17.2

(a) Strong Entity Types

A **strong entity type** is an independent entity with a primary key. When deriving relations:

- Create a relation (table) for each strong entity type.
- The attributes of the entity become columns of the relation.
- The primary key of the entity is designated as the primary key of the relation.

(b) Weak Entity Types

A **weak entity type** depends on a strong entity type and lacks a primary key of its own. For weak entities:

- Create a relation for each weak entity.
- Include all attributes of the weak entity.
- Include the primary key of the owner (strong) entity as a foreign key.
- The primary key of the weak entity relation is a composite key consisting of the primary key of the owner entity and the partial key (discriminator) of the weak entity.

(c) One-to-Many (1:*) Binary Relationship Types

In a *1: relationship** between entities A and B (A has one, and B has many):

- Add the primary key of the "one" side (entity A) as a foreign key in the relation representing the "many" side (entity B).
- No separate relation is needed for the relationship unless it has attributes of its own, in which case those attributes would also be added to entity B's relation.

(d) One-to-One (1:1) Binary Relationship Types

In a **1:1 relationship** between entities A and B:

- Either add the primary key of A as a foreign key in B, or the primary key of B as a foreign key in A.
- Ensure uniqueness constraints on the foreign key to enforce the 1:1 relationship.
- If the relationship has attributes, create a separate relation with foreign keys referencing both A and B as a composite primary key.

(e) One-to-One (1:1) Recursive Relationship Types

In a **1:1 recursive relationship** (where an entity is related to itself):

- Add a foreign key within the same relation referencing the primary key of the entity.
- Ensure a uniqueness constraint on the foreign key to enforce the 1:1 constraint.
- If there are attributes associated with the relationship, create a separate relation with foreign keys to the same primary key (entity ID) and its related attributes.

(f) Superclass/Subclass Relationship Types

For **superclass/subclass relationships** (inheritance), use one of these approaches:

1. **Single Table Inheritance (Flattening)**: Create one table for all subclasses, including superclass attributes. Use a type discriminator column to distinguish each subclass.
2. **Class Table Inheritance**: Create separate tables for each subclass and one for the superclass, linking them via primary keys (shared key inheritance).
3. **Concrete Table Inheritance**: Create a separate table for each subclass, duplicating attributes from the superclass into each subclass table.

(g) Many-to-Many (*:*) Binary Relationship Types

In a **many-to-many (M**

) relationship between entities A and B:

- Create a new relation to represent the relationship.
- Include the primary keys of both entities A and B as foreign keys in this relation.
- The primary key of this relation is typically the combination of these two foreign keys.
- If the relationship has attributes, include them in this new relation as well.

(h) Complex Relationship Types

For **complex relationships** involving more than two entities:

- Create an associative relation that includes foreign keys referencing each participating entity.
- Use a composite primary key made up of the foreign keys or a surrogate key if needed.
- Include any relationship-specific attributes in this relation.

(i) Multi-Valued Attributes

For **multi-valued attributes** (attributes that can have multiple values for a single entity):

- Create a separate relation to represent the multi-valued attribute.
- Include the primary key of the entity as a foreign key in the new relation.
- The primary key of this relation is the combination of the foreign key and the multi-valued attribute itself or add an identifier if necessary.

17.3 Discuss how the technique of normalization can be used to validate the relations derived from the conceptual data model.

The logical data model can be validated using the technique of normalization and against the transactions that the model is required to support. Normalization is used to improve the model so that it satisfies various constraints that avoid unnecessary duplication of data. Normalization ensures that the resultant model is a closer model of the enterprise that it serves, it is consistent and has minimal redundancy and maximum stability.

17.8

Customer (customerNo, customerName, customerStreet, customerCity,

customerState, customerZipCode, custTelNo, custFaxNo, DOB,

maritalStatus, creditRating)

Primary Key customerNo

Alternate Key custTelNo

Alternate Key custFaxNo

Employee (employeeNo, title, firstName, middleName, lastName, address, workTelExt, homeTelNo, empEmailAddress, socialSecurityNumber, DOB, position, sex, salary, dateStarted)

Primary Key employeeNo

Alternate Key socialSecurityNumber

Invoice (invoiceNo, dateRaised, datePaid, creditCardNo, holdersName, expiryDate,

orderNo, pMethodNo)

Primary Key invoiceNo

Foreign Key orderNo references Order(orderNo)

Foreign Key pMethodNo references PaymentMethod(pMethodNo)

Order (orderNo, orderDate, billingStreet, billingCity, billingState, billingZipCode, promisedDate, status, customerNo, employeeNo)

Primary Key orderNo

Foreign Key customerNo references Customer(customerNo)

Foreign Key employeeNo references Employee(employeeNo)

OrderDetail (orderNo, productNo, quantityOrdered)

Primary Key orderNo, productNo

Foreign Key orderNo references Order(orderNo)

Foreign Key productNo references Product(ProductNo)

PaymentMethod (pMethodNo, paymentMethod)

Primary Key pMethodNo

Product (productNo, productName, serialNo, unitPrice, quantityOnHand, reorderLevel, reorderQuantity, reorderLeadTime)

Primary Key productNo

Alternate Key serialNo

Shipment (shipmentNo, quantity, shipmentDate, completeStatus, orderNo, productNo, employeeNo, sMethodNo)

Primary Key shipmentNo

Foreign Key orderNo, productNo references OrderDetail(orderNo, productNo)

Foreign Key employeeNo references Employee(employeeNo)

Foreign Key sMethodNo references ShipmentMethod(sMethodNo)

ShipmentMethod (sMethodNo, shipmentMethod)

Primary Key sMethodNo