5.29 Define the three types of recursive binary relationships, and give an example of each, other than the ones shown in this text.

In the Chapter 4 Review Questions, we illustrated the three types of recursive models by using the example of an AGENT at the Pacific Northwest Real Estate Agency (PNREA), and we'll continue with that example. The three types of recursive binary relationships are the standard 1:1, 1:N, and N:M.

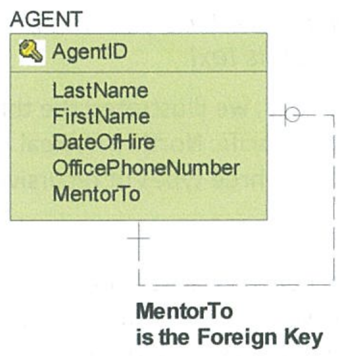
1:1 Recursive: Agents at the NWREA are assigned as a mentor to *one* other agent. Each NWREA mentor can only work with one agent at a time. Further, each agent is only mentored by one PNREA agent. However, although being mentored is required, being a mentor is not. Therefore, each agent mentor is optionally associated with exactly one other mentored agent.

1:N Recursive: The PNREA has changed the rules about mentoring. Agents at the NWREA may be assigned as a mentor to *one or more* other agents. Now each NWREA mentor can work with several agents at a time. However, each agent being mentored is still mentored by only one PNREA agent. The rule that being mentored is required, although being a mentor is not, still applies. Therefore, each agent mentor is optionally associated with many other mentored agents.

N:M Recursive; If agents in the community where PNREA is located want to acquire property themselves, they are required to use another agent (not necessarily at PNREA) to act as the selling agent on the deal. If an agent makes several purchases, he or she can use a different selling agent for each deal. Further that agent may be the selling agent in other deals. However, no agent is required to buy property, so working with a selling agent is optional. Similarly, no agent is required to be the selling agent in this type of a transaction. Therefore, each agent is optionally associated with many other selling agents, and may optionally be a selling agent to many other agents.

5.30 Show how to represent the 1:1 recursive relationship in your answer to question 5.29. How does this differ from the representation of 1:1 nonrecursive relationships?

For the 1:1 recursive relationship, we can place one of two new columns in AGENT—MentorTo or MentoredBy—as a foreign key. In this case MentoredTo will be used. Note that this limits us to a 1:1 relationship. The column MentoredBy will allow a 1:N recursive relationship, and we'll use it in Question 5.32



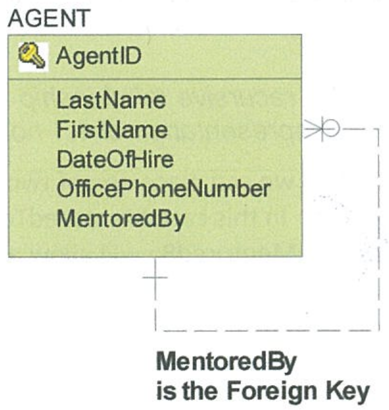
This is really not that different, we're placing the primary key of one row into another row as a foreign key; it's just in the same table rather than in a different table. With a 1:1 relationship, the foreign key may appear in only one row.

5.31 *Code an SQL statement that creates a table with all columns from the parent and child tables in your answer to question 5.30.*

```
SELECT    *
FROM      AGENT AS A, AGENT AS M
WHERE     A.AgentID = M.MentorTo;
```

5.32 *Show how to represent a 1:N recursive relationship in your answer to question 5.29. How does this differ from the representation of 1:N nonrecursive relationships?*

For the 1:N recursive relationship, place the foreign key MentorBy into AGENT. MentoredBy has the AgentID of the agent who does the mentoring.



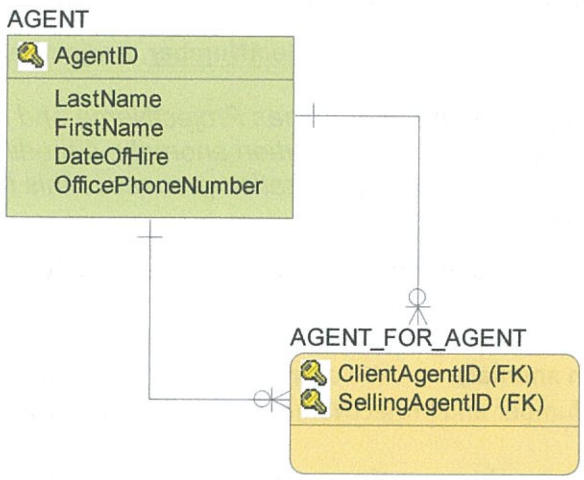
This is really not that different. We are placing the primary key of one row into another row as a foreign key; it's just in the same table rather than in a different table. With a 1:N relationship, the foreign key may appear in one or more rows.

5.33 *Code an SQL statement that creates a table with all columns from the parent and child tables in your answer to question 5.32.*

```
SELECT   *
FROM     AGENT AS A, AGENT AS M
WHERE    A.AgentID = M.MentoredBy;
```

5.34 *Show how to represent the M:N recursive relationship in your answer to question 5.29. How does this differ from the representation of M:N nonrecursive relationships?*

For the M:N recursive relationship, create an intersection table, say, AGENT_FOR_AGENT, with ClientAgentID and SellingAgentID as its columns. ClientAgentID contains the AgentID of the agent buying the property, and SellingAgentID contains the AgentID of the agent acting as the selling agent. There may be several rows in the intersection table for a given agent.



This is basically the same technique used for any intersection table relation used for any M:N relationships. The only difference is that values in the intersection table come from only one source relation.

5.35 *Code an SQL statement that creates a table with all columns from the parent and child tables in your answer to question 5.34. Code an SQL statement using a left outer join that creates a table with all columns from the parent and child tables. Explain the difference between these two SQL statements.*

The basic SQL join of the two tables is:

```
SELECT   *
FROM     AGENT AS C, AGENT_FOR_AGENT as AFA, AGENT AS S
WHERE    C.AgentID = AFA.ClientAgentID
   AND   AFA.SellingID = S.AgentID
```

The SQL left outer join of the two tables is:

```
SELECT    *
FROM      AGENT AS C LEFT JOIN AGENT_FOR_AGENT as AFA
          ON C.AgentID = AFA.ClientAgentID
             JOIN AGENT AS S
                ON AFA.SellingID = S.AgentID;
```

The first query will only show agents who have purchased property. The second query statement will show all agents, regardless of whether or not they have purchased property.

ANSWERS TO EXERCISES

5.36 *Consider the following table, which holds data about employee project assignments:*

ASSIGNMENT (<u>EmployeeNumber</u>, <u>ProjectNumber</u>, ProjectName, HoursWorked)

Assume that ProjectNumber determines ProjectName and explain why this relation is not normalized. Demonstrate an insertion anomaly, a modification anomaly, and a deletion anomaly. Apply the normalization process to this relation. State the referential integrity constraint.

The ASSIGN relation is not normalized because there is a determinant, ProjectNumber, which is not a candidate key.

- **Insertion anomaly:** We cannot record the correspondence between a ProjectNumber and ProjectName until we have at least one assignment for that project.
- **Update anomaly:** If a project changes its name, there are potentially many rows that will have to be updated.
- **Deletion anomaly:** Deleting the last assignment for a project will lose the correspondence between ProjectNumber and ProjectName.

Normalized relations:

ASSIGN (<u>EmployeeNumber</u>, <u>*ProjectNumber*</u>, HoursWorked)

PROJECT (<u>ProjectNumber</u>, ProjectName)

where

ProjectNumber in ASSIGN must exist in ProjectNumber in PROJECT.