第3章 创建表和视图

本章描述了如何才能在 DB2 通用数据库中创建和操作表和视图。读者可通过图表和示例来研究表和视图的关系。

本章包括:

- 创建表和创建视图
- 插入数据
- 更改数据
- 删除数据
- 使用视图来处理数据

创建表

使用 CREATE TABLE 语句创建自己的表，指定列名和类型以及*约束*。约束在第51页的『用约束和触发器实施商业规则』中进行讨论。

下列语句创建一个名称为 PERS 的表，该表与 STAFF 表类似，但有一个附加列 BIRTH_DATE。

```
CREATE TABLE PERS
    ( ID              SMALLINT        NOT NULL,
      NAME            VARCHAR(9),
      DEPT            SMALLINT WITH DEFAULT 10,
      JOB             CHAR(5),
      YEARS           SMALLINT,
      SALARY          DECIMAL(7,2),
      COMM            DECIMAL(7,2),
      BIRTH_DATE      DATE)
```

此语句创建一个不包含数据的表。下一节将描述如何将数据插入新表。

如示例中所示，对每一列都指定了名称和数据类型。数据类型在第4页的『数据类型』中进行讨论。 NOT NULL 是可选的，可以指定它以表示列中不允许有空值。缺省值也是可选的。

可以在 CREATE TABLE 语句中指定许多其他选项，如*唯一约束*或*参考约束*。有关所有选项的详情，参见 *SQL Reference* 中的 CREATE TABLE 语句。

插入数据

当创建新表时，新表不包含任何数据。要将新的行输入表中，使用 INSERT 语句。此语句有两种通用格式:

- 通过第一种格式，您可使用 VALUES 子句来指定一行或多行的列值。以下三个示例均使用此通用格式将数据插入表中。
- 通过另一种格式，您可指定全*查询*而不是指定 VALUES，来标识其他表和 / 或视图中的行的列。

全查询是 INSERT 或 CREATE VIEW 语句中所使用的选择语句、或者是跟在谓词后面的选择语句。括在括号中的全查询通常称为*子查询*。

根据创建表时已选择的缺省选项，对于每个插入的行，为每一列提供一个值或者接受一个缺省值。各种数据类型的缺省值在 *SQL Reference* 中进行讨论。

下列语句使用 VALUES 子句将一行数据插入 PERS 表中:

```
INSERT INTO PERS
   VALUES (12, 'Harris', 20, 'Sales', 5, 18000, 1000, '1950-1-1')
```

以下语句使用 VALUES 子句将三行插入 PERS 表中，此表中仅 ID、姓名以及工作是已知的。如果列定义为 NOT NULL 且没有缺省值，则必须为该列指定一个值。

CREATE TABLE 语句中的列定义上的 NOT NULL 子句可以用单词 WITH DEFAULT 扩充。如果某一列定义为 NOT NULL WITH DEFAULT 或常数缺省值（如 WITH DEFAULT 10），并且您未在列列表中指定该列，则缺省值插入至已插入行的该列中。例如，在 CREATE TABLE 语句中，仅为 DEPT 列指定了缺省值并将该值定义为 10。因此，部门号 (DEPT) 设置为 10，而任何其他未显式给定值的列都设置为空。

```
INSERT INTO PERS (NAME, JOB, ID)
   VALUES ('Swagerman', 'Prgmr', 500),
          ('Limoges', 'Prgmr', 510),
          ('Li', 'Prgmr', 520)
```

下列语句返回插入的结果:

```
SELECT *
   FROM PERS

ID     NAME      DEPT   JOB   YEARS  SALARY    COMM      BIRTH_DATE
------ --------- ------ ----- ------ --------- --------- ----------
    12 Harris        20 Sales      5  18000.00   1000.00 01/01/1950
```

```
500 Swagerman     10 Prgmr      -         -         - -
510 Limoges       10 Prgmr      -         -         - -
520 Li            10 Prgmr      -         -         - -
```

注意，在此情况下，并未给每个列指定值。空值显示为破折号 (–)。为此，列名列表的次序和数据类型都必须与 VALUES 子句中提供的值对应。如果省略列名列表（如第一个示例中那样），则 VALUES 之后的数据值列表的次序必须与它们所插入至的表中的列次序相同，值的数目必须等于表中列的数目。

每个值必须与它所插入至的列的数据类型相容。如果某列定义为可空，且未指定该列的值，则将空值赋给插入行中的该列。

因为未给行中的那些列指定值，所以下列示例将空值插入 YEARS、COMM 和 BIRTH_DATE 中。

```
INSERT INTO PERS (ID, NAME, JOB, DEPT, SALARY)
   VALUES (410, 'Perna', 'Sales', 20, 20000)
```

INSERT 语句的第二种格式对于用另一表中的行的值来填充表非常方便。如上所述，指定全查询而不是指定 VALUES，以标识其他表和／或视图中的行中的列。

下列示例从 STAFF 表中选择部门 38 的成员的数据，并将它插入 PERS 表中:

```
INSERT INTO PERS (ID, NAME, DEPT, JOB, YEARS, SALARY)
   SELECT ID, NAME, DEPT, JOB, YEARS, SALARY
      FROM STAFF
      WHERE DEPT = 38
```

在此插入之后，下列 SELECT 语句与 INSERT 语句中全查询产生的结果相同。

```
SELECT ID, NAME, DEPT, JOB, YEARS, SALARY
   FROM PERS
   WHERE DEPT = 38
```

结果为:

```
ID     NAME      DEPT   JOB   YEARS  SALARY
------ --------- ------ ----- ------ ---------
    30 Marenghi      38 Mgr        5  17506.75
    40 O'Brien       38 Sales      6  18006.00
    60 Quigley       38 Sales      -  16808.30
   120 Naughton      38 Clerk      -  12954.75
   180 Abrahams      38 Clerk      3  12009.75
```

更改数据

使用 UPDATE 语句来更改表中的数据。使用此语句，可以更改满足 WHERE 子句搜索条件的每行中的一列或多列的值。

下列示例更新 ID 为 410 的雇员的信息:

```
UPDATE PERS
   SET JOB='Prgmr', SALARY = SALARY + 300
   WHERE ID = 410
```

SET 子句指定要更新的列并提供值。

WHERE 子句是可选的，它指定要更新的行。如果省略 WHERE 子句，则数据库管理程序用您提供的值更新表或视图中的每一行。

在此示例中，首先命名表 (PERS)，然后指定要更新行的条件。员工号码 410 的信息已更改: 该雇员的工作职位更改为 Prgmr，工资增加了 $300。

可以通过包括应用于两行或更多行的 WHERE 子句来更改多行数据。下列示例给每个销售员的工资增加 15%:

```
UPDATE PERS
   SET SALARY = SALARY * 1.15
   WHERE JOB = 'Sales'
```

删除数据

使用 DELETE 语句，根据在 WHERE 子句中指定的搜索条件从表中删除数据行。下列示例删除其中雇员 ID 为 120 的行:

```
DELETE FROM PERS
   WHERE ID = 120
```

WHERE 子句是可选的，它指定要删除的行。如果省略 WHERE 子句，则数据库管理程序删除表或视图中的所有行。

可以使用 DELETE 语句删除多行。下列示例删除其中雇员部门 (DEPT) 为 20 的所有行:

```
DELETE FROM PERS
   WHERE DEPT = 20
```

当删除某一行时，是除去整个行，而不是除去该行中的特定列值。

要删除表的定义及其内容，可发出 DROP TABLE 语句，如 *SQL Reference* 中所述。

创建视图

如在第3页的『视图』中所讨论的那样，视图提供在一个或多个表中查看数据的替代方法。通过创建视图，可以对想让各种用户查看的信息进行限制。下列图表显示视图和表之间的关系。

在图2中，View_A 被限制为只可存取 TABLE_A 的列 AC1 和 AC2。

View_AB 允许存取 TABLE_A 中的列 AC3 和 TABLE_B 中的列 BC2。

通过创建 View_A，将用户可以具有的存取权限制为 TABLE_A，而通过创建 VIEW_AB，则是将存取权限制为两个表中的某些列。

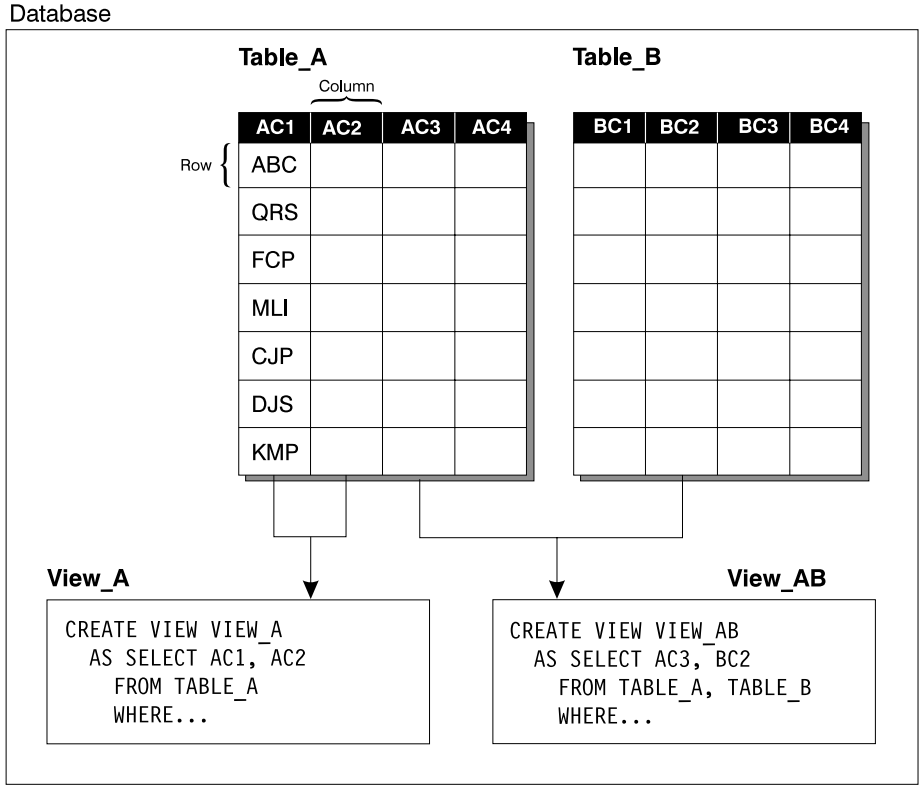

图 2. 表和视图之间的关系

下列语句创建 STAFF 表中部门 20 内的非经理人员的视图，其中工资和佣金不通过基表显示。

```
CREATE VIEW STAFF_ONLY
   AS SELECT ID, NAME, DEPT, JOB, YEARS
         FROM STAFF
         WHERE JOB <> 'Mgr' AND DEPT=20
```

在创建视图之后，下列语句显示视图的内容:

```
SELECT *
   FROM STAFF_ONLY
```

此语句产生下列结果:

```
ID     NAME      DEPT   JOB   YEARS
------ --------- ------ ----- ------
    20 Pernal        20 Sales      8
    80 James         20 Clerk      -
   190 Sneider       20 Clerk      8
```

作为一个更深入的示例，我们可使用 STAFF 和 ORG 表来创建一个列出每个部门名称和部门经理姓名的视图。 以下语句创建此视图:

```
CREATE VIEW DEPARTMENT_MGRS
   AS SELECT NAME, DEPTNAME
         FROM STAFF, ORG
         WHERE MANAGER = ID
```

创建视图时，可以使用 WITH CHECK OPTION 子句，将附加约束添加到通过视图对表进行的插入和更新。此子句导致数据库管理程序验证对视图的任何更新或插入是否符合该视图的定义，并拒绝那些不符合定义的更新或插入。如果省略此子句，则不检查违反视图定义的插入和更新。有关 WITH CHECK OPTION 如何起作用的详情，参考 *SQL Reference* 中的 CREATE VIEW 语句。

使用视图来处理数据

象 SELECT 语句一样，INSERT、DELETE 以及 UPDATE 语句应用于视图，就好象视图是一个实表那样。这些语句处理基本基表中的数据。因此当再次存取该视图时，使用最新的基表对它进行计算。如果不使用 WITH CHECK OPTION 子句，则使用视图修改的数据可能不在视图的重复存取中出现，原因是该数据可能不再符合原来的视图定义。

以下示例将更新应用于视图 FIXED_INCOME ：

```
CREATE VIEW FIXED_INCOME (LNAME, DEPART, JOBTITLE, NEWSALARY)
   AS SELECT NAME, DEPT, JOB, SALARY
         FROM PERS
         WHERE JOB <> 'Sales' WITH CHECK OPTION

UPDATE FIXED_INCOME
   SET NEWSALARY = SALARY * 1.10
   WHERE LNAME = 'Li'
```

除了校验选项以外，先前视图中的更新等效于对基表 PERS 的更新:

```
UPDATE PERS
   SET SALARY = SALARY * 1.10
   WHERE NAME = 'Li'
     AND JOB <> 'Sales'
```

注意，由于视图是在 CREATE VIEW FIXED_INCOME 中对约束 JOB <> 'Sales' 使用 WITH CHECK OPTION 创建的，所以当 Limoges 调去做销售时不允许下列更新:

```
UPDATE FIXED_INCOME
   SET JOBTITLE = 'Sales'
   WHERE LNAME = 'Limoges'
```

不能更新由表达式 SALARY + COMM 或 SALARY * 1.25 定义的列。如果定义一个包含一个或多个这样的列的视图，则拥有者不具有对这些列的 UPDATE 特权。在包含这样的列的视图上不允许 INSERT 语句，但允许 DELETE 语句。

现在我们讨论一个没有任何一列定义为 NOT NULL 的表 PERS。可以通过 FIXED_INCOME 视图将行插入表 PERS 中，即使该视图不包含基本表 PERS 的 ID、YEARS、COMM 或 BIRTHDATE。整个视图中看不到的列被适当地设置为空值或缺省值。

但是表 PERS 确实已将列 ID 定义为 NOT NULL。如果尝试通过 FIXED_INCOME 视图插入行，则系统试图将空值插入在整个视图中“看不到”的所有 PERS 列。由于 ID 列未包括在视图中并且该列不允许空值，所以系统不允许通过该视图进行插入。

有关修改视图的规则和限制，参考 *SQL Reference* 中的 CREATE VIEW 语句。