

Laboratory 3

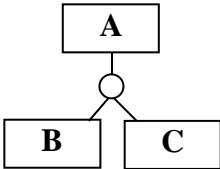
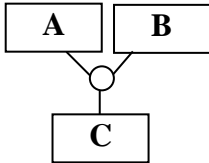
EER Modelling and Transformation

References: Lecture Notes: Topic 3, Elmasri and Navathe, 2017: Chapters 4 & 9

In this class you will design solutions that require the additional features available in Enhanced Entity Relationship (EER) Modelling, namely the ability to define specialisation/generalisation relationships and consequently inherit attributes.

Helpful Tips:

Relational Table Transformation- 9 Step (First 7 steps are in previous lab)

Step	ER-Model	Relational Model
8	<u>Specialisation/ generalisation relationship</u>	
8a	<i>N relations: superclass + subclasses</i> 	3 tables A, B, C A(A.PK, A.attrs) B(A.PK, B.attrs) C(A.PK, C.attrs) For: total or partial, disjoint or overlapping
8b	<i>N relations: subclasses only</i>	2 tables B, C B(A.PK, A.attrs, B.attrs) C(A.PK, A.attrs, C.attrs) For: total
8c	<i>1 relation + 1 type attribute</i>	1 table A A(A.attrs, B.attrs, C.attrs, aType) For: disjoint, NULL for B.attrs (or C.attrs)
8d	<i>1 relation + N type attributes</i>	1 table A A(A.attrs, B.attrs, C.attrs, isB, isC) For: overlapping
9	<i>Category/Union type relationship</i> 	3 tables A, B, C A(A.PK, A.attrs, C.PK) B(B.PK, B.attrs, C.PK) C(C.PK, C.attrs)

Repeat steps 2-7 for all tables created in Step 8 & 9

IMPORTANT: Steps 8 and 9 can be combined with Step 1 to create all necessary tables (before transforming their relationships in steps 2-7)

Exercise: EER Diagrams and Transformation

Create an EER-diagram to model the following problem and perform the mapping to translate your diagram into relational tables. For any specialisation/generalisation relationships, state your reasons for using the mapping technique you have chosen.

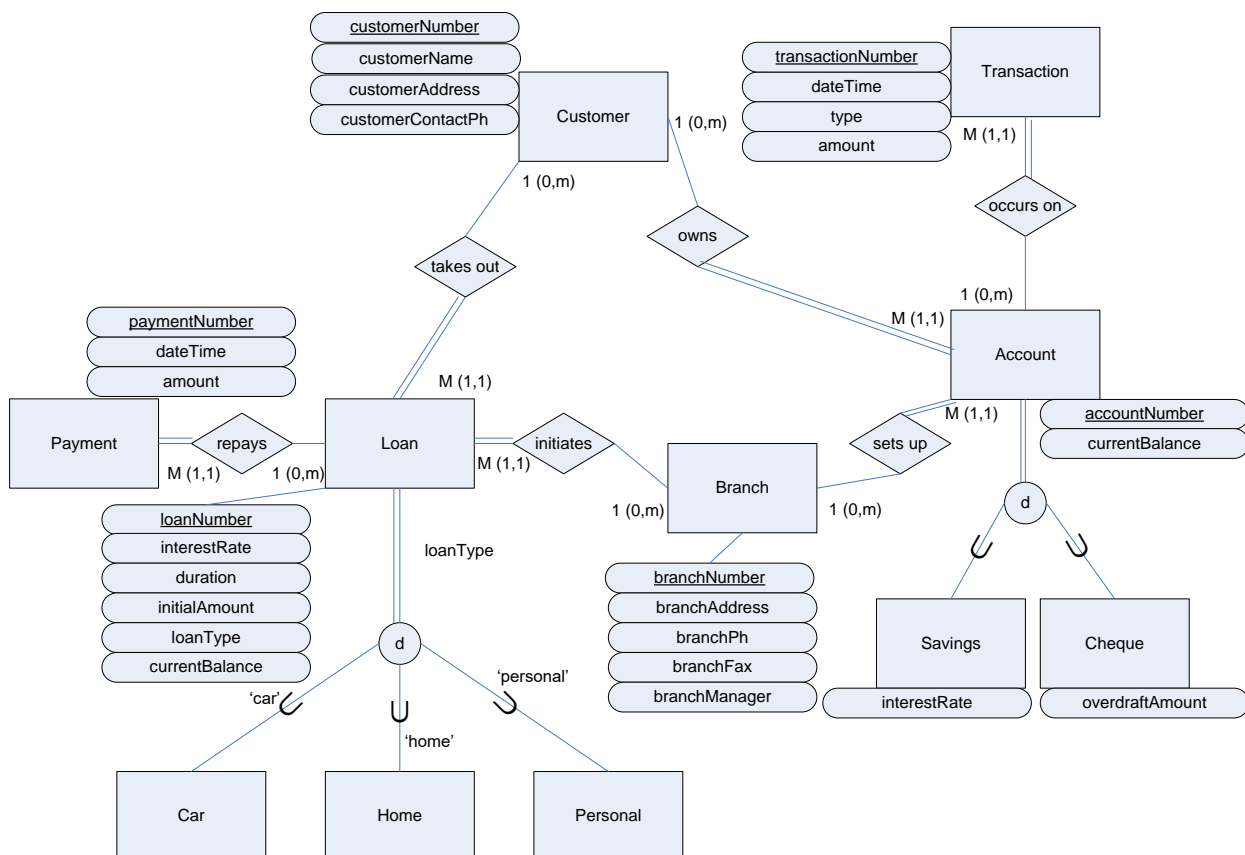
1. A bank has a number of branches. Each branch manages its own transaction accounts and loans. There are two types of transaction accounts: savings accounts and cheque accounts; and three types of loans: car loans, home loans and personal loans. Each loan has its own rate of interest, duration and loan amount. A savings account has an associated rate of interest and a cheque account has an overdraft limit.

For both savings accounts and cheque accounts the branches need to keep track of deposits and withdrawals and the current balances. The transaction information required includes the transaction amount and the date and time of the transaction. Similar information is required for loan repayments.

Customer information includes: customer number, name, address and phone number. A customer may have many transaction accounts and loans, however, each transaction account and loan may belong to only one customer.

Branch information includes: branch number, address, phone number, fax number and the name of the branch manager.

Answer:



*Car, Home and Personal are **predicate-defined** or **condition-defined** subclasses. This means that the members of each subclass are determined by placing a condition on the values of some attribute of the superclass – in this case loanType. loanType becomes the **defining predicate** of the subclass. This condition is a constraint specifying that exactly those entities of the LOAN entity type whose attribute value for loanType is 'Car' belong to the CAR subclass etc. A predicate-defined subclass is represented by writing the predicate condition next to the line that connects the subclass to the specialisation circle as has been done in the EER diagram above.*

*If all subclasses in a specialisation have their membership condition on the same attribute of the superclass (as is the case above), the specialisation itself is called an **attribute-defined specialisation**, and the attribute is called the **defining attribute** of the specialisation. This is indicated above by the placement of the defining attribute name next to the arc from the circle to the superclass.*

*Note: if there is no condition for determining membership in a subclass, the subclass is called **user-defined**.*

*“Two other constraints may apply to a specialisation. The first is the **disjointness constraint** (also called the **disjointedness constraint**) which specifies that the subclasses of the specialisation must be disjoint. This means that an entity can be a member of at most one of the subclasses of the specialisation. A specialisation that is attribute-defined implies the disjointness constraint if the attribute used to define the membership predicate is single-valued the **d** in the circle stands for disjoint.”*

*“If the subclasses are not constrained to be disjoint, their sets of entities may **overlap**; that is, the same (real-world) entity may be a member of more than one subclass of the specialisation. This case, which is the default, is displayed by placing an **o** in the circle...”*

*“The second constraint on specialisation is called the **completeness constraint**, which may be total or partial. A **total specialisation** constraint specifies that every entity in the superclass must be a member of at least one subclass in the specialisation... This is shown in EER diagrams by using a double line to connect the superclass to the circle. A single line is used to display a **partial specialisation**, which allows an entity not to belong to any of the subclasses.”*

Relational Transformation

Step 1: Mapping of Regular Entity Types (include superclasses)

CUSTOMER (customerNumber, customerName, customerAddress, customerContactPh)
 BRANCH (branchNumber, branchAddress, branchPh, branchFax, branchManager)
 PAYMENT (paymentNumber, dateTime, amount)
 TRANSACTION (transactionNumber, dateTime, amount, type)
 LOAN (loanNumber, interestRate, duration, initialAmount, currentBalance, loanType)
 ACCOUNT (accountNumber, currentBalance)

Step 4: Mapping of Binary 1:M Relationship Types

PAYMENT (paymentNumber, dateTime, amount, loanNumber)
 TRANSACTION (transactionNumber, dateTime, amount, type, accountNumber)
 LOAN (loanNumber, interestRate, duration, initialAmount, currentBalance, loanType, branchNumber, customerNumber)

ACCOUNT (accountNumber, currentBalance, branchNumber, customerNumber)

Step 8: Mapping of Specialisation/Generalisation Relationship Types

For the purposes of outlining what each of the options mean, we present all four options (8A, B, C, and D) here. But for every specialisation, we must select ONE.

Option 8a: Multiple Relations – Superclass and subclasses

Create a relation for the superclass which contains the primary key and all other attributes of the superclass. Create a relation for each of the subclasses, which includes all attributes of that subclass and where the primary key is the primary key of the superclass.

“This option works for any specialisation (total or partial, disjoint or overlapping).”

ACCOUNT (accountNumber, currentBalance, branchNumber, customerNumber)

SAVINGS_ACCOUNT (accountNumber, interestRate)

CHEQUE_ACCOUNT (accountNumber, overdraftAmount)

For account specialisation, we select 8A.

Option 8b: Multiple Relations – Subclass relations only

Create a relation for each subclass which contains the attributes of the superclass and the attributes of that subclass. The primary key is the primary key of the superclass.

“This option only works for a specialisation whose subclasses are total (every entity in the superclass must belong to (at least) one of the subclasses). If the specialisation is overlapping, an entity may be duplicated in several relations.”

SAVINGS_ACCOUNT (accountNumber, currentBalance, interestRate, branchNumber, customerNumber)

CHEQUE_ACCOUNT (accountNumber, currentBalance, overdraftAmount, branchNumber, customerNumber,)

NOTE: If this option is taken, the ACCOUNT table from Step 1 must be removed.

Option 8c: Single relation with one type attribute

*Create a single relation that has as its attributes all the attributes of the superclass, plus all the attributes of each of the subclasses, plus an additional attribute called **type**. The **type** attribute is the discriminating attribute that identifies the subclass to which each tuple belongs, if any. The primary key of the relation is the primary key of the superclass.*

“This option works only for a specialisation whose subclasses are disjoint, and has the potential for generating many NULL values if many specific attributes exist in the subclasses.”

LOAN (loanNumber, interestRate, duration, amount, loanType, currentBalance, customerNumber, branchNumber)

Option 8c is used because none of the subclasses of loan have their own attributes. Therefore, the relationship is best modelled using a type attribute. In fact there was no need to model

this using specialisation/generalisation it could have been modelled in the same way as TRANSACTION.

Option 8d: Single relation with multiple type attributes

*Create a single relation that has as its attributes all the attributes of the superclass, plus all the attributes of each of the subclasses, plus multiple **type** attributes. Each of the **type** attributes is a **Boolean** indicating whether a tuple belongs to a particular subclass.*

“This option works for a specialisation whose subclasses are overlapping (but will also work for a disjoint specialisation).” (Elmasri and Navathe: 280, 2011)

AFTER STEP 9 YOU WILL NEED TO CHECK THROUGH STEPS 2 – 7 AGAIN FOR THE TABLES CREATED IN STEPS 8 AND 9

FINAL TABLES: - USING OPTION 8B FOR ACCOUNT AND 8C FOR LOAN

CUSTOMER (customerNumber, customerName, customerAddress, customerContactPh)

BRANCH (branchNumber, branchAddress, branchPh, branchFax, branchManager)

PAYMENT (paymentNumber, dateTime, amount, *loanNumber*)

TRANSACTION (transactionNumber, dateTime, amount, type, *accountNumber*)

ACCOUNT (accountNumber, currentBalance, *branchNumber*, *customerNumber*)

SAVINGS_ACCOUNT (*accountNumber*, interestRate)

CHEQUE_ACCOUNT (*accountNumber*, overdraftAmount)

LOAN (loanNumber, interestRate, duration, amount, loanType, currentBalance, *customerNumber*, *branchNumber*)

2. An art museum wants to catalogue all the works in its collection. The collection currently consists of three different types of works: paintings, sculptures and tapestries. All works are catalogued using the following information: a unique catalogue number, artist (if known), year of creation (if known), country of origin, title of the work and a description of the work. Paintings are further described by: the type of paint used (oil, watercolour, etc), the material on which they are painted (paper, canvas, wood, etc) and the style in which they are painted (abstract, impressionist, etc). Sculptures are further described by the material used to create them (wood, stone, marble, bronze, etc) as well as their height and weight. Tapestries are further described by the material used for their warp threads (wool, cotton, etc) and weft threads (wool, cotton, silk, gold, silver, etc).

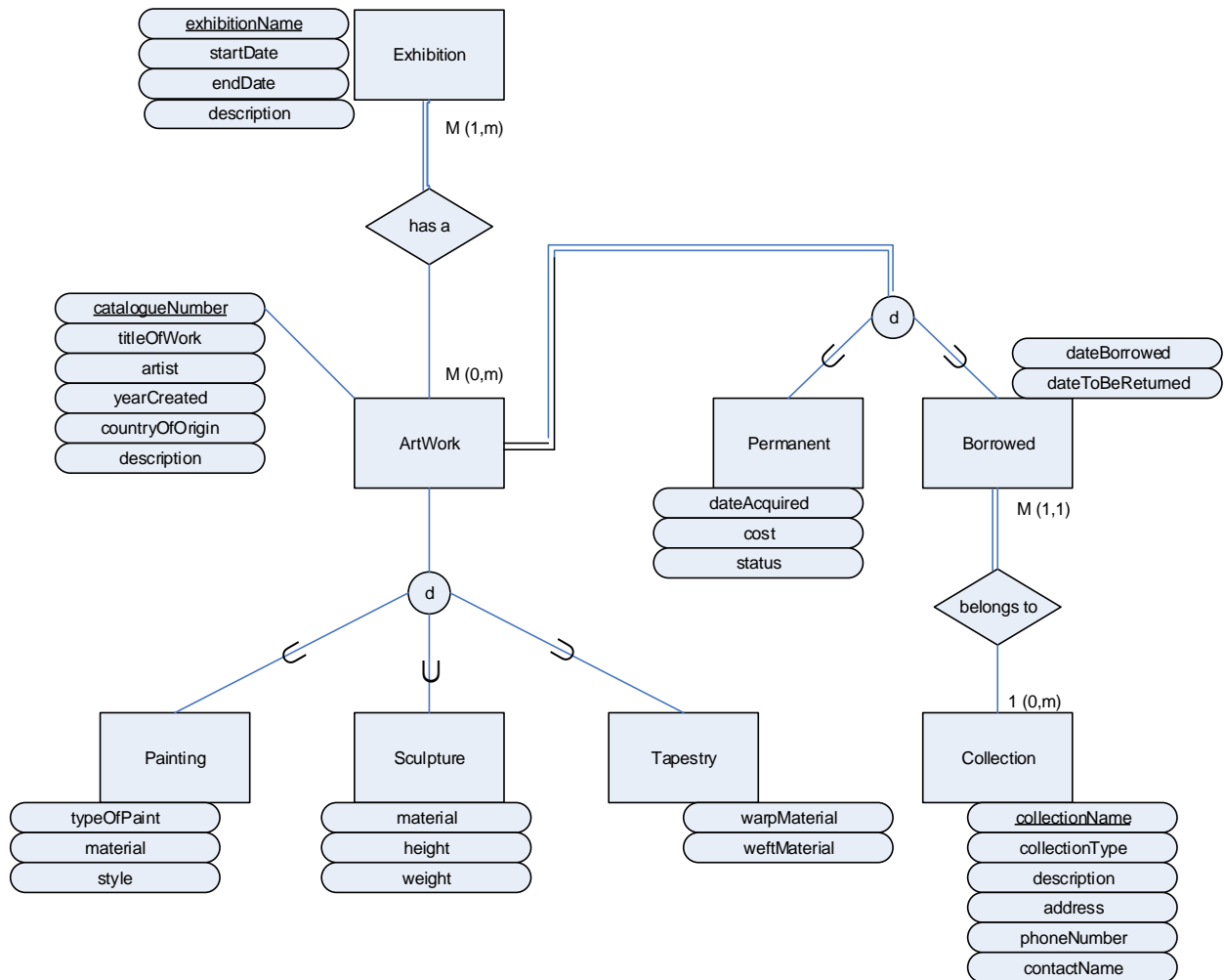
Each work is also categorised according to whether it belongs to the museum's permanent collection or is on loan from another collection. Works in the permanent collection require the date they were acquired, their status (on display, on loan or in storage), and their cost to be recorded. For borrowed works information on the collection they belong to, the date they were borrowed and the date they are to be returned is required. Borrowed works may be of different types to those in the museum's collection.

The collections works are borrowed from can be either other museums or private collections. Information to be kept regarding the collections includes: the name of the collection (unique), the type, a description of the collection, address, phone number, and contact name.

The museum also wants to keep track of which works have been displayed in each of its many exhibitions throughout year. In addition to the works on display, exhibition information includes the name of the exhibition (unique), a description, and its start and end date.¹

¹ Adapted from Elmasri R. and Navathe S.B., *Database Systems*, 2014.

Answer:



Relational Transformation

Step 1: Mapping of Regular Entity Types (including superclasses)

EXHIBITION (exhibitionName, startDate, endDate)

COLLECTION (collectionName, collectionType, description, address, phoneNumber, contactName)

ARTWORK (catalogueNumber, titleOfWork, artist, yearCreated, countryOfOrigin, description)

Step 4: Mapping of Binary 1:M Relationship Types

See step before final tables

Step 5: Mapping of Binary M:M Relationship Types

IN_EXHIBITION(exhibitionName, catalogueNumber)

Step 8: Mapping of Specialisation/Generalisation Relationship Types

Option 8a: Multiple Relations – Superclass and subclasses

ARTWORK (catalogueNumber, titleOfWork, artist, yearCreated, countryOfOrigin, description)

PAINTING (catalogueNumber, , typeOfPaint, material, style)

SCULPTURE (catalogueNumber, , material, height, weight)

TAPESTRY (catalogueNumber, , warpMaterial, weftMaterial)

Option 8a has been chosen so that information relating to those works borrowed from other collections that are of different types to the collection in this museum can be recorded.

PERMANENT (catalogueNumber, dateAcquired, cost, status)

BORROWED (catalogueNumber, dateBorrowed, dateToBeReturned)

Option 8b: Multiple Relations – Subclass relations only

This option is not used in this mapping

Option 8c: Single relation with one type attribute

This option is not used in this mapping

Option 8d: Single relation with multiple type attributes

This option is not used in this mapping

REPEAT STEPS 2 – 7 FOR THE TABLES CREATED IN STEPS 8 AND 9

Step 4: 1:M

BORROWED (catalogueNumber, dateBorrowed, dateToBeReturned, *collectionName*)

FINAL TABLES:

EXHIBITION (exhibitionName, startDate, endDate)

COLLECTION (collectionName, collectionType, description, address, phoneNumber, contactName)

ARTWORK (catalogueNumber, titleOfWork, artist, yearCreated, countryOfOrigin, description)

PAINTING (catalogueNumber, , typeOfPaint, material, style)

SCULPTURE (catalogueNumber, , material, height, weight)

TAPESTRY (catalogueNumber, , warpMaterial, weftMaterial)

PERMANENT (catalogueNumber, dateAcquired, cost, status)

BORROWED (catalogueNumber, dateBorrowed, dateToBeReturned, *collectionName*)

IN_EXHIBITION(exhibitionName, catalogueNumber)

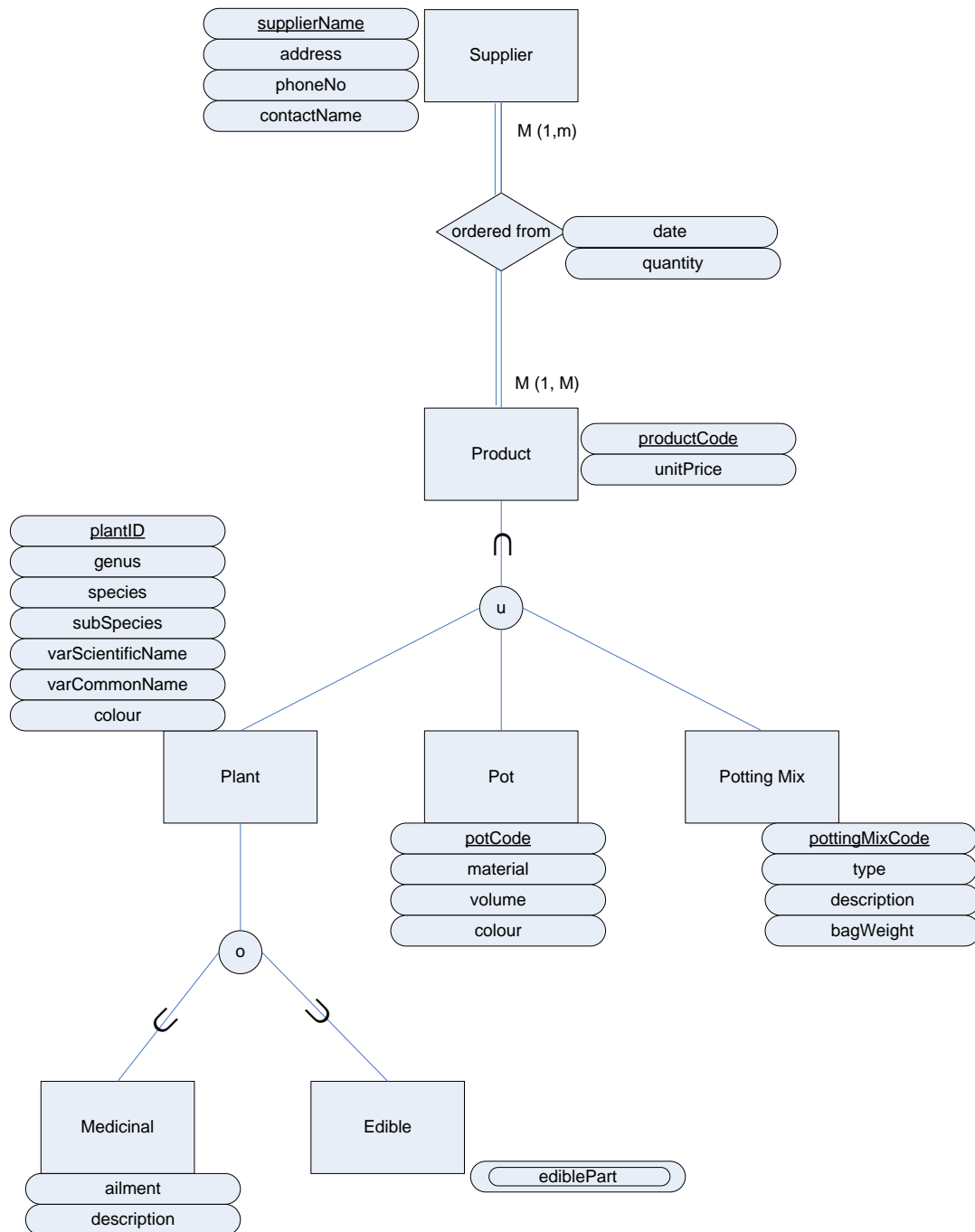
3. Natasha's Native Nursery has commissioned you to create a database design that will help Natasha co-ordinate the ordering of stock. The Nursery's stock consists of three distinct types of product: plants, pots and potting mixes. Each type of plant has a unique id, genus, species, subspecies, variety and flower colour (if flowering). For the variety both the scientific name and common name need to be stored.

Information regarding pots includes the pot code, volume, material (terracotta, plastic etc) and colour. Information on potting mixes includes the potting mix code, the potting mix type (general purpose, special purpose etc), a description of the ingredients and the weight of each bag.

Natasha needs to keep track of which suppliers she has bought her stock from so that she can easily place an order when the stock is running low. Supplier information includes the supplier company name, address, phone number and contact person. When Natasha places an order she needs to enter the supplier company name, order date, product code(s), product(s) unit price and quantity required. Note that Natasha's Nursery also grows her own plants, makes custom made plant pots and potting mixes

In addition to information regarding the plant type, Natasha likes to provide her customers with information regarding whether or not a plant is edible, whether it has any medicinal properties, or whether it is purely ornamental. Edible plants are further described by which part of the plant is edible (flower, leaf, stem, all). Medicinal plants are further described by which ailment they can assist in treating (for the purposes of this exercise assume one ailment per plant) and a description of how the plant is to be used (eaten, applied to skin, boiled and drunk etc).

Answer:



NOTE:

Some students might want to add another subclass ORNAMENTAL.

Relational Transformation

Step 1: Mapping of Regular Entity Types (include any superclasses)

SUPPLIER (supplierName, address, phoneNo, contactName)

PLANT (plantCode, genus, species, subSpecies, varScientificName, varCommonName, colour)

POT (potCode, material, volume, colour)

POTTING_MIX(pottingMixCode, type, description, bagWeight)

Step 8: Mapping of Specialisation/Generalisation Relationship Types

Two of the possible options are shown below. We eventually select option 8A for this example.

Option 8a: Multiple Relations – Superclass and subclasses

PLANT (plantCode, genus, species, subSpecies, varScientificName, varCommonName, colour)

MEDICINAL(plantCode, medicalAilment, medicinalDescription)

EDIBLE(plantCode)

Option 8d: One Relation with multiple type attributes

PLANT (plantID, genus, species, subSpecies, varScientificName, varCommonName, colour, isMedicinal, isEdible, medicinalAilment, medicinalDescription)

Step 9: Mapping of Categories (Union Types)

*"For mapping a category whose defining superclasses have different keys, it is customary to specify a new key attribute, called a **surrogate key**, when creating a relation to correspond to the category. The keys of the defining classes are different, so we cannot use any one of them exclusively to identify all entities in the category."*

The category becomes an entity with the surrogate key and the surrogate key becomes a foreign key in the defining superclasses. If the defining superclasses have the same key, that key becomes the primary key of the category.

PRODUCT (productCode, unitPrice)

PLANT (plantID, genus, species, subSpecies, varScientificName, varCommonName, colour, *productCode*) – Using 8d

POT (potCode, material, volume, colour, *productCode*)

POTTING_MIX(pottingMixCode, type, description, bagWeight, *productCode*)

Here the assumption is made that the productCode relates to the supplier, where as the primary keys of Plant, Pot and Potting_Mix are defined by the Nursery.

REPEAT STEPS 2 – 7 FOR THE TABLES CREATED IN STEPS 8 AND 9

Step 5: Mapping of Binary M:M Relationship Types

ORDER(productCode, supplierName, quantity, date)

Step 6: Multi-valued Attributes

EDIBLE(plantCode, ediblePart)

FINAL TABLES:

SUPPLIER (supplierName, address, phoneNo, contactName)

ORDER(productCode, supplierName, quantity, date)

PRODUCT (productCode, unitPrice)

PLANT (plantID, genus, species, subSpecies, varScientificName, varCommonName,
colour, *productCode*)
POT (potCode, material, volume, colour, *productCode*)
POTTING_MIX(pottingMixCode, type, description, bagWeight, *productCode*)
MEDICINAL(*plantCode*, medicalAilment, medicinalDescription)
EDIBLE(*plantCode*, ediblePart)