Day 4

*** Aggregate Functions in SQL :**

- هى عبارة عن Built in Functions بتستخدم لعمل العمليات الحسابية على الـ Column Values

- الـ Aggregate Functions عبارة عن 5 Functions :

(COUNT , SUM , AVG , MAX , MIN)

- الـ Aggregate Function مش بتاخد الـ NULL معاها فى الإعتبار

1) COUNT Function :

- تستخدم لحساب عدد الـ Records فى الـ table

2) SUM Function:

- تستخدم لحساب Sum of Records value فى الـ Table

2) AVG Function:

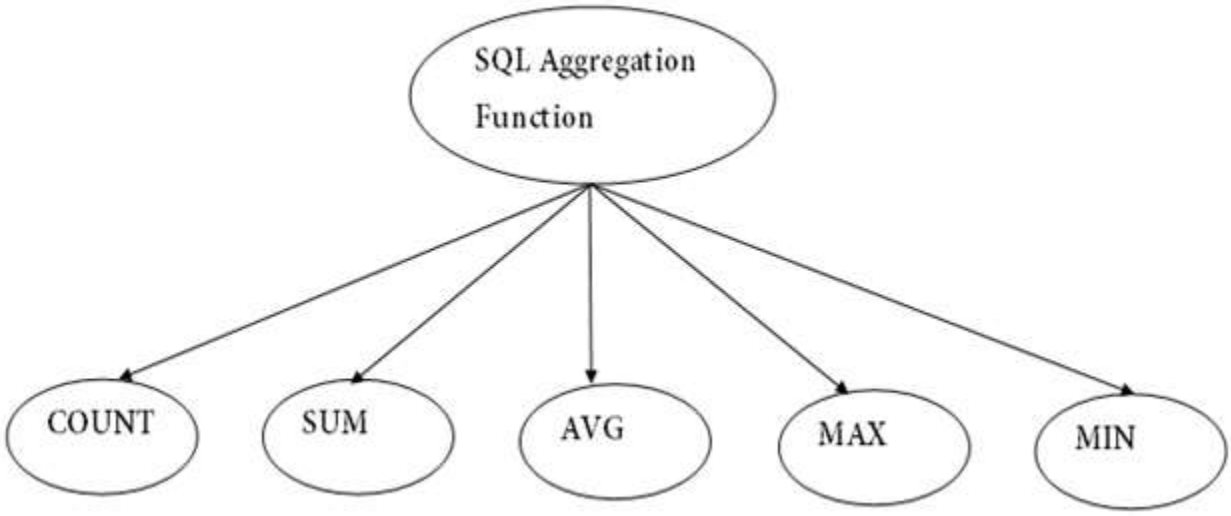
- تستخدم لحساب Average of Records value فى الـ Table

2) MAX Function:

- تستخدم لحساب الـ Maximum Value فى الـ Table

2) MIN Function:

- تستخدم لحساب Minimum value فى الـ Table



- ممكن أستخدم الـ Where Clause مع الـ Aggregate Functions وبتأثر على قيمة الـ Values لأن ممكن بشرط الـ Where هيعمل Except لـ Records اللى بتحقق الشرط وبالتالى هيأثر فى قيمة الناتج

- فى حالة إن إستخدمت Column تانى مع الـ Aggregate Function ببقى فى حالة إن لازم أستخدم الـ Group by لأن كدا أنا بقسم الـ Tables لـ Groups وكل Group بيتم حساب الـ Aggregate المستخدمة له

EX:

```
Select Max(Column 1) , Column 2
From Table
Group by Column 2
```

- هنا هيتم تقسيم الـ Table إلى Groups بناءاً على الـ Column 2 Values (كل Value عبارة عن Group) وبعدها هيتم حساب الـ Max Value لكل Group

- الفكرة فى إلزامية إستخدام الـ Group by هو إن مكنش ينفع أعرض Single value ناتجة من الـ Aggregate Function وأعرض جنبها الـ Column Values فى نفس الـ Query بدون ما أستخدم فكرة الـ grouping

---

- الـ Where Clause بتستخدم عشان أ Except Record's value أنا مش عاوزها معايا فى الـ Query

- فى حالة الـ grouping لو أنا محتاج أ Except Group بستخدم Having Clause لأن مينفعش Where

Where → Row Conditions , using Column)

(Having → Group Conditions , using Aggregate(Column)

- فى حالة الـ Where , Having فى نفس الـ Query الترتيب بيكون كالأتى

(Where → Group by → Having)

بيشيل الـ records' Value اللى بتحقق شرط الـ Where Clause وبعدين باقى الـ table يعمله Grouping على الـ Column الموجود فى الـ group by و بعدها يستخدم الـ Groups اللى محققة شرط الـ Having

*** Subqueries in SQL :**

- فكرتها هى إستخدام Query جوا Query تانى

- بيكون الـ output for Q1 هو الـ Input for Q2

- بنفذ الـ Inner Queries الأول وبعدها Outer Queries

- ممكن يتكتب فى أى مكان بحيث الـ Subquery يخرجلى Value أنا محتاجها فى الـ Query الأساسى

- لو نفس الحاجة تتعمل بـ Join و Subquery بختار الـ Join لأنها أسرع من الـ Subquery وبالتالى يحسن فى الـ Performance (أخر حاجة تفكر فيها هيا الـ Subquery)

- الـ Subqueries ممكن تيجى مع الـ DML عادى

(مثلاً أنا عاوز أمسح Courses من الـ Student Course Table بس للطلبة اللى ساكنين فى القاهرة وبالتالى هستخدم Subquery يجبلى الـ IDs الخاصة بطلبة القاهرة وبعدين أستخدم الـ Ids دى فى إن أحدد الطلبة من الـ Student course Table) " تتعمل بالـ Join برضو "

*** Unions in SQL :**

- فكرة الـ Union هى عرض الـ Tables مع بعض فى نفس الـ Batch

- على عكس الـ Join ، الـ Union ممكن تستخدم فيها tables مفيش بينهم Relationship

```
Select Column_1
From Table_1
Select Column_2
From Table_2
```

- الـ Two queries اللى فوق هيعرض كل query فى batch

- باستخدام الـ Union أنا هقدر أعرض الـ Two Queries فى one batch

- شروط الـ Union : إن يكون عدد الـ Columns واحد مع مراعاة نفس الـ Data Types

- UNION Family :

1) Union all :

- بتعمل Union لـ Two Tables بدون شروط على Duplication

(بيسمح بتكرار الـ Values)

1) Union:

- بتعمل Union لـ Two Tables بشرط الترتيب و عدم وجود الـ Duplication

(مش بيسمح بتكرار الـ Values) (الـ Result أقل ولكن أبطئ من الـ union all)

1) Intersect :

- بيختار الـ values اللى مشتركة بين الـ Two Tables

- بيرتب ويمنع الـ Duplication

1) Except :

- بيختار الـ values اللى موجودة فى Table 1 ومش موجودة فى Table 2

- بيرتب ويمنع الـ Duplication

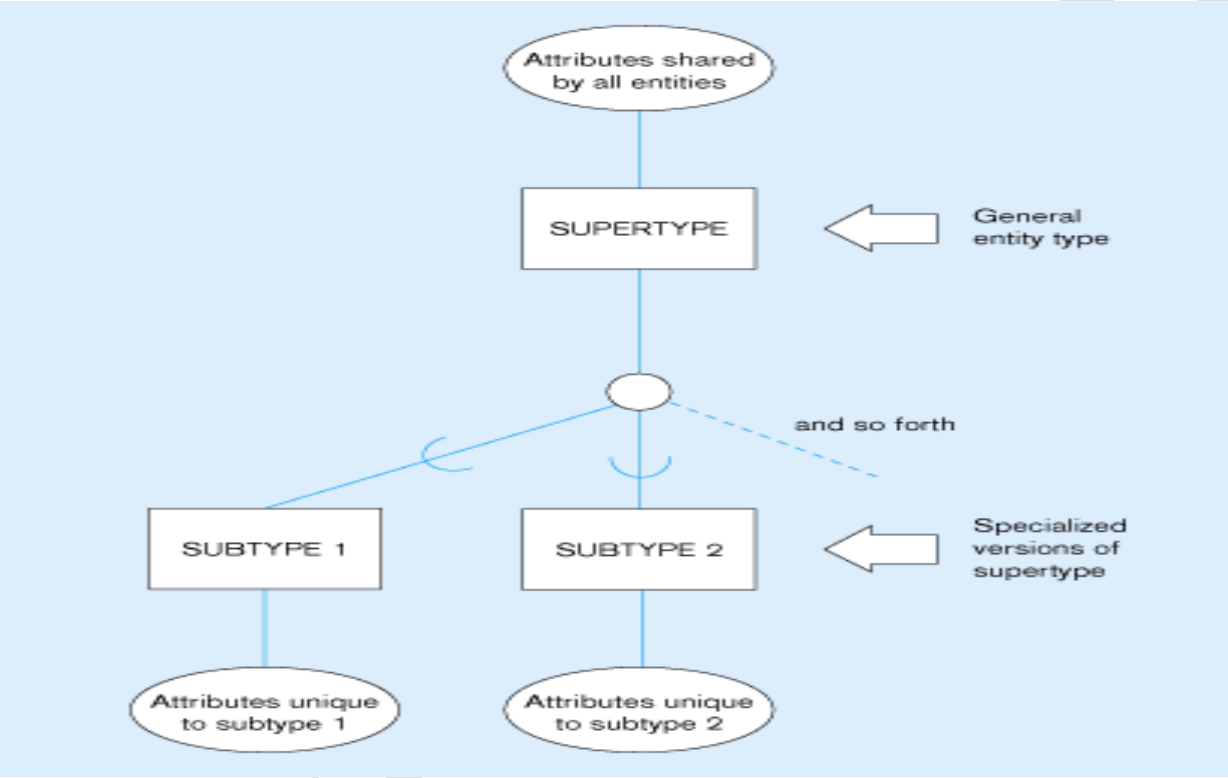
- ممكن أستخدم فكرة الـ UNION مع Multiple Tables بحيث الـ Mechanism هيعمل Union لأول Two Tables والـ Output هيتعمله UNION مع Table 3 ..... وهكذا

*** EERD (Enhanced Entity Relationship Diagram) :**

- هى عبارة عن ERDولكن معه الـ Inheritance

- لو كان عندى Two Entities من نفس النوع وبينهم Column متشابهة و Columns مختلفة ، أقدر أعملهم Inheritance فى الـ EERD

- الـ Basic Notation لـ Inheritance in EERD :



- الـ Inheritance هو أول شئ بعمله وبعد كدا باقى الـ ERD

- الـ Super Entity بتكون من إختراعك (مش موجودة فى الـ Req Doc)

- الـ Subtype هو عبارة عن Special Type of Super

-الـ Sub بيرث كل الـ attributes اللى موجودة فى الـ Super بالإضافة إلى الـ Attributes الخاصة بيه

- لو الـ Relationship موجودة فى كل الـ Subtypes بالتالى ممكن تربطها مع الـ Super

(شرط الكل ، لو مش الكل تبقى موجودة مع كل Subtype مشترك فى العلاقة)

*** Inheritance Constraints :**

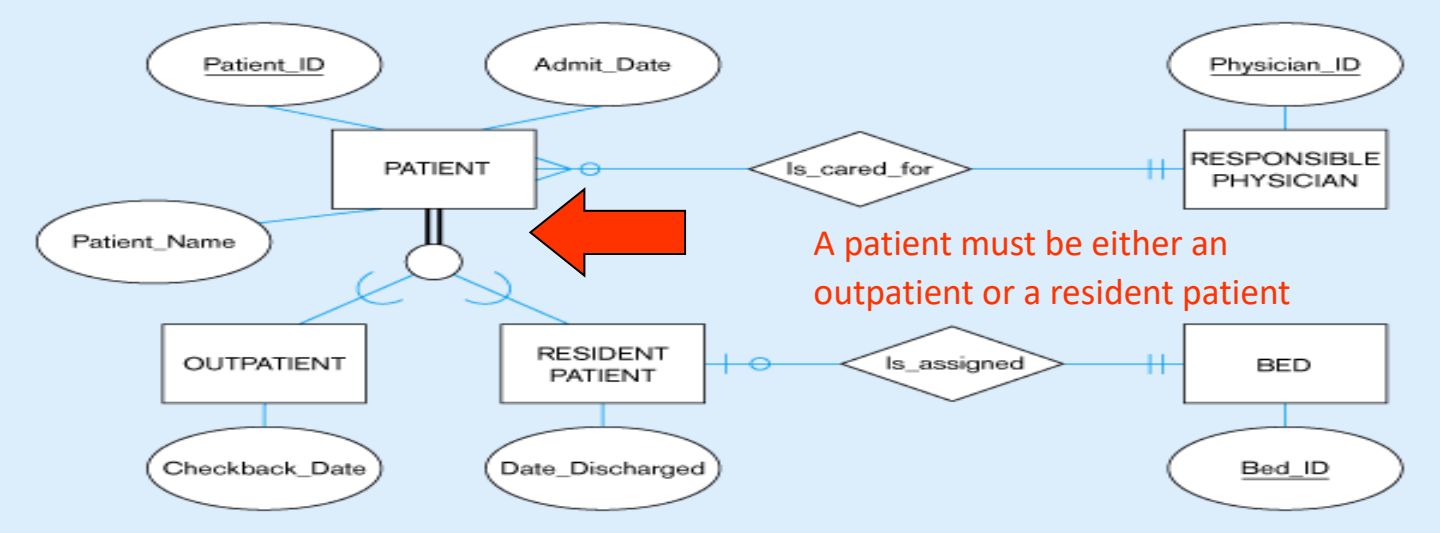
1) Completeness Constraints :

1-1) Total Specialization Rule :

- فى حالة إن الـ Super متكون من أنواع الـ Subtypes الموجودة فى الـ Diagram بس

- الـ Completeness كاملة

- بيتمثل بـ Double lines

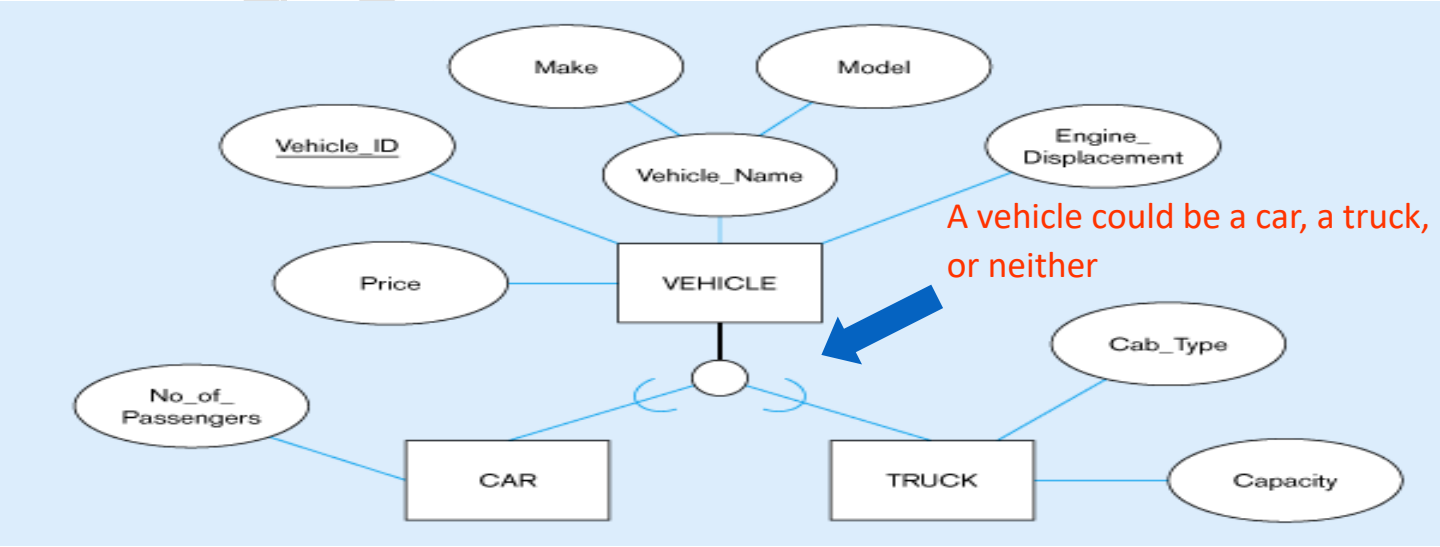


1-2) Partial Specialization Rule:

- فى حالة إن فيه Subtype مش موجود فى الـ Diagram ولكنه مكون لـ Super برضو

- الـ completeness ناقصة Subtype أو أكتر مش موجود فى الـ Diagram

- بيتمثل بـ Single Line



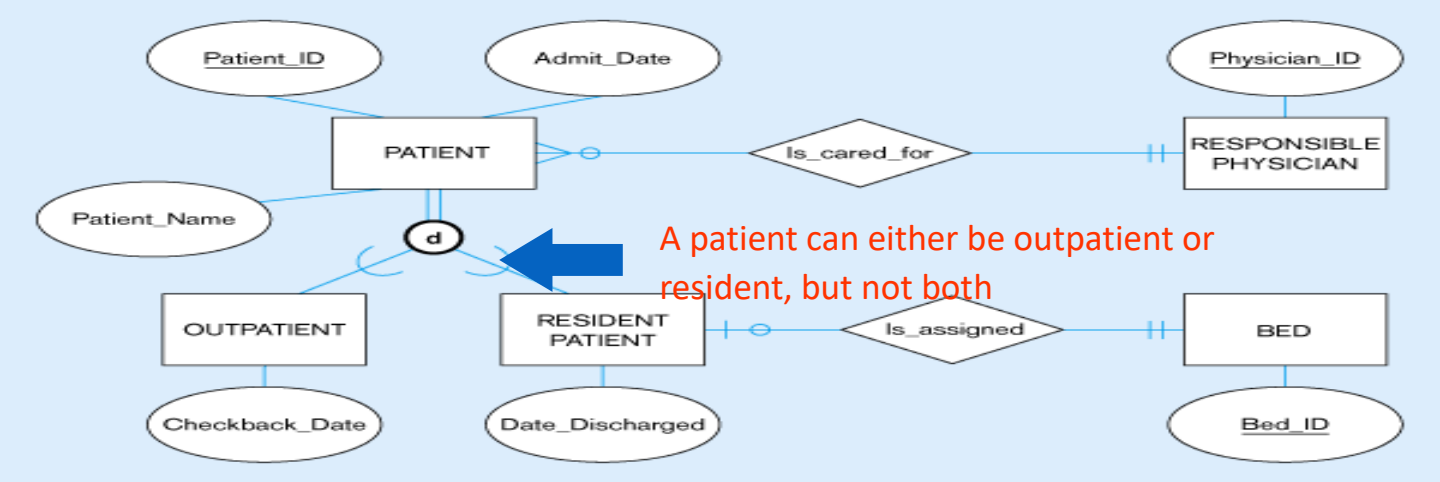
1) Disjointness Constraints :

2-1) Total Disjoint :

- كل Row من الـ Super بيتمثل بنوع واحد من الـ subtype (مفيش Record ممكن يكون موجود فى 2 Subtypes)

- One parent → one Child

- بيتمثل بحرف الـ D

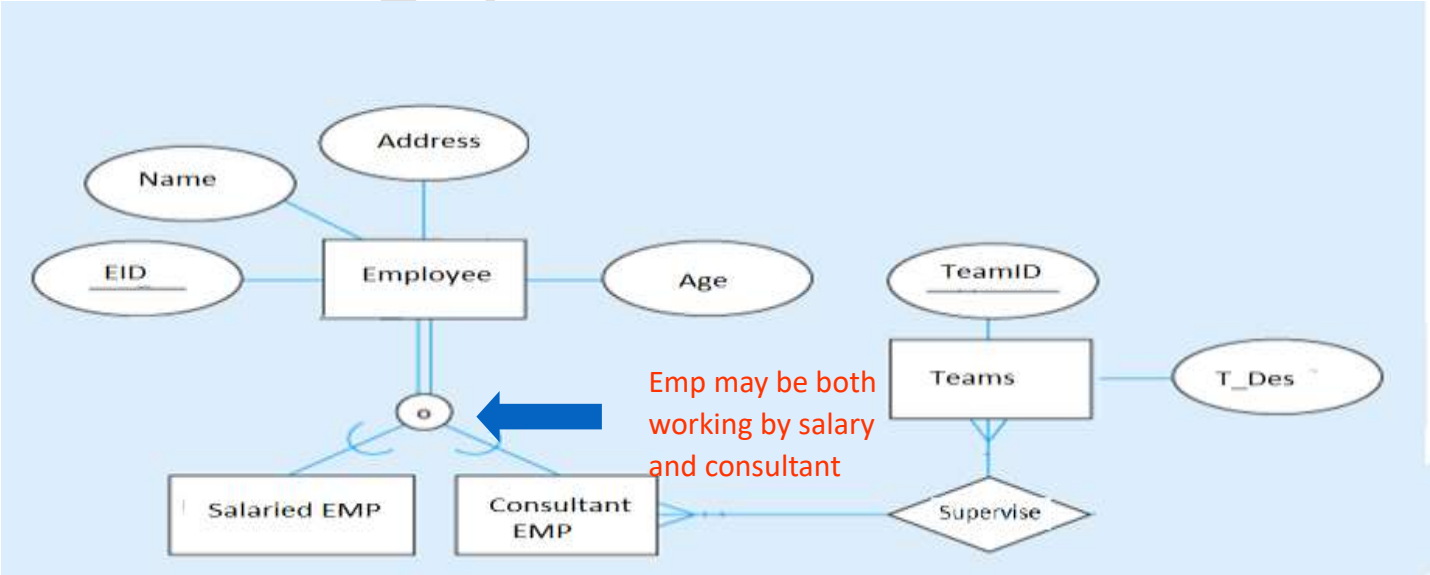


2-2) Overlap Rule :

- الـ Row ممكن يكون موجود فى Subtype 1 و Subtype 2

- One parent → Multiple Childs

- بيتمثل بحرف الـ O



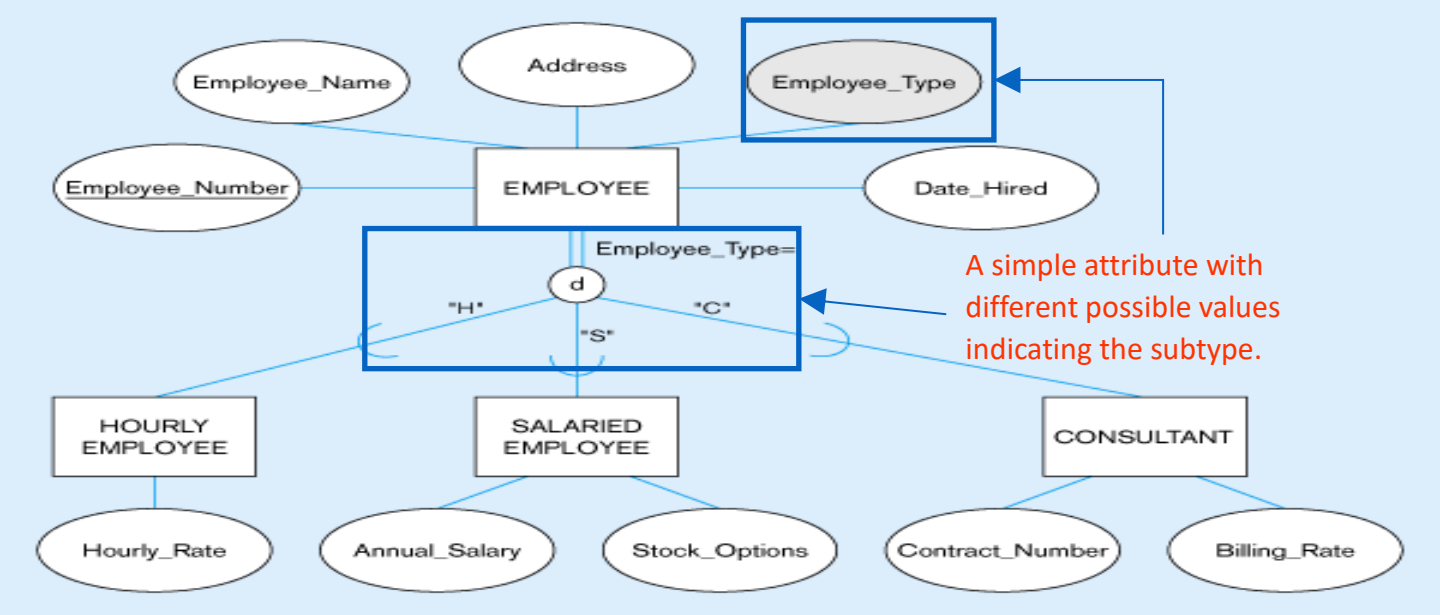
*** Discriminator :**

- فكرتها هى عمل Column بيحدد الـ Type of Sub والـ values أنا بحددها (كل Value تمثل Type)

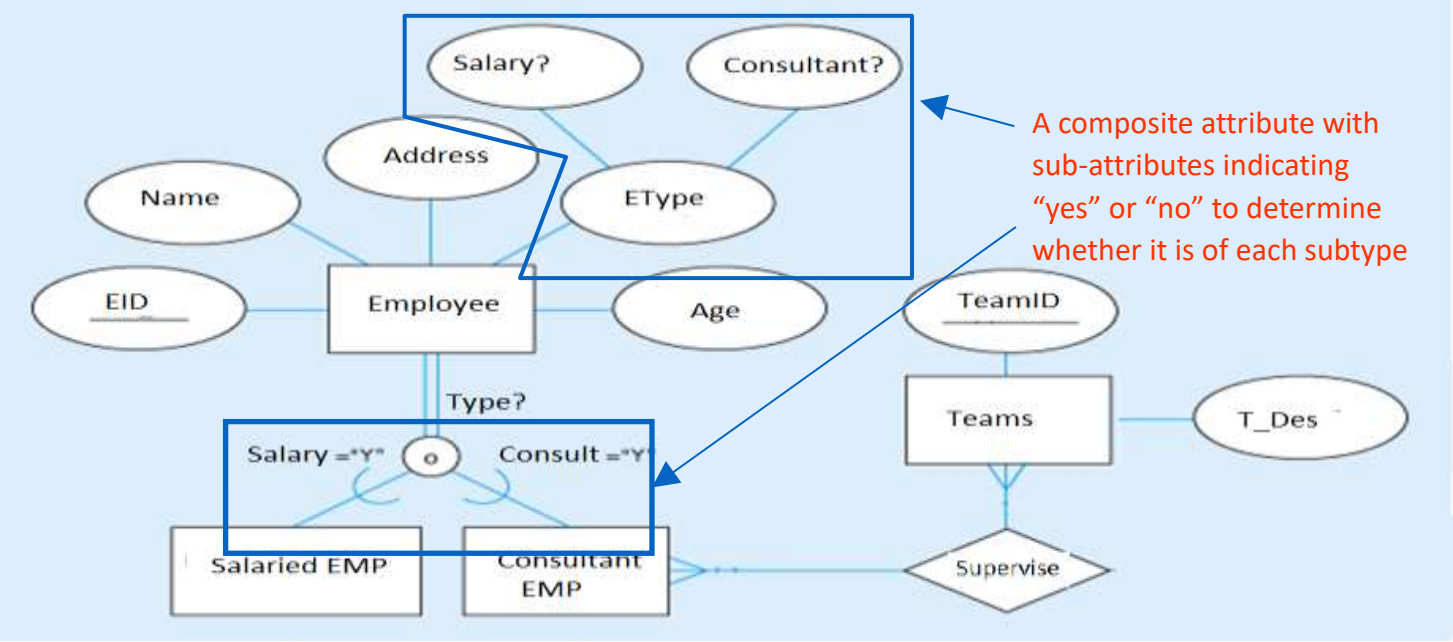
- بديل الـ Joins (كنت هضطر أعمل Join بين الـ Super والـ Sub)

- ممكن الـ Discriminator يكون Composite ودى فى حالة الـ Overlap عشان أقدر أحدد أنهو نوع بالظبط (لأن الـ Record ممكن يكون فى كذا نوع)

Disjoint Rule (Simple Discriminator) :



Overlap Rule (composite Discriminator) :



*** Mapping :**

- بيتم تحويل الـ Inheritance كل Entity بيتحول لـ table

- فى الـ Subs الـ Pk بيبقى عبارة عن primary و Foreign فى نفس الوقت
