implementation to define. Note that A1, A2, and so forth may be individuals, like John in IT department or Mary in marketing; but they may also be applications or programs that want to access a database.

In SQL2, the same effect can be accomplished by having the DBA issue a CREATE SCHEMA command, as follows:

```sql
CREATE SCHEMA EXAMPLE AUTHORIZATION A1;
```

User account A1 can now create tables under the schema called EXAMPLE. To continue our example, suppose that A1 creates the two base relations EMPLOYEE and DEPARTMENT shown in Figure 30.1; A1 is then the owner of these two relations and hence has all the relation privileges on each of them.

Next, suppose that account A1 wants to grant to account A2 the privilege to insert and delete tuples in both of these relations. However, A1 does not want A2 to be able to propagate these privileges to additional accounts. A1 can issue the following command:

```sql
GRANT INSERT, DELETE ON EMPLOYEE, DEPARTMENT TO A2;
```

Notice that the owner account A1 of a relation automatically has the GRANT OPTION, allowing it to grant privileges on the relation to other accounts. However, account A2 cannot grant INSERT and DELETE privileges on the EMPLOYEE and DEPARTMENT tables because A2 was not given the GRANT OPTION in the preceding command.

Next, suppose that A1 wants to allow account A3 to retrieve information from either of the two tables and also to be able to propagate the SELECT privilege to other accounts. A1 can issue the following command:

```sql
GRANT SELECT ON EMPLOYEE, DEPARTMENT TO A3 WITH GRANT OPTION;
```

The clause WITH GRANT OPTION means that A3 can now propagate the privilege to other accounts by using GRANT. For example, A3 can grant the SELECT privilege on the EMPLOYEE relation to A4 by issuing the following command:

```sql
GRANT SELECT ON EMPLOYEE TO A4;
```

Notice that A4 cannot propagate the SELECT privilege to other accounts because the GRANT OPTION was not given to A4.

Now suppose that A1 decides to revoke the SELECT privilege on the EMPLOYEE relation from A3; A1 then can issue this command:

```sql
REVOKE SELECT ON EMPLOYEE FROM A3;
```
must now revoke the SELECT privilege on EMPLOYEE from A3, and it *automatically revoke* the SELECT privilege on EMPLOYEE from A4. Because A3 granted that privilege to A4, but A3 does not have the privilege

pose that A1 wants to give back to A3 a limited capability to SELECT from EMPLOYEE relation and wants to allow A3 to be able to propagate the privilege. This is to retrieve only the Name, Bdate, and Address attributes and only those with Dno = 5. A1 then can create the following view:

```
CREATE VIEW A3EMPLOYEE AS
  SELECT Name, Bdate, Address
  FROM EMPLOYEE
  WHERE Dno = 5;
```

View is created, A1 can grant SELECT on the view A3EMPLOYEE to A3

```
GRANT SELECT ON A3EMPLOYEE TO A3 WITH GRANT OPTION;
```

pose that A1 wants to allow A4 to update only the Salary attribute of A1 can then issue the following command:

```
UPDATE ON EMPLOYEE (Salary) TO A4;
```

E and INSERT privileges can specify particular attributes that may be inserted in a relation. Other privileges (SELECT, DELETE) are not attributes, because this specificity can easily be controlled by creating the appropriate views that include only the desired attributes and granting the corresponding privileges on the views. However, because updating views is not always possible (see the UPDATE and INSERT privileges are given the option to specify the attributes of a base relation that may be updated.

Specifying Limits on Propagation of Privileges

to limit the propagation of privileges have been developed, although not yet been implemented in most DBMSs and are not a part of SQL. Horizontal propagation to an integer number *i* means that an account *B* can grant the privilege to at most *i* other accounts. Vertical propagation is more complicated; it limits the depth of the granting of privilege with a vertical propagation of zero is equivalent to privilege with no GRANT OPTION. If account *A* grants a privilege to *B* with the vertical propagation set to an integer number *j* > 0, this means *B* has the GRANT OPTION on that privilege, but *B* can grant the privilege only with a vertical propagation less than *j*. In effect, propagation limits the sequence of GRANT OPTIONS that can be given from to the next based on a single original grant of the privilege.

Illustrate horizontal and vertical propagation limits—which are not currently in SQL or other relational systems—with an example. Suppose