Logical Database Design

Adapted from Chapter 17 (Connolly & Begg)

Logical database design

Logical data model process: represent a conceptual data model using a specific data model.

We will be using the relational data model, so we will be representing the conceptual data as relations.

Steps to the logical database design

1. Derive relations for logical data model
2. Validate relations using normalization
3. Validate relations against user transactions
4. Check integrity constraints
5. Review logical data model with user

Derive relations for ...

1. Strong entity types
2. Weak entity types
3. 1 to many (1:*) binary relationship types
4. 1 to 1 (1:1) binary relationship types
5. 1 to 1 (1:1) recursive relationship types
6. Superclass /subclass relationship types
7. Many to Many (*:*) binary relationship types
8. Complex relationship types
9. Multi-values attributes

Maps to a relation

Strong entity	Create a relation that contains all simple attributes
Weak entity	Create a relation that contains all simple attributes - primary key must take into account the owner entity's key
: binary relationship	Create a relation for the relationship, including all relationship attributes. Each entity in the relation is a foreign key in the relationship's relation.
1:1 binary relationship Mandatory participation Optional participation Both entities optional	 Combine entities into 1 relation Define a foreign key for relation associated with mandatory participation Your choice for representation (either can have FK)
Multi-valued attributes	Define a relation for the multi-valued attribute and create a foreign key to the relation representing the containing entity
Complex relationship	Create a relation for the relationship, including all relationship attributes. Each entity in the relation is a foreign key in the relationship's relation.

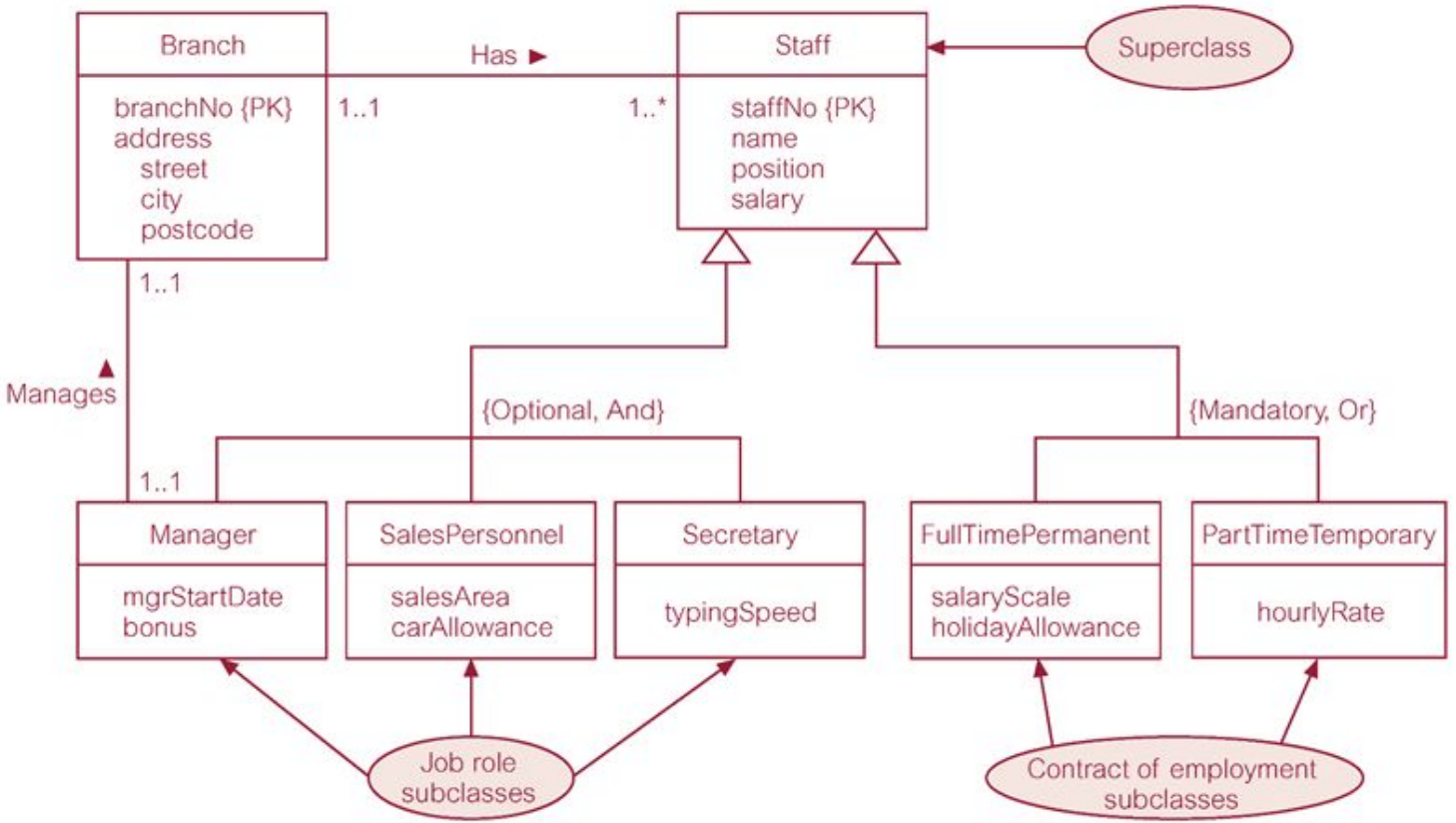
Maps to a foreign key

Entity/Relationship	Mapping to logical design
1:* binary relationship	Define a foreign key on the “many” side. It points to a candidate key on the “1” side”. All relationship attributes are stored in the “many” relationship. No relation necessary for relationship.
: binary relationship	Create a relation for the relationship, including all relationship attributes. Each entity in the relation is a foreign key in the relationship’s relation.
1:1 binary relationship Optional participation Both entities optional	Define a foreign key for relation associated with mandatory participation Your choice for representation (either can have FK)
Multi-valued attributes	Define a relation for the multi-valued attribute and create a foreign key to the relation representing the containing entity

Superclass/Subclass conversion

Participation	Disjoint constraint	Mapping to logical design
Mandatory	Nondisjoint (AND)	Single relation with 1 or more attributes acting as a discriminator for the subclasses
Mandatory	Disjoint (OR)	Many relations one for each subclass/superclass combination
Optional	Nondisjoint (AND)	Two relations, 1 relation for the superclass and 1 relation for all of the subclasses, subclass needs a discriminating attribute to differentiate type of subclass
Optional	Disjoint (OR)	Many relations, one relation for the superclass, one relation for each subclass

Classwork: create relations for UML



Normalization is covered

In a separate presentation.

Check integrity constraints

Types of integrity constraints

1. Identifying attributes that are required
   a. For each column decide if it needs to have a value
2. Attribute domain constraints
   a. List or describe the legal values for each attribute (NULL allowed?)
3. Multiplicity
   a. Ensure the relationship constraints are properly represented
4. Entity integrity
   a. Primary key attributes cannot hold a NULL value
5. Referential integrity
   a. Foreign key created in the child tuple linking to existing parent tuple
6. General constraints

Referential integrity defines DB behavior

Define the desired database behavior to ensure that a child relation NEVER references a parent relation instance that does not exist.

Review changes to the child relation.

1. CREATE a new record in the child relation
   a. If all foreign key attributes are NULL no check.
   b. If not NULL ensure parent tuple exists
2. UPDATE a foreign key attribute in the child relation
   a. Same as above
3. DELETE a record from the child relation
   a. Operation cannot violate referential integrity.

Referential integrity defines DB behavior

Review changes to the parent relation.

1. UPDATE a primary key attribute in the relation
   a. Identify the child tuples in the other table referencing this instance
   b. May choose to not allow update (ON UPDATE RESTRICT)
   c. May choose to allow UPDATE to parent relation to propagate to child (ON UPDATE CASCADE)
   d. May choose to remove the link between the 2 entities (ON UPDATE SET NULL or ON UPDATE SET DEFAULT)
2. DELETE a record from the parent relation
   a. Same as above except DELETE as oppose to UPDATE
3. CREATE a record in the parent relation
   a. No check to be done

Classwork: convert to a logical db design