8.2

**Superclass of a subclass.** Some entities types may have one or more attributes in common. They could be candidates for a superclass. Basically, some attributes or relationships may apply to all entity types, but other attributes and relationships among those groups of "subclass candidates" are very different. In this case, these subclasses form a bigger entity type, A.K.A the superclass.

For example, sedans, SUVs, trucks are three different entity types (subclass), but they all belong to a superclass: vehicle. The VIN number is a shared attribute type among those subclasses. So as the color, odometer, etc. But trucks have capacity of goods as its own special attribute, sedan and SUV does not. As a result, the vehicle is a superclass, while the sedans, SUVs, and trucks are subclasses.

**Superclass/subclass relationship**. Relationship between a superclass and any one of its subclass is the superclass/subclass relationship.

**IS-A relationship**. If the concept of the superclass and its subclass exists, there must be some relationship between them. The relationship between a superclass and its subclass is often referred as is-a relationship.

For example, in the previous vehicle model, we can say a sedan **is a** vehicle.

**Specialization** is a process of defining a set of subclass of an entity type. In other words, after specialization, a special set of subclass is defined on basis of some unique characteristic of the entities in the superclass.

For example, if we want the vehicle that can carry lots of goods, then the truck subclass is defined after specialization using the characteristic of capability of carrying lots of goods.

**Generalization** is a reverse process of abstraction in which differences between several entity types are suppressed, shared attributes are listed, and generalized into a single superclass.

For example, the sedan, SUV, and trucks can form the vehicle superclass after generalization of common attributes, such as the VIN, odometer, colors, owners, etc.

**Category**. Sometimes, we may need to connect a subclass to more than one superclass. This may happen because for different superclass, different entity types are presented. In this case, the subclass will represent a collection of objects that is a subset of the distinct entity types; such subclass is called a category (/union).

**Specific (local) attributes** are these features that are not commonly shared among all the subclasses.

For example, the trucks may have the carrying capacity (for goods) as its specific local attribute. Meanwhile, the sedans and SUVs do not have that attribute.

**Specific relationships** are some relationships only for a subclass of the superclass, but not for all subclasses nor for the superclass. Such relationships are called specific relationships.

For example, carries_goods can be a specific relationship between trucks and companies (suppose it is an entity), but not between sedans/SUVs and companies.

8.3

**Mechanism of Attribute/Relationship Inheritance**.

Members of a subclass inherit all attributes of superclass entity. This is because the attributes represent some real-world features all the subclasses share in common. Likely, subclass entity also inherits all relationships in which superclass involves.

**Benefits**:

It allows specification of particular attributes in relation to its being present in a subclass and still retains characteristics that it possessed being part of the superclass. Besides, it helps to decrease redundancy in representing attributes and relationships.

8.5

**User-defined specializations**. When there is no condition for determining membership of all subclasses of a specialization, the specialization is called user-defined specialization. Membership in this case is determined by database users when they apply some operation. Membership is not specified by any condition that may be evaluated automatically.

**Attribute-defined specializations**. If all subclasses of a specialization have their membership condition on the same attribute of the superclass, the specialization itself is called and attribute-defined specialization.

**Difference**: Membership in the former case is defined automatically, using some condition. On the other hand, membership in the latter case is not automatically defined. Some users have to apply their rules/commands to determine membership.

8.19

Entity Types:

1. LIBRARY_MEMBER

2. BOOK

3. STAFF_MEMBER

Relationship types:

1. ISSUE_CARD

2. ISSUE_NOTICE

3. ISSUE_BOOK

4. GET_DESCRIPTION

Aggregation:

1. All entity types are aggregation of constituent attributes as can be seen from EER diagram.

2. Relationship types that have member attributes (see figure) are also aggregation.

Identification:

1. All entity types and Relationship type are identified by names.

2. Each entity of entity type is identified differently by:

a. LIBRARY_MEMBER: SSN

b. BOOK: Key (Title, Bind, Language, ISBN)

c. STAFF_MEMBER: SSN

Specialization/ generalization:

1. Specialization of STAFF_MEMBER on basis of Designation. This is a partial disjoint specialization.

2. Specialization of BOOK on basis of In_Library. This is a total disjoint specialization.

3. Specialization of IN_LIBRARY_BOOK on basis of Rentable. This is a total disjoint specialization.

Other Constraints that may pose in future:

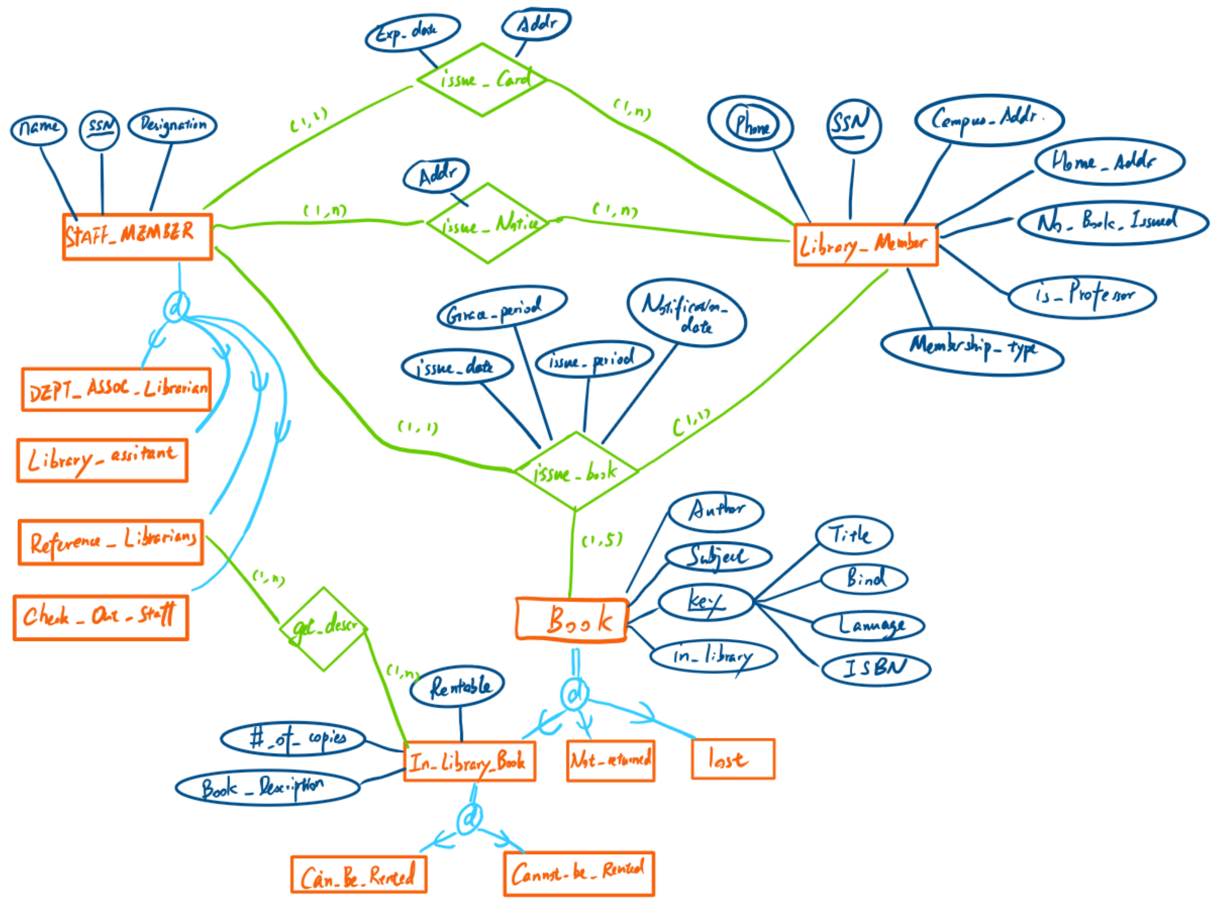
1. Fine that will be charged for a lost card.

2. Fine that will be charged for damaged/lost book.

3. Privileges that may be entitled to a particular group of users. (eg: Prof has higher priv than student)

4. ISBN for a book may have 13-digit-length and 11-digit-length at the same time. (So far, we just assume we only use the 13-digit-length)

5. Expiry period of lost membership card. (One’s membership will expire after that period.)



Notice: catalog is recorded by DBMS, because it is a meta-data type. Borrowing activity is recorded through the relationship: issue_book.

8.26

a) **Wrong**! E1 and E2 are overlapping specializations. Suppose A belongs to both E1 and E2, then A has a relationship R with E3, according to the E2-R-E3. But since A is in E1 as well, E1 must have some relationship with E3. So, it is a wrong EER diagram.

On the other hand, if there is a condition that any overlapping entity is prohibited from being related to E3, then this EER diagram could be correct.

b) **Correct**! Consider E1 is a brother and E2 is a sister. R could be fall_in_love. Although this kind of love is prohibited by law, this model is still possible for research purpose.

c) **Correct** if we **assume** these is a superclass above the picture! Consider E1 to be manager, and E3 to be managed workers. Since a boss could has his/her own boss, the EER diagram is very realistic to represent the employee organization in a company.