ROLLUP 分组产生包含常规分组行和*小计行*的结果集。 CUBE 分组产生包含来自 ROLLUP 和交叉制表行中的行的结果集。

所以对于 ROLLUP，可获取每人每月的销售额及每月的销售总额，以及全年销售总额。对于 CUBE，将包括每人销售总额的附加行。

参见 *SQL Reference* 以了解详情。

递归查询

*递归查询*是迭代使用结果数据来确定进一步结果的查询。可以把这想象成在一棵树上或一幅图中来回移动。

实际示例包括保留系统、网络规划和调度。

递归查询是使用包括引用自己名称的的公共表表达式来编写的。

参见 *SQL Reference* 以获取递归查询的示例。

OLAP 函数

联机分析处理 (OLAP) 函数对一*窗口*数据执行列函数操作。此窗口可指定分区内的行分区、行排序或*聚合组*。聚合组允许用户指定参与计算的行（相对于当前行）。使用这样的窗口允许象计算累计总和及滑动平均值这样的操作。

除允许用户为现存列函数（如 SUM 和 AVG ）指定窗口外， OLAP 函数还可执行排序（RANK 和 DENSE_RANK）操作，并提供行号编制 (ROW_NUMBER)，给出特定的行分区和行排序。

以下示例查询根据工资给出了部门内的雇员级别，并显示部门（对于第 15 部门和第 38 部门）内的工资的累计总和:

```
SELECT NAME, DEPT,
       RANK () OVER (PARTITION BY DEPT ORDER BY SALARY DESC) AS RANK,
       SUM (SALARY) OVER (PARTITION BY DEPT
                          ORDER BY SALARY DESC
                          ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW)
           AS CUMULATIVE_SUM
  FROM STAFF
  WHERE DEPT IN (15,38)
  ORDER BY DEPT, RANK
```

此语句产生下列结果:

```
NAME       DEPT      RANK       CUMULATIVE_SUM
---------- --------- ---------- --------------------
Hanes      15                 1              20659.80
Rothman    15                 2              37162.63
Ngan       15                 3              49670.83
Kermisch   15                 4              61929.33
O'Brien    38                 1              18006.00
Marenghi   38                 2              35512.75
Quigley    38                 3              52321.05
Naughton   38                 4              65275.80
Abrahams   38                 5              77285.55
```

第8章 定制和增强数据操作

本章简要介绍了 DB2 通用数据库中的*面向对象的扩充*。使用面向对象的扩充有很多优点。*用户定义类型 (UDT)* 增加了可用于应用程序的数据类型集，而*用户定义函数 (UDF)* 允许创建应用程序特定的函数。 UDF 通过提供类型的一致行为和封装来充当 UDT 的*方法*。

然后讨论*专用寄存器*和*系统目录*。专用寄存器是数据库管理程序定义的存储区；它们用来存储 SQL 语句可引用的信息。专用寄存器是在连接时建立的，且专门用于该应用程序的处理。系统目录包含关于数据库对象的逻辑结构和物理结构的信息。

本章包括:

- 用户定义类型
- 用户定义函数
- 大对象 (LOB)
- 专用寄存器
- 目录视图的介绍

上面主题的详细讨论超出了本书的范围，但在 *SQL Reference* 和*管理指南*中给出了详细讨论。

用户定义类型

*单值类型*是与现存类型（它的“源”类型）共享其内部表示的用户定义数据类型，但对于大多数运算来说，认为单值类型是独立和不兼容的。例如，您可能想定义年龄类型、重量类型以及高度类型，所有这些类型都有相当不同的语义，但都使用内部数据类型 INTEGER 作为它们的内部表示。

下列示例说明了命名为 PAY 的单值类型的创建:

```
CREATE DISTINCT TYPE PAY AS DECIMAL(9,2) WITH COMPARISONS
```

虽然 PAY 有与内部数据类型 DECIMAL(9,2) 相同的表示，但还是认为它是与 DECIMAL(9,2) 或任何其他类型不可比的独立类型。它只能与相同的单值类型比较。并且，会影响到按 DECIMAL 使用的运算符和函数将在此不适用。例如，具有 PAY 数据类型的值不能与具有 INTEGER 数据类型的值相乘。因此，您必须编写只应用于 PAY 数据类型的函数。

使用单值数据类型可限制偶然错误。例如，如果将表 EMPLOYEE 的 SALARY 列定义为 PAY 数据类型，则不能将该列添加至 COMM，即使它们的源类型相同。

单值数据类型支持类型转换。源类型可以转换为单值数据类型，单值数据类型也可以转换为源类型。例如，如果将表 EMPLOYEE 的 SALARY 列定义为 PAY 数据类型，则下列示例将不会在比较运算符处出错。

```
SELECT * FROM EMPLOYEE
   WHERE DECIMAL(SALARY) = 41250
```

DECIMAL(SALARY) 返回一个十进制数据类型。相反地，数字数据类型可以转换为 PAY 类型。例如，可以使用 PAY(41250) 来转换数字 41250 的类型。

用户定义函数

如第27页的『使用函数』中所述，DB2 通用数据库提供内部函数和用户定义函数 (UDF)。然而，此函数集从不会满足所有需求。您常常需要为特别的任务创建定制函数。用户定义函数允许您创建定制函数。

有四种类型的用户定义函数: *有源*（或模板）、*外部标量*，*外部表*和 *OLE DB 外部表*。

本节涉及有源类型和外部标量类型。有关外部表类型和 OLE DB 表类型的详情，参见*SQL Reference*。

源用户定义函数允许用户定义类型有选择地引用另一个已为数据库所知的内部函数或用户定义函数。您可以既使用标量函数又使用列函数。

在下一个示例中，用户定义函数（称为 MAX）是根据内部 MAX 列函数创建的，该内部 MAX 列函数采用 DECIMAL 数据类型作为输入。 MAX UDF 采用 PAY 类型作为输入且返回一个 PAY 类型作为输出。

```
CREATE FUNCTION MAX(PAY) RETURNS PAY
   SOURCE MAX(DECIMAL)
```

外部用户定义函数由用户用程序设计语言编写。有*外部标量函数*和*外部表函数*，在 *SQL Reference* 中讨论了这两个函数。

又例如，假定您已编写了一个计算字符串中字数的函数，则您可以使用 CREATE FUNCTION 语句以名称 WORDCOUNT 向数据库注册该函数。然后就可在 SQL 语句中使用此函数。

以下语句返回雇员号和他们简历的 ASCII 格式的字数。WORDCOUNT 是用户已向数据库注册并且现正在语句中使用的外部标量函数。

```
SELECT EMPNO, WORDCOUNT(RESUME)
   FROM EMP_RESUME
   WHERE RESUME_FORMAT = 'ascii'
```

有关编写用户定义函数的详情，参考 *Application Development Guide*。

大对象 (LOB)

术语*大对象*及其缩写词 *LOB* 用于表示三种数据类型：BLOB、CLOB 或 DBCLOB。这些类型可以包含诸如音频、图片以及文档等对象的大量数据。

二进制大对象 (BLOB) 是变长字符串，以字节进行量度，最大长度可达 2G 字节。BLOB 主要用来保存非传统数据，如图片、声音以及混合媒体等。

字符大对象 (CLOB) 是变长字符串，以字节进行量度，最大长度可达 2G 字节。CLOB 用于存储大的单字节字符集数据，如文档等。CLOB 被认为是字符串。

双字节字符大对象 (DBCLOB) 是最大长度可达 2G 字节的双字节字符变长字符串（1 073 741 823 个双字节字符）。 DBCLOB 用于存储大的双字节字符集数据，如文档等。DBCLOB 被认为是图形字符串。

操作大对象 (LOB)

由于 LOB 值可以很大，所以将它们从数据库服务器传送至客户机应用程序可能要花费一些时间。然而，一般一次处理 LOB 值的一部分，而不是将它们作为一个整体处理。对于应用程序不需要（或不想要）将整个 LOB 值存储在应用程序内存中的那些情况，应用程序可以通过*大对象定位器*变量引用此值。

然后后续语句可以使用定位器对数据执行操作，而不必检索整个大对象。定位器变量用来减少应用程序的存储器需求，并通过减少客户机与服务器之间的数据流而改进性能。

另一个机制是*文件引用*变量。它们用来直接对文件检索大对象或直接从文件来更新表中的大对象。文件引用变量用来减少应用程序的存储器需求，因为这些变量不必存储大对象数据。有关详情，参考 *Application Development Guide* 和 *SQL Reference*。

专用寄存器

*专用寄存器*是由数据库管理程序为连接定义的存储区，用于存储可以用 SQL 语句引用的信息。下列是几个较常用的专用寄存器的示例。有关所有专用寄存器的列表及详情，参考 *SQL Reference*。

- CURRENT DATE: 保存对应于执行 SQL 语句时的日时钟的日期。
- CURRENT FUNCTION PATH: 保存一个指定用于分解函数和数据类型引用的函数路径的值。
- CURRENT SERVER: 指定当前应用程序服务器。
- CURRENT TIME: 保存对应于执行 SQL 语句时的日时钟的时间。
- CURRENT TIMESTAMP: 指定对应于执行 SQL 语句时的日时钟的时间戳记。
- CURRENT TIMEZONE: 指定世界时与应用程序服务器本地时间之间的差别。
- USER: 指定运行期权限 ID。

您可以用 VALUES 语句显示专用寄存器的内容。例如:

```
VALUES (CURRENT TIMESTAMP)
```

也可以使用:

```
SELECT CURRENT TIMESTAMP FROM ORG
```

这将为表中的每一行项返回 TIMESTAMP。

目录视图的介绍

DB2 为每个数据库创建并维护一个系统目录表扩充集。这些表包含关于数据库对象（如表、视图、程序包、参考完整性关系、函数、单值类型以及触发器）的逻辑结构和物理结构的信息。这些表是在创建数据库时创建的，并在正常操作过程中得到更新。不能显式创建或卸下它们，但可以查询和查看其内容。

有关详情，参考 *SQL Reference*。

从系统目录中选择行

目录视图象任何其他数据库视图一样。可以使用 SQL 语句来查看数据，确切地说，就是使用与查看系统中任何其他视图的相同方式来查看数据。

可以在 SYSCAT.TABLES 目录中找到关于表的非常有用的信息。要查找已创建的现存表的名称，发出一个类似以下语句的语句:

```
SELECT TABNAME, TYPE, CREATE_TIME
   FROM SYSCAT.TABLES
   WHERE DEFINER = USER
```

此语句产生下列结果:

```
TABNAME             TYPE CREATE_TIME
------------------- ---- --------------------------
ORG                 T    1999-07-21-13.42.55.128005
STAFF               T    1999-07-21-13.42.55.609001
DEPARTMENT          T    1999-07-21-13.42.56.069001
EMPLOYEE            T    1999-07-21-13.42.56.310001
EMP_ACT             T    1999-07-21-13.42.56.710001
PROJECT             T    1999-07-21-13.42.57.051001
EMP_PHOTO           T    1999-07-21-13.42.57.361001
EMP_RESUME          T    1999-07-21-13.42.59.154001
SALES               T    1999-07-21-13.42.59.855001
CL_SCHED            T    1999-07-21-13.43.00.025002
IN_TRAY             T    1999-07-21-13.43.00.055001
```

下列列表包括与本书讨论的与主题有关的目录视图。还有很多其他目录视图，在 *SQL Reference*和*管理指南*手册中详细列出了它们。

说明	目录视图
检查约束	SYSCAT.CHECKS
列	SYSCAT.COLUMNS
检查约束引用的列	SYSCAT.COLCHECKS
关键字中使用的列	SYSCAT.KEYCOLUSE
数据类型	SYSCAT.DATATYPES
函数参数或函数结果	SYSCAT.FUNCPARMS
参考约束	SYSCAT.REFERENCES
模式	SYSCAT.SCHEMATA
表约束	SYSCAT.TABCONST
表	SYSCAT.TABLES
触发器	SYSCAT.TRIGGERS
用户定义函数	SYSCAT.FUNCTIONS
视图	SYSCAT.VIEWS