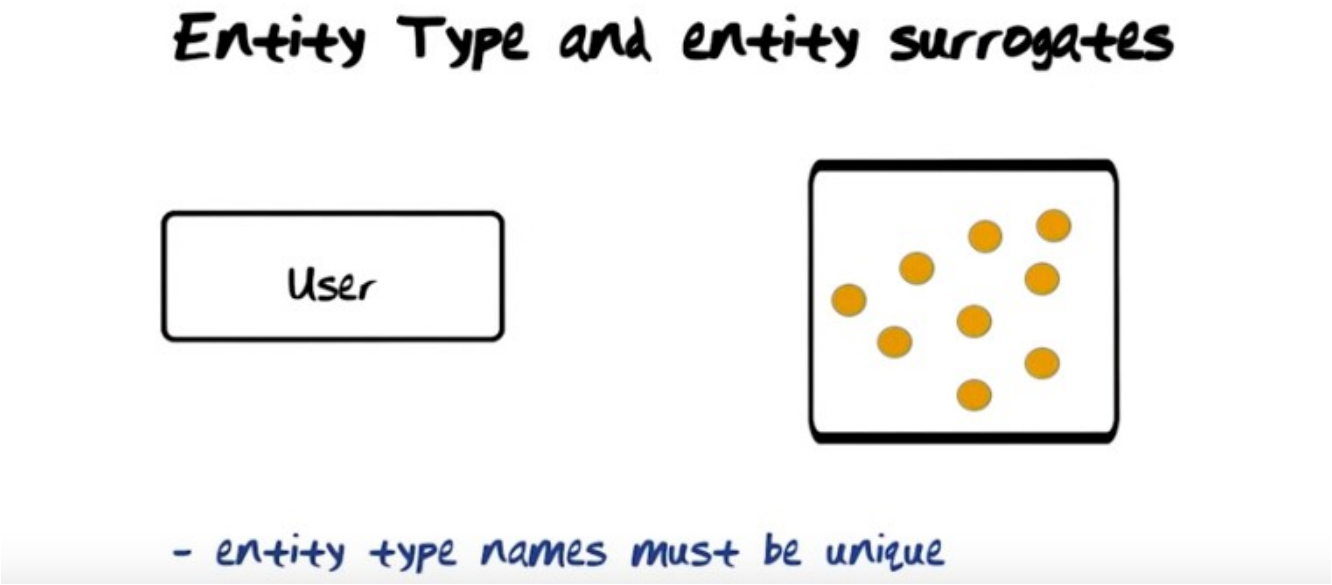
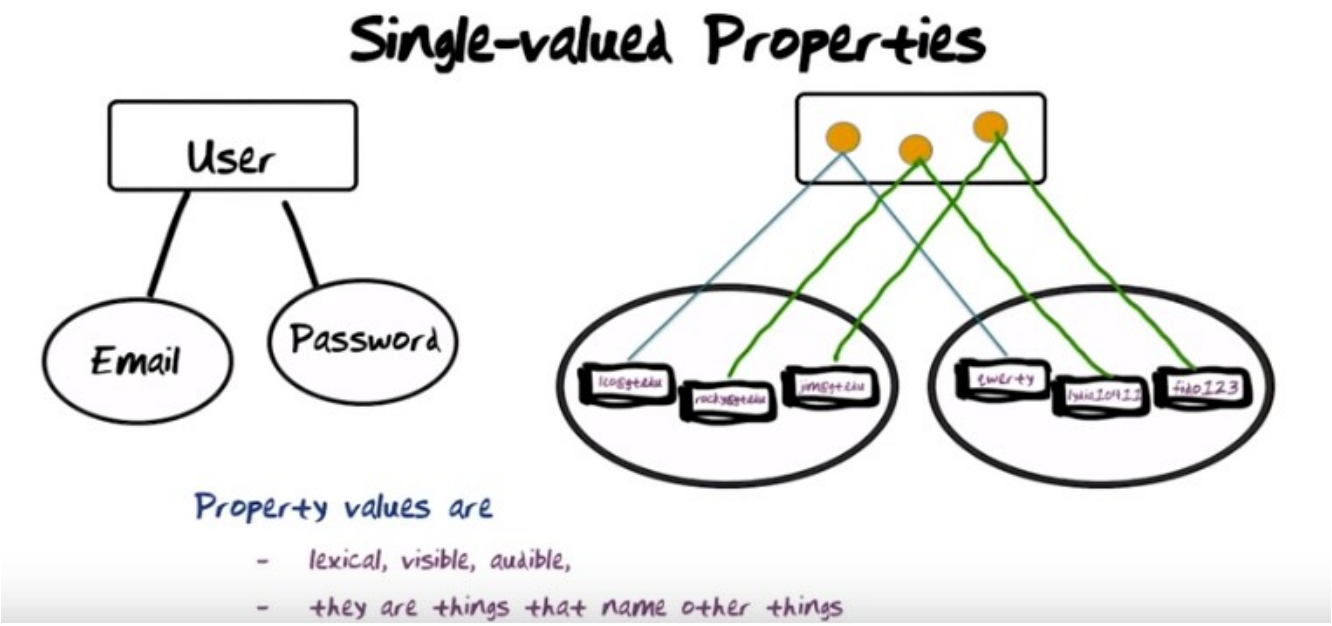
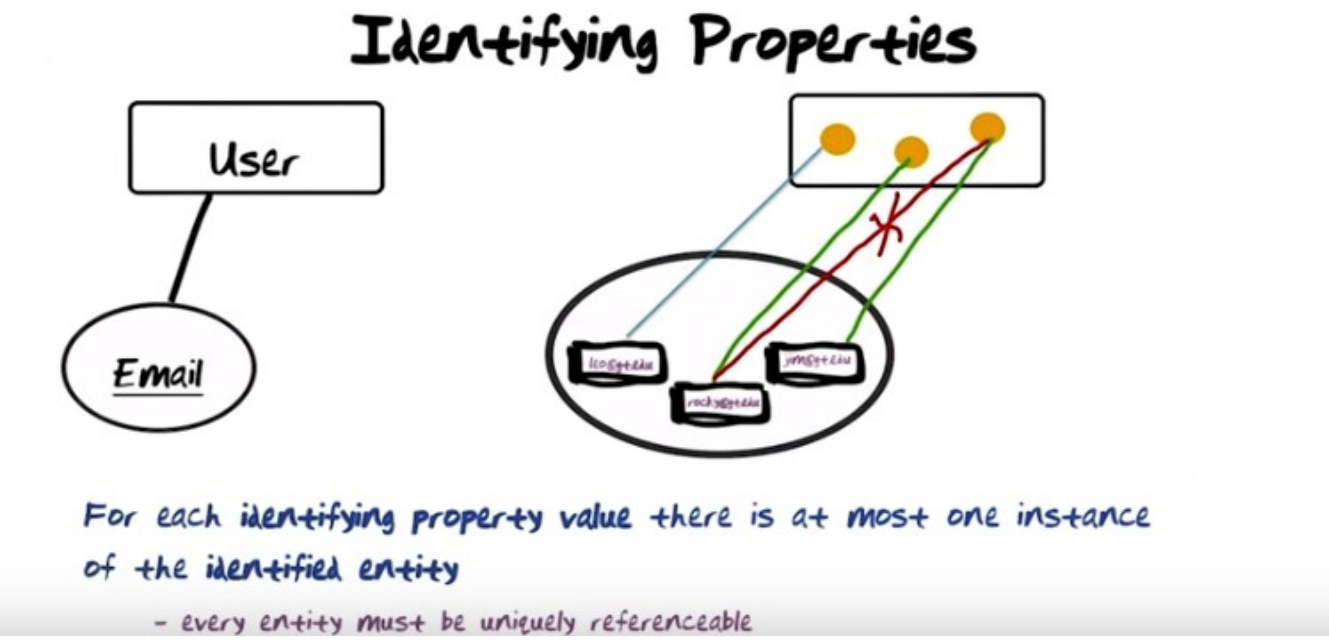
The extended entity-relationship model is on one data model that is particularly good at helping us fix and represent a perception of reality.



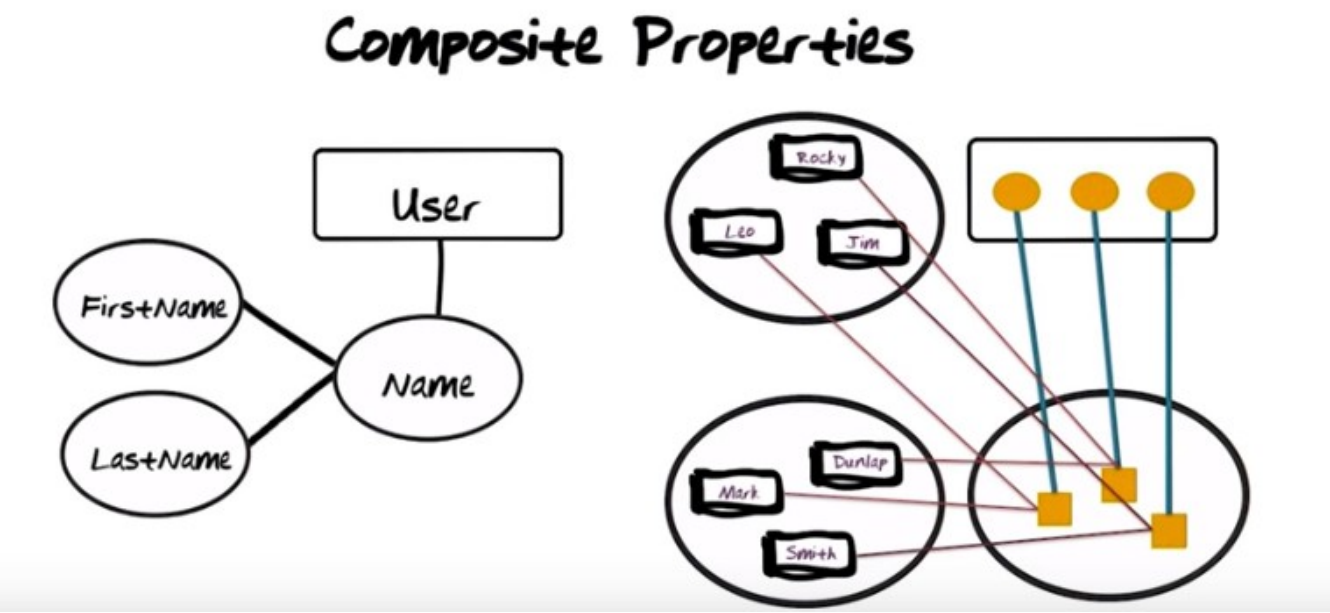
The above figure: The first symbol of the extended entity-relationship model I'm gonna introduce to you is that often entity type (左邊的 User 方框) (重心在 entity 上, 不在 type 上). This is a time in varying representation of a set of users. For each one of the symbols that I'm gonna introduce to you I'm also gonna show you what the semantics (語義) is. This symbol (左邊的 User 方框) is part of the extended entity-relationship model. This (右邊的黃點方框) is not. This is just my attempt at explaning to you what it means. So at any given point in time, there is a set of users that we have captured in our model. Each one of these dots (右邊的黃點) represent such a user. We have previously talked about surrogates. Think about these as surrogates. They are the susbstance inside the system that represent users out in the real world. So here I got 9 users represented. It is important that within one entity relationship diagram all entity type names must be unique, so it is not possible to have multiple entity types, for example with the name User.



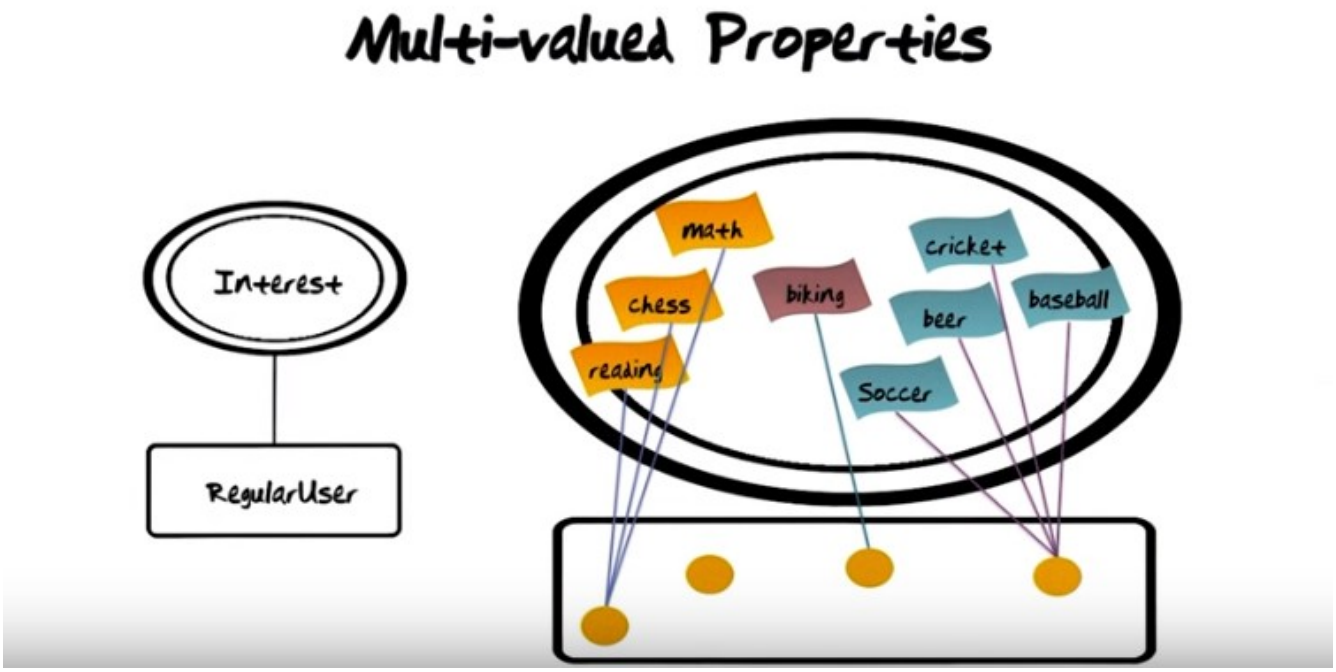
上面左半圖: Property types are represented by ellipses (重心在 property 上, 不在 type 上). These ellipses carry names. So for the User entity type we just introducecd, I have here shown you two single valued properties (看後面的 multi valued propterties 就知道是甚麼意思了). The fact that this is a single line ellipses signifies that it is a single valued property. First property here has the name email, second property here has the name password.
上面右半圖: So let us look at the instance diagram. So here you see there are three instances of User (黃點), and the first user is Leo and that user has the password "quwerty". The second user is ….
注意上面左半圖是一個方框下跟兩個 ellipses, 右半圖也是一個方框下跟兩個 ellipses. 所以右半圖是左半圖的 instance.



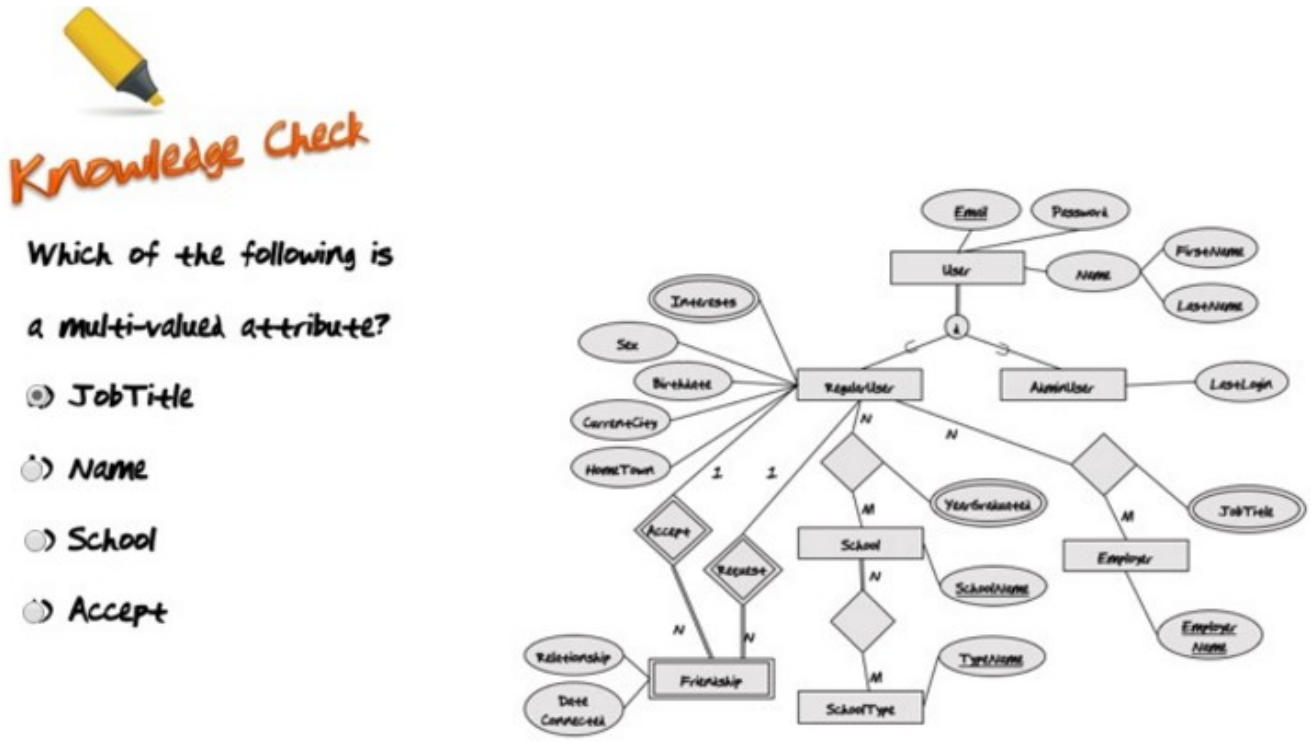
The above figure: Some property types are identifying property types that shown by underling the name of the property type. The constraint that email be identifying property has the following implication. When you point to a particular value (leo@gt.edu), there can be only a single entity instance identified by that email value.

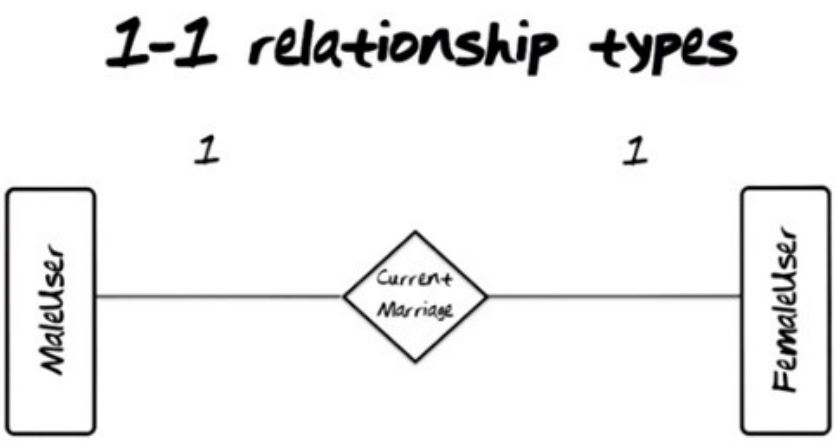


The above pic: A composite property type. Here again we have the entity type User. User has a property of Name. But in this case Name is composed from two properties, namely FirstName and LastName. So let's look at the instance diagram....

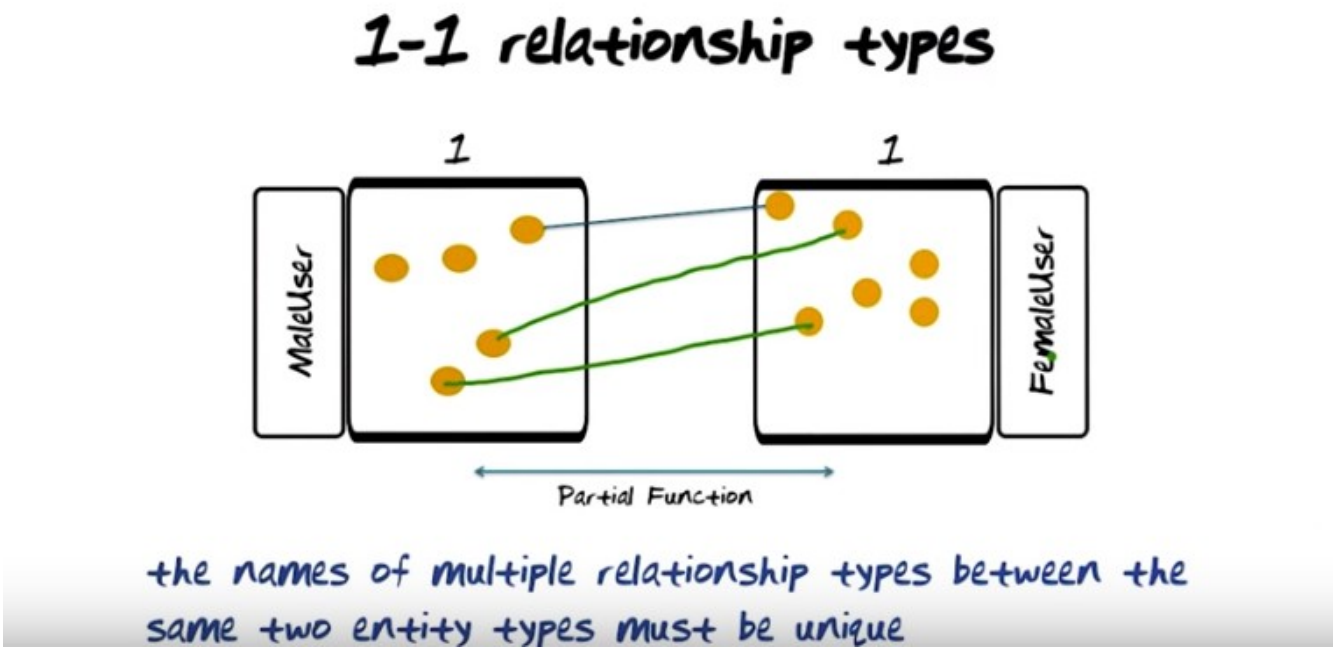


The above pic: multi-valued property types are modeled by the double ellipses. Let's look at the instance diagram. So here you have your regular users. So this one multivalue consists of reading, chess, and math, those are the interests of the first user.

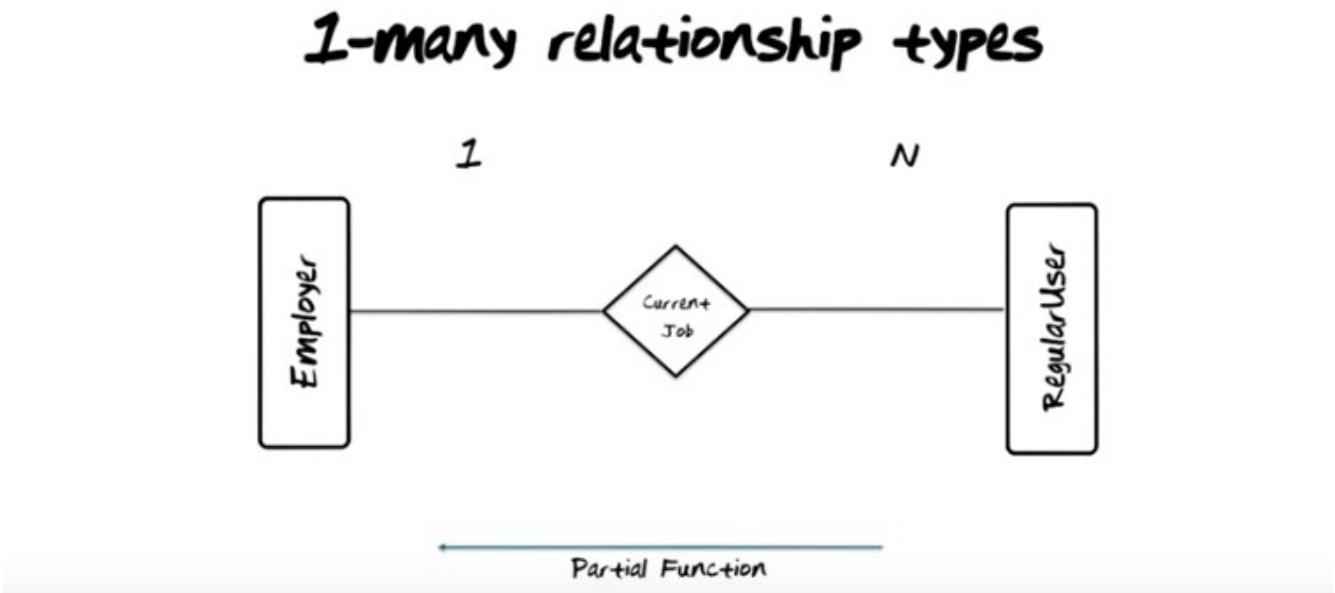


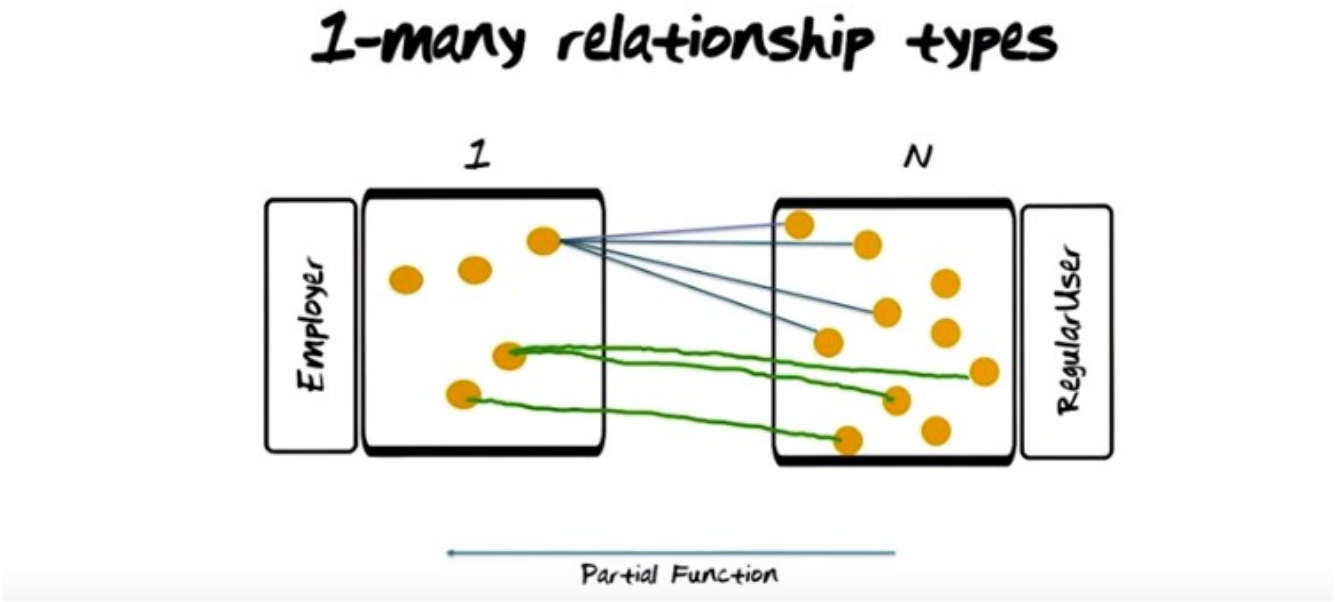


The above pic: Let's now turn our attention to relationship types. Relationship types are represented by diamonds. So here in this example we have two entity types: MaleUser and FemaleUser and they are connected by the relationship type CurrentMarriage. This notation here (那個 1) is the cardinality (from 發音, cardinality: 集合元素的數目) of that relationship. It says that this relationship CurrentMarriage is a one to one relationship type. Let's add the instance diagram to see what that means (下圖).

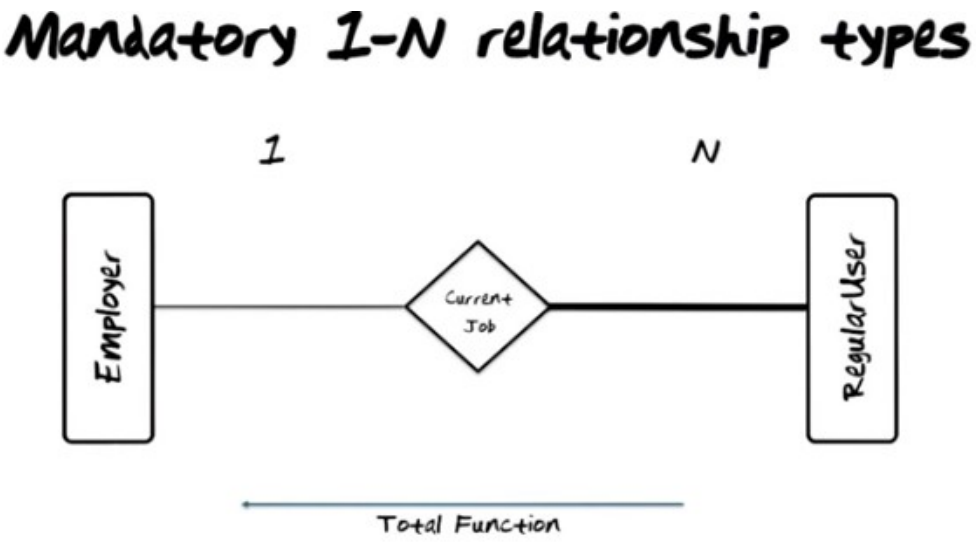


The above pic: Notice that there are couple of male users here that are currently not married, and notice there are three female users here that are currently not married. 所以叫 partial function. If we were to have two relationship types between the same two entity types, then they would clearly have to have different names, so we can distinguish between them.

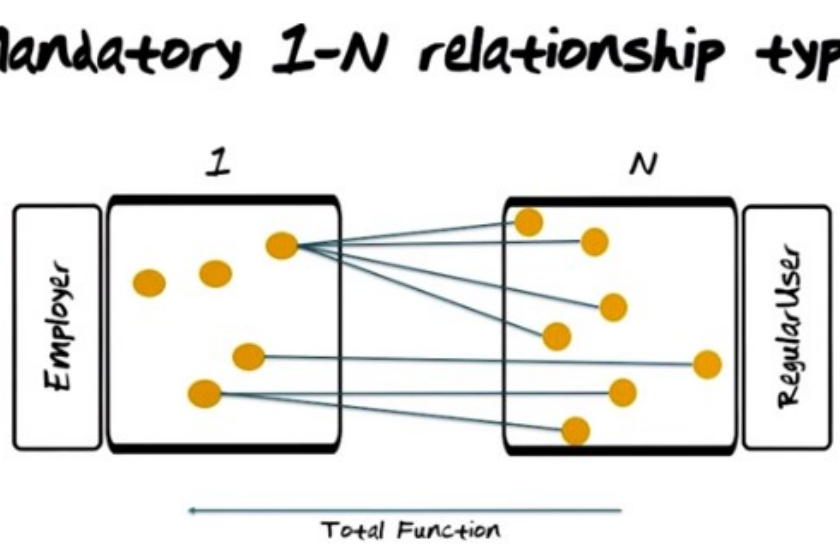




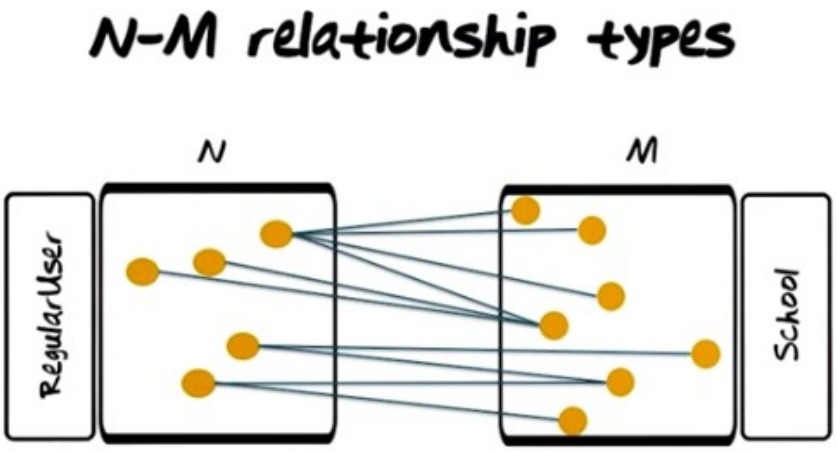
The above pic: 注意沒要求右邊一定是 N 個(即可以是 1:4, 也可以是 1:2, …).
Textbook xournal p245: The cardinality ratio for a binary relationship specifies the maximum number of relationship instances that an entity can participate in. For example, in the WORKS_FOR binary relationship type, DEPARTMENT : EMPLOYEE is of cardinality ratio 1:N (N stands for any number of related entities (zero or more)), meaning that each department can be related to (that is, employs) any number of employees, but an employee can be related to (work for) only one department. This means that for this particular relationship WORKS_FOR , a particular department entity can be related to any number of employees (N indicates there is no maximum number). On the other hand, an employee can be related to a maximum of one department. The possible cardinality ratios for binary relationship types are 1:1, 1:N, N:1, and M:N.

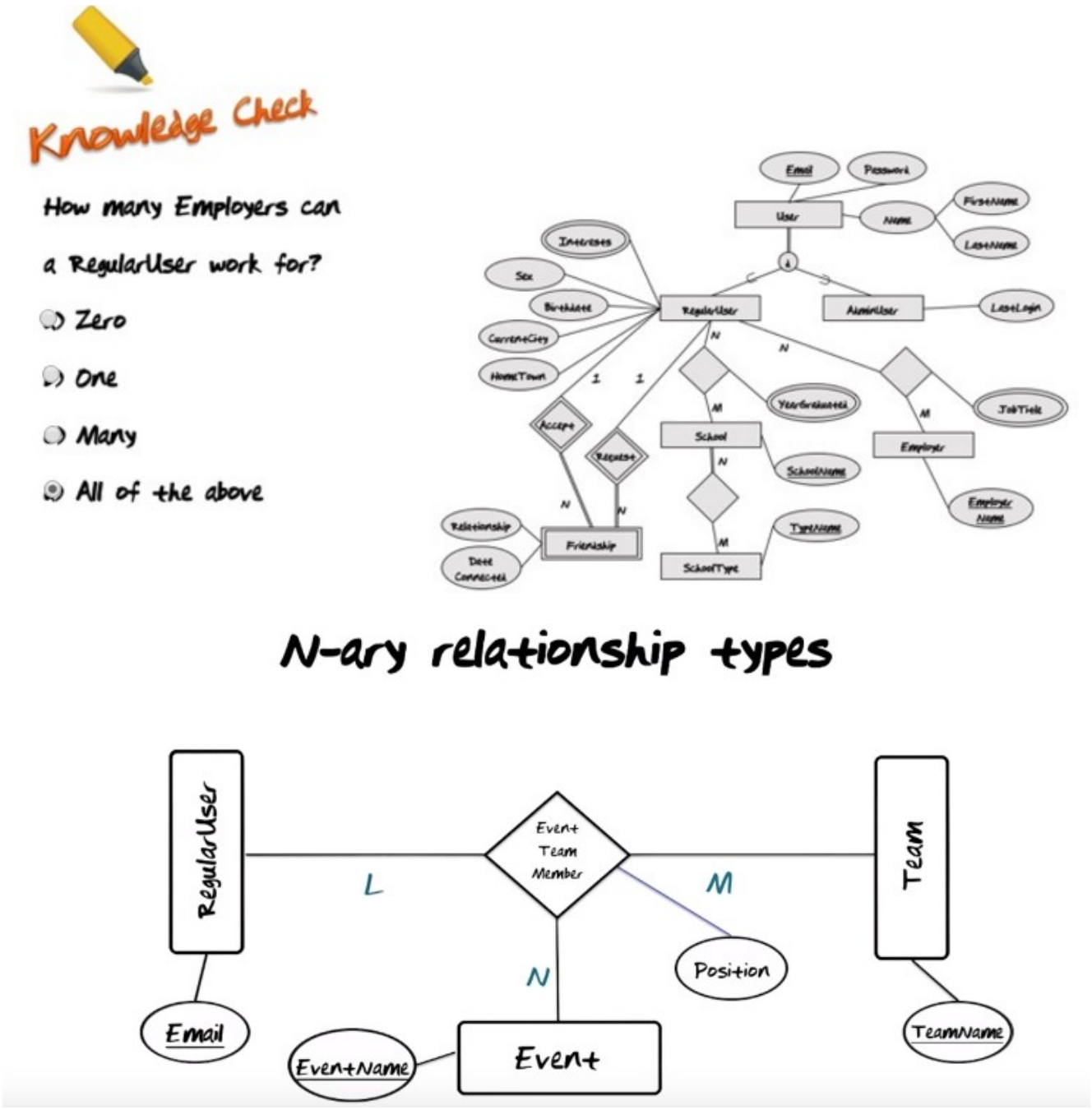


注意上圖中的粗線.



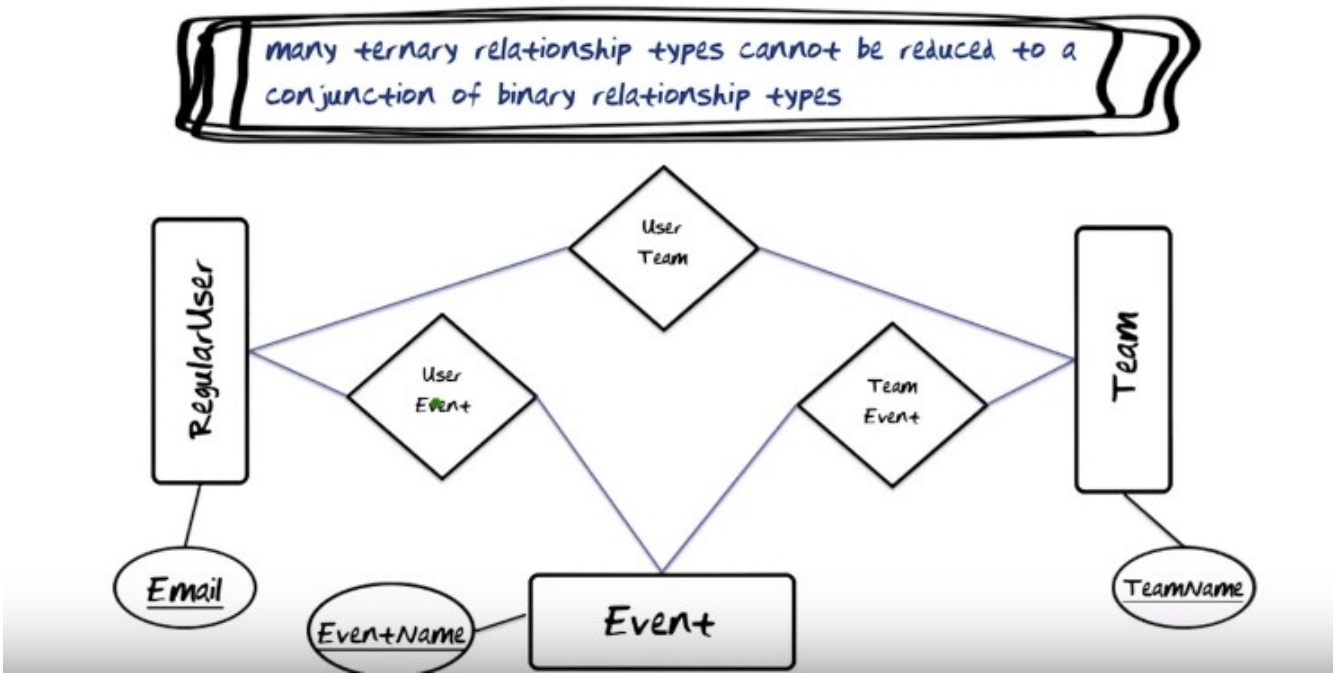
The above pic: All the RegularUsers that are instances of that type that they must be hooke up via that relationship to an employer. 注意從右到左是 total function.



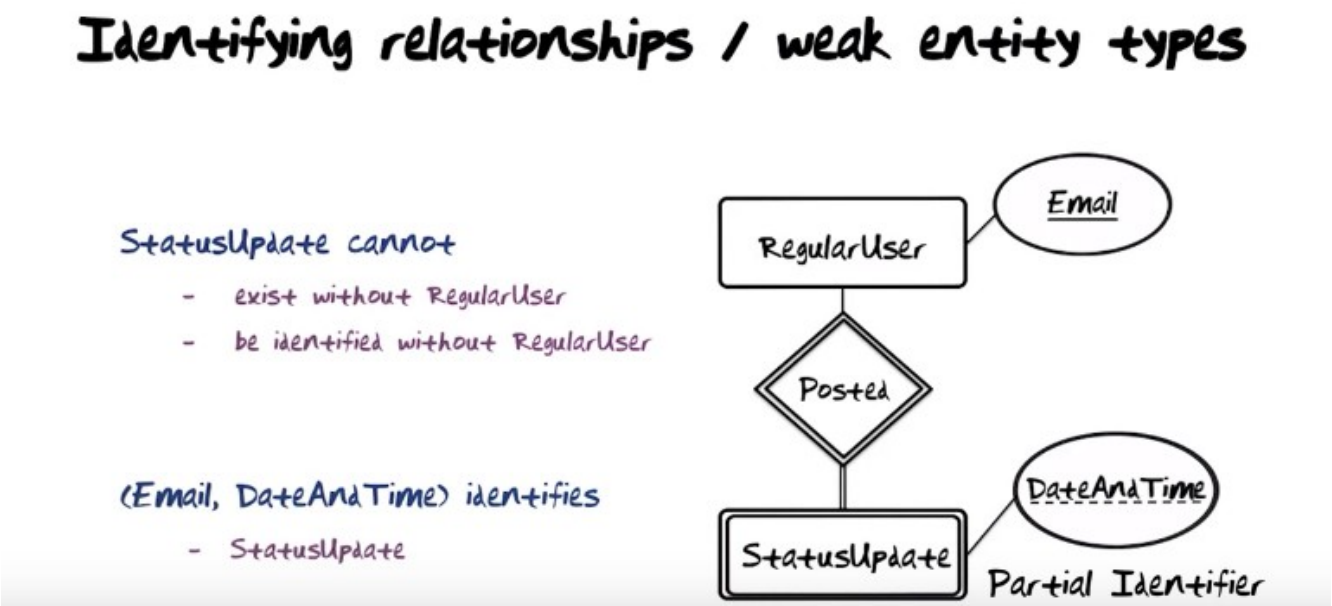


So far all the relationship types we've looked at have been binary. We're now goint to look at an n-ary relationship type within greater than two. In this case here the EventTeamMember relationship type is a n n-ary relationship type. The sematics of this n-ary relationship type is somewhat complicated. Let us fix email to be a single value. Let us fix event name to be a single value. It means that a particular RegularUser in a particular event may participate on many teams. Now let's fix event name and let us fix team name. Again since these are the identifiers (注意 Email, EventName, TeamName 都有下劃線), EventName is going to give us a single Event and TeamName is going to give us a single Team. Holding onto that pair you see that there are many RegularUsers. Finally let's now fix Email and TeamName.... So the meaning of an instance of this relationship type EventTeamMember really is the following: a particular RegularUser on a particular regular Team participate in one particular Event. The implication of that is that you need Email and EventName and TeamName to identify precisely a

single instance of this relationship type. It's quite rare that yo usee examples of relationship types that are degree higher than two. I think one of the reasons for that is that they are difficult to understand, they are difficult to explain. The problem is that it is not always possible to take an n-ary relationship type with n 3 or greater and decompose it into a conjunction of binary relationships. We will look at that next.



The above pic: 意思就是前面那個 n-ary relationship type 的圖 不能轉化成上圖. n-ary relationship type 的圖 表示的是一個 a particular RegularUser on a particular regular Team participate in one particular Event, 意思要比 上圖 narrow. (ternary: 三個的)

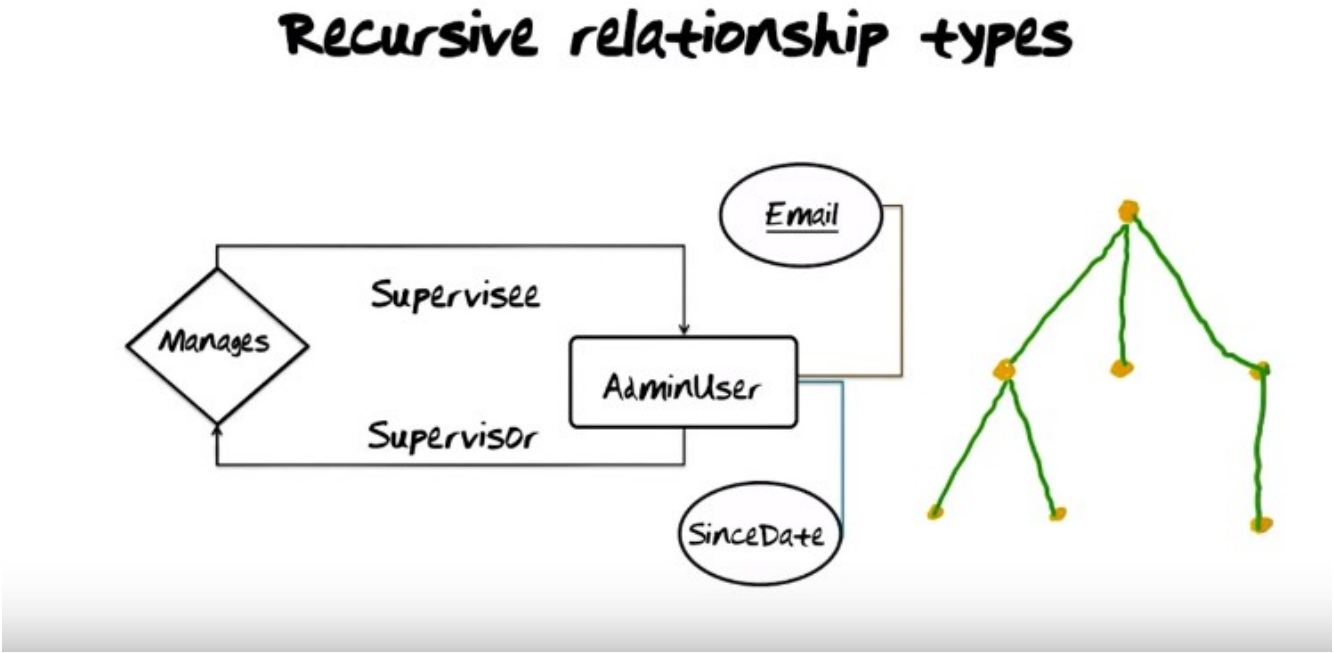


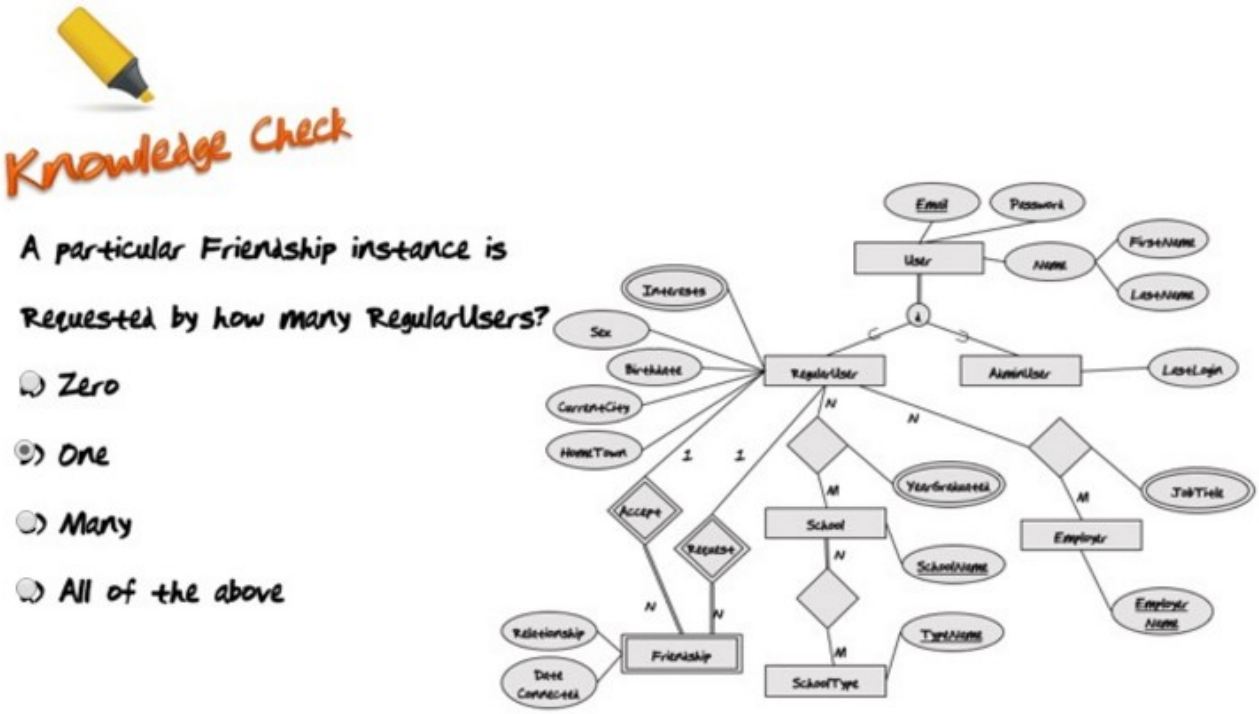
上圖的意思就是: 同一個 RegularUser 可以 post 多個 StatusUpdate, 每一個 StatusUpdate 都有它自己的 DateAndTime (同一個 RegularUser post 的 StatusUpdate 的 DateAndTime 是互不相同的). 而不同的

RegularUser 可以在同一個 DateAndTime post 不同的 StatusUpdate, 所以僅靠 DateAndTime 是定不出到底是哪個 StatusUpdate. 我們需要 Email 和 DateAndTime 兩個信息來定出到底是哪個 StatusUpdate. So StatusUpdate can not exist without being hooked up to a RegularUser. That's why we call that entity type (即 StatusUpdate, 注意不是 DateAndTime) weak. And we say that this is an identifying relationship type because in order to get to a unique StatusUpdate we need to go through the email of RegularUser and combine that with DateAndTime. The fact that this one is not in and by itself a unique identifier illustrated by the dotted line here and this is called a partial identifier.

Textbook xournal p248: In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines. The partial key atribute is underlined with a dashed or dotted line. p152 的表中說: 雙線 entity 表示 weak entity, 雙線 relationship 表示 indentifying relationship. 關於 indentifying relationship, textbook p248 的定義為: we call the relationship type that relates a weak entity type to its owner the identifying relationship of the weak entity type.

Textbook xournal p247: total paritcipation is displayed as a double line connecting the participating entity type to the relationship. 所以連接 relationship 和 entity 的雙線應該就是前面 Mandatory 1-N relationship types 中的粗線, 表示的就是 mandatory 的 relationship. 注意 textbook xournal p248 說了: A weak entity type always has a total participation constraint with respect to its identifying relationship.

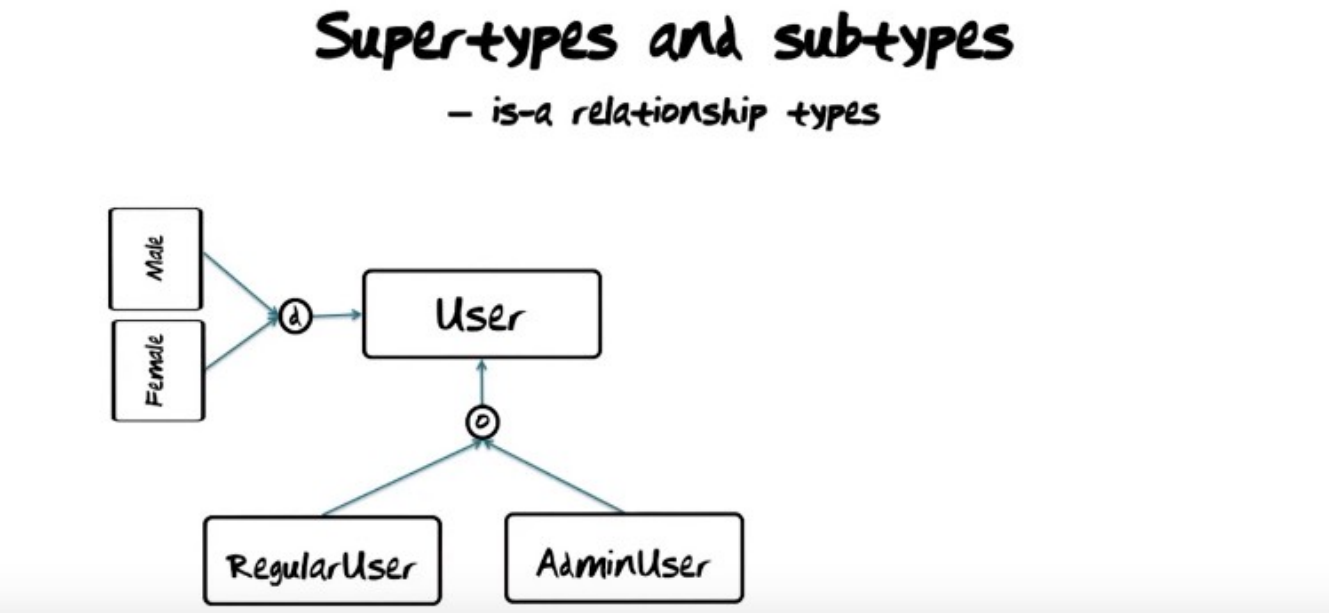




上圖的意思即 AdminUser supervise 它自己, 所以是 recursive. 右邊的樹的每一個節點是一個 AdminUser 的 instance. 一個節點是它子節點的 supervisor, 是它父節點的 supervisee.

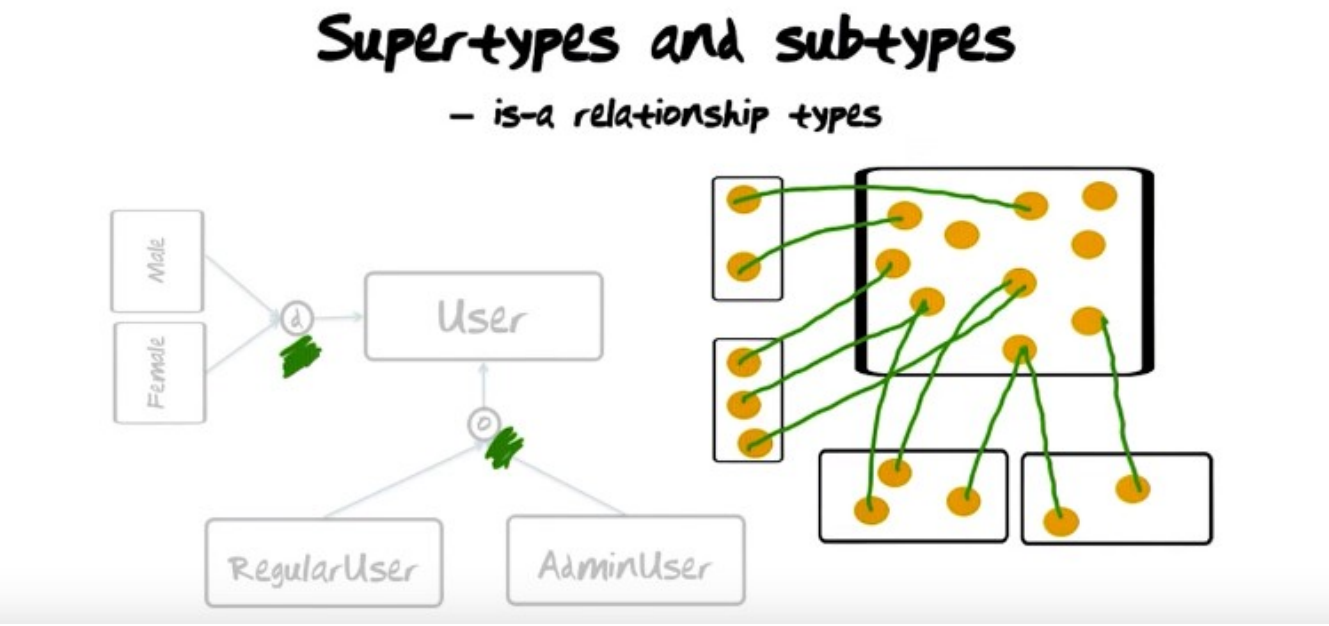
另外, 上圖(看下圖更清楚)中的 RegularUser 和 School 之間的 relationship 還帶了一個 property, 即 YearGraduated. 它表示的意思見我的 note 6. Methodology III DESIGN, 這裡為了方便, 將其 copy 過來了(還將那裡的圖也 copy 過來了).

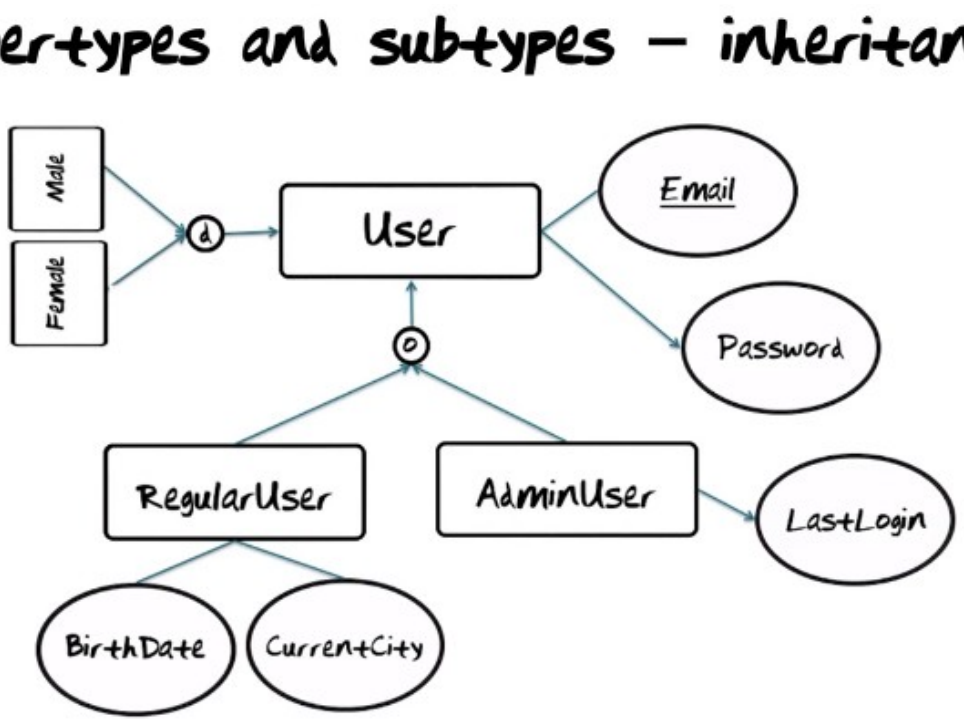
What is a little bit different about this mapping is that for each one of these instances, there is a YearGraudated associated with it. One reason for that could be that you might attend, for example, Georgia Tech twice, once for your bachelors and once for your masters. In each one of those, there will of course be a unique YearGraduated (這就是 EER 圖中 relationship 帶 property(應該都是 multi-valued property)時的意思).



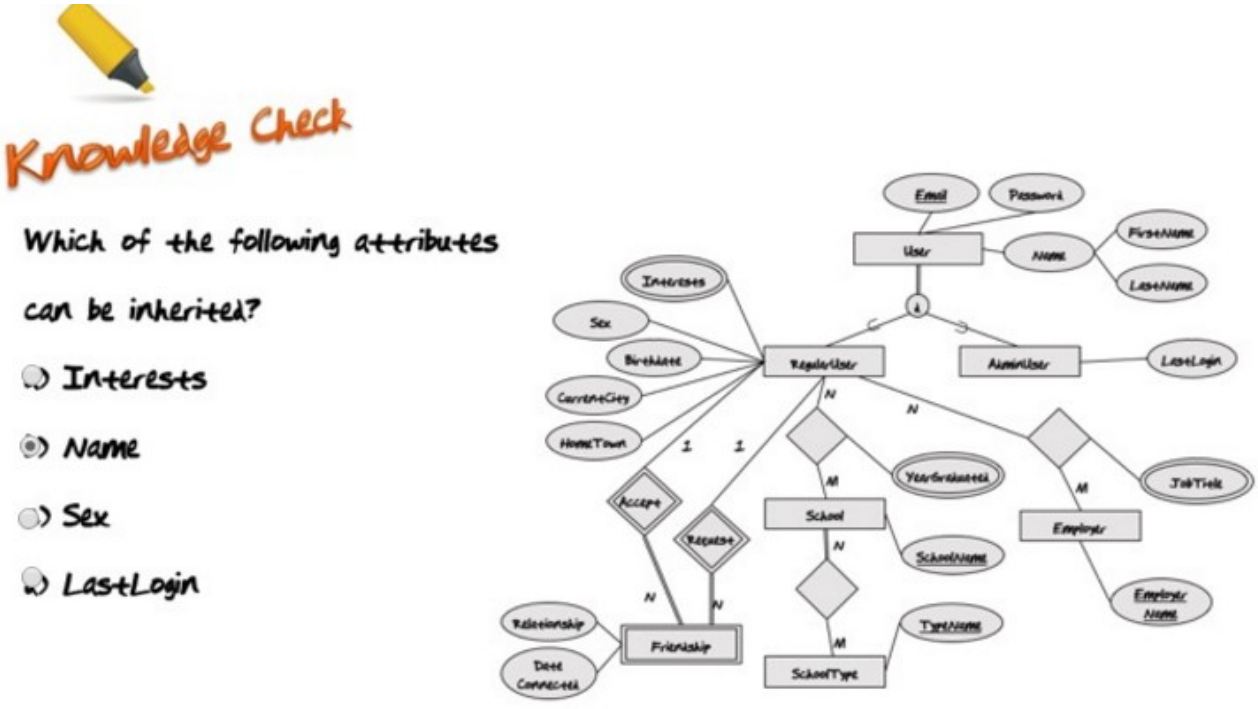
上圖中的 RegularUser 也是 User, AdminUser 也是 User, Male 也是 User, Female 也是 User. d 表示一個 User 不能同時為 Male 和 Female (textbook 說 d 表示 disjoint), o 表示一個 User 可以同時為 RegularUser 和 AdminUser (textbook 說 o 表示 overlapping), 見下圖.

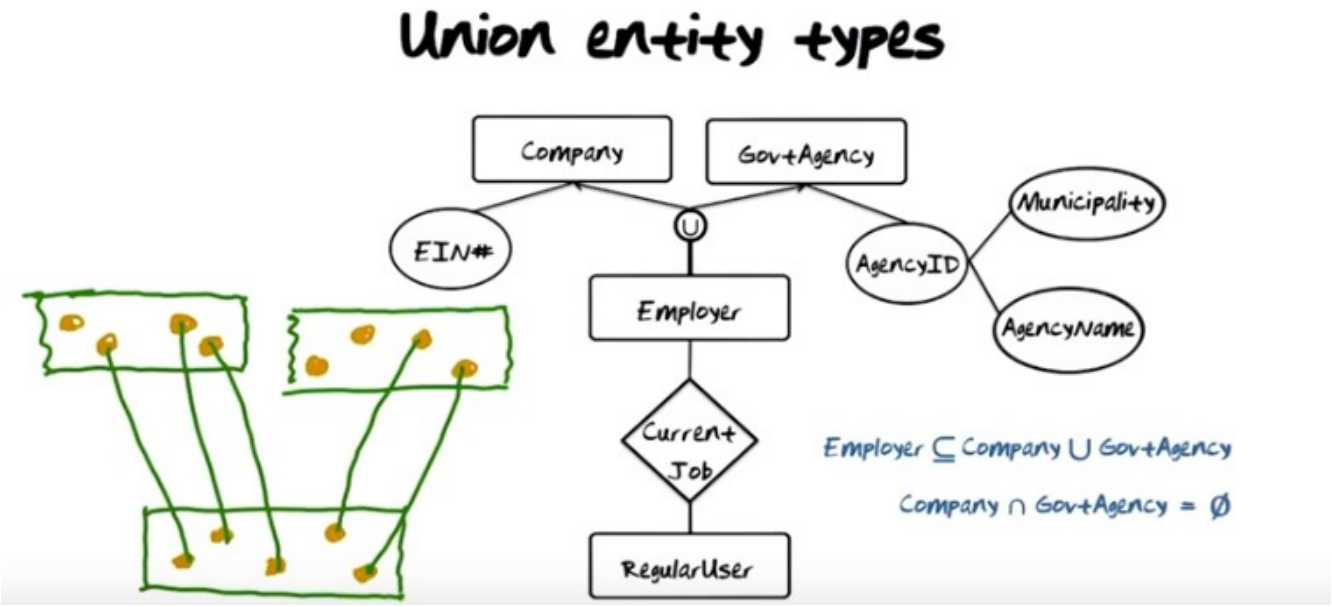
Textbook xournal p282: A total specialization constraint specifies that every entity in the superclass must be a member of at least one subclass in the specilization. This is shown in EER diagrams by using a double line (或在本 note 中為粗線) to connect the superclass to the circle. A single line is used to display a partial specilization, which allows an entity not to belong to any of the subclasses.





上圖的意思是, RegularUser, AdminUser, Male, Female 都是 User 的 subtype, 它們從 User 中 inherite Email 和 Password. 注意 d 和 o 就是 supertype-subtype 的標志, 下面的 quiz 中就會用到.



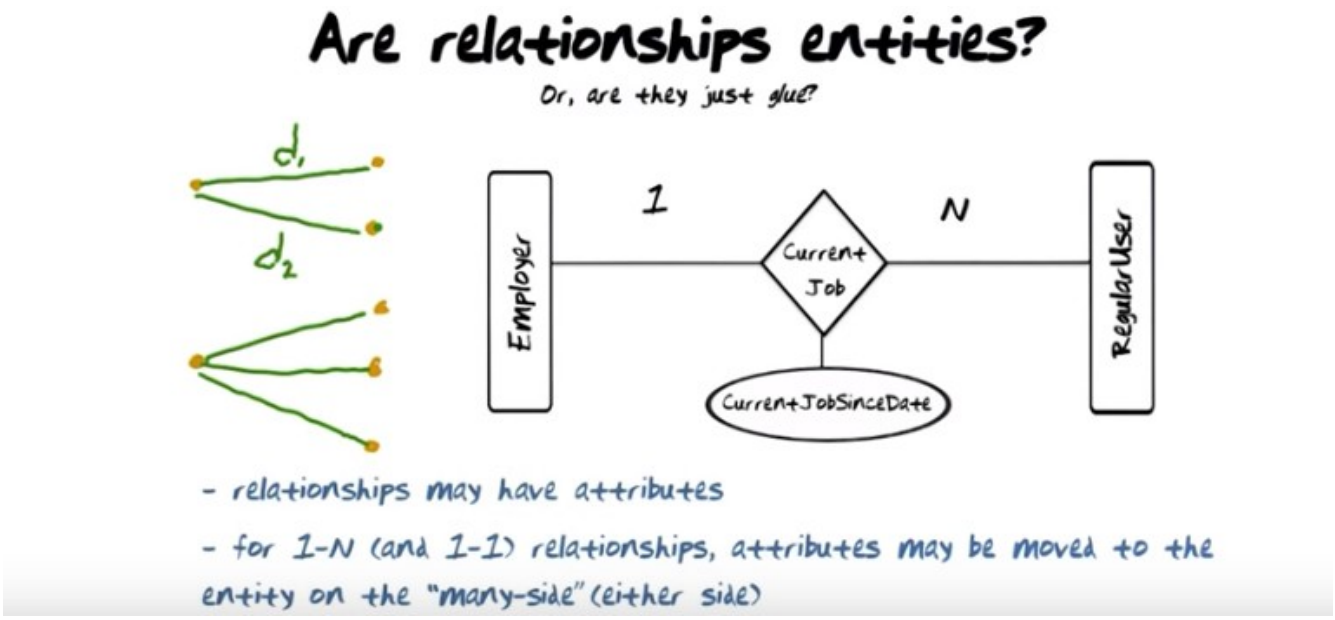


The above figure: The point I want to make here is that the employer may be one of two different kinds of entities. It could either be a company which is the employer, or it could be a government agency which is the employer. 圖中的 U 就是並集的意思. 注意 Company 和 GovtAgency 沒有交集.

Thing?
Relationship?
Property?

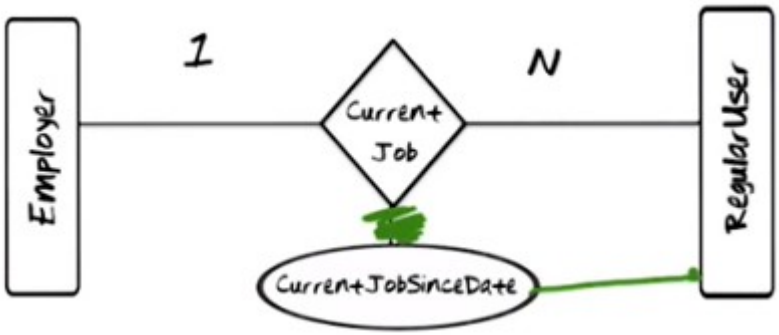
Is the EER Model supporting the fundamental types of abstraction?

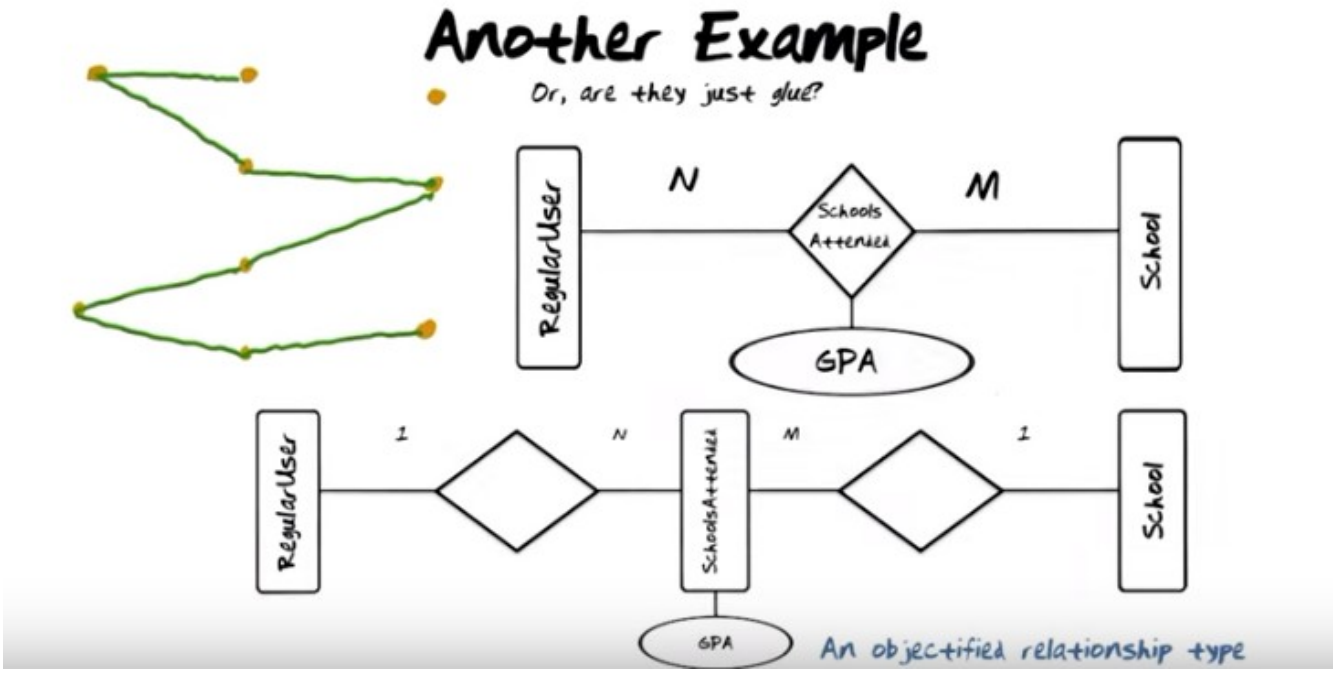
What would the type of a query on the EER Model be?



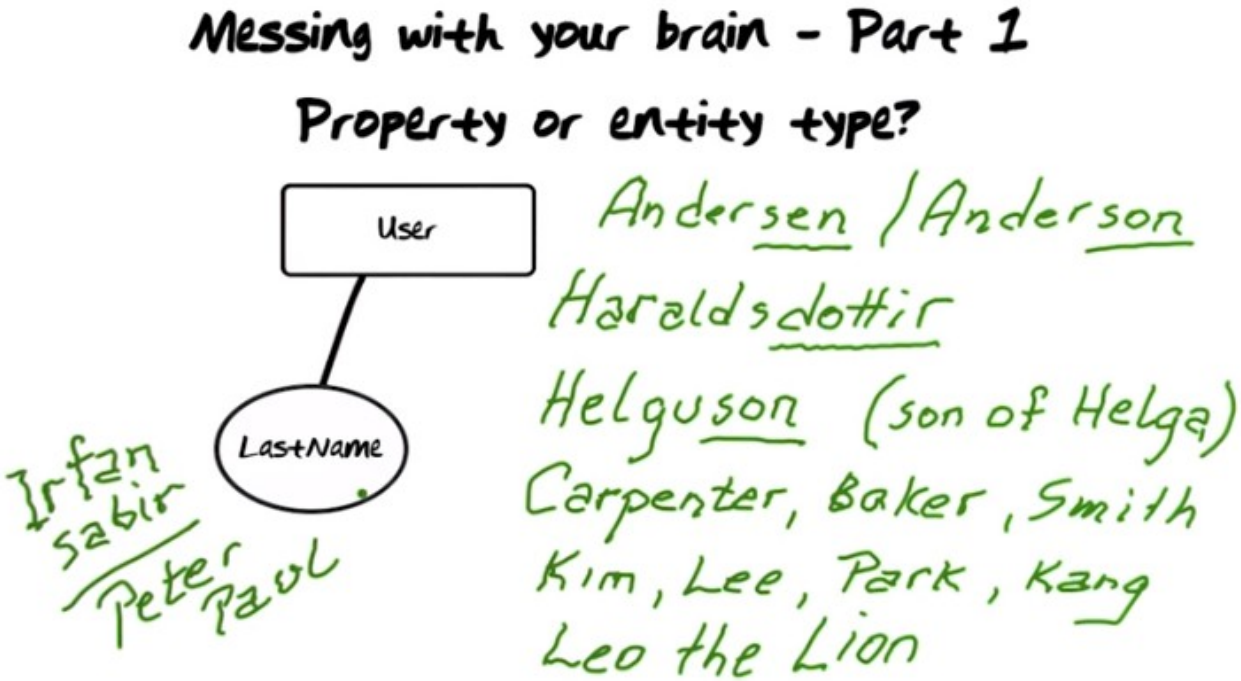
上圖中，老師並沒回答 Are relationships entities 這個問題．左邊的黃點為 Employer, 右邊的黃點為 RegularUser, d1 和 d2 即 CurrentJobSinceDate 的值, solution 即將 d1 和 d2 移到對應的 RegularUser 中去, 即下圖那樣:

.



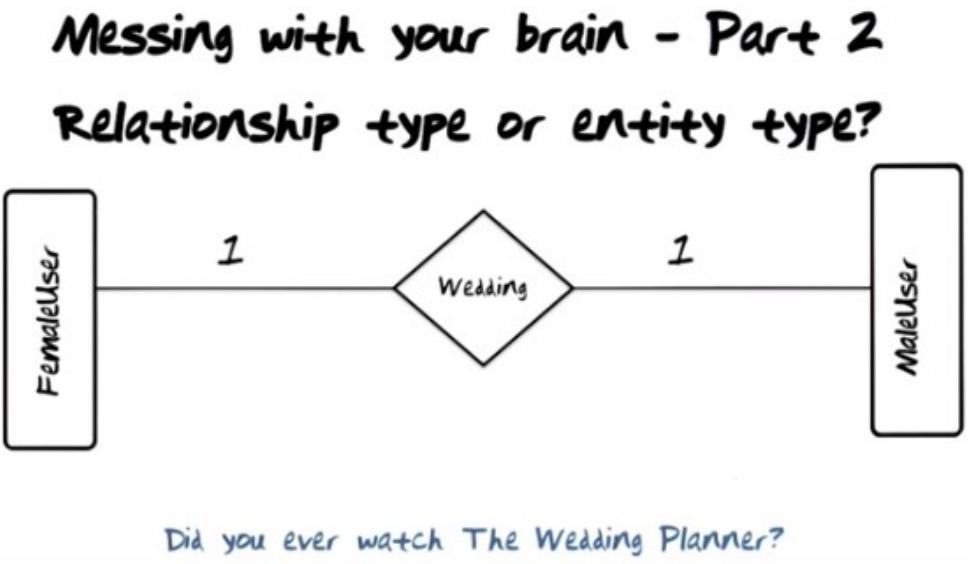


The above figure: what needs to happen is the SchoolsAttended need to be an entity type as opposed to a relationship type. We call that an objectified relationship type. 所以就弄成下半圖那樣了. 上半圖中, 要求的是 a many to many relationship type between RegularUser and School. 下半圖也實現了這麼個要求, 可從左邊的黃點中看出, 黃點圖即為下半圖的 instance, 黃點共三列, 左列為 RegurlarUser, 中列為 SchoolsAttended, 右列為 School. 可以看出, 一個 RegurlarUser 黃點可以 map 到多個 School, 一個 School 也可以 map 到多個 RegularUser.

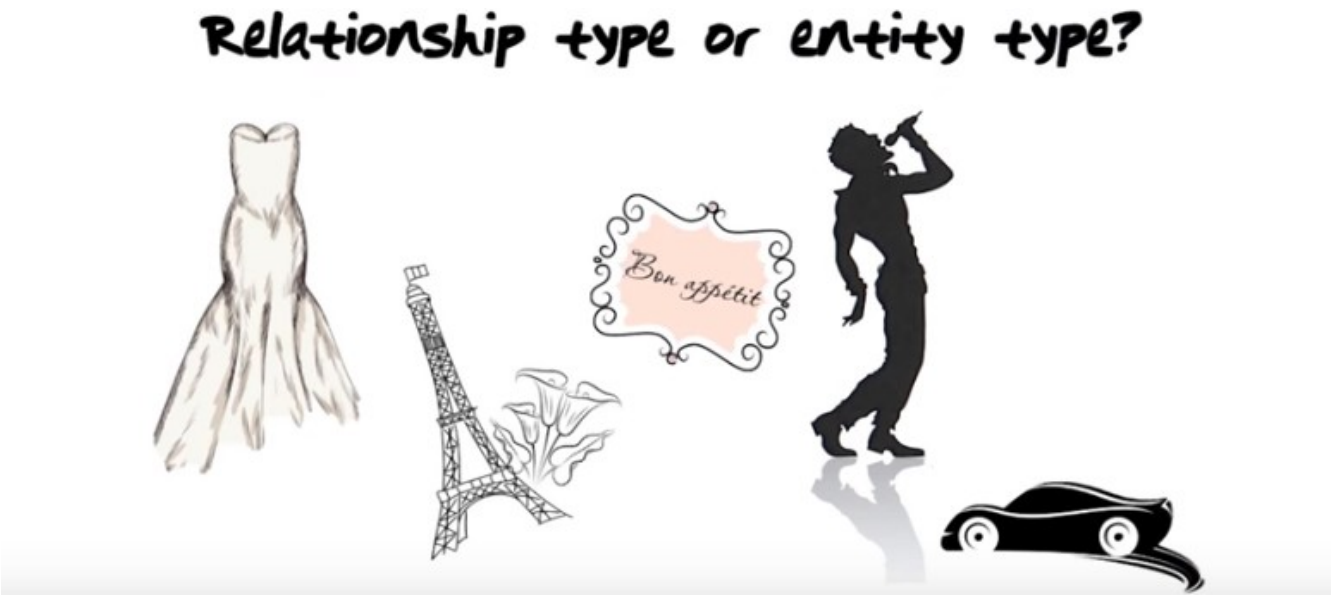


The above figure: Well most of you probably guessed that User is an entity type and LastName is property type. 那些名字的例子, 分別表明這些名字來自父係命名, 職業, 韓國, 宗教等. So do you still think

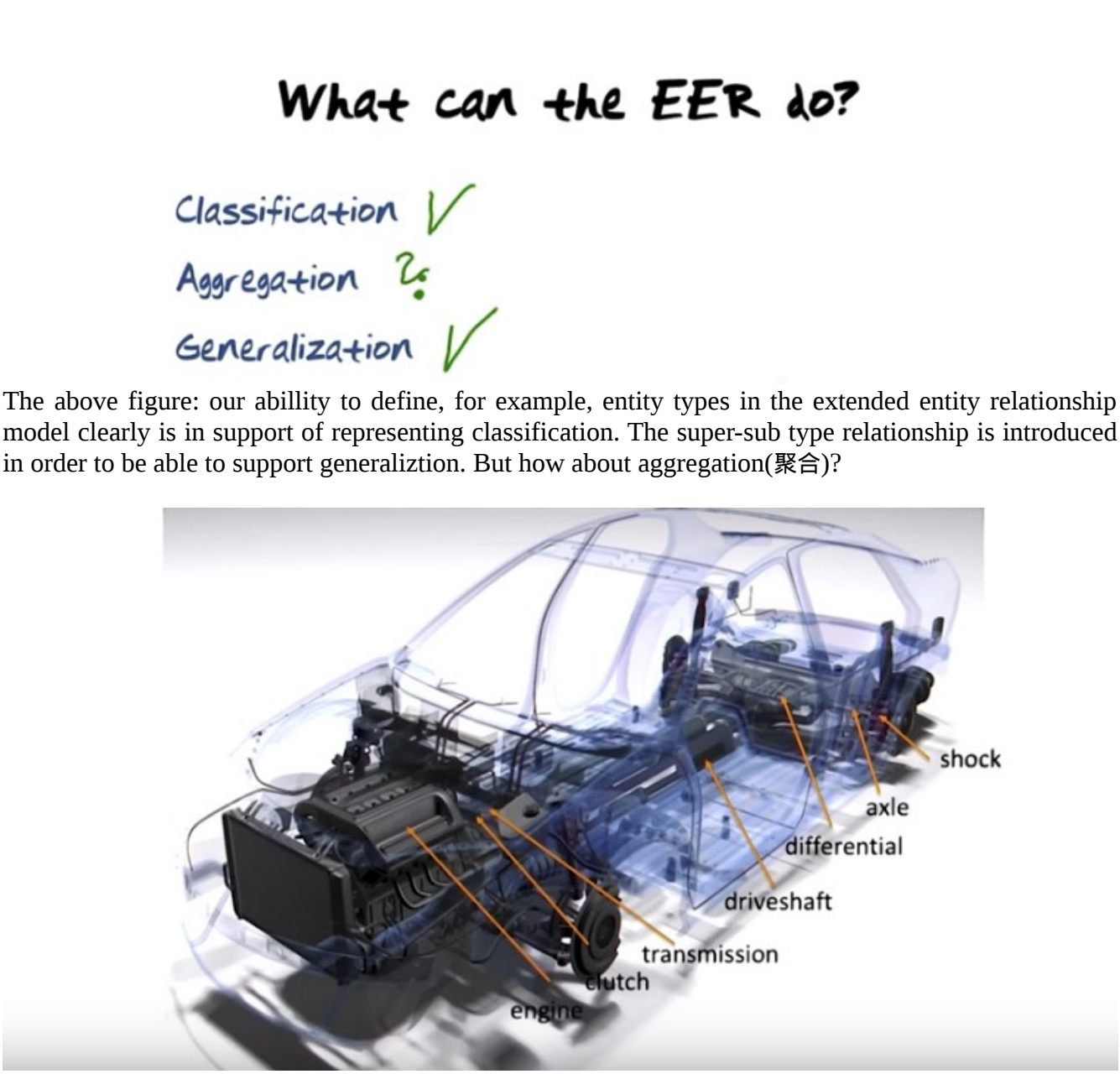
that LastName is a property, or do you think that that's the central entity type in a database dealing with genealogy(系譜,家系)?



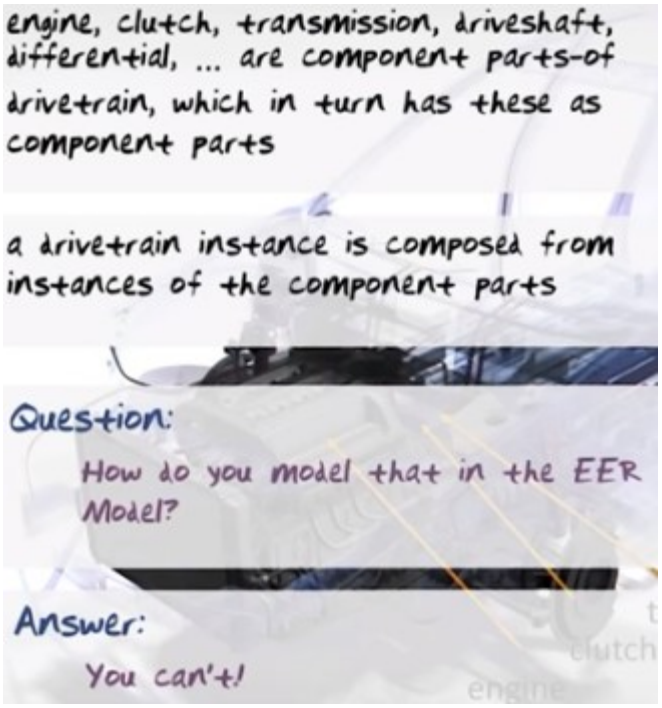
上圖的 Did you ever watch The Wedding Planner 意思是說有的情況下, Wedding 有可能也不是 relashionship, 而是 entity. 詳見下圖:



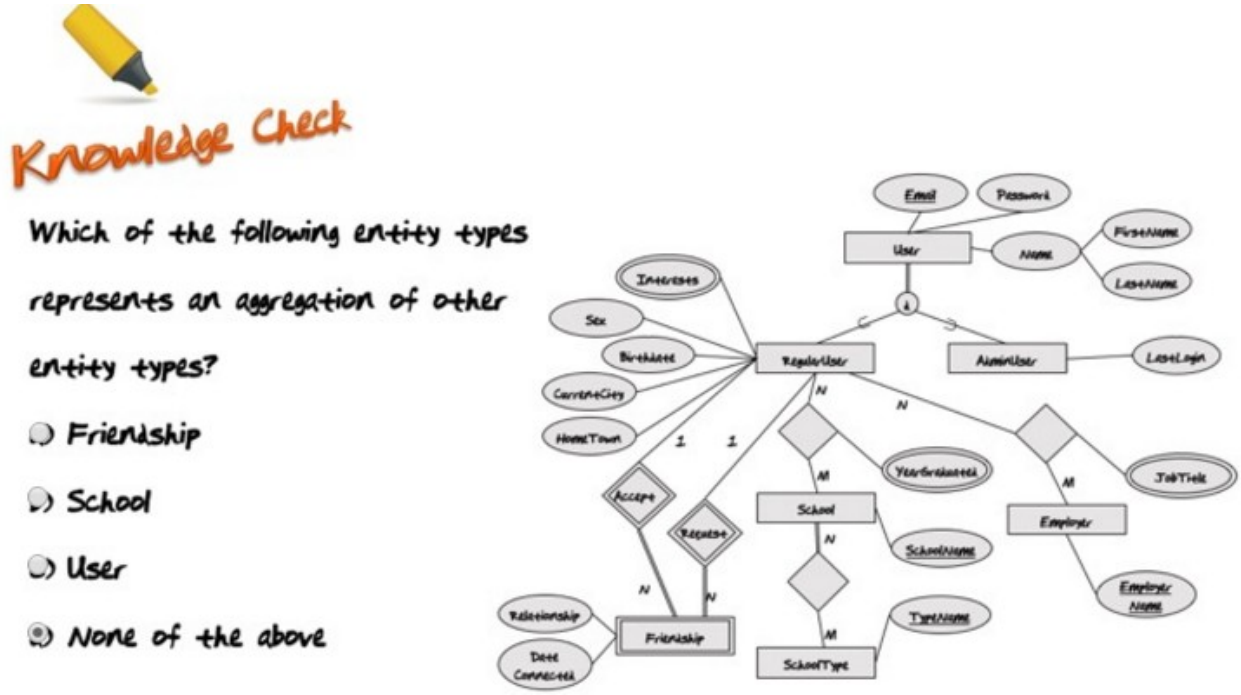
The above figure: From the Wedding Planner's perspective, these are all properties about the all central entity type of this whole database, namely their wedding. So what are we illurstrating with these two basic example is that context is very important when you fix a perception of reality (此處可以深刻理解 fix a perception of reality 是甚麼意思) and want to represent it.



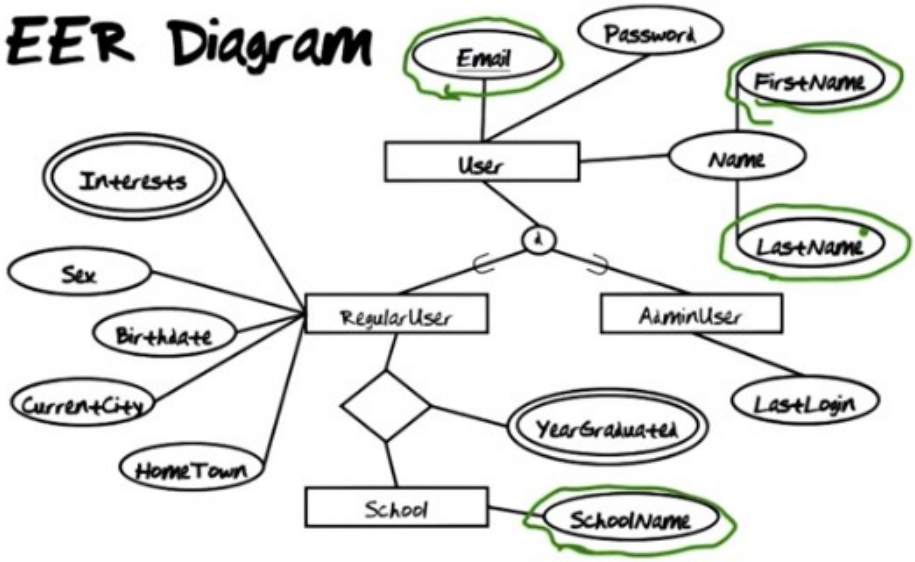
The above figure: our abillity to define, for example, entity types in the extended entity relationship model clearly is in support of representing classification. The super-sub type relationship is introduced in order to be able to support generaliztion. But how about aggregation(聚合)?



上兩圖說的就是 car, 而不是 train. So the Entity-Relationship (無 Extended, 不過無所謂) model does ont explicity support aggregation. Of course you can fauch(意) it and use existing types, relationship types etc. But that's like playing Objected Oriented Programming in C. It might be a good idea, but it's not supported by any kind of tool.



What's the Result Type of a Query?



The above figure: Let's say we want to print a list of information about RegularUsers in the schools they attended. The information we would likt to come out in the result could be say, the email address... (即劃綠圈的那幾個). What's the type of that query? Well, it's a list of properties. But a list of properties is not a type. It's not an entity type, it's not a relationship type, …. So there is no type to capture this result. The essential observation that comes out of that is that since the result doesn't have a type, there is not way that we can take that result and contiue to operate on it with a query language. So the query that I just asked does not have a type and therefore what I used was not a closed query language. Query langauges have to be closed. It's the only way we can formulate high-level ideas and ask high level questions. Later on when we look at relational database, you will see elegant powerful examples of colsed query languages. In my opinion, the lack of a generally agreed upon closed query langauge for the Extended-Entity Relationship Model is the reason that database management systems are not based on the Extended-Entity Relationship Model.

The Extended-Entity Relationship Model is great for fixing and representing a perception of reality. However, there is no commercial database system that implements the model directly. All most all the commercial systems implement the relational data model. So what I want to do is to show you how to map Extended-Entity Relationship diagrams to relations. For you to understand that mapping, I first need to define for you what a relation is.

The Relational Model

Theoretical Foundation

Relational Model

- Data structures
- Constraints
- Operations
  - Algebra
  - Calculus
    - tuple calculus (SQL)
    - domain calculus (QBE)

Data Structures

- a domain, D, is a set of atomic values (a type)

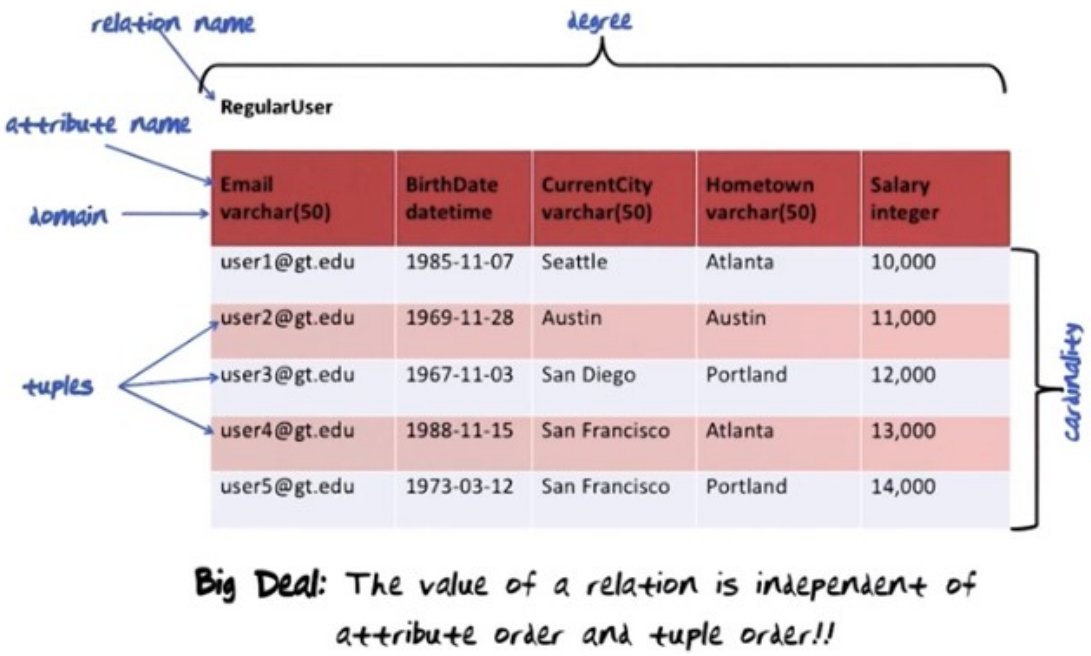
- a relation, R, is a subset of the set of ordered n-tuples,

$$R \subseteq \{<d_1, d_2, \ldots, d_n> \mid d_i \in D_i, i=1, \ldots, n\}$$

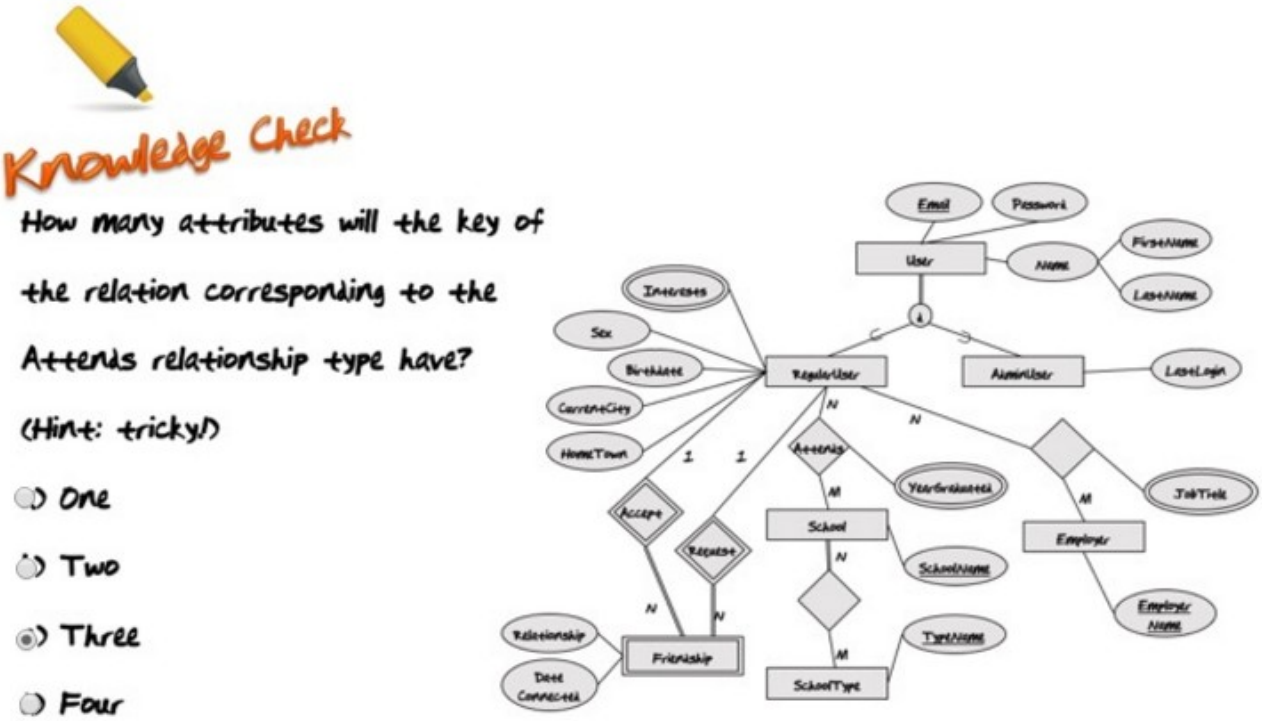
- an attribute, A, is a unique name given to a domain in a relation

helping us interpret domain values

The above figure: as opposed to the EER model, … there is only one structure in relational database, namely relations. Atomic values that from the point of view of the database management system has no meaning inside of it. In other words, nothing is coded in the values, they are just values without meaning that we want to represent. 上面第一句的 a type 是指'a set of atomic values'.



上圖中, 整個表就是一個 relation, 每一列是一個 attribute(總列數叫 degree), 每一行是一個 tuple(總行數叫 cardinality). vachar[50], datatime 和 integer 這些數据類型就叫 domain, 注意上上圖中的 a type.



From Piazza: the instructors' answer:

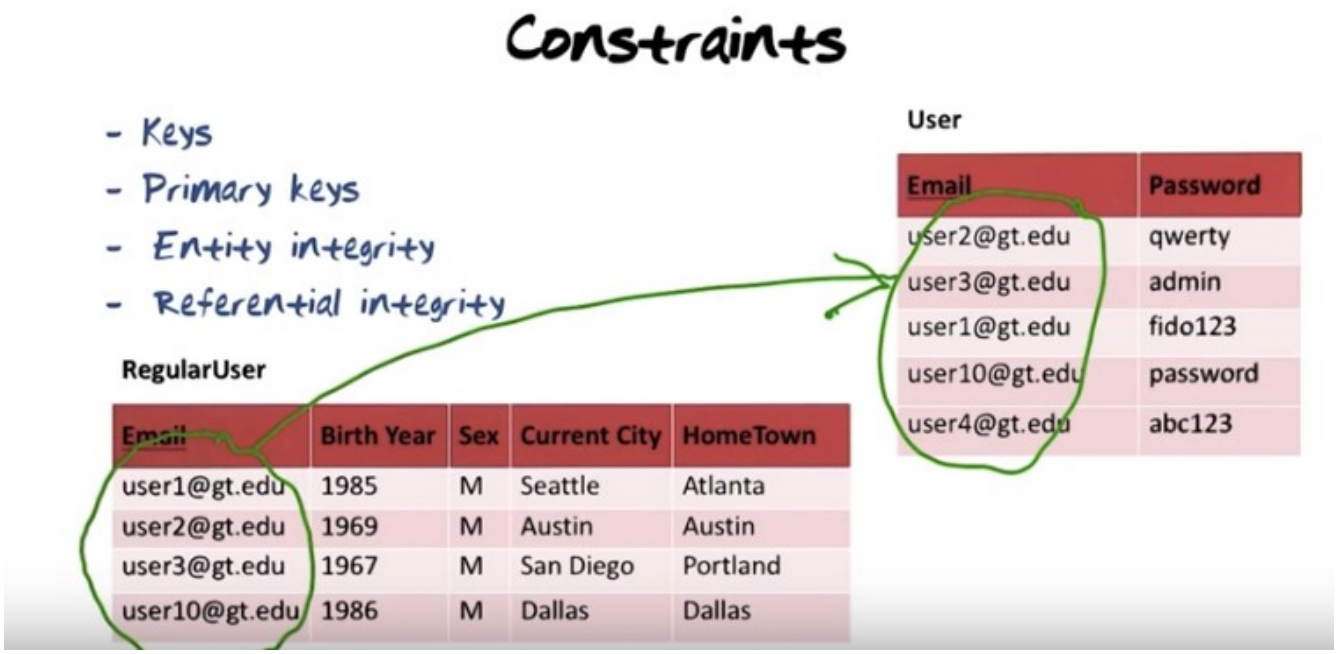
The question expects you to tell the number of attributes required to identify a single entry in the relationship table.

Here, we are not modeling the degree earned but each 'attendance / degree' at the school uniquely using the year of graduation from the school (which is a multi-valued attribute). The value of the attribute

'year_graduated' forms a part of the key thus (user_id, school_id, year_graduated) together form what we call a compound key and help identify each 'attendance' at the school uniquely.

For eg.
A user 'u' attended school 's' and graduated in 2008. (u, s, 2008)
'u' attended 's' again and graduated in 2013. (u, s, 2013)

Hence 3; Hence tricky.



Here we have chosen to use emails as the unique identifiers for users as opposed to, for example, using surrogates. Email is called the primary key of the user table. There is an important consequence of being a primary key and that is that no value of email is allowed to be the null value. There is another important implication and that is whenever something which is a primary key somewhere else like email here(即 User 這個表中), when that's used in another table like here(即 RegularUser 這個表中), the following must be obeyed: the set of email addresses that appeared in this table(即 RegularUser 這個表) must be a subset of the set of email addresses that appear here(即 User 這個表). So this(即 RegularUser 這個表) must be a subset of this(即 User 這個表).