [6] Maintenance

- What is the Project?

- هو عبارة عن حاجة بيتم إنشائها لحل مشكلة معينة ويكون له بداية وله نهاية بمعنى أنها مؤقتة عكس الـ department فهو حاجة ثابتة داخل الشركة

- What is the Project Management?

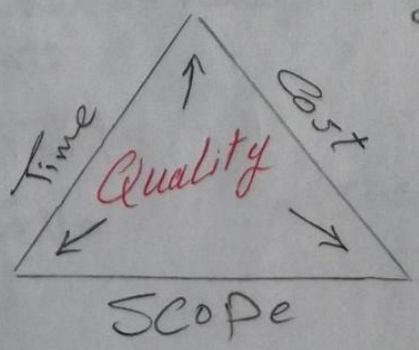
- هو مجموعة الـ Tools والـ Techniques اللي هيتم استخدامها للتعامل مع المشروع [1] معرفة الـ Scope وإزاي من خلاله هعرف الـ Requirements المطلوبة [2] عمل الـ Schedule وده الخروج بـ Time estimates والـ base Line للمشروع مثلاً (المشروع دا هياخد سنة أو سنتين) [3] تحديد الـ Resources إيه هنحتاجها في المشروع مثلاً (equipments) [4] تحديد الـ Cost ووضع Cost baseline [5] وسائل الإتصال Communication داخل المشروع وأساليب الإعلان عن حاجة [6] تحديد الـ Quality standards إيه هنشتغل عليها وإزاي هعمل audit مراجعة وضمان بتطبيق المعايير في الـ quality وإزاي هعمل assurance ضمان لجودة المنتج [7] عمل سجل الخاطر إللي ممكن مواجهتها في المشروع Risk Register وتجنبها. [8] أسلوب عمل Procurment والعقود Contracts.

- Predictive Vs Adaptive

- Predictive :- كل الخطوات واضحة بالكامل من الأول فأنت الخطوة إللي بتعملها مبترجعها شي تاني زي الشلال كده Waterfall وبالتالي كل ما يترتب عليه واضح.

- Adaptive :- أنا بيقى معايا جزء من الـ Requirements وبالباقي مش واضح وبالتالي بجزء المشروع إلى أجزاء صغيرة وأخلص جزء ولما يظهر Requirements جديدة بشتغل عليها في sprint جديد

- Project Management Triangle



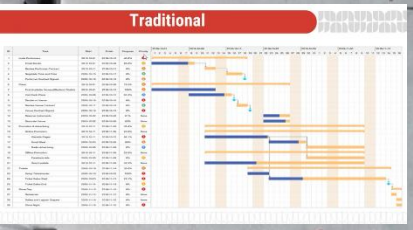
* إزاي تحقق الـ Scope المطلوب في الوقت المحدد بالـ Cost المحددة مع تحقيق الـ Quality standards المطلوب.

Waterfall Vs Agile
What is Agile?
Agile Values

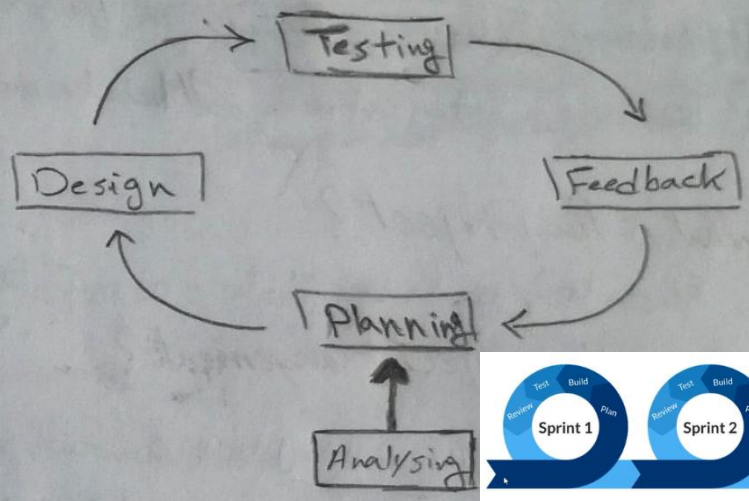
- Waterfall Vs Agile

Waterfall

Requirments → Design → implementation → Verification → Maintenance



Agile



What is Agile?

- A project management approach based on delivering requirments iteratively and incrementally (sprint).

← بمعنى كل شوية هسلمك حجات صغيرة بس كتيير وبشكل تكراري وتدريجي وله قيمة.

تكراري

تدريجي وله قيمة

Agile Values :-

1. روح التعاون والتواصل بين الأفراد من الـ Team سواء أفراد الـ Team أو الـ Customer أهم من الـ Process والأدوات بمعنى مش كل شوية هتقول لـ أهل العقد بيقص على فى حالة طلب العميل أو فى حالة طلب حد من أعضاء الـ Team حاجة أو تأخير فى حاجة تقول أهل القانون بيقص على أو بيقع عليك عقوبة كذا.

2. أكون مهتم إنه أطلع حاجة شغالة أكتر من أكون مهتم بالـ Documentation.

3. عقود الـ Agile بيبقى فيها نوع من المرونة للتغيرات إلى بيطلبها العميل بمعنى منقفش عند العميل.

4. لو جه Change مهم يكون له الأولوية على الحاجة إلى المفروض تتعمل طبقاً للـ Plan الموضوعة.

Agile Scrum Framework

Product owner

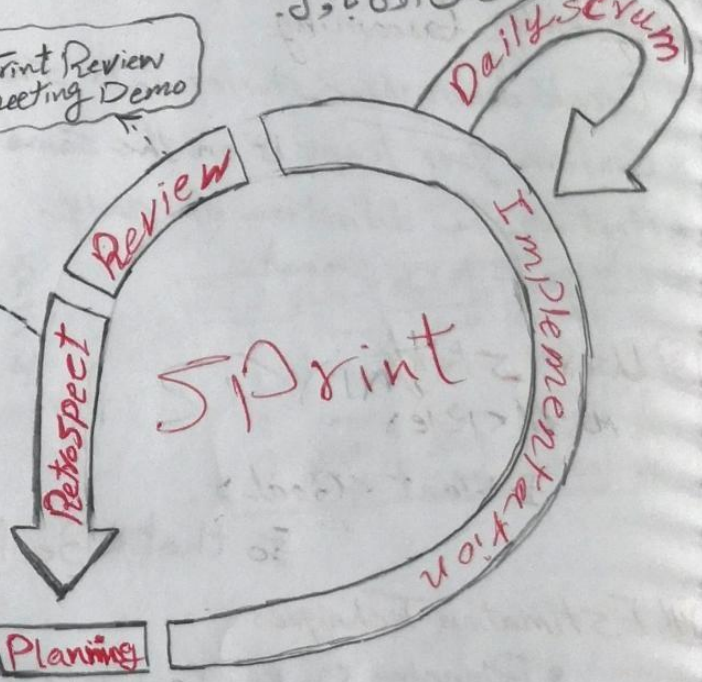
Scrum master

بيسهل ويبسر ويعلم
Team mentor

Scrum Team members

Stakeholders

عبارة عن اجتماع يومي لا يزيد عن ربع ساعة
بيتم مناقشة إلي هيتعمل خلال اليوم ومناقشة
العوائق أول بأول.



Sprint Retrospective Meeting

دا الـ meeting إلي بيتم مناقشة فيه
المشاكل إلي واجهه الـ Team من الـ Sprint
وإزاي نتفاداها من الـ Sprint القادم
وكمان نعمل أوت بالعجات الحلوه عشان
نطورها.

Backlog

بنحط فيه الحجات إلي كنا شغالين عليها
وفيها مشاكل بالنسبة للـ owner عشان
نرجع نعيد الـ Sprint تاني.

Project Vesion
الرؤية العامة

Epics
صور كبيرة
محتاجه تتفصل
أكتر

User stories
حكايه كل مستخدم
مع الـ system والمطلوب
تنفيذه

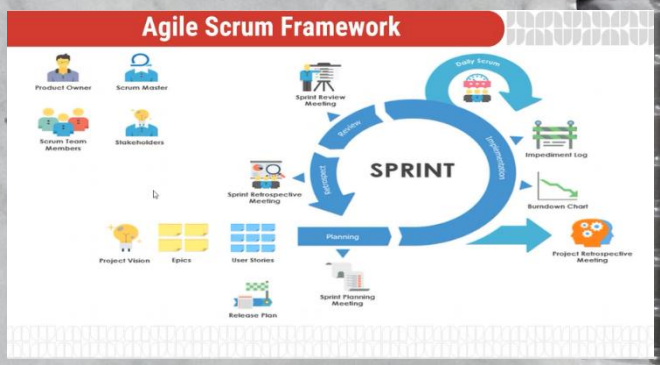
Release Plan

بتحدد فيه الأهداف والحجم والمدة
إلي هتشتغل بيها في كل Sprint

Backlog Grooming

بنتناقش
في الحاجه إلي هنشتغل
فيها ونشوف إيه المطلوب نعمله وإزاي نعمله.

Sprint Planning Meeting



Backlog Grooming:-

* Breakdown user stories into smaller tasks.
* Confirm your team is on the same page.
* Meeting the definition of ready.
* Assign story points

Important Tools

* Jira
* Trello
* ClickUp

Agile scrum steps

1 Sprint Planning Meeting

Start of the sprint -> What is to be built during the sprint?
How the team will built it?

2 Backlog Grooming

* Break down user stories into smaller tasks.
* Confirm your team is on the same page.
* Meeting the difinition of ready.
* Assign story points

3 User story

As A <Role>
I Want <Goal>
So that <Benifits>

4 Estimation Techniques

* Planning Poker (0, 1, 2, 3, 5, 8, 13 ---)
* T-shirt sizes (Small, medium, Large ---)
* Dot Voting
* Bucket system.

5 Daily standup Meeting

* What did you do yesterday?
* What will you do today?
* Are there any impediments in your way?
عوائق

6 Sprint Review Meeting

* What is done and What is not done?
* Demo the work.
* Review key metrics.
* Review and Revise product backlog.

7 Sprint Retrospective

* Prepare
* What went well?
* What needs improvement?
* Next steps.

21/3/2023
Day 4

SQL: Day 4
Aggregate Functions
Group by

Aggregate Function

Aggregate Function:

Cout, Max, Min, Avg, Sum

```
Select sum(salary)
From employee ;
```

```
Select Min(salary), Max(salary)
From Employee;
```

```
Select Count(e_Id)
From employee ;
```

Count(*) → 5
Count (E_Id) → 5
Count (E_name) → 4

```
Select Avg(salary)
From employee;
```

E_Id	E_name	Salary	add	D_Id
1	Ahmed	2000	Cairo	10
2	Ali	3000	Cairo	20
3	eman	500	alex	10
4	ola	300	alex	20
5	null	100	alex	10

Group By

لو إنت معاك Agg Fun في ال Query ومعاك تمام Coulmn في ال Select لازم تستخدم group by بالنسبة لل Coulmn دا.

لاحظ:

```
Select Min(salary), D_Id
From employee
groupby D_Id;
```

→

Min(salary)	D_Id
100	10
300	20

```
Select Count(E_Id), Add
From Employee
Where D_Id IN(10,20)
group by Add ;
```

→

Count(E_Id)	Add
2	Cairo
3	Alex

```
Select Min(Salary), D_Id
From Employee
Where Add Like '_a%'
group by D_Id ;
```

→

Min(Salary)	D_Id
2000	10
3000	20

لاحظ: تم إستخدام Where Filter قبل تنفيذ groub by ولذلك لا تؤثر على groupby

Having

```
select Sum(Salary), D_Id
From Employee
group by D_Id
having Count(E_Id) >= 3;
```

Sum(Salary)	D_Id
2600	10

لاحظ - يتم إستخدام having كـ Filter بعد تنفيذ group by ولذلك تؤثر على نتيجة group by.

```
select Sum(Salary), D_Id
From employee
where add Like '_a%'
group by D_Id
having Sum(Salary) > 2500
```

Sum(Salary)	D_Id
3000	20

Having
Subqueries

Subqueries

هو عملية إستخدام Query بداخل Query أي بمعنى إستخدام (Query) كـ input as بداخل Query تانى.

```
select *
From Student
where St_age < (select Avg(St_age) From Student);
```

لاحظ تم عمل Query بداخل Query وذلك لتجنب إستخدام aggegrate Function مع Where.

```
select dept_name
From Department
where dept_Id in (select distinct(dept_Id)
                  From Student
                  where dept_Id is not null);
```

← في بعض الأحيان بنضطر لعمل أكثر من Join داخل الـ Query ولذلك بيتم إستخدام الـ Subquery. تسهيلاً وتسريعاً للوقت.

Subquery + DML

```
Delete From Stud_Course
where Stud_Id in (select Std_Id
                  From Student
                  Where St_address = 'Cairo');
```

Union Family
Union all
Union
Intersect
Except

Union Family

(union all , union , intersect , except)

union all

```
Select Stud_Fname , Stud_id From Student
union all
Select instruc_Fname , instruc_Id From Instructor;
```

← يتم إستخدام union all لعمل Merge لل outputs
أكثر من Query لجعلهم في output واحد
بشرط تطابق الـ output في Data type
وعدد الـ output ولذلك لكل جدول لهم

Ahmed	1
medo	2
Amr	3
medo	4
May	1
Ahmed	2
Noor	3
May	4

كما بالمثال :- تعمل union all على دمج Stud_Fname مع Instru_Fname في نفس الـ Column
ودمج Stud_Id مع Instruc_Id في نفس الـ Column
في output واحد عبارة عن جدول مكون من الـ 2 Columns
ممكن أن يحتوي على null أو قيم مكررة .

union

union ← تعمل عمل union all لكن لا تسمح بالتكرار بحيث تكون
الـ output ← Distinct
Unique
ومترتب أبجدي

```
Select Stud_Fname From Student
union
Select instru_Fname From instructor;
```

Ahmed
medo
Amr
May
Noor

intersect

intersect ← تستخدم لعمل تقاطع بين الـ output أي جدول الـ outputs
يحتوي على القيم المتشابهة بين الـ Tow Columns .

```
Select Stu_Fname From Student
intersect
Select instruc_Fname From instructor;
```

← هنا بالمثال الـ output سيكون عبارة عن Table
مكون من Column واحد يحتوي على الأسماء
المتشابهة بين الـ Stud والـ Instruc .

Ahmed

← فقط لأن الاسم الوحيد
المشترك بين الطلاب والمعيدين

except

except ← تستخدم لإخراج القيم الناتجة من الـ Query الأول بشرط أن تكون
غير موجودة في الـ output الـ Query الثاني

```
Select St_Fname From Student
except
Select ins_Fname From instructor;
```

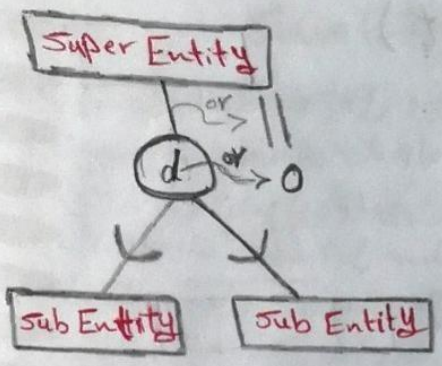
medo
Amr

← لاحظ تم إخراج "Ahmed" لأنه
موجود في instructor أيضاً
وتم إخراج "medo" لعدم التكرار .

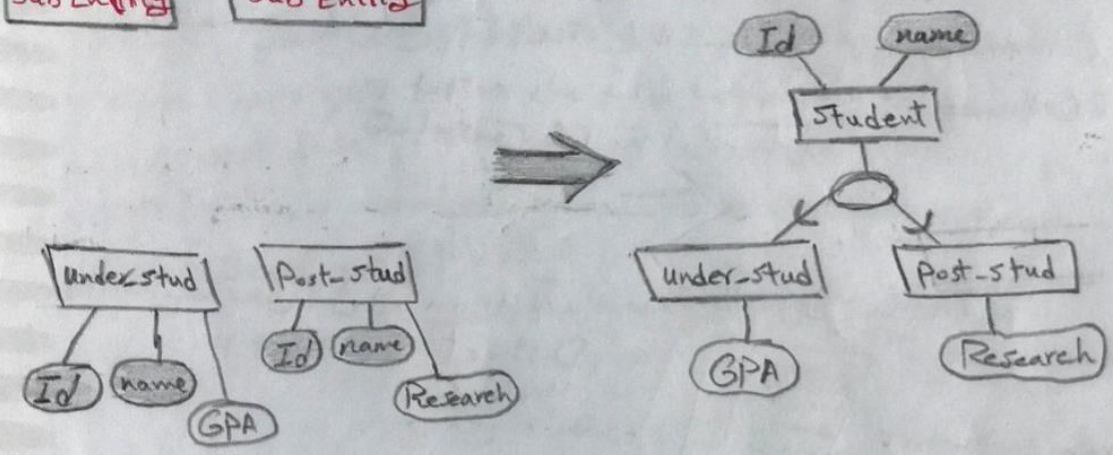
EERD (Enhanced ERD)
Super Entities / Sub Entities
Relationships

EERD

ERD + Inheritance



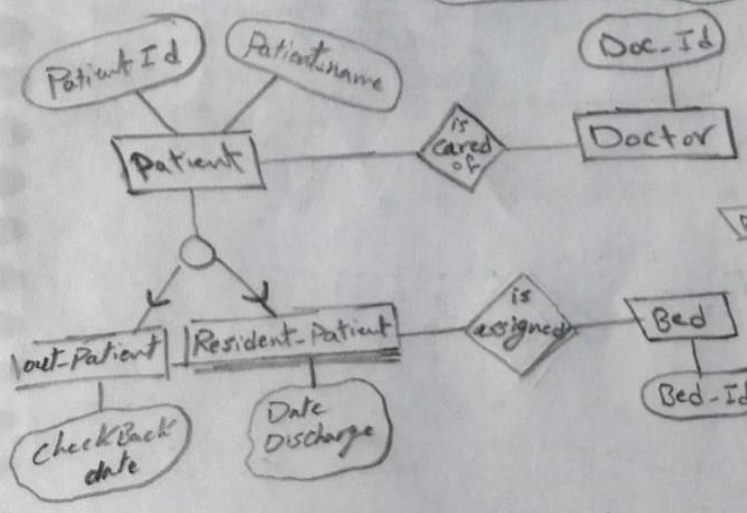
* لما يبقى عندك كذا Entity بينهم Attributes متشابهة وأخرى مختلفة
قبل ما ندخل في مرحلة تحديد الـ Relations والـ Participation والـ Cardinality
بيتم عمل إختبار الـ inheritance الـ intities وذلك لتفادي تكرار
الداتا المتشابهة عن طريق خلق New Entity (Super Entity)
يجمع الـ Attributes المشتركة بين الـ old Entites (Sub Entities).



Sub Entities ← هو عبارة عن Entities موجودة وبينهم Attributes
متشابهة ، فبيتم إستخلاص هذة الـ attributes المتشابهة
ووضعها في New entity (Super Entity) ويظل الـ Sub Entity
موجوداً محتفظاً فقط بالـ attributes الخاصة به ومرتبط بـ
الـ Super Entity

Super Entity ← هو عبارة عن New Entity من نفس نوع الـ Sub Entities الموجودة
تم وضعه ليجمع الـ Attributes المتشابهة بين الـ Sub Entites مع
حفاظ الـ Sub Entites على الـ Specific attributes الخاصة بهم فقط

Super Entities / Sub Entities Relationships



لاحظ ⊛ الـ relation بين الـ Patient والـ Doctor
هي بين الـ Super Entity والـ Doctor وبالتالي
هي General بمعنى أن الـ Doctor له
relation مع الـ Sub entities out_patient و Resident_patient

⊛ الـ relation بين Residut patient والـ Bed
هي بين Sub Entity والـ Bed وبالتالي هنا
الـ relation تخص فقط الـ Resident Patient

Constraints in SuperEntity

Constraints in Super Entity
Completeness Constraints
Total Specialization (||)
Partial Specialization (|)
Disjointness Constraints
Disjoint Rule (d)
Overlap Rule (o)

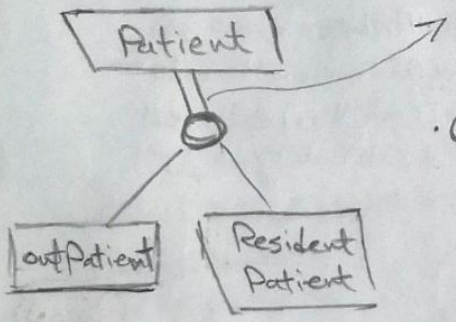
Completeness Constraints:

* Total Specialization (||)
* Partial Specialization (|)

Disjointness Constraints:

* Disjoint Rule (d)
* Overlap Rule (o)

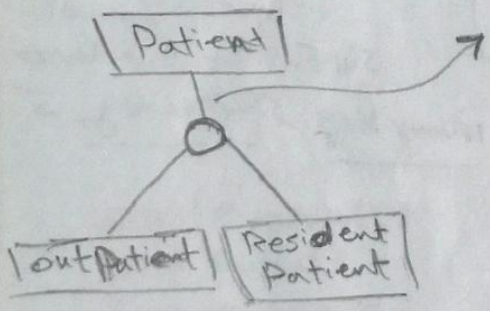
Total Specialization (||)



*هنا تعني أن كل الـ Patient مكونة من النوعين الموجودين فقط يا إما Resident Patient أو outPatient ولا يوجد نوع آخر من المرضى.

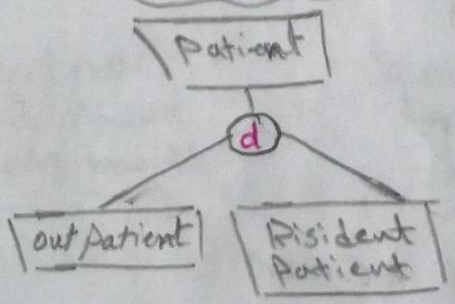
ولذلك تم تمثيلها بخطين ||.

Partial Specialization (|)



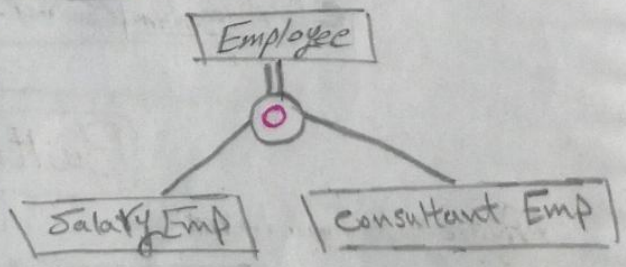
*هنا تعني أن Patient الـ SuperEntity تحتوي على أنواع أخرى من المرضى غير الـ Resident Patient و out Patient ولذلك تم تمثيلها بخط واحد (|)

Disjoint Rule (d)



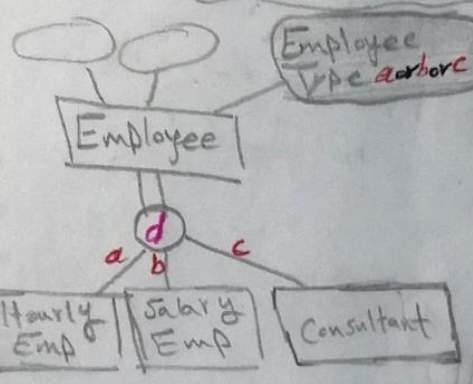
*هنا يعني أن لا يوجد احتمال أن يصبح الـ Risident Patient out patient

Overlap Rule (o)



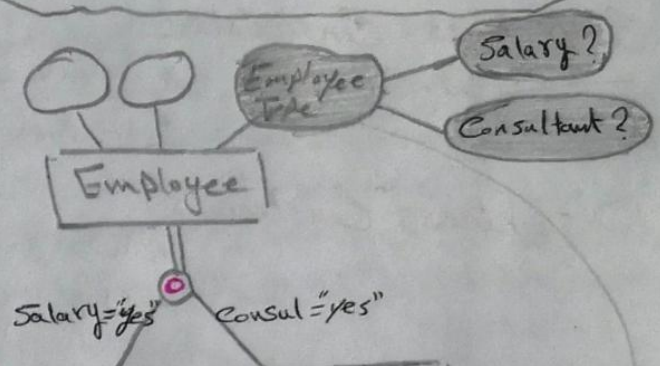
*هنا يعني أن يوجد احتمال أن يكون هناك موظف يأخذ راتب as Salary Emp وفي نفس الوقت يكون استشاري في مشروع آخر ويكون له مرتب آخر as Consultant Emp

SubEntity discriminator (disjoint Rule)



هو عبارة عن simple attribute يتم وضعه لتحديد نوعية ال Emp المشار إليها دون الحاجة لعمل Join بين ال SuperEntity و Sup Entity

SubEntity discriminator (overlap Rule)

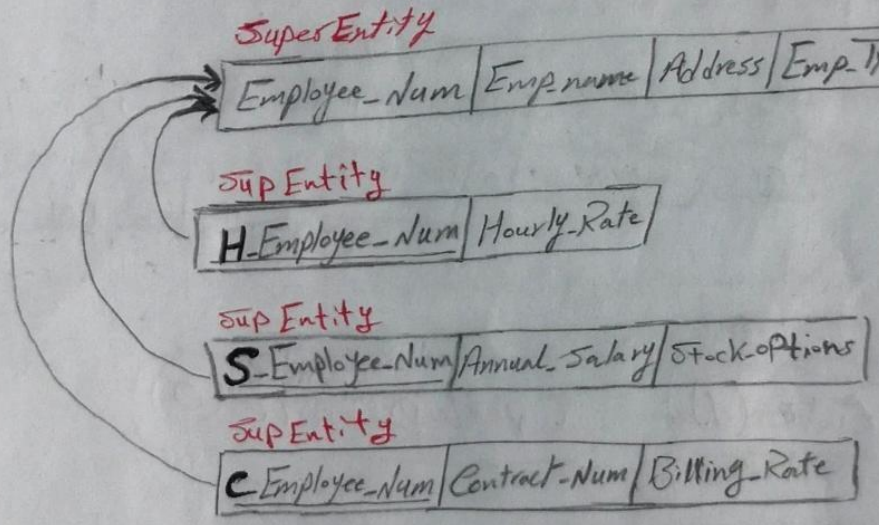


هو عبارة عن Composite Attribute مكون من simple attributes تشير إلى نوع ال Emp المشار إليها وذلك لعدم استخدام Join بين أكثر من Sub Entity و ال Super Entity.

Mapping SuperEntity/SupEntity Relationships

لاحظ

تم استخدام ال Primary Key الخاص بـ SuperEntity ووضعه في ال Sup Entities كـ Primary Key و Forign Key



Builtin_Functions

* Aggregate Functions
* getdate() * Isnull * Coalesce * Concat() * Convert
* Select year(getdate()) → عشان أجيب السنة
* Select Month(getdate()) → " " الشهر
* Select Substring(St_Fname, 1, 3) → عشان أجيب أول ٣ حروف من الكلمة
  From Student;
* Select db_name() → عشان أجيب ال Database
* Select suser_name() → عشان أجيب ال user

Mapping of Super Entities / Sub Entities Relationships

Built in Functions(substring(),Year(),Month(),Database name,user name)

View Template Explorer (Video 02:56:10)

24/3/2023
Day 5

Select Top( )

```
Select top(3) *
From Student ;
```

هاتلي أول ٣ صفوف من Student Table

```
Select top(3) St_Fname
From Student ;
```

هاتلي أول ٣ أسامي من ال Student Table

```
Select top(3)
From Student
Where St_address = 'Alex' ;
```

هاتلي أول ٣ أسامي ميكونوا ال add بتاعهم Alex

```
Select top(2) salary
From Instructor
order by salary desc
```

عشان أجيب أعلى ٢ salary

SQL: Day 5
SQL Server Engine Presentation(Video 02:32)
Setup Instance Vs Services(Video 15:00)

```
Select Top( ) --- with ties
|
|
order by ----- ;
```

```
Select top(3) with ties *
From Student
order by St_age ;
```

	Id	Name	Age
1	13		null
2	1		16
3	2		21
4	10		21

هنا هتلاحظ إن في بعض الأحيان رغم أنا قايل Top(3) هتلاقي ال output أكتر من (3) وذلك بسبب ال Ties أي (الديل) بمعنى إنه هيروح يطلع أول ٣ زي ما أنت قايله وهيجيب للصف الأخير له لقى إنه متكرر في ال Table هيجيب تكراراته معاه بمعنى مثلا هيطلع أول ٣ وهيرتبهم بالنسبة لل St_age وهيجيب على الأخير له لقى مثلا عمره ٢١ سنة هيشوف في ال Table هل في حد تاني عمره ٢١ سنة لو لقى هيظهره.

Select, newid( )

```
Select *, newid( )
From Student ;
```

				newid
-	-	-	-	175074
-	-	-	-	350772
-	-	-	-	7402
-	-	-	-	57-3

لما تجيب قدام كل صف Id بكون unique على مستوى ال Server وكمان كل مرة بعمل run للكود ال Id هيتغير بمعنى هيكون random

```
Select *
From Student
order by newid( ) ;
```

هنا هاتلي تظهر ال Data الموجودة في ال Student Table بس بطريقة عشوائية من غير ترتيب كالعادة بال St_Id

```
Select top(3) *
From Student
order by newid( ) ;
```

هنا هيجيب أول ٣ صفوف من Student Table بس بطريقة عشوائية يعني كل مرة بعمل run للكود هيظهر ٣ مختلفين.

Execution Orders

1) From
2) Join
3) On
4) Where
5) group by
6) Having
7) Select
8) order by
9) Top

① بعرف الأول ال Tables اللي هشتغل عليها
② بعمل Filter لل rows
③ بقسم البيانات إلى groups
④ بعمل Filter لل groups
⑤ تحديد المراد عرضه وإظهاره
⑥ ترتيب المراد عرضه
⑦ بعمل Top للمراد عرضه

تسلسل تنفيذ الدوال داخل الأكواد وأولوية التنفيذ.

```
Select *
From ( Select St_Fname + ' ' + St_Lname as Fullname
       From Student ) as Newtable
Where Fullname = 'Ahmed Ali'
```

← لاحظ هنا الـ Newtable هو Table
هتعمل في الـ runtime فقط
عشان أحط فيه الـ output بتاع
الـ subquery فقط عشان أستخدمه
أحط فيه الـ Fullname مؤقتاً واستخدمه
في الـ mainquery.

← أي subquery يجي فيه From لازم يتعمل له elias name.

DB_Objects [Table_ View_ Function_ SP_ Rule]

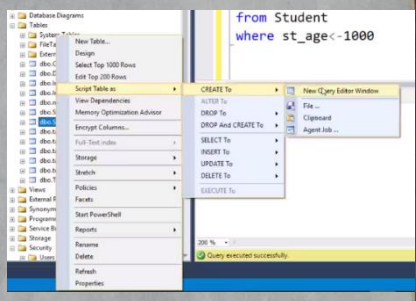
The Object Path:-

[ServerName] . [DBName] . [SchemaName] . [Object Name]

```
Select *
From Student;
```

⇒

```
Select *
From [DESKTOP_VF50P25].iTi.dbo.Student;
```




```
Select *
From Company_SD.dbo.Project;
```

لاحظ هنا رغم إني عامل use
لـ iTi DB إلا إني أقدر
أجيب Table من DB تانية
عن طريق تحديد الـ Path

```
Select dname
From Company_SD.dbo.Departments
union all
Select dept_name
From Department;
```

لاحظ إننا ممكن نعمل Join أو union
بين Data موجودة في DB (iTi)
و Data موجودة في DB تانية
(Company_SD)

```
Select ... into New_Table
From ...
```

```
Select * into table_2
From Student;
```

هنا أول حاجة عملها من
From عرف مين إلي هيختار وحدد
عن طريق Select إلي هيأخذها
ثم عملها نسخة داخل Table
عن طريق into.

```
Select * into Company_SD.dbo.Student
From Student;
```

هنا هو عمل نسخة من Student Table من الـ iTi_DB
في Database تانية الـ Company_SD.

```
Select St_id, St_Sname into Alex_Stud
From Student
Where St_address = 'Alex';
```

هنا هو بياخد حتة من جدول بيعمل بيها
جدول جديد تاني.

```
Select * into Table_3
From Student
Where 1=2
```

هنا الـ Condition مستحيل يتحقق لأن $2 \neq 1$
وبالتالي هيروح يدور الصفوف إلي هينسخها من الـ Data
مش هيلاقي الشرط متحقق وبالتالي مش هينسخ
الداتا وهينسخ الـ Table بس فاضي.

The Object Path

Insert Data Ways

Create [new_table] from [old_Table]

1)Extract the Script of the old table with constrains (Video 01:24:30).

2)(Select .....Into[new_table]From [old_table])

(Copy & Past)Tables with

Insert based on Select
Using Having without Group
by(Video 1:28:10)
Data Types

```
insert into ----
Select --- From ----
```

Insert into table3
Select * From Student;

لما أصب أنقل Data بين 2 Tables موجودين
وبا بتكون ال Data فقط من غير ال Structure

لو حظ هنا Table3 هو جدول موجود بالفعل
وعاوز أنقل ليه ال Data الموجودة بداخل
جدول ال Student. أو ممكن أنقل Columns محددة

Insert into table3
Select St_Id, St_Fname
From Student;

هنا هينقل
الداتا الموجودة
في العمودين
المحددين فقط.

```
Select Agg_Fun
From ----
Having Agg_Fun
```

Select Sum(Salary)
From instructor
Having Count(ins_id) < 100
where.... error

لو حظ يمكن إستخدام Having
بدون group by في حالات
إن Select معاها Agg فقط
ولا يسمح إستخدام where مع Agg.

Data types

Numeric DT

Bit → bool , 0:1 , true : false
tinyint → 1 Byte -128 : +127 unsigned 0 : 255
Smallint → 2 Byte -32768 : +32767 unsigned 0 : 65555
int → 4 Byte
Bigint → 8 Byte

Decimal DT

Smallmony → 4 Byte □□□.0000 ← 4 علامات عشرية
mony → 8 Byte □□□.0000
real → □□□.0000000 علامات عشرية
Float → □□□.000000000000000
Decimal → dec(5,2) رقم مكون من 5 أرقام منهم 2 علامات عشرية
123.87 ✓
12.089 ✗

character DT

Char(□) → [□ Fixed Length character] | Ahmed → 10 | Ali → 10 |
Varchar(□) → [□ Variable length character] | Ahmed → 5 | Ali → 3 |
nchar( ) → unicode يسمح بكتابة أي لغة ويعمل لكل لغة كود
nVarchar( ) →
nVarchar(max) → unlimited للكتابة وانت مش عارف هتكتب قد إيه.

DateTime

Date → MM/DD/yyyy
Time → hh:mm:□.000 → يعطي الثواني لحد ٣ أعداد جزء من الثانية
Time(4→7) → hh:mm:□.0000000
SmallDateTime → MM/DD/yyyy hh:mm:00
DateTime → زي Small بس تبدي ثواني لحد ٣ أجزاء من الثانية
DateTime2(Z) → زي Small بس تبدي ثواني لحد Z أجزاء من الثانية.
DateTimeoffset → 11/24/2020 10:30 +2:00
TimeZone

Binary → 011110 , 111100
Image → Path
→ Binary

XML
unique_identifier
SQL_variant

Ranking Functions

Ranking Functions
Row_Number () / Dense_Rank ()
N tile ()
Denis_Rank () Vs Rank()

```
Row_Number( ) over(order by ---- desc/Asec) as RN
Dense_Rank( ) over(order by ---- desc/Asec) as DR
N Tile (Group) over(order by ---- desc/Asec) as G
Rank( )
```

lua

StudentName	Score	DenseRank
Emily	90	1
Sarah	90	1
David	85	2
John	80	3
Michael	75	4

lua

StudentName	Score	Rank
Emily	90	1
Sarah	90	1
David	85	3
John	80	4
Michael	75	5

E_Id	E_name	E_salary	D_Id	RN	DR	G
15	Ahmed	10000	10	1	1	1
14	Ali	10 000	10	2	1	1
12	Eman	9000	10	3	2	1
2	nada	9000	10	4	2	1
2	reem	9000	10	5	2	1
3	Khalid	8000	10	6	3	2
7	Mohammd	7000	20	7	4	2
8	Sayed	7000	20	8	4	2
6	Hassan	6000	20	9	5	2
5	omar	6000	20	10	5	2
9	Sally	5000	30	11	6	3
10	Shaimaa	4000	30	12	7	3
11	Hana	4000	30	13	7	3
12	Lama	3000	30	14	8	3

```
Select *
From (Select * , Row_Number( ) over(order by E_salary desc) as RN
             , Dense_rank( ) over(order by E_salary desc) as DR
             , N Tile (3)  over(order by E_salary desc) as G
        From Employee ) as Newtable
Where RN = 1 ->
or RN = 3 ->
or RN <= 2 ->
or DR = 1 ->
or DR <= 2 ->
or G = 1 ->
```

- RN = 1 → ماهو بيانات الموظف صاحب أعلى مرتب ؟
- RN = 3 → ماهو بيانات الموظف صاحب ثالث أعلى مرتب ؟
- RN <= 2 → ماهو بيانات الموظفين الحاصلين على أعلى مرتبين بغض النظر عن تكرار الراتب ؟
- DR = 1 → ماهو بيانات الموظفين الحاصلين على أعلى مرتب ؟
- DR <= 2 → ماهو بيانات الموظفين الحاصلين على أعلى مرتبين مع تكرار الموظفين لنفس الراتب ؟
- G = 1 → ماهي بيانات الموظفين الموجودين في الجروب الأول ؟

لاحظ: N Tile تعمل على تقسيم الـ Data إلى Groups متساوية في العدد وإذا كان عدد الـ Rows لا يقبل القسمة على عدد الـ Groups فتقوم بتقسيم الـ Data بحيث يكون الجروب الأخير أقل بـ Row واحد على الأكثر من الجروب الذي يسبقه وإذا لم نستطيع إنه نقوم بتقسيم الداتا بحيث يكون الجروب الأخير وما قبله أقل بـ Row واحد على الأكثر من باقي الـ Groups بحيث غير مسموح وجود جروب أقل من باقي الـ Groups بأكثر من Row واحد.

Row_Number( ) over (partition by ---- order by ---- desc/Asce) as RN

Dense_Rank ( ) over (Partition by ---- order by ---- desc/Asce) as DR

NTile (Num of Groups) over (Partition by ---- order by ---- desc/Asce) as G

Rank( )

لازم تقوم بتقسيم الـ Table Partition by ....
إلى Groups بحسب الـ Column الموجود بعد by
بحيث تكون الـ Row_Number و Dense_Rank و NTiles
على مستوى المجموعة وليس الـ Table بالكامل

E_Id	E_name	E_salary	D_Id	RN	DR
15	Ahmed	10000	10	1	1
14	Ali	10 000	10	2	1
12	eman	9000	10	3	2
1	Nada	9000	20	1	1
7	Sayed	7000	20	2	2
6	omar	7000	20	3	2
9	Sally	5000	30	1	1
10	hana	5000	30	2	1
11	Lama	3000	30	3	2

```
Select *
From ( Select * , Row_Number( ) over (Partition by D_Id order by E_salary desc) as RN
        , Row_Number( ) over (Partition by D_Id order by E_salary desc) as DR
    From Employee ) as NewTable
where RN = 1
RN = 3
DR = 1
DR <= 2
```

where RN = 1 ← مع عدم تكرار الموظفين بنفس المرتب

س١: ماهي بيانات الموظف الحاصل على أعلى مرتب داخل كل قسم

RN = 3 ← مع عدم تكرار الموظفين بنفس المرتب

س٢: ماهي بيانات الموظف الحاصل على ثالث أعلى مرتب داخل كل قسم؟

DR = 1 ← مع تكرار الموظفين الحاصلين على نفس المرتب

س٣: ماهي بيانات الموظفين الحاصلين على أعلى مرتب داخل كل قسم؟

DR <= 2 ← " " "

س٤: ماهي بيانات الموظفين الحاصلين على أعلى مرتبين داخل كل قسم؟

Case When
IIF()
Formate()

```
Case When ---- then -----
        else ----
End as ----
```

```
Select E-name, Salary, Case
                When Salary>=3000 then 'high salary'
                When Salary< 3000 then 'Low'
                else 'No value'
                End as NewSalary
From Employee;
```

E-name	Salary	NewSalary
Ahmed	10 000	High
Ali	2000	Low
Sally	null	No value
⋮	⋮	⋮

```
Update Employee
        set Salary =
                Case
                When Salary >=3000 then Salary*1.10
                else Salary*1.20
                End ;
```

iiF(Condition, Yes, No)

```
Select E-name, iiF(Salary>=3000, 'high', 'Low')
From Employee;
```

Formate(Date, 'dd-MM-yyyy')

```
Select Format(E.Birthdate, 'dddd-MMMM-yyyy')
From Employee;
```

'dd-MM-yyyy' / 'ddd-MMM-yy' / 'dddd' / 'MMMM' / 'h:mm:ss'
'HH' / 'hh tt' pm/am / 'dd-MM-yyyy hh:mm:ss tt'

26-3-2023
Day 6

SQL: Day 6
Database Properties (mdf, ldf, File Groups) (Video 00:2:00)
Database Properties (mdf,

Data Base

- Physical Representation
  - Physical Files
    - عبارة عن ملفات تحتفظ على الهارد
    - LdF File — الفايل اللي حايل الـ Queries والـ Transactions على الجهاز
    - Mdf File — الفايل اللي حايل الداتا على الجهاز
    - عبارة عن ملفات تحتفظ على الهارد عشان ما تبقى عليها الداتا وممكن تكون مساحة التخزين أكتر من هارد وكل Ndf موجود على هارد مختلف عن الآخر وهذا بيعمل على رفع الـ Performance لأن الـ operations تعمل بالتوازي بمعنى أن تكون أكتر من هارد شغال في نفس الوقت وبالتالي أكتر من operation شغالة في نفس الوقت
    - .Ndf
    - .Ndf
    - .Ndf
- Logical Representation
  - File Group — عبارة عن ملفات بتحتفظ على SQL
  - Primary F.g
  - FG2 ← student
  - FG3 ← Dept
  - FG4

join

لما تيجي تcreate جداول حطهم علي hard discs مختلفين عشان من مميزاته:

1- يعمل علي زياده الperformance لان لما تيجي تعمل joins بين جدولين الهاردين بيشتغلوا مع بعض في نفس الوقت .

1- لما تيجي تعمل backup ممكن تعمله علي file group واحد مش ال database كلها ودا ساعتها هيبقي خفيف مقارنه بال database كلها .

Create Schema

```
Create schema Sales ;
```
→ هنا هيعمل schema جديد اسمها Sales بالإضافة للـ schema الإفتراضية dbo.

```
Alter schema Sales transfer department ;
```
→ القيام بنقل الـ department Table من dbo إلى Sales .

```
creat table student
(Id int,
name varchar(20)) ;
```
→ هنا سوف نقوم بعمل جدول الـ student داخل dbo لأنه طالما لم يكتب الـ schema فتكون الإفتراضية

```
creat table HR.student
(id int,
name varchar(20)) ;
```
→ هنا هو ذكر الـ schema (HR) وبالتالي يتم وضع الجدول داخل الـ HR schema

لاحظ وجود أكتر من جدول يحمل نفس الاسم student في نفس الـ DB لكن في schemas مختلفة

الschema تعمل علي حل 3 مشاكل :

2- اقدر اعمل 2objects بنفس الاسم في نفس ال database لكن في 2 Schemas مختلفين. ومش شرط يكون جدولين لا ممكن يكون (table & view)

3- تنظيم ال tables في logical groups لتسهيل التعامل معها.

4- اقدر اربط user معين ب Schema كامله مش table معين.

Create Synonym

```
Create synonym HE
For HumanResources.EmployeeDepartmentHistory ;
```
→ * هنا نقوم بعمل shortcut لكل ما هو جنب For بحيث بدلا من كل مرة أحتاج أكتب الإسم والـ schema في كل Query هنعوض مكانه ب HE فهو عبارة عن alias name للـ Table لكن بيتخزن داخل الـ synonym .

```
select E_Id
from HE ;
```

```
Alter synonym SD
For HumanResources ---------- ;
```
→ الإسم الجديد

* بتغييرنا كما لما نحتاج نغير الـ Table من الـ queries لكن مستخدم في Queries كتير فوجدوه لكن بإستخدام الـ synonym عن طريق Alter وقت نيم تغير الـ Table .

Drop

```
drop table ---- ;
```
هنا هيقوم بمسح الـ Table بالكامل الـ data + Meta data

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

Delete

```
delete from ----
where -
```
تقوم بمسح الـ data فقط

هنا where تأتي مع delete ولا تأتي مع Truncate

عادة ما تكون delete أبطأ من Truncate لكن أنضف لأنها تمسح بعمل RollBack عكس Truncate لا تمسح

Truncate

```
truncate table ----
```
هنا نقوم بمسح الـ data فقط زي delete .

* يتم إستخدامها في حالة وجود data كبيرة وعايز أحذفها أنا في بعد المسح .

Truncate بتعمل Reset لل identity في الـ Table اما delete لا مبتعملش .

Data Base Integrity

	Domain integrity	Entity integrity	Refrential integrity
	Range of Values (Columns)	Uniqueness (Rows)	Relationship
DB Constraints	- Data type - Default - Allow null - Not null - Check Constraint	- PK Constraint - Unique Constraint Allow one null value	- Foreign key Constraint
DB objects	- Rules - Triggers	- Index - Triggers	- Triggers

دا بنعمله لما اجي create table بحط condition داخل الquery زي مثلا تكون الأعمار بتاعت الطلبه من 20 الي 30.

الUnique key زي ال PKلكن ب Allow one null ويستخدمه زي مثلا بحط مرتبات كل الموظفين لكن بخفي مرتب المدير وبخليه null

What types of constrains on database?
1-Check Constraint
2- PK Constraint
3-Uniqe key Constraint
4-Foreign key Constraint
5- Custom Constraint → SP or trigger زي مثلا لما اجي اعمل عشان أقول امي الموظف يترقي مثلا زي مثلا لما أقول لما يعدي عليه 10 سنين واشتغل ف 5 Different Project.

```
create table emp
( eid int
  netsalary as (isnull(Sal,0)+isnull(overtime,0)) Persisted
  age as (year(getdate()) - year(Birthdate)) Persisted
)
```

error

لاحظ Day6 2:18:10
netsalary في حالة الPersisted
الFormula غير مكتوب بها
Function متغيرة مع الوقت
زي ما مكتوب داخل ال age
ال getdate() هي Function متغيرة
تعطي تاريخ اليوم زي بعض أنها
تتغير يوميا وبالتالي سوف يضطر
ال system إلى حساب ال Formula
يوميا لإختلاف التاريخ ولذلك
لا يسمح بعمل Persisted ويعطي error

Create Constraint

```
Create Table emp
( e_id int identity(1,1),
  ename varchar(20),
  eadd varchar(20) default 'Alex',
  hiredate date default getdate(),
  Sal int,
  overtime int,
  netsal as (isnull(Sal,0) + isnull(overtime,0)) Persisted,
  BD date,
  age as (year(getdate()) - year(BD)),
  gender varchar(1),
  houxrate int notnull,
  did int,
  Constraint C1 primary key (e_id, e_name),
  Constraint C2 unique (Sal),
  Constraint C3 unique (overtime),
  Constraint C4 check (Sal >1000),
  Constraint C5 check (e_add in ('Cairo','Mansoura','Alex')),
  Constraint C6 check (gender = 'F' or gender = 'M'),
  Constraint C7 check (overtime between 100 and 500),
  Constraint C8 foreign key (did) references dept(dept_id)
                on delete [redacted] set null on update cascade
);
```

Database Integrity
Domain Integrity (Columns)
Entity Integrity (Raws)
Referential Integrity (Relationship)

Day6 2:23:03
لاحظ في C2 ، C3 تم جعل Sal و overtime unique كل على حده ولم يتم دمجهم في Constraint واحد يجمعهم ربما عشان لا يمر بطهشه ببعض ونفهم ان القيمتين مع بعض unique

لاحظ ممكن عمل كذا Constraint على نفس الcolumn زي C2 و C4 C3 و C7

لاحظ ممكن عمل كذا check Constraint

لاحظ تم عمل C8 كبديل لل Diagram

Alter table emp add Constraint C100 Check (hour_rate>100) ,
Alter table emp drop Constraint C3 ,

لاحظ
- ممكن أضيف Constraint على Table موجود وفيه data لكن لازم أتأكد إن الـ Constraint متوافق مع الـ data القديمة.
- ممكن أمسح الـ Constraint عن طريق Alter .

Day 6
2:34:31

Rule

- Constraint $\xrightarrow[\text{على}]{\text{يطبق}}$ New Data
- Constraint $\xrightarrow{\text{يكون}}$ Shared between Tables
  Not only one Table
- create New datatype $\xrightarrow[\text{مربوط}]{\text{يكون}}$ Constrain $\xrightarrow[\text{لـ}]{\text{يكون}}$ Default Value

بيتم إستخدامه في حالة إني عاوز أعمل Constraint يطبق على الـ New data فقط أو عاوز أعمل Constrain بيتم إستخدامه في أكثر من Table أو عاوز أعمل datatype جديدة وأربطه بـ Constraint

لاحظ
* لاحظ هنا أنا عاوز أعمل Constraint على الـ sal من جدول emp والجدول فيه بالفعل rule وتم وضع بداخله المتغير (X) ومن ثم تم إستخدام الـ stored procedure لتعمل على تغيير (X) بـ الـ emp.sal عن طريق sp_bindrule .
* لاحظ هنا تم إستخدام نفس الـ rule على Table تاني وهو الـ instructor .

- create rule r1 as @X > 1000
- sp_bindrule r1, 'emp.sal'
- sp_bindrule r1, 'instructor.sal'

- sp_unbindrule 'emp.sal'
- sp_unbindrule 'instructor.sal'
- drop rule r1

لاحظ
عند حذف الـ rule لابد أولاً عمل sp_unbindrule للـ Tables إلى الـ rule مطبقة عليها ومن ثم نقوم بعمل drop rule .

لاحظ
لو قمنا بتطبيق rule 2 على الـ Column وكان مطبق عليه rule1 سوف يقوم بعمل over_ride للـ rule1 القديمة ويطبق الـ rule2 الجديدة وبالعكس الـ Constraint بحيث الـ Column مسموح أطبق عليه كذا Constraint ولكن مسموح أطبق عليه rule واحد فقط .

Default

- create default def1 as 5000
- sp_bindedefault def1, 'emp.sal'
- sp_bindedefault def1, 'instructor.sal'

- sp_unbindedefault 'emp.sal'
- sp_unbindedefault 'instructor.sal'
- drop default def1

لاحظ
هنا لو حابب أخلي الـ default برة الـ Table بحيث استخدمها زي الـ rule كده على أكثر من Table لاحظ كمان عند حذفها لازم الأول أعمل unbinde للـ stored procedure الموجودة

Add Constraint to table
Drop Constraint
Create Rule
Create Default

Create Datatype

--Create Datatype

باسم → ComplexDT يكون → (int) وكمان → (>1000)

default value 5000 ولو

لاحظ

الهدف من عمل ال rule وال default هو عمل New datatype داخل ال DB ويكون لهذا ال datatype نوع و default value ويتم تطبيقها على أكثر من table ولا تتعارض مع الداتا القديمة

```
Create rule r1 as @X > 1000
Create default def1 as 5000
sp_addtype ComplexDT, 'int'
sp_bindrule r1, ComplexDT
sp_bindefault def1, ComplexDT
```

Exmp

```
create table test3
(id int,
name Varchar(20)
salary ComplexDT
);
```

لاحظ

ال column بيطبق عليه rule واحد فقط لأن الهدف من ال rule هو عمل new datatype وبالتالي مبيفرقش يكون ال column كذا datatype.

Create Data Type

Day 7
28/3/2023

SQL: Day 7
Variables
Local Variables

Variables

Local Variables @

يستخدم في الـ
batch
Function
Stored Procedure

* To initialize values inside the Variable:

1. Declare @X Int
2. * Set @X = 100
   * Select @X = 100
   * Select @X = age From Student Where id = 1
   * Update student Set F_name = 'omar', @X = age where id = 9
3. Select @X

لاحظ: عشان نعمل Variable لازم نعملها declare الأول عشان نعرف نوعها

لاحظ: عشان نحط Value الـ Variable ويظهر معاك لازم نعمل الـ 3 steps مع بعض في نفس الـ Patch

Global Variables @@

هي عبارة عن متغيرات مخزنة داخل الـ SQL ويتم استخدامها بدون عمل ليها declare ولا بيوضع لها قيمة الهدف منها العرض فقط

Select @@ servername
بتعرض اسم السيرفر

Select @@ rowcount
بتعرض عدد الـ rows الـ affected في آخر run statment

Select @@ Version
بتعرض نسخة الـ SQL Server إلى شغال عليها

Select @@ error
بتعرض رقم الـ error massage إلى ظهرت في آخر run statment أنا عملتها ولو مكنتش حصل error هتبين zero(0)

Select @@ identity
بيعرض آخر identity دخل للـ Table من آخر Insert نفسه بيعرض رقم آخر value دخلت الـ Table ولو الـ Table مفهوش identity وجبت أعلى Select @@ هتطلع null.

Select @X = @@ rowcount ✓
Select @@ rowcount = @X ✗✗

Local Variables

```
declare @X int
set @X=10
Select @X
```

```
declare @X int
Select @X = 10
select @X
```

```
declare @X int = (Select avg(st_age) From student)
select @X
```

لما باجي ا assign قيمه للVariable باستخدم ال Select في 3 احتمالات :

1- الSelect هترجع قيمه ← خلاص الVariable هياخد القيمه دي.
2- الSelect مرجعتش ناتج ← الvariable هيحتفظ ب اخر قيمه كانت جواه ومش هيتحط ب Null.
3- الSelect هترجع Array وال Variable معرف انه ياخد one value ← الVariable هياخد اخر قيمه ف ال array.
4- الSelect هترجع Array وال Variable معرف انه Table ← الVariable هاي Assign الoutput دا في الtable.

```
declare @y int
Select @y = st_age From student where st_id = 6
Select @y
```

لاحظ: هنا هو هيروح جيب الـ age من الجدول ويحطها كـ Value داخل الـ Variable ويعرض الـ Variable

وهنا في ٣ سيناريوهات

1. إن الـ Query طلعت أكثر ناتج من الجدول كان الـ Variable فيتا قيمة من البداية وساتها الـ Variable هيعرض الـ Value المحفوظة جواه.
2. إن الـ Query طلعت كذا ناتج عبارة عن array ساتها الـ Variable هياخد آخر قيمة في الـ array.
3. إن الـ Query طلعت ناتج واحد زي المثال وبالتالي الـ Variable هياخد الناتج كـ Value جواه.

variable = 1 2 4 (20, 21, 22)

```
declare @y int, @name varchar(20)
Select @y = st_age, @name = st_Fname From student where st_id = 7
Select @y, @name;
```

لاحظ: لا يمكن استخدام Select للعرض والـ insert في نفس الوقت.
حيث كنت أكتب Select @y = st_age , st_fname معشان كدا تم عمل الـ Variable الآخر.
insert — عرض

```
declare @Z int
update student set st_name = 'Ali', @Z = dept_id
Where st_id = 7
Select @Z;
```

لاحظ: هنا هو قام باستخدام الـ Variable مع الـ update
حيث إنه هيعمل update لاسم الطالب مراحب الـ id رقم 7 وهيغير إسمه 'Ali'
وهيروح يشوف رقم الـ department بتاعه (dept_id) ويحطه كـ Value داخل الـ Variable
وبعدين الـ Variable.

```
declare @t table (X int)
insert into @t
Select St_id From Student where St_address='Alex'
Select * From @t;
```

X
1
5
7
10

لاحظ: ممكن أعمل Variable من النوع Table وباستخدام (insert based on Select) أقدر أملأ الجدول بالداتا وساعتها الجدول دا هيكون في الـ memory

```
declare @X int
Select top(@X) *
From Student;
```

لاحظ: تم إستخدام الـ Variable بداخل الـ Top عشان كل مرة الـ Variable تتغير يفرض Top مختلف.

```
Select 'Select * From Student'
```
output → 'Select * From Student'

```
Execute('Select* From Student')
```
output →

id	name	add	gender	age
-	-	-	-	-
-	-	-	-	-

لاحظ: الـ execute تقوم بتحويل الـ String إلى Query لو ينفع وبعد كده تعمل run

---

```
declare @Col Varchar(20) = '*', @tab Varchar(20) = 'Student'
Select @Col From @tab
execute('Select'+@Col+'From'+@tab);
```

لاحظ: هنا تم إستخدام execute عشان أقدر أحول الـ String لـ Query وتشتغل عشان الـ From كان بيها String.

Global Variables

```
Select @@Servername
```
→ هيظهر إسم السيرفر

```
Select @@Version
```
→ هيظهر نسخة الـ SQL Server

```
update Student
  set St_age+=1
Select @@RowCOUNT
```

لاحظ: تم إستخدام RowCount لإظهار عدد الـ Rows affected من الـ Query

```
Select * From Stud
go
Select @@error
```

لاحظ: لا يوجد Table باسم Stud وبالتالي الـ Query سوف تعطي error وباستخدام الـ @@error هقدر أحصل على رقم الـ error massage ولو كان الـ Query طلع error كان @@error طلع Zero (0)

لاحظ: تم إستخدام go بين الـ Two queries وعشان أقدر أعمل run للـ Two queries مع بعض في نفس الوقت.

Control of flow statment

IF

```
declare @X int
update Student
  set St_age+=1
Select @X=@@RowCOUNT
iF @X>0
    begin
    Select 'multi rows Affected'
    End
else
    begin
    Select 'No Rows Affected'
    End;
```

لاحظ: تم إستخدام الـ Variable (@X) عشان يرجع القيمة الناتجة من الـ Global Variable (@@RowCount)

لاحظ: تم إستخدام iF Statment وبداخلها [begin--End] ودول بيتم إستخدامهم مع iF لما يكون بداخلها كود مكون من أكتر من سطر بدلاً من في باقي لغات البرمجة ومفيش مشكلة لما أستخدم begin--End وأنا عندي سطر واحد فقط بداخل iF.

كيرلي براكتس
Carely Practice

Execute Vs Select
Global Variables

IF Exists

```
-- if exists (select name from sys.tables
              where name = 'students')
    select 'table is existed'
-- else
    create table students
    (id    int,
     name varchar(20)
    );
```

لاحظ: هنا أول حاجة لازم تاخد بالك منها هي (sys.) ودي بقا عبارة عن built in schemas داخل الـ DB محفوظ بداخلها كل الـ meta data الخاصة بالـ DB يعني محفوظ بداخلها كل أسماء الـ Tables

لاحظ: تم إستخدام if exists لعمل check على الـ Database أولا ومعرفة هل اسم الـ Table إللي عاوز أعمله موجود بالفعل ولا لأ فلو كان موجود هيطلعلي massage إنه موجود ولو مكنش موجود هينفذ ما بداخل else.

IF Not Exists

```
-- if not exists (select deptid from student where dept_id=20)
   and not exists (select dept_id from Instructor where dept_id=20)
      Delete from department where dept_id=20
-- else
      select 'table has relationship'
```

لاحظ: هو هنا هيروح يشوف لو مفيش dept_id=20 في الـ student و instructor هيعمل delete dept_id=20 من الـ department لاحظ هنا أنا قولت لو مفيش لـ نفس (if not exists) عكس if أنا بقول (if) لو في كنا إعمل كذا

لاحظ: أنا عملت المثال الموجود في (if not exists) باستخدام Try_Catch إحنا في Try بنحط الـ query وبنقله نفذها وعشان في إحتمالية إن يحصل error بنروح نعمل Catch عشان لو حصل error يوقف تنفيذ الـ query ومبيعرضش في الـ app وباستخدام الـ Functions مع Select ممكن نعرف بيانات الـ error دي.

Try ... Catch

```
-- begin try
      delete from department where dept_id=20
-- End try
-- begin catch
      select 'Error'
      select ErrorLine(), Error_Number(), Error_massage()
-- End catch
```

While

```
-- declare @X int=10
   -- while @X<=20
      -- begin
            set @X+=1
            -- if @X = 14
                  continue
            -- if @X=16
                  break
            -- select @X
      -- End
```

لاحظ: * في SQL لعمل Loop بيتم إستخدام while

* Continue: بتخلي الـ أكمل شوية وترجع تاني لفوق عند الـ Condition عشان كدا 15 مخرجتش لما وصلت @X=14 شرط IF الأولى تحقق فالتالي دخلت لـ Continue فرجعت لفوق بعد ما أنزل أطبع رقم 14

The output
11, 12, 13, 15

If Exists
If Not Exists
Try.... Catch
While

batch ... script ... transaction

batch

- Insert ——
- update ——
- delete ——

← هو عبارة عن مجموعة من Queries ملهاش علاقة ببعض بس ممكن أحددهم مع بعض وأعملهم run مع بعض ومبيأثرش على بعض.

لاحظ: من ال batch لو فى Query فيها مشكلة وباقى ال Queries مفهمش هيعملوا run عادى وملهوش علاقة بال Queries إلى فيها مشكلة.

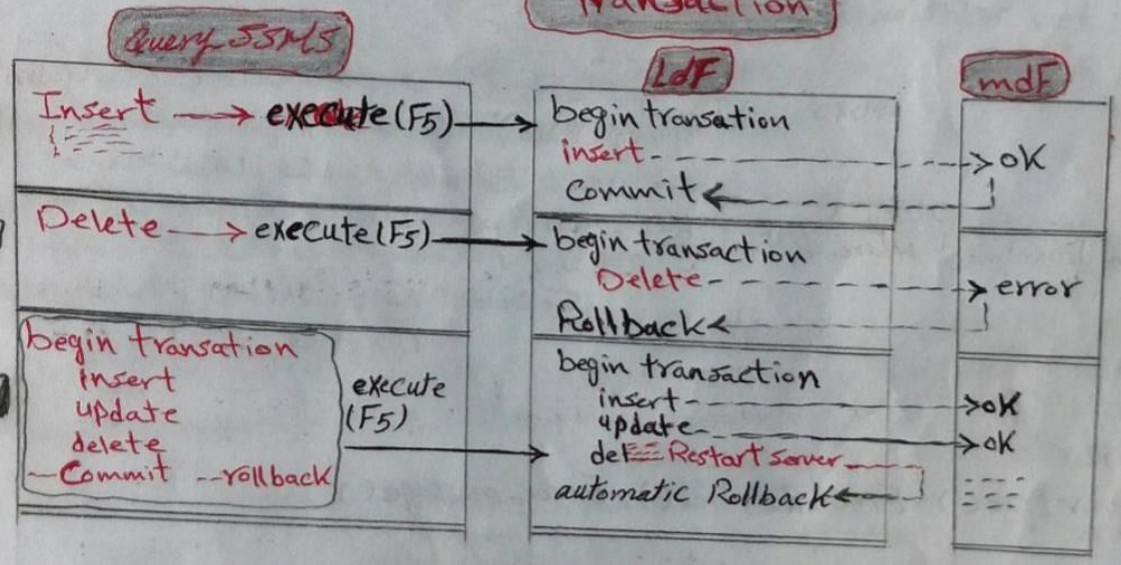
script

Create table
go
drop table
go
creat rule
go
SP_bindrule

← هو عبارة أيضاً عن مجموعة من ال Queries ملهوش علاقة ببعض بس مقدرش أعملهم run بعض وعشان كدا بفصل بينهم ب go وغالبا بتبقى ال Queries إلى ليها علاقة ب Metadata

Batch
Script
Transaction (Query → Ldf → mdf)

Transaction



لاحظ: بستخدم ال Transaction فى حالة إنى لازم أعمل run لكذا Query فى نفس الوقت وميفعش واحدة تشتغل والباقى لا زى مثلاً سحب الفلوس من ال atm لازم كل ال process تشتغل مع بعض وإن يوقف وميفذش حاجة يعنى منفعش تخصم الفلوس من الحساب وتعطل وتطلعش فلوس، عشان كدا باستخدم Transaction واستخدم باخلى commit أو rollback لو ملقاش الكلمتين دول هيعمل automatic Rollback.

```
begin try
    begin transaction
        insert into child values (1)
        insert into child values (2)
        insert into child values (3)
    Commit
End try
begin catch
    rollback
End catch
```

لاحظ: عشان commit, rollback مبيتحطوش عشوائياً كدا داخل ال Transaction إستخدمت try — catch وداهيجلية يروح ينفذ إلى جوا ال Transaction ولو ملقاش error هيعمل commit ولو فى error هيدخل catch ويعمل rollback.

Functions

Scalar Functions

* Built in Functions

- null → isnull( ), Coalesce( ), NulliF( )
- system → db_name( ), Suser_name( )
- Convert → Convert( ), Cast( ), Formate( )
- String → Substring( ), upper( ), lower( ), Len( )
- Date → getdate( ), Year( ), Month( ), day( )
- Agg → Count( ), max( ), min( ), Avg( ), Sum( )
- Power → Power( ), Log( ), Sin( ), Cos( ), tan( )
- ranking → Row_number( ), Dense_rank( ), Ntile( )
- Logical → IIF( ), Choose( )
- Windowing → Lead( ), Lag( ), First_value( ), Last value( )

* User Defined Functions

- Scalar Function
  ↳ Return One Value.
- Inline table Function
  ↳ Return Table $\xrightarrow[\text{بتكون من}]{\text{body}}$ Select فقط
- Multi statement table Valued Function
  ↳ Return Table $\xrightarrow[\text{بتكون من}]{\text{body}}$ Select + Logical Function (IF_while_declare)

لاحظ
1. لازم كل الـ Functions تـ Return value.
2. كل الـ Functions بيتكتب جواها Selects بس مينفعش أكتب جواها (Insert, update, delete).
3. أي Function إنت هتعملها لازم تبقى عارف هتعملها إيه

Built in Functions

Select getdate( );

```
Select isnull(St_name, ' ')
From Student;
```

لاحظ: في Isnull هو هيروح يجيب الـ St_name لو لقاها
ب null هيرجع مكانها ' '

```
Select Coalesce( St_Fname, St_Lname, Address)
From Student;
```

في Coalesce هو هيروح يجيب الـ St_Fname لو لقاها ب null
هيروح يجيب St_Lname ولو لقاها ب null هيروح
يجيب الـ Address.

```
Select upper(St_name), Lower(St_lname)
From Student;
```

لاحظ: الاسم بيتعرض زي ما إنكتب جوا الـ DB الحروف
الـ lower هتطلع small والحروف الـ upper بتطلع capital
إلا لو أنا زي ما عملت هنا استخدمت الـ upper و lower كـ Functions
وطلبت منه ينفذهم ويعرضهم بـ Format معين.

```
Select Len (St_Fname), St_Fname
From Student;
```

Len	St_Fname
5	Ahmed
3	Ali
5	Sally

لاحظ: Len بتحدد طول الكلمة من خلال
عدد حروف الكلمة أو الجملة.

```
Select Max (St_Fname)
From Student;
```

لاحظ: كل حرف على الـ Key board بيبقى ليه رقم متخزن بيه ولما أقول
هات Max وبدخله حروف بيروح يشوف أول حرف من كل كلمة والكلمة
إلى أول حرف فيها له أكبر رقم هتبقى max ولو في كلمتين بيبدأوا بنفس الحرف
هيروح يشوف الحرف التاني من الكلمتين والكبير يأخذ الكلمة تبعته Max.

```
Select top(1) St_Fname
From Student
order by Len(St_Fname) desc;
```

هنا أنا كنت عاوز أعرض أطول كلمة
أو جملة موجودة بالـ St_Fname
فرتبت الأول بـ order by على حسب
الـ Len( ) ثم اخترت top( ).

Functions
Bult in Functions (upper,Lean.....)

```
Select power(salary, 2)
From Instructor;
```

لاحظ: هو هنا هيجيب الـ salary ويرفعه لـ (2).

```
Select Convert(varchar(20), getdate()), 101)
```

البديل

```
Select format(getdate(), 'dd-MM-yyyy')
```

Lag ---- Lead ----- Firstvalue --- Last value Functions

Lag ___ over ___ order by
Lead ___ over ___ order by

```
Select Sname
    , grade
    , X = Lag(grade) over(order by grade)
    , Y = Lead(grade) over(order by grade)
From Student;
```

لاحظ: الداتا اللي في الجدول مترتبة غلط ولازم تبقى مترتبة تصاعدي عشان الـ order by

Sname	grade	X	Y
omar	80	null	84
Sally	84	80	90
Ahmed	90	84	70
Hassan	70	90	95
Heba	95	70	null

لاحظ: هنا Lag موجود داخلها الـ grade وبالتالي هيرجع تجيب الـ grade تبعت الطالب إلي فوقه في الجدول وتحطه في (X) أما lead هترجع تجيب الـ grade إلي فوقه في الجدول وتحطه قدام Y

← لو جيت تبص على Sally هتلاقيها جايبة 84 وإلي فوقها جايب 80 وإلي تحتها جايب 90.

لاحظ: هنا أنا جبت الدرجة إلي قبلي وإلي بعدي بغض النظر نفس المادة ولا لا، أنا جيبت إلي فوق وإلي بعدي في الجدول وخلاص

Lag ___ over ___ Partition by ___ order by
Lead ___ over ___ Partition by ___ order by

```
Select Sname
    , grade
    , C-name
    , X = Lag(grade) over(Partition by C-name order by grade)
    , Y = Lead(grade) over(Partition by C-name order by grade)
From Student;
```

لاحظ: هنا باستخدام Partition by قدرت أقسم الطلبة مجموعات على حسب كل مادة ودرجات كل مادة بالطلبة إلي فيها بقوا منفصلين عن درجات الطلبة في المواد التانية وبعدها باستخدام lag_lead قدرت أجيب درجة الطالب إلي فوق وإلي تحت لكل طالب بس اللي في نفس المادة.

Sname	grade	C-name	X	Y
mohamed	80	C-Programing	null	84
Sally	84	C-Programing	80	100
Ahmed	100	C-Programing	84	null
Ahmed	97	HTML	null	110
Heba	110	HTML	97	120
omar	120	HTML	110	null

FIRST_VALUE ___ over ___ order by
LAST_VALUE ___ over ___ order by

```
Select Sname
    , grade
    , X = LAG(grade) over(order by grade)
    , Y = LEAD(grade) over(order by grade)
    , First = First_value(grade) over(order by grade)
    , Last = Last_value(grade) over(order by grade)
From Student;
```

لاحظ: First value / Last value بيرجعوا القيمة إلي في أول صف وآخر صف في الجدول وده على حسب الـ order by اللي حاطه.

Sname	grade	X	Y	First	Last
Mohamed	80	null	60	80	100
omar	60	80	70	80	100
Sally	70	60	85	80	100
Ahmed	85	70	90	80	100
hassan	90	85	100	80	100
heba	100	90	null	80	100

لاحظ: الداتا اللي في الجدول مترتبة غلط ولازم تبقى مترتبة تصاعدي عشان الـ order by.

Lag Function / Lead Functions
First Value Functions / Last

FIRST_VALUE — over — Partition by — order by
LAST_VALUE — over — Partition by — order by

```
Select S_name, grade, C_name
    , X = LAG(S_name) over(Partition by C_name order by grade)
    , Y = LEAD(S_name) over(Partition by C_name order by grade)
    , First = FIRST_VALUE(S_name) over(Partition by C_name order by grade)
    , Last = LAST_VALUE(S_name) over(Partition by C_name order by grade)
From Student;
```

S_name	grade	C_name	X	Y	First	Last
mohammed	80	C-Programing	null	Sally	mohammed	Ahmed
Sally	84	C-Program	mohammed	Ahmed	mohammed	Ahmed
Ahmed	100	C-Program	Sally	null	mohammed	Ahmed
Ahmed	97	HTML	null	Heba	Ahmed	omar
Heba	110	HTML	Ahmed	omar	Ahmed	omar
omar	120	HTML	Heba	null	Ahmed	omar

User Defined Functions
Scalar Function
Inline Table Function

Create My own Functions

Scalar Function

```
Create Function getsname (@id int)
Returns Varchar(20)
begin
    declare @name varchar(20)
    Select @name = St_name From Student Where St_id = @id
    return @name
End
```

لاحظ * هنا أنا بعمل Function بتاخد Id بتطلع الـ St_name
* عشان تطلع one output تبقى Scalar Function .
* كمان ممكن أحط الـ Schema جنب اسم الـ Function زي مثلاً dbo.getsname أو HR.getsname .

```
Select dbo.getsname(7)
```
مثلاً 3

لاحظ عشان أعمل Call للـ Function إلى أنا كاتبها لازم أحط الـ Schema جنب الـ Function .

Inline Table Function

```
Create Function Getinst (@d_id int)
returns table
as
return
(
    Select ins_name, salary*12 as totalsalary
    From Instructor
    where dept_id = @d_id
)
```

لاحظ * هنا أنا بعمل Function بتاخد رقم القسم d_id
ترجع جدول فيه ins_name والـ annualsal
وعشان ترجع جدول والـ Function body موجود فيه Select بس بقت من النوع Inline Table
ولكن من Multi Statement .
* لما جيت أجيب الـ annual salary ضربت salary Column * 12 وكمان لازم أديلها اسم جديد
عشان الـ Function لازم تطلع جدول أسماء الـ Columns واضحة .

```
Select * From Getinst(7);
Select ins_name From Getinst(7);
Select Sum(totalsalary) From Getinst(7);
```
مثلاً 10

لاحظ لما جيت أعمل Call للـ Function كمان لازم أعمل Select * عشان هي عبارة عن Table من الأول والآخر
ولازم الـ Table يجي بعد From .

Multi Statement Table Function

```
create Function getstuds (@Format Varchar(20))
returns @t table
        (
            id int,
            Ename varchar(20)
        )
as
begin
    iF @Format='First'
        insert into @t
        Select St_id, St_Fname From Student
    else iF @Format = 'Last'
        insert into @t
        Select St_id, St_Lname From student
    else iF @Format='Full'
        insert into @t
        Select St_id, St_Fname+' ' + St_Lname From Student
    return
End

Select * From getstuds ('First');
Select * From getstuds ('Last');
Select * From getstuds ('Full');
```

لاحظ هنا أنا بعمل Function أديها اسم الطالب
تطلعلي ال id, name وكده الاسم إلي هيطلع
على حسب هل ال First ولا Last ولا ال Full
وده على حسب الكلمة إلي هاديها وإلا هيطلع
جملة if.

Multi-Statement Table

لاحظ في ال Inline Table Function كنت بعمل Variable واحد وبستخدمه كـ input
وعشان بستخدم Select بس جوا النوع دا فال output دايماً بيبقى معروف ال Table structure
تبعك فكنت مش محتاج أعمل أنا Table ال output وأعمله Variable زي هنا

هنا بقى أنا بعمل متغيرين واحد هيكون ال Function (@Format) وده عشان استخدمه
في ال logical Functions وعمل طريقة هتحدد ال output وبعد كده عمل طريق ال Varible التاني (@t)
أقدر أملى الجدول إلي أنا عملته بال Data المحددة كـ output عن طريق طريقة Insert
based on Select.

Day 8
21/1/2023

SQL: Day 8
Index
Clustered index
Non-Clustered

Index

* تقدر إنه عندي Table without P.K وعاوز أحفظها معناة إنه الـ Data معمولها Save كـ Heap بمعنى إنه الداتا دي not sorted ودا بيترتب على إنتا لما يجي يعمل Select أو Search مش حاجة بيعمل Table Scan بمعنى إنه مضطر يمشي على الـ Table من أوله لآخره عشان مش قادر يوصل للداتا إلى بدور عليها حتى لو كانت الـ Data دي في أول Row في الـ Table.

* لاحظ في الحالة دي الـ Insert هنا هتبقى سريعة جداً لأنه لما يجي يدخل أي حاجة جديدة يرميها في آخر الـ Table لعدم وجود ترتيب.

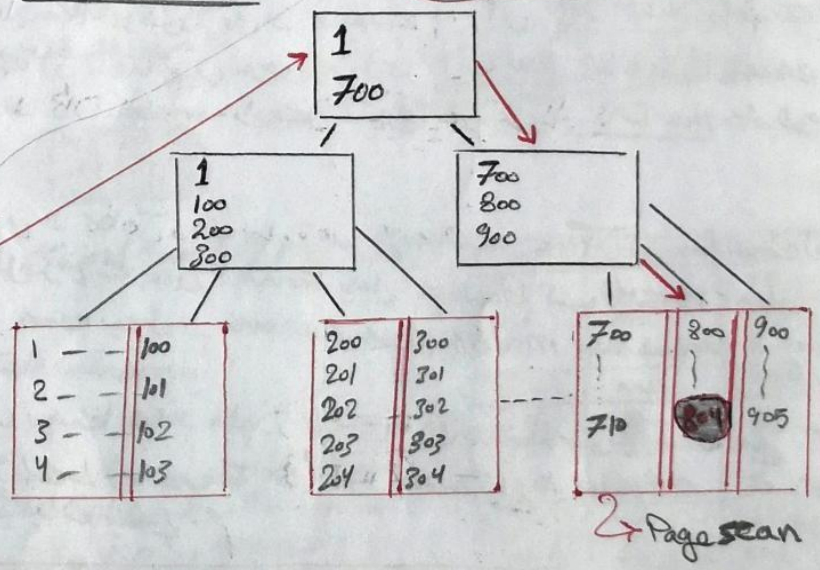
Clustered Index

* Table is Sorted with Primary Key.
* Search with P.K

P.K → Clustered Index

S-Id	S-name	S-age
1	Ahmed	21
2	Ali	24
4	Eman	23
7	Omar	25

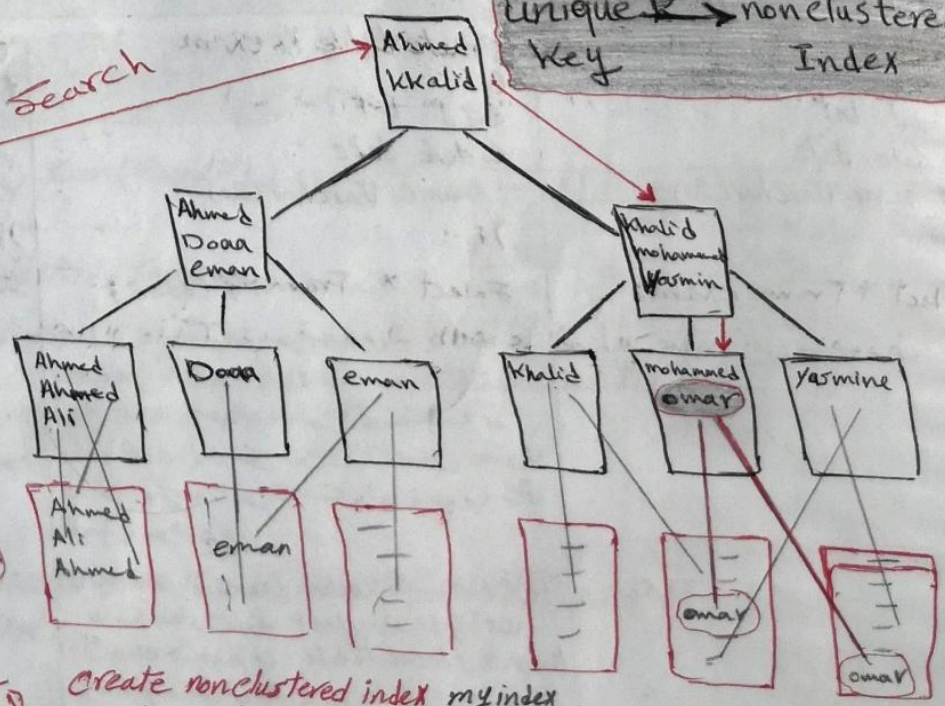
Select *
From Student
where S-Id = 804



NonClustered Index

Select *
From Student
where name = omar

unique Key → non clustered Index



لاحظ
* أقدر أعمل أكتر من nonCluster
لكن مقدرش أعمل غير Cluster واحد وغالباً بيبقى مع الـ P.K
* بيتفضل أكتر من الـ Indexs لأنه بيستهلك الـ memory.

لاحظ
لو أنا بـ Create Table مفيش P.K ممكن أجعل الـ Cluster على أي Culmns زي مثلاً الـ age.

لاحظ
P.K → Constraint → clustered index
unique → Constraint → non clustered index
بمعنى إنه لو أنا بـ Create Table وفيه unique و P.K فتلقائي الـ P.K هيبقى Clustered والـ unique هيبقى تلقائي non Clustered.

```
create nonclustered index myindex
on Student (St_Fname);

create nonclustered index myindex_2
on Student (St_Lname);

create unique index i4
on Student (St_age);
```

⊛ عشان تحدد إيه الـ Culmns إلى بتستخدم كتير والمفروض تعمل عليها هي الـ Index مكان تستخدم

SQL Server Profiler and SQL Server Tuning Advisor

SQL Server Profiler: لما بتشغلها بيعمل Trace File وده File بيتسجل فيه كل العمليات إلى عملتها على الـ SQL

SQL Server Tuning Advisor: هو عبارة عن برنامج بدخله الـ Trace File الخارج من الـ Profiler بيعمل عليه Analysis وبيقولي إيه الـ Culmns إلى المفروض أعمل عليها الـ Indexs.

System Databases

1) الـ master :- فيها كل الـ Configration الخاصة بالـ Server يعني إن فيها كل أسامي الـ users وكل الـ permissions والـ Passwords الخاصة بهم وكل أسامي الـ DBs وأماكنهم على الهارد. ولو ضاعت أو اتمسحت مش هتقدر أـ Connect على الـ Server

2) Meta data الخاصة بالـ Server

2) model :- هي عبارة عن الـ template الموجودة على الـ Server إلى أي New DB بتاخد منها Image. يعني لو أنا عاوز user أو Table يبقى موجود في أي New DB أنا هعملها create يبقى أروح أعمله الأول في الـ model DB - ولازم تأثيرها يبقى على الـ New DB مش الموجودة بالفعل.

2) Templet DB

3) msdb :- هي المسؤولة عن التحكم في الـ Jobs ودي عبارة عن Query ب time أو Condition لما بيتحقق فيه action بيعمل زي مثلا عاوز الـ Server يعمل Backup الساعة 12:00 أو عاوز أعمل Backup لما عدد الـ users يوصل لـ 1000. هنا بقى الـ msdb يبدأ ب run الـ Query دي.

4) tempdb :- هي عبارة عن DB إلى بيحفظ بها الـ Data المؤقتة إلى هتقعد شوية وتتمسح زي مثلا لما تحب تعمل Local Table (* Tablename) أو الـ output بتاع الـ subquery إلى هتستخدمه في الـ mainquery وبعد كدا هتتمسح.

Physical & Local & Global Tables

```
create table exam
(
e_Id int
edate date
e_name Varchar(20)
);
Select * From exam;
```

هنا الـ Table بقى موجود في الـ آآ

```
create table * exam
(
e_Id int
e_date date
e_name Varchar(20)
);
Select * From * exam;
```

هنا الـ Table بقى موجود في الـ tempdb وهيفضل موجود طول ما الـ Session أو الـ Query page مفتوحة ومتقفلتش وده Local لأنه موجود بس على مستوى الـ Query page لو فتحت واحدة جديدة مش هيبقى موجود.

⊛ زي مثلا الـ online exam إنت بتدخل تحل الأسئلة وتضغط Next والتاني بتاعت الـ exam بتتحفظ في local Table في temp db وهتفضل تعدل في حاجات لحد ما تخلص وتضغط submit الـ Table هتنقل إلى الـ DB وبعدين physical وكدا أبقى ما جهدتش الـ DB وروحت ليها مرة واحدة بس.

```
create table ** exam
(
e_Id int
e_date date
e_name Varchar(20)
);
Select * From ** exam;
```

هنا الـ Table ده يعني كل الناس الموجودة على الـ Server هيقدروا يستخدموه وهيفضل موجود لحد ما كل الـ users الموجودين على الـ Server يخرجوا وطبعا هيبقى موجود بس مش زي الـ Local على الـ Temp db.

* زي مثلا لما كل الـ users يدخلوا meeting وكلهم بيقرا شغالين على الـ Table ده وقبل ما يخلصوا الـ meeting يحفظوا الـ Table في الـ DB

SQL Server Profiler (Video 39:00)
SQL Server Tuning Advisor (Video 39:00)
Types of System Databases (master, model, msdb, tempdb)
Physical & Local &

Rollup

```
Select Salesmanname as name, Sum(quantity) as 'quantities'
From Sales
Group by rollup(Salesmanname);
```

name	quantities
Ahmed	165
Ali	405
Khaled	220
null	790

لاحظ: Rollup بتاخد نفس الناتج بتاع الـ groub by وتضيف عليه نفسه الـ agg Func الموجودة ولاحظ هتشتغل على نفس الـ Column اللي عليه الـ agg.

```
Select product_id, Salesmanname, Sum(quantity) as "Qty"
From Sales
group by rollup(Product_id, Salesmanname);
```

Product_id	Salesmanname	Qty
1	Ahmed	35
1	Ali	145
1	Khaled	30
1	null	210
2	Ahmed	70
2	Ali	90
2	Khaled	70
2	null	230
null	null	440

لاحظ: خد بالك من حاجة إن Rollup بتشتغل مع الـ Column الأول الموجود في group by ولو تلاحظ هنا الـ Column الأول هو الـ product وعشان كده الـ Rollup جابت الكميات كـ sum بالنسبة للـ product_id.

Cube

```
Select Product_id, Salesmanname, Sum(quantity) as 'Qty'
From Sales
group by Cube(Product_id, Salesmanname);
```

Product_id	Salesmanname	Qty
1	Ahmed	35
1	Ali	145
1	Khaled	30
1	null	210
2	Ahmed	70
2	Ali	90
2	Khaled	70
2	null	230
null	null	440
null	Ahmed	105
null	Ali	235
null	Khaled	100

لاحظ: هنا خد بالك الـ Cube بيشتغل على كل الـ Columns الموجودين مع group by وده هتلاحظ من الجدول وده مختلف عن Rollup اللي بيشتغل مع أول Column مع group by بس.

Grouping Sets

```
Select Product_id, Salesmanname, Sum(quantity) as 'Qty'
From Sales
group by grouping sets (product_id, Salesmanname)
order by Salesmanname;
```

Product_id	Salesmanname	Qty
1	null	210
2	null	230
null	Ahmed	105
null	Ali	235
null	Khaled	100

لاحظ: هنا groupby بتوقف grouping sets عن العمل لو تلاحظ الجدول.

Pivot

```
Select *
From Sales
Pivot (Sum(quantity) For Salesmanname In ([Ahmed],[Khaled],[Ali])) as PVT
```

Product_id	Ahmed	Ali	Khaled
1	35	80	145
2	70	70	90

لاحظ: الـ Pivot هو عبارة عن groupby + rotation للـ Columns

لو تلاحظ هنا هو أخد أسماء العاملين من الـ rows وعملهم Columns + عمل groupby للـ product_id بالنسبة ليهم.

Rollup
Cube
Grouping sets
Pivot

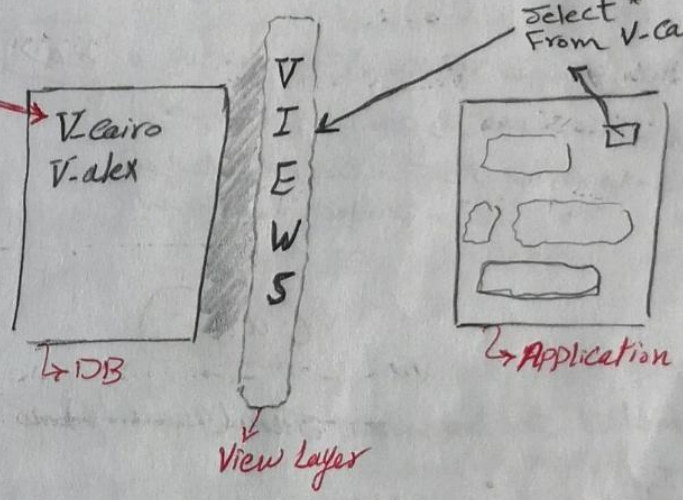
View

Standard View definition

- Is a Select Statement.
- Specify user view of data.
- Hide DB objects.
- Limit access of data.
- Simplify Construction of complex queries.
- has no Parameter.
- has no DML Queries inside its body.
- Standard View can be considered as virtual table
- Only Index View can increase Performance.

- ال View هو عبارة عن Select Statement
- ال View بيحدد لكل user ال View of data الخاصة بيه
- بيعمل hide ال DB عشان يزود ال Security الخاصة بيها.
- لو عندك Join statement أو Subquery بتستخدمه كتير ممكن تعملهم جوة ال View وأعمل كل ال Views بدل ما أنفذ أعمل Join كتير.
- ال View مبيأخدش Parameter عكس ال Function بيأخد.
- ال View و Function مبيكتبش جوة ال body بتاعهم DML Queries

```
Create View V_Cairo
as
Select id, name, address
From Student
where address = 'Cairo';
```



لاحظ ال Standard View بيجيب ال Data من DB أو DBs مختلفين بس شرط كلهم على نفس ال Server

Partitioned View

ال Partitioned View هو عبارة عن View بيجيب ال data من أكتر من Server مختلفين.

```
Create View V_Studs
as
Select *
From Mans_Server.ITI.dbo.Students
union all
Select *
From Sohag_Server.ITI2.HR.Students;
```

لاحظ هنا ال View جايب ال Data من سيرفرين مختلفين Mans_Server و Sohag_Server. وممكن كمان ال View على Server وبيشتغل على Data من على Two Servers مختلفين.

```
Create View V_alex (Sid, Sname, Sadd)
as
Select St_id, St_Fname, St_address
From Student
where St_address = 'alex';

Select S_name From V_alex;
```

لاحظ هنا أنا إديت elias_name ال Columns الموجودة وأخفيت أسماء ال Columns الحقيقية عشان أزود ال Security على ال DB.

لاحظ كمان ال View هنا من النوع Standard عشان جاب ال Data من نفس ال Server إلى شغال عليه.

لاحظ لما أجي أعمل elias name مينفعش أكتب ال (S_id) لازم (Sid)

```
Create View V_all
as
Select * From V_Cairo
union all
Select * From V_alex;
```

لاحظ هنا أنا عملت View كبير مكون من Two Views أو أكتر عن طريق union all.

لاحظ كل View بيبقى له schema وعن طريق ال access بيتحدد الحدود من ال users إلى يقدر يتعامل معاه.

```
alter schema hr transfer V_all;
```

View
Standard View
Partitioned View

```
create view V-Join(Sid, Sname, did, dname)
as
Select S.St-id, S.St-fname, d.dept-id, d.dept-name
From Student S join department d
on d.dept-id = S.dept-id;

Select * From V-Join;
```

لاحظ: هنا أنا استخدمت View عشان أحط فيها Join عشان مش كل شوية 1 و 2 أكتب الـ Query كلها وفيها الـ Join وأروح بسه أكتب Select * From(View).

```
create view V-grades
as
Select Sname, dname, grade
From V-join V Join Stud_Course SC
on V.Sid = SC.St_Id;

Select * From V-grades;
```

لاحظ: هنا أنا قدرت أعمل Join بين View و Table وحطيتهم في New View حيث إني قدرت أعمل Join بين الـ V-Join View والـ Stud_Course table.

Sp_helptext ()
Encryption
DML on the View

Sp_helptext '....'

```
Sp_helptext 'V-join';
```

لاحظ: Sp_helptext ← لو عملت run ليها هتطلعلك الكود إلي إنكتب بيه الـ View

---

With Encryption

```
Alter View V-Join (Sid, Sname, did, dname)
with encryption
as
Select ---
;

Sp_helptext 'V-Join';
```

لاحظ: عند إستخدام with encryption داخل الـ View سوف يمنع ظهور الكود عند إستخدام Sp_helptext أو حتى عند إستخراج الـ Script الـ DB وده بيعمل زيادة الـ security الخاص بالـ DB.

DML on the View

لاحظ: أنا أقدر أعمل عمليات الـ DML (Insert, update, delete) على View موجود بالفعل وكمان تقدر تستخدم الـ DML في الـ body بتاع الـ View وأنا بعمل create

DML → View of one table

```
create View V-Cairo (Sid, Sname, Sadd)
as
Select St-id, St-name, St-address
From Student
where St-address = 'Cairo'
with check option;

Insert into V-Cairo
Values (821, 'Ali', 'Cairo');
```

لاحظ: في النوع الأول دا الـ View جايب data من Table واحد.

لاحظ: ينفع أعمل عمليات الـ DML في الـ View From one table بس بشروط إن الـ Colums إلي مش ظاهرة في الـ View لكن موجودة في الـ table الأصلي تكون بت allow أي حاجة من ٤ حاجات: 1 allow null 2 allow identity 3 Default Value 4 derived Value

لاحظ: ولازم أكون بـ Insert بنفس عدد الـ Columns الموجودة في الـ View.

لاحظ: إستخدام With check option كأنك عملت check constraint وده بيظهر كل ما يجي تعمل Insert بحيث إنه لازم يتأكد الأول من الـ Condition إلي مع Where زي مثلا هنا لازم هيتأكد إن الـ address إلي مدخله يكون 'Cairo' مش حاجة تانية