Answers to Review Questions

1. What is an entity supertype, and why is it used?

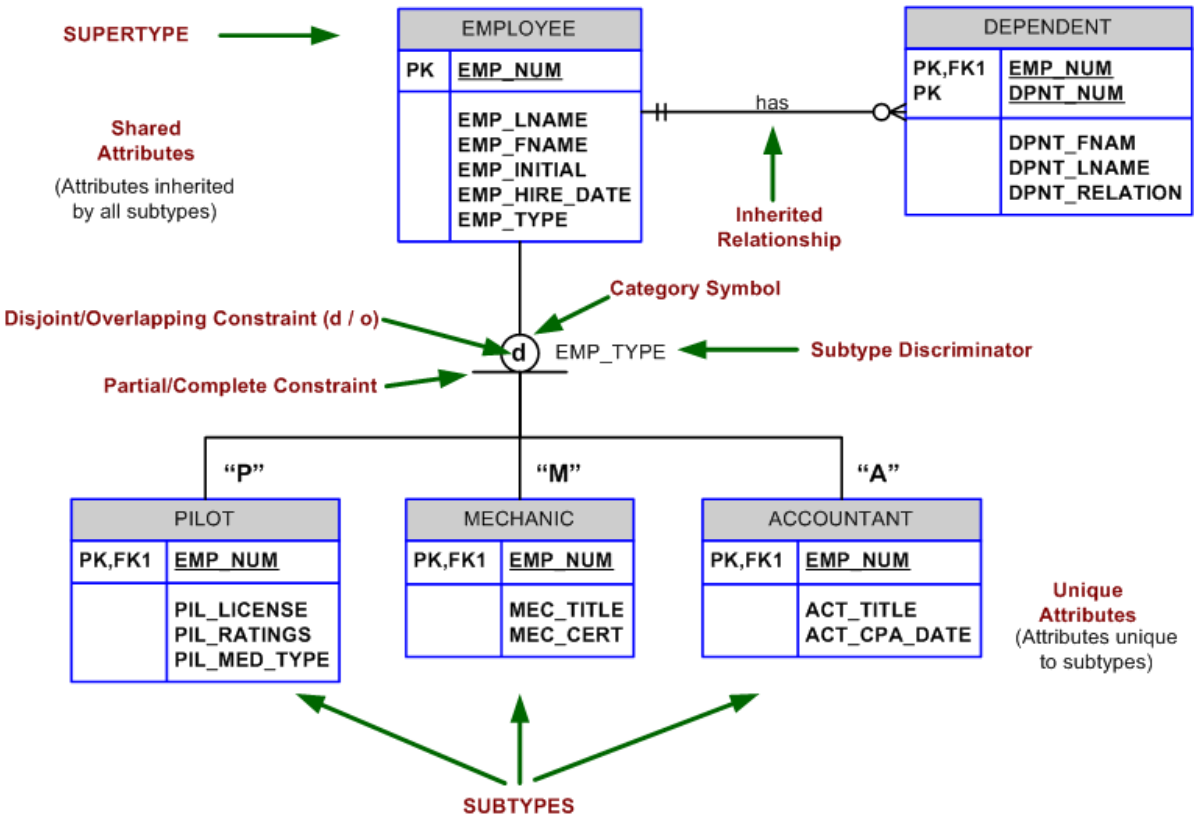
An entity supertype is a generic entity type that is related to one or more entity subtypes, where the entity supertype contains the common characteristics and the entity subtypes contain the unique characteristics of each entity subtype. The reason for using supertypes is to minimize the number of nulls and to minimize the likelihood of redundant relationships.

2. What kinds of data would you store in an entity subtype?

An entity subtype is a more specific entity type that is related to an entity supertype, where the entity supertype contains the common characteristics and the entity subtypes contain the unique characteristics of each entity subtype. The entity subtype will store the data that is specific to the entity; that is, attributes that are unique the subtype.

3. What is a specialization hierarchy?

A **specialization hierarchy** depicts the arrangement of higher-level entity supertypes (parent entities) and lower-level entity subtypes (child entities). To answer the question precisely, we have used the text's Figure 5.2. (We have reproduced the figure on the next page for your convenience.) Figure 5.2 shows the specialization hierarchy formed by an EMPLOYEE supertype and three entity subtypes—PILOT, MECHANIC, and ACCOUNTANT.



(Text) FIGURE 5.2 A Specialization Hierarchy

The specialization hierarchy shown in Figure 5.2 reflects the 1:1 relationship between EMPLOYEE and its subtypes. For example, a PILOT subtype occurrence is related to one instance of the EMPLOYEE supertype and a MECHANIC subtype occurrence is related to one instance of the EMPLOYEE supertype.

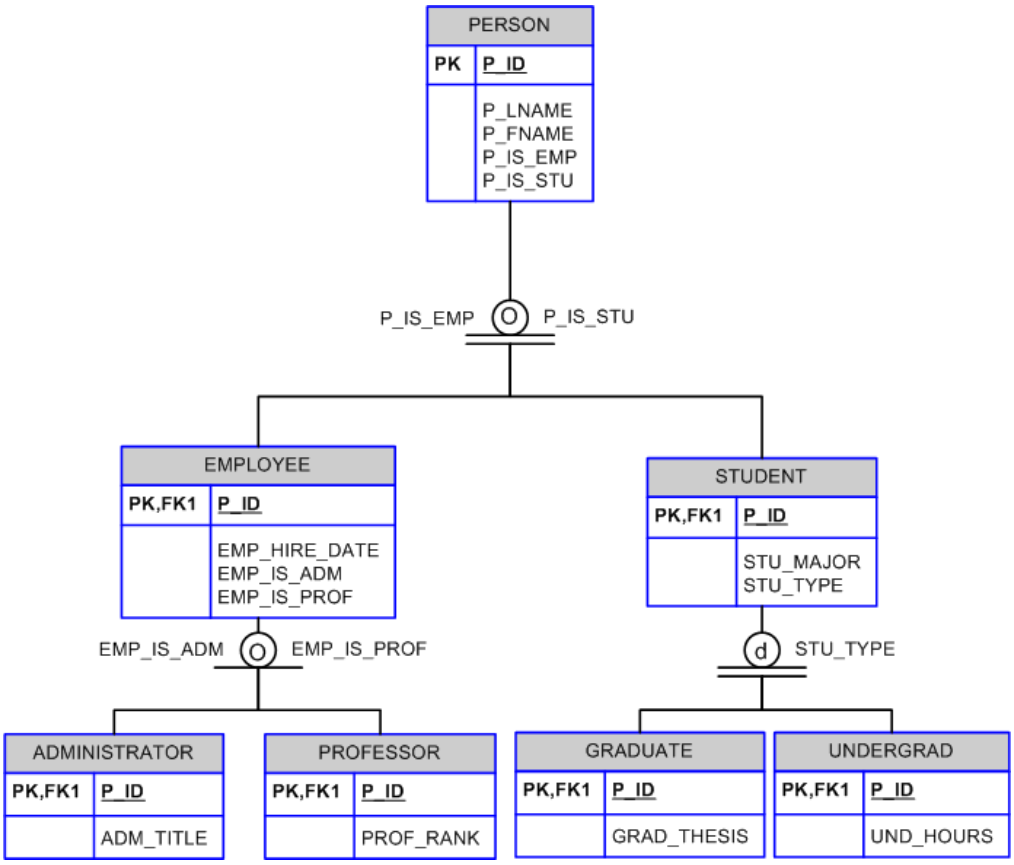
4. **What is a subtype discriminator? Given an example of its use.**

A subtype discriminator is the attribute in the supertype entity that is used to determine to which entity subtype the supertype occurrence is related. For any given supertype occurrence, the value of the subtype discriminator will determine which subtype the supertype occurrence is related to. For example, an EMPLOYEE supertype may include the EMP_TYPE value "P" to indicate the PROFESSOR subtype.

5. **What is an overlapping subtype? Give an example.**

Overlapping subtypes are subtypes that contain non-unique subsets of the supertype entity set; that is, each entity instance of the supertype may appear in more than one subtype. For example, in a university environment, a person may be an employee or a student or both. In turn, an employee may be a professor as well as an administrator. Because an employee also may be a student, STUDENT and EMPLOYEE are overlapping subtypes of the supertype PERSON, just as PROFESSOR and ADMINISTRATOR are overlapping subtypes of the supertype EMPLOYEE. The

text's Figure 5.4 (reproduced next for your convenience) illustrates overlapping subtypes with the use of the letter **O** inside the category shape.



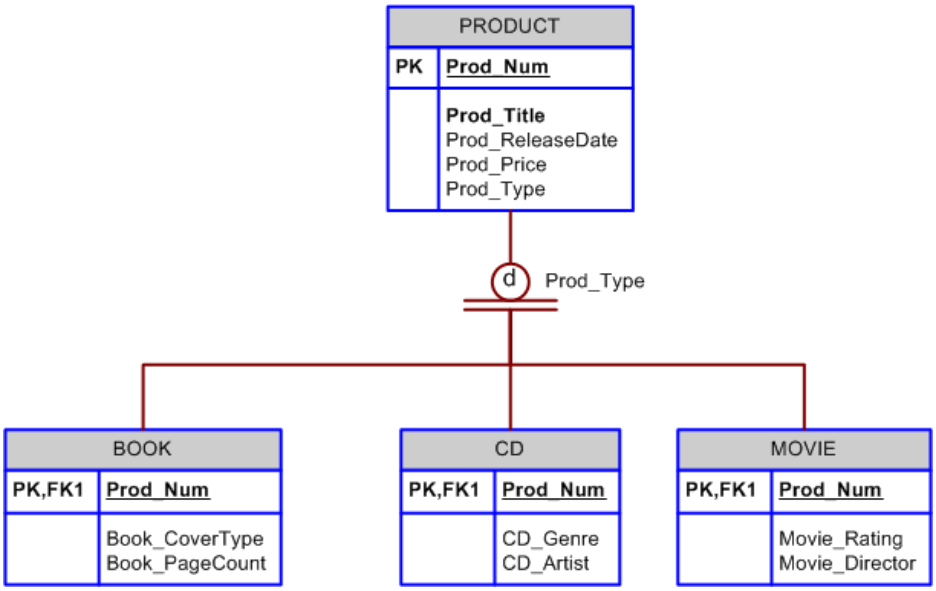
(Text) FIGURE 5.4 Specialization Hierarchy with Overlapping Subtypes

6. What is the difference between *partial completeness* and *total completeness*?

Partial completeness means that not every supertype occurrence is a member of a subtype; that is, there may be some supertype occurrences that are not members of any subtype. **Total completeness** means that every supertype occurrence must be a member of at least one subtype.

For questions 7 – 9, refer to Figure Q5.7

FIGURE Q5.7 The PRODUCT data model



7. List all of the attributes of a movie.

Recall that the subtype inherits all of the attributes and relationships of the supertype. Therefore, all of the attributes of a subtype include the common attributes from the supertype plus the unique (unique to that subtype) attributes from the subtype. All of the attributes of a movie would be:

- Prod_Num
- Prod_Title
- Prod_ReleaseDate
- Prod_Price
- Prod_Type
- Movie_Rating
- Movie_Director

8. According to the data model, is it required that every entity instance in the PRODUCT table be associated with an entity instance in the CD table? Why or why not?

No. The completeness constraint for the data model shows a total completeness constraint from PRODUCT to the subtypes. However, the total completeness constraint indicates that every instance in the supertype (PRODUCT) must be associated with one row in **some** subtype, not all subtypes. Since the subtypes are designated as disjoint, or exclusive, then every row in the supertype is associated a row

in only one subtype. For some products that subtype will be CD, but for other products the subtype will be either Movie or Book.

9. Is it possible for a book to appear in the BOOK table without appearing in the PRODUCT table? Why or why not?

No. Subtypes can only exist within the context of a supertype.

10. What is an *entity cluster,* and what advantages are derived from its use?

An entity cluster is a "virtual" entity type used to represent multiple entities and relationships in the ERD. An entity cluster is formed by combining multiple interrelated entities into a single abstract entity object. An entity cluster is considered "virtual" or "abstract" in the sense that it is not actually an entity in the final ERD, but rather a temporary entity used to represent multiple entities and relationships with the purpose of simplifying the ERD and thus enhancing its readability.

11. What primary key characteristics are considered desirable? Explain *why* each characteristic is considered desirable.

Desirable PK characteristics are summarized in the text's Table 5.3, reproduced below for your convenience. The table also includes the reason why each characteristic is desirable. (See the *Rationale* column.)

PK Characteristic	Rationale
Unique values	The PK must uniquely identify each entity instance. A primary key must be able to guarantee unique values. It cannot contain nulls.
Nonintelligent	The PK should not have embedded semantic meaning. An attribute with embedded semantic meaning is probably better used as a descriptive characteristic of the entity rather than as an identifier. In other words, a student ID of "650973" would be preferred over "Smith, Martha L." as a primary key identifier.
No change over time	If an attribute has semantic meaning, it may be subject to updates. This is why names do not make good primary keys. If you have "Vickie Smith" as the primary key, what happens when she gets married? If a primary key is subject to change, the foreign key values must be updated, thus adding to the database work load. Furthermore, changing a primary key value means that you are basically changing the identity of an entity.
Preferably single-attribute	A primary key should have the minimum number of attributes possible. Single-attribute primary keys are desirable but not required. Single-attribute primary keys simplify the implementation of foreign keys. Having multiple-attribute primary keys can cause primary keys of related entities to grow through the possible addition of many

	attributes, thus adding to the database work load and making (application) coding more cumbersome.
Preferably numeric	Unique values can be better managed when they are numeric because the database can use internal routines to implement a "counter-style" attribute that automatically increments values with the addition of each new row. In fact, most database systems include the ability to use special constructs, such as Autonumber in MS Access, to support self-incrementing primary key attributes.
Security complaint	The selected primary key must not be composed of any attribute(s) that might be considered a security risk or violation. For example, using a Social Security number as a PK in an EMPLOYEE table is not a good idea.

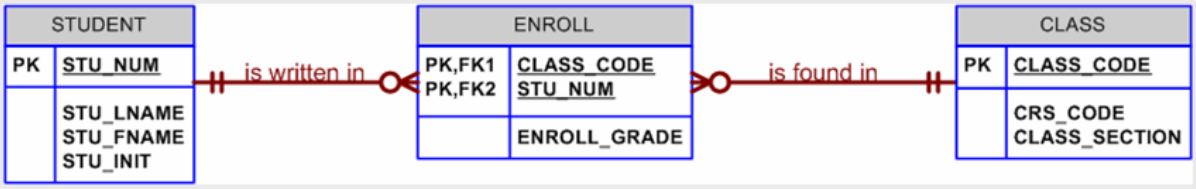
TABLE 5.3 Desirable Primary Key Characteristics

12. Under what circumstances are composite primary keys appropriate?

Composite primary keys are particularly useful in two cases:

- As identifiers of composite entities, where each primary key combination is allowed only once in the M:N relationship.
- As identifiers of weak entities, where the weak entity has a strong identifying relationship with the parent entity.

To illustrate the first case, assume that you have a STUDENT entity set and a CLASS entity set. In addition, assume that those two sets are related in a M:N relationship via an ENROLL entity set in which each student/class combination may appear only once in the composite entity. The text's Figure 5.6 (reproduced here for your convenience) shows the ERD to represent such a relationship.



Database name: Ch06_TinyCollege

Table name: STUDENT
(first four fields)

STU_NUM	STU_LNAME	STU_FNAME	STU_INIT
321452	Bowser	William	C
324257	Smithson	Anne	K
324258	Brewer	Juliette	
324269	Oblonski	Walter	H
324273	Smith	John	D
324274	Katinga	Raphael	P
324291	Robertson	Gerald	T
324299	Smith	John	B

Table name: ENROLL

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS
(first three fields)

CLASS_CODE	CRS_CODE	CLASS_SECTION
10012	ACCT-211	1
10013	ACCT-211	2
10014	ACCT-211	3
10015	ACCT-212	1
10016	ACCT-212	2
10017	CIS-220	1
10018	CIS-220	2
10019	CIS-220	3
10020	CIS-420	1
10021	QM-261	1
10022	QM-261	2
10023	QM-362	1
10024	QM-362	2
10025	MATH-243	1

(Text) FIGURE 5.6 M:N Relationship Between Student and Class

As shown in the text's Figure 5.6, the composite primary key automatically provides the benefit of ensuring that there cannot be duplicate values—that is, it ensures that the same student cannot enroll more than once in the same class.

In the second case, a weak entity in a strong identifying relationship with a parent entity is normally used to represent one of two cases:

1. *A real-world object that is existent dependent on another real-world object*. Those types of objects are distinguishable in the real world. A dependent and an employee are two separate people who exist independent of each other. However, such objects can exist in the model only when they relate to each other in a strong identifying relationship. For example, the relationship between EMPLOYEE and DEPENDENT is one of existence dependency in which the primary key of the dependent entity is a composite key that contains the key of the parent entity.
2. *A real-world object that is represented in the data model as two separate entities in a strong identifying relationship*. For example, the real-world invoice object is represented by two entities in a data model: INVOICE and LINE. Clearly, the LINE entity does not exist in the real world as an independent object, but rather as part of an INVOICE.

In both cases, having a strong identifying relationship ensures that the dependent entity can exist only when it is related to the parent entity. In summary, the selection of a composite primary key for composite and weak entity types provides benefits that enhance the integrity and consistency of the model.

13. What is a surrogate primary key, and when would you use one?

A surrogate primary key is an "artificial" PK that is used to uniquely identify each entity occurrence when there is no good natural key available or when the "natural" PK would include multiple attributes. A surrogate PK is also used if the natural PK would be a long text variable. The reason for using a surrogate PK is to ensure entity integrity, to simplify application development – by making queries simpler – to ensure query efficiency – for example, a query based on a simple numeric attribute is much faster than one based on a 200-bit character string -- and to ensure that relationships between entities can be created more easily than would be the case with a composite PK that may have to be used as a FK in a related entity.

14. When implementing a 1:1 relationship, where should you place the foreign key if one side is mandatory and one side is optional? Should the foreign key be mandatory or optional?

Section 5.4.1 provides a detailed discussion. The text's Table 5.5, reproduced here for your convenience, shows the rationale for selecting the foreign key in a 1:1 relationship based on the relationship properties in the ERD.

Case	ER Relationship Constraints	Action
I	One side is mandatory and the other side is optional.	Place the PK of the entity on the mandatory side in the entity on the optional side as a FK and make the FK mandatory.
II	Both sides are optional.	Select the FK that causes the fewest number of nulls or place the FK in the entity in which the (relationship) role is played.
III	Both sides are mandatory.	See Case II or consider revising your model to ensure that the two entities do not belong together in a single entity.

TABLE 5.5 Selection of Foreign Key in a 1:1 Relationship

15. What are time-variant data, and how would you deal with such data from a database design point of view?

As the label implies, time variant data are time-sensitive. For example, if a university wants to keep track of the history of all administrative appointments by date of appointment and date of termination, you see time-variant data at work.

16. What is the most common design trap, and how does it occur?

A design trap occurs when a relationship is improperly or incompletely identified and therefore, it is represented in a way that is not consistent with the real world. The most common design trap is known as a *fan trap*. A fan trap occurs when you have one entity

in two 1:M relationships to other entities, thus producing an association among the other entities that is not expressed in the model.