1.1 DB2 UDB 的版本

DB2 针对于不同的用户需求，提供了多种版本：

● 企业服务器版（ Enterprise Server Edition ）

该版本通常用于支持大规模的企业级应用程序以及大型企业级数据仓库，它提供了最大程度的连接性，并且可以与异构平台上的 DB2 数据库和第三方厂商的数据库产品共享数据资源。

●工作组服务器版（ Workgroup Server Edition ）

该版本通常用于支持部门级应用程序或者支持那些不需要存取驻留在 OS/400、VM/VSE 和 OS/390 平台上的远程数据库的应用程序。

●个人版（ Personal Edition）

该版本通常被单机用户使用， 功能完备但不能响应远程的数据库请求。 该版本只能运行在 Linux 和 Windows 操作系统上。

● Everyplace

该版本是专门为移动计算环境设计的，允许移动用户通过个人数字助理 (PDA) 和掌上电脑等手持设备存取企业中的 DB2 数据源。DB2 Everyplace 可以执行在包括 Palm OS、Linux 、Windows CE 、Neutrino、PocketPC 和 Symbian 在内的多种操作系统上。

注： DB2 企业服务器版是以前的 DB2 UDB 企业版（ DB2 UDB Enterprise Edition ）和 DB2 UDB 企业扩展版(DB2 UDB Enterprise-Extended Edition) 的合并。以前的 DB2 卫星版已经被合并到新发行的 DB2 个人版当中。

1.2 DB2 的连接性

DB2 提供了很多种方法可以连接到 DB2 数据库和非 DB2 数据库。

● DB2客户端

在要存取 DB2 数据库的远程用户的工作站上必须安装 DB2 客户端组件。

● DB2 Connect

对于 UNIX 和 Intel 平台上的数据库应用程序，如果需要存取 OS/400、VM/VSE 和 OS/390 系统上的 DB2 数据库，则需要 DB2 Connect 的支持。

注意：任何平台上的数据库应用程序对 UNIX 和 Intel 平台上的数据库进行存取，都不需要 DB2 Connect 的支持。

● DB2 Relational Connect

DB2 Relational Connect 允许 DB2 客户端在 DB2 数据库和异构数据库（如 Oracle）之间进行存取和表的连接（ Join）操作。

1.2.1 DB2 客户端

针对不同的应用需求， DB2 客户端产品有：

● DB2运行时间客户端（ DB2 Runtime Client ）

在 DB2 应用程序开发完成以后，只需要在每个要运行 DB2 应用程序的工作站上安装 DB2 运行时间客户端即可。

DB2 运行时间客户端提供了与 DB2 服务器和 DB2 Connect 服务器进行通信的功能。

DB2 运行时间客户端可以在远程客户端上交互式地执行 SQL 语句对 DB2 服务器和 DB2 Connect 服务器上的数据进行存取。

DB2 运行时间客户端可以运行使用 ODBC 或 OLE DB 接口开发的应用程序或运行使用 JDBC 或者 SQLJ 接口开发的 JAVA 应用程序来存取 DB2 数据库。

● DB2管理客户端（ DB2 Administration Client ）

DB2 管理客户端中除了包含 DB2 运行时间客户端的全部功能之外，还包含所有的 DB2 图形化管理工具。

● DB2应用程序开发客户端（ DB2 Application Development Client ）

DB2 应用程序开发客户端中包含了开发 DB2 数据库应用程序所需要的一组开发工具，用于满足数据库应用程序开发人员的需求。 DB2 应用程序开发客户端包括所有的 DB2 图形化管理工具，并且具备 DB2 运行时间客户端的全部功能。

● DB2瘦客户端（ DB2 Thin Client ）

DB2 可以支持使 Type 4 的 JDBC 驱动程序的 Java客户端，这样就可以直接利用服务器端的客户端组件对 DB2 服务器进行存取。在这种情况下不需要 DB2 运行时间客户端。

1.2.2 DB2 Connect

通过 DB2 Connect 进行存取的 DRDA 应用程序服务器可以是 OS/390、VM/VSE 或者 OS/400 系统上的任何 DB2 服务器。

DB2 Connect 分为服务器版（又被称为企业版）和单用户版（又被称为个人版）两个版本。

DB2 Connect 企业版支持多个客户端同时通过它对主机数据进行存取，它可以将从远程客户端提交过来的数据库请求传递到相应的 DRDA 应用程序服务器上。远程客户端可以通过任何支持的网络协议与 DB2 Connect 交流。

DB2 Connect 个人版可以被安装在 Linux 和 Windows 平台，客户端应用程序只能从安装了 DB2 Connect 个人版的系统上提交对主机数据库的访问请求。

.3 应用程序开发

DB2 提供了丰富多样的应用程序开发环境。 DB2 应用开发环境有两种版本：

● DB2个人开发版（ PDE）—可以运行于 Linux 和 Windows 平台。

● DB2通用开发版（ UDE）— 适用于所有服务器平台。

1.4 DB2 管理工具

数据库管理员可以使用很多 DB2 提供的图形化工具来完成 DB2 数据库的管理工作。

1.4.1. 控制中心

控制中心是 DB2 管理工具的核心。它向用户提供了完成几乎所有典型的数据库管理任务所需的工具。

1.4 DB2 管理工具

数据库管理员可以使用很多 DB2 提供的图形化工具来完成 DB2 数据库的管理工作。

1.4.1. 控制中心

控制中心是 DB2 管理工具的核心。它向用户提供了完成几乎所有典型的数据库管理任务所需的工具。

1.4.2 命令中心

命令中心提供了一个交互式的图形化界面，允许用户输入 SQL 命令和 DB2 命令、执行命令、察看执行结果和语句解释信息。 由于提供了强大功能和多方面的灵活性， 命令中心成为用户输入文本命令的常用方法。命令中心会记录当前会话中所有执行过的语句和命令

1.4.3 任务中心

任务中心被用于创建、调度和管理包含了 SQL 语句、 DB2 命令和操作系统命令的命令脚本。

1.4.4 开发中心

在第 8 版 DB2 中，开发中心取代了以前版本中的存储过程生成器。对于开发存储过程、用户自定义函数等例程，开发中心提供了一个易用的界面。

1.4.5 健康中心

DB2 提供了一系列的工具来使数据库能够实现自我管理。 自我管理和资源调节 （智能） 数据库技术能够实现数据库操作在配置、调节和管理方面更高程度的自治。

健康中心是一个服务器端的工具， 它甚至可以在没有用户干预的情况下对 DB2 实例的健康状况进行监控。

1.5 DB2 OLAP Server

DB2 OLAP Server 是一个可伸缩的、强有力的联机分析处理（ OLAP ）软件，通过它，用户可以对企业的数据进行非常复杂的计划和分析，并基于分析结果做出决策。

依赖于应用程序的需求，多维立方体既可以被存储在 DB2 数据库中以增加 SQL 存取的灵活性，也可以进行多维存储以优化性能。

2.1 表

表是数据记录未排序的集合，包含列和行（通常称为记录）。每列都基于一个数据类型。表一旦创建并填入数据，就可在 DML 语句的 FROM 和 INTO 子句中被引用。有三种表类型：

- 永久表（基表）
- 临时（说明）表
- 临时（派生）表

（1）创建新表：

```
create table tabname(col1 type1 [not null] [primary key],col2 type2 [not null],..)
```

（2） 根据已有的表创建新表：

A：create table tab_new like tab_old

B：create table tab_new as select col1,col2 … from tab_old definition only

（3）修改表：

增加一个列 ：

Alter table tabname add column col type

列增加后将不能删除。 DB2 中列加上后数据类型也不能改变，唯一能改变的是增加 varchar 类型的长度。

添加主键：

Alter table tabname add primary key(col)

删除主键：

Alter table tabname drop primary key(col)

（4）删除表：

drop table tabname

2.2 视图

视图是从一个或多个表或视图生成的虚拟表（或称逻辑表），可用 CREATE VIEW 命令创建。当检索数据时可代替基表。

一旦定义了视图，就可以和基表一样使用，使用 DML 语句如 SELECT、INSERT 、和 DELETE 来存取。

当视图中显示的数据被修改后，在后台表中的数据也相应地修改了。视图本身并不存放真正的数据，在数据库中它只有一个定义。

视图可用于限制对敏感数据的存取而对其它数据则允许进行更宽松地存取。

视图可以是可删除的、可更新的、可插入的以及只读的。不同的类别表明了在使用视图时所能允许的 SQL 操作。

通过视图可以使应用程序获得表数据的一个子集并验证插入或更新的数据。视图中列的名字可以和基表中相应的列名不同。视图的使用为应用程序和终端用户查看表中数据提供了一个灵活的方式。

CRAETE VIEW 语句的例子如下所示。 基表 EMPLOYEE 有 SALARY 和 COMM 两列。出于安全性的原因，这个视图中只包含 ID、NAME 、DEPT、JOB 和 HIREDATE 列。并且只显示 DEPTNO 为 10 的那个部门的员工的信息。

```
CREATE VIEW EMP_VIEW1
(EMPID,EMPNAME,DEPTNO,JOBTITLE,HIREDATE)
AS SELECT ID,NAME,DEPT,JOB,HIREDATE FROM EMPLOYEE
WHERE DEPT=10;
```

从上面的例子可以看到， 视图可包含 WHERE 子句以限制对某些行的存取； 视图中也可含有列的一个子集，以限制对某些列数据的存取。

视图中列的名字可以和基表中相应的列名不同，表名和视图名都有一个相关联的模式。

2.3 索引

用户对数据库最频繁的操作是进行数据查询。 一般情况下， 数据库在进行查询操作时需要对整个表进行数据搜索。当表中的数据很多时搜索数据就需要很长的时间。

索引是与单个表相关的物理对象。 任何永久表或已声明的临时表都可以定义它们的索引， 但不可以在视图上建索引。可以为单个表定义多个索引。

索引是根据指定的一列或多列的内容对行进行排序。 索引主要用于提高查询效率， 但索引也可以用于逻辑数据设计。例如，主键不允许在同一列中输入相同的值，从而保证了没有一行数据是一样的。

索引可以定义为惟一的或非惟一的。 非惟一的索引允许重复的键值。 惟一的索引只允许列表中出现一个键值。

索引是使用 CREATE INDEX SQL 语句创建的。为支持主键或惟一性约束，也可以隐式创建索引。

2.4 模式

模式是数据库实体，一个模式表示 DB2 数据库中命名对象的一个集合。模式名实际上是数据对象的全限制名称的一部分。当使用 CREATE <db object> 定义数据库对象时，限制符或模式名出现在数据库对象名称中。

模式名可以使用 CREATE SCHEMA 语句显式地创建，并指定一个用户作为模式名的拥有者。如果用户 BERT 想要创建模式名称为 DB2 的表，那么 DBADM 可以使用控制中心或者以下语句为 BERT 创建模式。因为 BERT 拥有这个模式，所以他可以在模式中创建对象。

```
create schema db2 authorization bert
```

如果数据库对象的创建者在数据库对象定义中不包括模式名， 那么将使用创建者的授权 ID 作为模式名（假设没有收回创建者的 IMPLICIT_SCHEMA 权限）创建对象。

例如，某个用户 Mark 使用语句 CREATE TABLE TABLE1(C1 CHAR(3)) 创建一个表。数据库对象完整的名称应为 MARK.TABLE1 。

2.5 别名

别名可以用于引用数据库中的表。如果一个应用程序包含有 SQL 语句，这些语句按照别名存取表，那么可以定义代表不同表的别名，而无需修改应用程序。可以为一个表或别名定义别名。

例如， 假设一个叫做 Dana 的用户创建名为 PRICES 的表，另一个用户 Austin 也创建了名为 PRICES 的表。这些表存储在同一个数据库中，命名为 DANA.PRICES 和 AUSTIN.PRICES 。

下面我们创建可以用于所举方案的别名对象。

```
create alias db2cert.prices for dana.prices
or
create alias db2cert.prices for Austin.prices
```

然后，就可以开发用 DB2CERT.PRICES 来引用表对象的应用程序了。应用程序将存取被别名定义为

DB2CERT.PRICES 的源的那个表。在一个数据库中，任何时候别名 DB2CERT.PRICES 只能有一个定义。例如，不能创建 DB2CERT.PRICES 两次。但是，相同的别名可以用在不同的数据库。

可同一个表或视图创建多个别名。

create alias andrew..prices for db2cert.prices

create alias geoff..prices for db2cert.prices

2.6 约束

在业务处理中，我们通常需要确保始终实施某些规则。例如，公司正式雇员的年龄必须在 16－65 岁之间。例如，性别的取值应仅为 "男"或"女"。

DB2 通用数据库为此提供了约束。唯一约束是禁止在表的一列或多列中出现重复值的规则。参照完整性约束（也称为引用完整性约束）确保在整个指定的表中数据一致性。表检查约束是一些条件，它们定义为表定义的一部分，限制一列或多列中使用的值。

（1）关键字

关键字是可用来标识或存取特定行的一组列。

由不止一列组成的关键字称为组合关键字。在具有组合关键字的表中，组合关键字中各列的排序不受这些列在表中排序的约束。

（2）唯一关键字

唯一关键字被定义为它的任何值都不相同。唯一关键字的列不能包含空值。在执行 INSERT 和 UPDATE 语句期间，数据库管理程序强制执行该约束。一个表可以有多个唯一关键字。唯一关键字是可选的，并且可在 CREATE TABLE 或 ALTER TABLE 语句中定义。

（3）主关键字

主关键字是一种唯一关键字，表定义的一部分。一个表不能有多个主关键字，并且主关键字的列不能包含空值。主关键字是可选的，并且可在 CREATE TABLE 或 ALTER TABLE 语句中定义。

（4）外关键字

外部关键字在参考约束的定义中指定。一个表可以有零个或多个外部关键字。如果组合外部关键字的值的任何部分为空，则该值为空。外部关键字是可选的，并且可在 CREATE TABLE 语句或 ALTER TABLE 语句中定义。

（5）唯一约束

唯一约束确保关键字的值在表中是唯一的。唯一约束是可选的，并且可以通过使用指定 PRIMARY KEY 或 UNIQUE 子句的 CREATE TABLE 或 ALTER TABLE 语句来定义唯一约束。例如，可在一个表的雇员编号列上定义一个唯一约束，以确保每个雇员有唯一的编号。

（6）参照完整性约束

通过定义唯一约束和外关键字，可以定义表与表之间的关系，从而实施某些业务规则。唯一关键和外部关键字约束的组合通常称为参照完整性约束。外关键字所引用的唯一约束称为父关键字。外部关键字表示特定的父关键字，或与特定的父关键字相关。

例如，某规则可能规定每个雇员（EMPLOYEE 表）必须属于某现存的部门（DEPARTMENT 表）。因此，将 EMPLOYEE 表中的"部门号"定义为外部关键字，而将 DEPARTMENT 表中的"部门号"定义为主关键字。

（7）表检查约束

表检查约束指定对于表的每行都要进行判定的条件。可对个别列指定检查约束。可使用 CREATE 或 ALTER TABLE 语句添加检查约束。

下列语句创建具有下列约束的表：
部门编号的值必须在范围 10 至 100 内
雇员的职务只能为下列之一： "Sales"、 "Mgr" 或 "Clerk"
1986 年之前雇用的每个雇员的工资必须超过 $40,500。

```
CREATE TABLE EMP
(ID SMALLINT NOT NULL,
NAME VARCHAR(9),
DEPT SMALLINT CHECK (DEPT BETWEEN 10 AND 100),
JOB CHAR(5) CHECK (JOB IN ('Sales', 'Mgr', 'Clerk')),
HIREDATE DATE,
SALARY DECIMAL(7,2),
COMM DECIMAL(7,2),
PRIMARY KEY (ID),
CONSTRAINT YEARSAL CHECK (YEAR(HIREDATE) >= 1986 OR SALARY > 40500) )
```

仅当条件判定为假时才会违反约束。例如，如果插入行的 DEPT 为空值，则插入继续进行而不出错，尽管 DEPT 的值应该像约束中定义的那样在 10 和 100 之间。
下列语句将一个约束添加至名为 COMP 的 EMPLOYEE 表中，该约束为雇员的总报酬必须超过 $15,000：

```
ALTER TABLE EMP
ADD CONSTRAINT COMP CHECK (SALARY + COMM > 15000)
```

将检查表中现存的行以确保这些行不违反新约束。可通过使用如下的 SET CONSTRAINTS 语句将此检查延期：

```
SET CONSTRAINTS FOR EMP OFF
ALTER TABLE EMP ADD CONSTRAINT COMP CHECK (SALARY + COMM > 15000)
SET CONSTRAINTS FOR EMP IMMEDIATE CHECKED
```

首先使用 SET CONSTRAINTS 语句以延期对表的约束检查。然后可将一个或多个约束添加至表而不检查这些约束。 接着再次发出 SET CONSTRAINTS 语句， 反过来将约束检查打开并执行任何延期的约束检查。

2.7 外连接

最常规的连接，即内（ inner）连接。内连接的结果集仅包含在连接表中相匹配的行。
当我们需要连接表中不满足连接条件的非匹配行的数值时，怎么办？这就需要外连接操作。外连接的结果集不仅包含符合连接条件的匹配行， 而且还包含不满足连接条件的非匹配行。 存在一些不同类型的外连接。
在深入讨论外连接的细节前，我们首先看看连接两张表的显式语法。我们将从使用该语法的内连接开始。
下面的连接实例给出了一个结果集， 该结果集包括 test_taken 表中的每个考生的名字、 电话号码以及最高得分。

```
SELECT fname, wphone, MAX(INTEGER(score))
FROM db2cert.candidate c
INNER JOIN db2cert.test_taken tt ON c.cid=tt.cid
```

```
GROUP BY fname, wphone
```

在该语法中，与连接操作一起，用户指出要进行连接的表和连接列。
观察上例中的内部连接操作符，它属于语句中 FROM 子句。 INNER JOIN （内连接）操作符规定了在该语句中将使用内部连接。关键词 ON 用来规定连接表的连接条件。在我们的例子中，连接条件基于连接列，即 candidate 表的 cid 列和 test_taken 表的 cid 列。
显式连接语法也允许用户指定一个外连接。

1）左外连接（ Left Outer Join ）
左外连接也称为左连接， 其结果集既包括连接表的匹配元组， 也包括左连接表 （左关系） 的所有元组 （行） 。
左连接表是连接操作语句中 LEFT OUTER JOIN 操作符左边的连接表。

注意：也可以使用 LEFT JOIN 指示一个左连接操作。
现在要求我们产生一个 candidate 表中出现的每个考生的名字、电话号码以及最高得分的报表。假如我们采用内连接，正如上例中所示的，报表仅包括在 test_taken 表中出现的那些考生的数据。
我们将要使用如下所示的左外连接实现该目的。

```
SELECT fname, wphone, MAX(INTEGER(score))
FROM db2cert.candidate c
LEFT OUTER JOIN db2cert.test_taken tt ON c.cid=tt.cid
GROUP BY fname, wphone
```

观察左外连接 SQL 语句的语法， LEFT OUTER JOIN 操作符用来表明左外连接操作。在该例中，结果集也包括那些不在 test_taken 表中的考生，对于那些考生， MAX （INTEGER （score））列将显示空值。

2）右外连接（ Right Outer Join ）
右外连接也称为右连接，其结果集既包括连接表的匹配元组，也包括右连接表的所有元组。右连接表是连接操作语句中 RIGHT OUTER JOIN 操作符右边的连接表。
使用右外连接的例子如下所示：

```
SELECT name, count(DISTINCT char(tt.cid))
FROM db2cert.test_taken tt
RIGHT OUTER JOIN db2cert.test t
ON tt.number = t.number
GROUP BY name
```

该例将获得 test 表中出现的所有考试的名字以及计划或已经参加该考试的考生人数。注意，可能有一些考试没有安排考生。使用内连接语句，用户将不能知悉那些没有安排考生的考试，使用右连接却可以做到这一点。

3）全外连接（ Full Outer Join ）
全外连接操作符产生的结果集不仅包含符合连接条件的匹配行，而且包括两个连接表中的所有记录。

2.8 检查约束

表检查约束在表级上实现数据完整性。一旦在表上定义了表检查约束，每个 UPDATE 和 INSERT 语句都会引起限制或约束的检查。如果违反了约束条件，数据记录将不被插入或更新，并且会返回一条 SQL 错误信息。

表检查约束可在表创建时定义，也可在以后使用 ALTER TABLE 语句时定义。

通过指定表中每行的一个或多个列的允许值， 表检查约束有助于实现对表中数据值的特定规则。 这可以节省应用开发人员的时间，因为每个数据值的有效性可以通过数据库而不是通过每个访问数据库的应用程序去执行。

（1）增加检查约束

当在已有数据的表上增加检查约束时，会发生下述两种情况之一：

● 所有的行都满足检查约束。

● 某些行或所有行都不满足检查约束。

第一种情况，所有行都满足检查约束，检查约束将成功创建。若不满足约束业务规则，插入或更新数据的尝试将被拒绝。

当某些行不满足检查约束时，将不创建该检查约束（即 ALTER TABLE 语句将失败）。如下所示， ALTER TABLE 语句在 EMPLOYEE 表上增加新的名为 check_job 的约束。 DB2 通过这个名字来告知 INSERT 或 UPDATE 语句失败时违反的是哪个约束。 CHECK 子句用于定义表检查约束。

```
ALTER TABLE EMPLOYEE
ADD CONSTRAINT check_job
CHECK (JOB IN(?Engineer ‘ , ‘ Sales ‘ , ‘ Manager ‘ ));
```

此处使用了 ALTER TABLE 语句， 因为表早已定义。 如果 EMPLOYEE 表中的值与定义的约束相冲突， 将不能成功完成 ALTER TABLE 语句。

可以关闭约束检查来增加一个新的约束。 SET INTEGRITY 语句可用于关闭一个或多个表检查约束和引用约束检查。当关闭一个表的检查约束时，它便处于 CHECK PENDING 状态，只允许对表进行有限的存取。例如，一旦表处于 CHECK PENDING 状态，就不允许在表上使用 SELECT，INSERT，UPDATE 和 DELETE 等语句。 SET INTEGRITY 语句的完整句法参见《 DB2 UDB V8.1 SQL 参考手册》。

注意：

1. SET CONSTRAINTS 语句在 DB2 Universal Database V6.1 中被 SET INTEGRITY 语句所代替。 SET CONTRAINTS 支持向后兼容。
2. 给每个约束（触发器，表检查或引用完整性）加上标签是个好主意。这对分析可能发生的错误尤其重要。

（2）修改检查约束

检查约束用于实现业务规则， 可能需要经常修改。例如， 当组织内部的业务规则修改时就需要修改检查约束。

不过， 没有用于修改检查约束的专门命令。 每当要修改一个检查约束时， 必须先删除它， 然后再新建一个。可在任何时刻删除检查约束，而不会影响表及表中的数据。

当删除一个检查约束时， 必须意识到由该约束执行的数据有效性检查将不再起作用。 用于删除约束的语句是 ALTER TABLE 。

下例显示如何修改一个已有的约束。在删除约束后，必须用新的定义来创建它。

```
ALTER TABLE EMPLOYEE DROP CONSTRAINT check_job;

ALTER TABLE EMPLOYEE
ADD CONSTRAINT check_job
CHECK (JOB IN (?OPERATOR‘，‘ CLERK‘));
```

3 数据库管理常识

3.1 事务

（1）工作单元 (unit of work)

为了确保数据库中数据的一致性 ,对于应用程序来说 ,常常必须一步完成一系列数据的变动。这就叫做工作单元。

工作单元是应用程序进程内可恢复的操作序列。 工作单元是应用程序用来确保在数据库中不引入不一致数据的基本机制。

在任何时刻， 应用程序进程都至少有一个工作单元， 在应用程序进程的生存期内可能包括了许多工作单元。

（2 ）事务

工作单元也被称为事务 (transaction)。任何一个成功地与数据库连接的应用程序都自动地启动一个事务。

一个事务将至少包含一条 SQL 语句。多条 SQL 语句可以组合在一起作为单个事务执行。 DB2 将追踪 SQL 语句存取或修改的任何数据。 这些数据将被永久地改变 —— 提交（ committed）或回复到原有状态 —— - 回滚（rolled back）。

事务在执行时遵循 -要么不做，要么全做的原则 ‖，即不允许事务被部分地执行，称为事务的原子性。事务的执行不仅要保证原子性，而且要保证一致性。

通过事务日志可以实现一致性。日志文件用来确保所有被提交的事务真正地应用到数据库。

事务从处理第一条 SQL 语句时隐含地开始。当显式或隐式地发布 COMMIT 或 ROLLBACK 语句后， 标志着事务的完成。

应用程序必须通过发出一条 COMMIT 或 ROLLBACK 语句去结束该事务。

COMMIT 语句告诉数据库管理系统立即对数据库实施事务中的所有数据库变动 (插入、 更新、 删除、 创建、变更、授权和撤消)。

ROLLBACK 语句告诉数据库管理器不实施这些变动，而将事务中已经发生影响的操作撤消，返回到开始该事务之前的原有状态。

3.2 DB2 内置数据类型的存储开销

SMALLINT 的数值范围从 -32768 至 32767。每个 SMALLINT 列占用两个字节的数据库存储空间。

INTEGER 数据类型的取值范围从 -2,147,483,648 至 2,147,483,647。每个 INTEGER 列需占用四字节的数据库存储空间。

BIGINT 的范围从 -9,223,372,036,854,775,808 至 9,223,372,036,854,775,807。每个 BIGINT 列使用八个字节的数据库存储空间。

DECIMAL 或 NUMERIC 数据类型用于表示带小数和整数部分的数。 DECIMAL 数据以压缩格式存储。一个 DECIMAL 数占用 p/2+1 个字节的存储空间，其中 p 是使用的精度（截去 p/2 的余数）。

REAL 数据类型是一个数的近似值。该近似值需要 32 位或 4 字节存储空间。使用 REAL 数据类型定义一个单精度数，它的长度必须定义在 1 到 24 之间。

DOUBLE 或 FLOAT 数据类型是数的近似值。该近似值需要 64 位或 8 字节的存储空间。使用 FLOAT 数据类型定义一个双精度数，它的长度必须定义在 25 和 53 之间。

定长字符串使用定义好的存储空间存放在数据库中。 如果被存储的数据总是具有相同的长度， 那么应当使用 CHAR 数据类型。如果数据没有使用所定义的存储空间，那么使用定长字符字段有可能潜在地浪费数据库内的磁盘空间。但是存储变长字符串会有一些额外开销。

3.3 系统编目表

DB2 中有一些特殊的表，用于存放数据库中所有对象的信息，它们在创建数据库（如使用 CREATE DATABASE 命令）时创建。这些表称为系统编目表，由 DB2 在查询过程中检查。系统编目表中包含的信息有：

- 表/索引定义
- 列数据类型
- 定义的约束
- 对象依赖
- 对象授权

当执行 SQL 数据定义语言时，系统编目表事实上已更新。在 DB2 数据库中有许多系统基表和视图，它们总是存放在 SYSCATSPACE 这个特殊的表空间中。

系统表还包含有关数据库中表的统计信息， 如存储在系统编目表中每个表所占用的物理页的数目。 这些统计信息时通过 RUNSTATS 命令来计算和更新的。

数据库对象授权，如 INSERT， SELECT 和 CONTROL ，也包含在系统编目表中。
当一个表删除后，它的描述将从编目表中删除，并且引用了该对象的包将变成无效。

4.DB2 安全性和存储控制

4.1 安全机制

在通过操作系统或者外部安全设施成功地进行了验证过程之后， DB2 内部的安全机制将控制对数据库对象的存取。

对数据库对象的存取控制主要是通过为用户指定某种权限（ authority ）或授予某些特权（ privilege）来进行的。

权限是一组高层次的用户权力，通常授予那些需要对数据库和实例进行管理和维护的用户。

特权是针对于某个数据库对象（如表、视图等）的用户权力，通常授予那些只需要对数据库对象进行存取的用户。例如，一个用户可以在一个表上具有 SELECT 和 UPDATE 特权。

4.2 权限

权限用于控制对实例内部对象的存取和能否执行某项管理任务。在 DB2 中，有 5 种类型的权限：

- SYSADM— 系统管理权限
- SYSCTRL— 系统控制权限
- SYSMAINT— 系统维护权限
- LOAD— 对表进行 LOAD 操作的权限
- DBADM— 数据库管理权限

1） SYSADM 权限

SYSADM 权限是 DB2 系统最高级别的权限。

具有 SYSADM 权限的用户可以执行任何 DB2 管理操作，也可以对所有的数据库对象进行存取。

SYSADM 用户组中的成员是唯一允许对 DB2 实例进行配置的用户。

2） SYSCTRL 权限

SYSCTRL 权限允许用户执行大多数的管理任务。但是，SYSCTRL 用户组中的成员不能对数据库对象进行存取，也不能对数据库管理器配置文件进行更改。

SYSCTRL 权限具有创建数据库的权力，如果具有 SYSCTRL 权限的用户创建了一个数据库，该用户自动具有该数据库的 DBADM 权限，可以对该数据库中的数据库对象进行存取。如果该用户需要对其它数据库中的数据库对象进行存取，必须对该用户进行显式授权。

具有 SYSCTRL 权限的用户具有 SYSMAINT 权限所包含的所有权力。

3） SYSMAINT 权限

SYSMAINT 权限允许用户执行系统的维护工作。 但是 SYSMAINT 用户组中的成员不能对用户数据进行存取。

4） LOAD 权限

被授予 LOAD 权限的用户可以在不具备 SYSADM 或者 DBADM 权限的情况下使用 LOAD 工具进行数据导入工作。该权限赋予了用户在数据库管理任务上更加细化的控制。

5） DBADM 权限

在数据库级别，最高的权限是 DBADM 权限。

具有 DBADM 权限的用户可以在数据库级别执行任何管理任务，比如创建和删除数据库对象、导入数据和监控数据库活动等。

DBADM 权限对数据库内的所有数据库对象具有绝对的控制权，除了可以查询、删除和创建任何表之外，还可以将相应数据库对象上的特权赋予其它用户。

数据库的创建者将自动被授予新创建的数据库上的 DBADM 权限。

4.3 特权

特权是创建或者存取数据库对象的权力。

有两条 SQL 语句可以用于特权管理 ：

- GRANT 语句用于授予用户特权
- REVOKE 语句用于撤销用户已有的特权

下面，我们来看看 DB2 如何处理对用户和用户组的授权。假设有一个名为 austin 的用户，我们想赋予该用户 candidate 表上的 SELECT 特权，则应当使用下列命令：

GRANT SELECT ON candidate TO austin GRANT SELECT ON candidate TO USER austin

如果在操作系统中不存在名为 austin 的用户组，以上两条语句都可以将 candidate 表上的 SELECT 特权授予用户 austin。

假定有一个名为 austin 的用户组， 我们想赋予该用户组 candidate 表上的 SELECT 特权， 则应当使用下列命令：

GRANT SELECT ON candidate TO austin GRANT SELECT ON candidate TO GROUP austin

如果在操作系统中不存在名为 austin 的用户，以上两条语句也都可以将 candidate 表上的 SELECT 特权授予用户组 austin。

但是，如果名为 austin 的用户和用户组同时存在，语句 GRANT SELECT ON candidate TO austin 将会引起混淆，系统会提示错误信息。因此，在授权时最好指明是对单个用户授权还是对用户组进行授权。下列语句在授权时就不会带有二义性：

GRANT SELECT ON candidate TO USER austin GRANT SELECT ON candidate TO GROUP austin

像 GRANT 命令一样， REVOKE 命令也带有 —USER 和 —GROUP 选项， 用以指明是撤销单个用户的权限还是撤销用户组的权限。

如果一个用户被授予表上的 CONTROL 特权，则该用户将隐式获得该表上的其它所有特权 （包括 INSERT、UPDATE 、DELETE 等），而且该用户还可以将除了 CONTROL 特权之外的其它所有特权授予其它用户。即使从该用户身上撤销了表上的 CONTROL 特权，该用户仍然拥有表上的其它所有特权，而且可以将这些特权赋予其它用户。如果需要限制该用户对表的存取，应当使用 REVOKE ALL 命令显式处理，具体命令如下：

REVOKE ALL ON tablename FROM username/groupname

5 并发性

数据库服务器对于一组最终用户而言， 其作用是一个存取数据的中心资源。 最终用户的数目可能是变化的，从一个用户到数千个用户。当多个用户访问相同的数据资源时，就必须建立关于数据记录的读取、插入、删除、更新的规则，以保证数据的完整性。

数据存取的规则由与 DB2 数据库相连接的每个应用程序设置。这些规则可以用以下两种方法建立：

显式控制 —— 使用一条 SQL 语句或 DB2 命令锁定资源

隐式控制 —— 通过指定一种隔离级别锁定资源

5.1 并发异常

在任何数据库环境中，数据完整性都是人们最关切心的事情。当数据被修改（包括插入和删除）时，数据库服务器必须保证数据的完整性。

执行的每条可执行 SQL 语句都是事务的一部分。一个事务将至少包含一条 SQL 语句。多条 SQL 语句可以组合在一起作为单个事务执行。

下面将讨论事务中出现的一些并发异常，包括：

- 丢失更新（ Lost Update）
- 未提交读（ Uncommitted read ）
- 不可重复读（ Nonrepeatable read）
- 幻象读（ Phantom read）

5.1.2 未提交读

未提交读（ UR）隔离级也称为 “脏读”，它可以用来存取其它应用程序的未提交的数据更新。

5.1.3 不可重复读

不可重复读是指在同一事务中，两次执行同样的 SELECT 语句，得到的结果集不同。非提交读应用程序不能保证可重复读。

为了避免这种类型的不可重复读，所有被查询的数据都应被锁定。假如用户希望被查询的数据不被修改，仅需将这些数据锁定就行了。 DB2 中，这称为读稳定性（ read stability）。假如用户希望所查询的数据在该事务中不再被更改，即可重复读（ repeatable read），这就需要附加的锁定。

5.1.4 幻象读

假如应用程序将同一查询执行两次， 将可能出现幻象读现象。第二次发出同一查询时， 返回结果可能新增了一些行。对于多数应用程序，这是一个可以接受的方案。例如，查询涉及寻找音乐会的所有空余座位，则重复发出该查询并获得一个较好的座位选择就是一个满足需求的特点。

根据不同的情况， 有时可能需要幻象读。 假如用户希望避免幻象读， 应用程序必需锁定全部可能限定的行。这将确保没有其它应用程序能够更新、 删除或插入可能会影响结果表的行。 这是一个需要理解的重要概念。

5.1.2 未提交读

未提交读（ UR）隔离级也称为 “脏读”，它可以用来存取其它应用程序的未提交的数据更新。

5.1.3 不可重复读

不可重复读是指在同一事务中，两次执行同样的 SELECT 语句，得到的结果集不同。非提交读应用程序不能保证可重复读。

为了避免这种类型的不可重复读，所有被查询的数据都应被锁定。假如用户希望被查询的数据不被修改，仅需将这些数据锁定就行了。 DB2 中，这称为读稳定性（ read stability）。假如用户希望所查询的数据在该

事务中不再被更改，即可重复读（ repeatable read），这就需要附加的锁定。

5.1.4 幻象读

假如应用程序将同一查询执行两次，将可能出现幻象读现象。第二次发出同一查询时，返回结果可能新增了一些行。对于多数应用程序，这是一个可以接受的方案。例如，查询涉及寻找音乐会的所有空余座位，则重复发出该查询并获得一个较好的座位选择就是一个满足需求的特点。

根据不同的情况，有时可能需要幻象读。假如用户希望避免幻象读，应用程序必需锁定全部可能限定的行。这将确保没有其它应用程序能够更新、删除或插入可能会影响结果表的行。这是一个需要理解的重要概念。

5.2 隔离级（ Isolation Levels）

在前面，已经讨论了一些可能的并发问题。DB2 通用数据库提供了不同的保护级别，将每一个数据库应用程序与被存取的数据隔离开来。

这些不同的保护级别即为不同的隔离级或锁定策略。DB2 支持的隔离级包括：

- 非提交读（ Uncommitted read ）
- 游标稳定性（ Cursor stability ）
- 读稳定性（ Read stability ）
- 可重复读（ Repeatable read）

将包绑定到数据库时，可以使用 PREP 或 BIND 命令的 ISOLATION 选项，定义嵌入式 SQL 的隔离级。假如没有指定隔离级，则采用游标稳定性的默认隔离级。

假如用户使用命令行处理器，用户可以使用 CHANGE ISOLATION LEVEL 命令改变隔离级。

对于 DB2 调用级接口（ DB2 CLI ），用户可以修改作为 DB2 调用级配置文件（ db2cli.ini ）的一部分的隔离级。

5.2.1 未提交读

未提交读（ UR）隔离级也称为 “脏读”，是 DB2 支持的最低级别的隔离级。它可以用来存取其它应用程序的未提交的数据更新。例如，使用未提交读隔离级的应用程序将返回查询的所有匹配行，即使该数据处于被更新的过程之中，并没有向数据库进行提交。

假如用户决定使用这一隔离级，用户的应用程序可能存取不正确的数据。未提交读事务持有的锁很少。当使用这种隔离级时，可能出现不可重复读和幻象读现象。

5.2.2 游标稳定性

游标稳定性（ CS，或称光标稳定性）隔离级锁定工作单元期间光标所在的任何行。对该行的锁定将保持到取出下一行记录或整个工作单元终止。假如更新某一行，该锁将保持到整个工作单元终止。当执行 COMMIT 或 ROLLBACK 语句时，该工作单元就终止了。

使用游标稳定性的应用程序不能读未提交数据。另外，应用程序锁定当前被取出的行，其它的应用程序都不能修改当前行的内容。

假如用户决定使用这一隔离级，　用户的应用程序读取的数据将保持一致性，　但不可重复读或幻象读状况仍可能存在。

5.2.3 读稳定性

读稳定性（ RS）隔离级锁定作为结果表中的那一部分行。假如用户有一张包含　10000 行记录的表，查询结果返回　10 行记录，则仅有　10 条记录被锁定。

使用读稳定性的应用程序不能读未提交的数据。　使用这种隔离级并不是仅有一行被锁定，　而是一个事务中所有被读取的行都会被锁定。其它的任何应用程序都不能修改这些行。

假如用户决定使用这一隔离级，　则在同一执行单元中多次执行同一查询时，　用户的应用程序得到相同的结果。这样可以保证在一个事务中即使多次读取同一行，得到的值不会改变，但读稳定性隔离级不能阻止通过插入操作在结果集中加入新行，即得到新增的幻象行。

5.2.4 可重复读

可重复读（ RR）隔离级是　DB2 中最高级别的隔离级，使用这种隔离级，不仅满足读取条件的行被锁定，应用程序在工作单元内所引用的所有行也将被锁定。无论结果集有多大，构造这个结果集所处理的所有行都持有该锁。表锁可以用来代替大量的因子锁（　factor）。

使用可重复读隔离级的应用程序无法读取并发应用中的未提交的数据。　假如用户决定采用这种隔离级，　那么用户的应用中就不会出现前面所讨论的各种问题（丢失更新、幻象读或不可重复读）。

5.2.5 隔离级的选择

选择合适的隔离级是很重要的，因为它不仅影响并发性，而且影响到应用软件的性能。用户的保护越多（使用的隔离级别越高），并发性就越低。

用户决定在应用程序中哪些并发问题是无法接受的，就选择能防止该问题出现的那种隔离级别：
当用户在只读型表上使用查询，或者仅使用　SELECT 语句，而并不在意是否从并发应用中获取了未提交数据时，则应使用未提交读隔离级。

当用户需要获得最大的并发度，　同时又只要见到并发应用的已被提交的数据，　则应使用游标稳定性隔离级。

当用户的应用程序在并发环境中操作时，则应使用读稳定性隔离级。这意味着在整个工作单元工作期间，被限定的行需要保持稳定。

假如用户需要在同一事务中结果集不发生改变，则需要使用可重复读隔离级。

5.3 锁定

DB2 通过隔离级来控制并发。在许多情况下，用户不需要采用直接行动来设置锁。通常来说，　DB2 根据隔离级的语义隐式地获取相应的锁。

5.3.1 锁属性

被锁定的资源叫做对象。用户可以显式锁定的对象仅有表和表空间。其它类型的数据库对象，诸如行、索引键和一些表， DB2 根据隔离级和处理情况隐式地获取相应的锁。被锁定的对象代表了锁的粒度。

锁的存取和规则由锁方式（ mode）限定。有些锁方式仅用来锁定表对象，而其它的锁方式用来锁定行对象。 DB2 使用如下的可锁定数据库对象的层次结构：

表空间
表
行

5.3.2 锁的升级（ Lock Escalation）

如果用户的应用程序修改了表中的许多行，最好对整个表有一个锁， 而不是在每一行上有多个锁。 对于每个锁， DB2 都需耗费内存，因此，假如一个表锁可以代替大量的行级锁，则节省的存储区域可以被其它应用程序使用。

当 DB2 代替用户将多个行级锁转换为表锁时，则称为锁的升级。 为了防止一些单个的锁占用太多的资源， DB2 将执行锁的升级，以避免资源浪费。

注意：每个 DB2 锁都将耗费同样大小的内存。

有两个数据库配置参数对锁升级有直接的影响。它们是：

LOCKLIST --- 定义分配给锁的内存空间大小。

MAXLOCKS --- 定义允许分配给单个应用程序的全部锁列表（ locklist ）的百分比。

锁升级有两种不同情况：

当某个应用程序超过 MAXLOCKS 配置参数定义的锁列表的百分比， 数据库管理器将试图通过为提出锁请求的应用程序授予表锁并释放行级锁来节省内存空间。

与数据库相连接的多个应用程序由于拥有大量的锁而占满了锁列表。 DB2 将试图通过获取表锁并释放行级锁来节省内存空间。

还应当注意，应用程序使用的隔离级也对锁的升级有影响，情况如下：

最初光标稳定性将获得行级锁。 假如需要，也能获得表级锁。 通常每个光标稳定性应用程序只需要数量很少的锁，因为它们只需要保证当前行中数据的完整性。

读稳定性锁定原来结果集中的所有行。

可重复读可以获取、 也可以不获取为确定结果集而读取的所有行上的行级锁， 假如不获取， 可以通过拥有表锁来代替。

5.3.3 锁定表语句（ Lock Table）

用户可以使用 LOCK TABLE （锁定表）语句重载（ override）初始锁方式的规则。它锁定指定的表，直到工作单元被提交或回滚。表可以用 SHARE MODE （共享方式）或 EXCLUSIVE MODE （排它方式）锁定。

当用 SHARE MODE 使用 LOCK TABLE 语句，其它任何程序都不能更新、删除或插入被锁定表中的数据。如果用户需要经常被并应用程序修改的表的快照，就可以使用这一语句去锁定该表的更改，而不对它的应用程序使用可重复读隔离级。

EXCLUSIVE MODE 比 SHARE MODE 更严格。它防止并发应用程序访问该表，进行读取、更新、删除和插入。如果用户准备更新表中的大部分内容，可以以排它方式（ EXCLUSIVE MODE ），而不是锁定每一行的方式使用 LOCK TABLE 语句。

6.1 控制中心和控制中心的其他工具

控制中心是 DB2 管理工具的核心。它向用户提供了完成几乎所有典型的数据库管理任务所需的工具。通过控制中心，用户可以方便地调用其它的数据库管理工具来进行数据库系统中的管理工作。具体的工具包括：

● 卫星管理中心（ Satellite Administration Center ）

卫星管理中心被用于管理多个 DB2 个人版或工作组服务器版的安装。

● 数据仓库中心（ Data Warehouse Center）

数据仓库中心被用于构建数据集市或数据仓库。

● 命令中心（ Command Center）

命令中心提供了一个交互式的图形化界面，允许用户输入 SQL 命令和 DB2 命令、执行命令、察看执行结果和语句解释信息。 由于提供了强大功能和多方面的灵活性， 命令中心成为用户输入文本命令的常用方法。另外，命令中心和任务中心之间可以进行互操作。

● 任务中心（ Task Center）

任务中心被用于创建、调度和管理包含了 SQL 语句、 DB2 命令和操作系统命令的命令脚本。

● 日志（ Journal）

日志工具保存了所有脚本的执行记录、 所有数据库管理器生成的信息以及 DB2 恢复历史文件中的信息。 日志工具通常被用于察看作业的执行结果、察看被调度脚本的内容、以及启动或终止作业。

● 许可证中心（ License Center）

许可证中心被用于管理 DB2 产品的许可证信息和检查当前并发的连接数目。

● 开发中心 (Development Center)

开发中心允许用户创建和测试 DB2 存储过程或用户自定义函数。

● 工具设置（ Tool Settings）

工具设置允许用户对 DB2 图形工具的某些设置选项进行配置。

6.2 日志

一些日志文件与每个数据库都相关。 当处理事务时， 处理情况都记录在日志文件中。 DB2 在数据库日志文件中记录下与该数据库相关的所有 SQL 语句的情况。

DB2 采用预写日志的方法。修改首先被写到日志文件，稍后，再对物理数据库表实施这些修改。

注意：在版本 8 中，日志文件可达 256GB，这样可以允许执行长事务而不会填满日志文件。 在某些事件中，事务会超出该限制，现在 DB2 会连接旧日志文件，因此在理论上用户永远不会用完日志空间。但是每个一定时间，应用应当提交他们的工作，因此日志的这种连接不会发生。

6.3 复制中心

复制中心是一个允许管理员设置和管理数据复制环境的图形工具。 复制预约定义作为脚本被存储在中央控制服务器中。设置一个复制预约的主要工作都可以通过复制中心完成，具体包括：

- 注册复制源
- 监控复制进程
- 运行 Capture 和 Apply 程序
- 定义警报条件

6.4 数据仓库管理器

除了 DB2 数据仓库中心提供的基本功能之外，集成在 DB2 UDB 中的数据仓库管理器还提供了以下功能：

- 数据仓库代理程序增强了数据仓库的功能，它主要负责管理源数据仓库和目标数据仓库中的数据流。

- 可以使用 Java 存储过程和用户自定义函数来进行高级数据变换，具体的功能包括 清除数据、生成数据透视表、生成主关键字和周期表以及计算各种统计信息等。

- 在内部集成了信息目录管理程序，可以引导用户找到进行决策所需的相关信息。

- 通过 DB2 Query Patroller 进行查询控制和工作负载分配。

- 生成能够满足绝大多数企业需要的报表。

注：数据仓库管理器通过对其他类型数据源、其他类型数据仓库、统计变换和集成的信息目录的支持扩展了数据仓库中心的功能。

6.5 信息目录中心

信息目录中心是一个可以帮助终端用户轻松地发现、 理解和访问公司中可用信息的图形化工具。 它由数据仓库管理员遵循商业视图模型构造出来，供用户浏览企业信息使用的。具体来说，它提供了以下功能：

- 通过与数据仓库中心以及其它分析和报表工具进行元数据（ metadata）交换来填充目录。

- 直接注册共享信息对象。

- 浏览和搜索数据对象以发现相关信息。

- 显示数据对象的元数据。

● 为终端用户启动提供信息的各种工具。

注：信息目录中心保存了终端用户需要存取的元数据。元数据为管理员和商业用户提供了数据仓库中所存储的数据的说明，信息目录中心可以与数据仓库中心自动进行元数据的交换。

6.6 Visual Explain

Visual Explain 是一个图形化的工具，可以将 DB2 为 SQL 语句生成的存取计划以图形的方式直观显示给用户，帮助用户对语句性能进行分析。

6.7 健康中心（ Health Center ）

IBM 的自治运算策略致力于建立一个能够自管理、自诊断和自治愈的 IT 基础架构。 DB2 提供了一系列的工具来使数据库能够实现自我管理。 自我管理和资源调节 （智能） 数据库技术能够实现数据库操作在配置、调节和管理方面更高程度的自治。

健康中心是一个服务器端的工具， 它甚至可以在没有用户干预的情况下对 DB2 实例的健康状况进行监控。当定义的阈值被超过了以后（比如日志空间不足），系统会发出警告信息。警告通知可以通过电子邮件或者传呼系统发送。如果不采用发送通知的方法，用户也可以采用命令脚本或者任务的形式预先定义一组行为在出现问题后自动执行。

图形用户界面允许管理员选择数据库对象来察看其详细信息、 为当前对象定义的警告以及为解决警告而建议采用的行为。

7UDP 产品环境、命令、配置

7.1 DB2 环境

DB2 环境管理着很多数据库相关要素，比如用于存取远程数据库的网络协议、应用程序搜索 DB2 相关文件的路径以及分配给数据库和应用程序使用的各种内存区的大小等因素都可以通过 DB2 环境来控制。 DB2 环境比较复杂，可以通过以下方式来控制：

●DB2 概要文件注册表（ DB2 Profile Registry ）

● 环境变量

● 配置参数

7.1.1 DB2 概要文件注册表

DB2 环境中的许多设置都可以通过在 DB2 概要文件中设定相应的注册表变量来完成。 DB2 概要文件注册表的作用在于统一 DB2 环境的设置方法，使得很多关键性的控制因素可以被集中地管理。

下面是一些 DB2 概要文件注册表变量的例子：

● DB2CODEPAGE

该注册表变量适用于所有平台上的 DB2 系统， 如果不设定， DB2 将根据操作系统的代码页设置来设定该注册表变量的值。

● DB2DBDFT

该注册表变量用于指定进行隐式连接（即在没有执行 CONNECT 命令的情况下进行数据库操作）的缺省数据库名。

● DB2COMM

该注册表变量适用于所有平台上的 DB2 服务器，它指定了 DB2 服务器可以使用什么网络协议与客户端通信。

● DB2NBADAPTERS

该注册表变量仅适用于 Windows 环境下的 DB2 服务器，它指定了使用哪块本地网卡来支持 NetBIOS 协议。

注：在对 DB2 概要文件注册表进行更改以后，不需要重新启动系统。

7.1.2 环境变量

有一些环境变量可以被存储在 DB2 概要文件中，也可以不被存储在 DB2 概要文件中（这具体依赖于操作系统）。 DB2 使用的系统环境变量必须被存储在操作系统的环境变量中，例如：

● DB2INSTANCE

该变量用于指定活动的 DB2 实例。

● DB2PATH

该变量用于指定 DB2 可执行文件所在的路径。

7.1.3 声明注册表变量和环境变量

所有的注册表变量都被存储在一些外部文件中，文件中包含了各个变量的名称和设定的值。但是，这些文件不能使用文本编辑工具直接编辑或操作。要想更改注册表变量的值，应使用 db2set 命令。

对注册表变量所进行的更改都会动态生效，不需要用户重新启动系统。在注册表变量被更改以后启动的 DB2 应用程序立刻就可以使用更改后的值进行操作。但是，如果用户对实例级的注册表变量进行了更改，则需要先执行 db2stop 命令来停止相应实例，然后再使用 db2start 命令来启动该实例，以使更改过的变量生效。

使用 db2set 命令加上 -all 选项，可以察看系统中设置的所有概要文件注册表变量。

下列是有关 db2set 命令用法的一些例子：

要设置当前实例的一个注册表变量：

db2set parameter=value

要为特定实例设定一个注册表变量：

db2set parameter=value –i instance_name

要设置一个全局级的注册表变量：

db2set parameter=value –g

要察看能够被设置的所有概要文件注册表变量的列表：

db2set -lr

7.2 DB2 常用的管理命令

实例级：

1）启动数据库管理器实例

db2start

2）停止数据库管理器实例

db2stop

3）获取数据库管理器配置设置

get dbm cfg

4）显示数据库管理器参数的当前值和延迟值

get dbm cfg show detail

5）返回 DB2INSTANCE 环境变量的值

get instance

6) 断开所有应用程序与数据库的连接

force application all

7) 以用户 <userid> 通过使用密码 <pwd> 与标识为 <node> 的远程实例连接

attach to <node> user <userid> using <pwd>

数据库级：

8）显式地以用户 <userid> 和密码 <pwd> 与数据库 <dbname> 连接

connect to <dbname> [[user <userid>] using <pwd>]

9）显式地激活数据库

activate database <dbname>

10）显式地使数据库失效

deactivate database <dbname>

11）断开与当前数据库的连接

connect reset

12) 显示数据库配置参数的当前值和延迟值

get db cfg show detail

13）显示数据库 <dbname> 的数据库配置设置

get db cfg for <dbname>

14）将数据库 <dbname> 的数据库配置参数 <p> 更新为值 <v>

update db cfg for <dbname> using <p> <v>

15）列出数据库中的表

list tables[for {user | all | system | schema <schemaname>}][show detail]

如果没有指定任何参数，则缺省情况是列出当前用户的表。

16）显示一个表或视图的列信息

describe table <tablename>

17）显示表空间的标识、名称、类型、内容和状态

list tablespaces [show detail]

18）显示用 <tablespace_id> 指定的表空间的容器信息

list tablespace containers for <tablespace_id> [show detail]

8 验证权限和特权

8.1 验证

安全性管理的第一个步骤就是核实用户身份，该过程被称为验证（ Authentication ）。验证是判断用户是否真是其所宣称的那个人的过程。

典型的验证手段是强迫用户提供用户名和口令。 在每次试图对本地或者远程数据库进行连接的时候， 用户都会被系统要求验证。 DB2 将会把用户名和口令传递给操作系统或者外部安全设施来进行检验。

在配置 DB2 客户端和服务器之间的连接时，用户可以指定在什么地方进行验证，比如，是在客户端还是在服务器端。 验证类型可以在参与连接的每一台机器上进行设置， 可以选择的验证类型依赖于所处的环境。下面是两种常见的环境：

- 客户端直接连接到 DB2 UDB 服务器
- 客户端通过 DB2 CONNECT 连接到主机数据库

8.1.1 客户端直接连接到 DB2 UDB 服务器

这种环境中包含两种机器： 服务器和客户端。 验证类型由存储在数据库服务器端的数据库管理器配置文件的相应参数值指定。

用户可以在从 DB2 客户端对数据库进行编目时指定验证类型。

需要注意的是，在编目数据库时指定的验证类型一定要与服务器端验证参数的值相匹配。

8.1.1.1 DB2 UDB 服务器端的验证类型

在服务器端，验证类型被定义在数据库管理器配置文件中。该文件用于控制特定的 DB2 实例，其包含的参数的值将会影响到该实例中所有的数据库。

因此， 如果为某个实例指定了验证类型， 那么该实例中的所有数据库都会使用这种验证类型， DB2 系统也会基于这种验证类型对所有连接到该实例中的数据库的用户进行验证。

可以在服务器端指定的验证类型包括：

- SERVER

该验证类型强迫所有的验证过程都将在服务器端进行，因此，在对实例或数据库进行连接时必须提供用户名和口令。 DB2 系统在将口令传递给服务器的过程中不对口令进行加密。

- SERVER_ENCRYPT

该验证类型与 SERVER 验证类型类似，所有的验证过程也将在服务器端进行。但是，所有的口令在被传递给服务器之前都将在客户端被加密。在口令传递到服务器以后，口令将被解密并进行验证。但是，如果在连接的同时要求更改现有口令， DB2 系统在将新口令传递给服务器的过程中不对新口令进行加密。

- CLIENT

该验证类型允许在客户端对用户进行验证，但它并不能保证验证过程肯定能够在客户端进行。

- KERBEROS

该验证类型既可以设置在客户端，也可以设置在服务器端，但是要求 DB2 系统所驻留的操作系统必须支持 Kerberos 安全协议。

Kerberos 安全协议是一种第三方的验证服务，它使用传统的加密手段来创建一个共享密钥，这个密钥将成为用户的凭证。无论是请求本地服务还是网络服务，系统都将使用它来验证用户的身份。该密钥消除了将用户名和口令以明文方式在网络中传输所带来的隐患。使用 Kerberos 安全协议允许对一个远程的 DB2 服务器进行单点登录。

- KRB_SERVER_ENCRYPT

该验证类型指定了 DB2 服务器可以接受的验证类型为 KERBEROS 或者 SERVER_ENCRYPT 。如果客户端的验证类型为 KERBEROS，则验证将通过 Kerberos 安全系统来进行；如果客户端的验证类型不是 KERBEROS，则系统缺省认为验证类型为 SERVER_ENCRYPT 。

8.1.1.2 DB2 客户端的验证类型

在为远程存取编目一个数据库的时候也可以指定验证类型，定义的验证类型将会保存在数据库目录中。

在对数据库进行编目时，并不一定要指定验证类型。如果没有指定，对于通过 DB2 Connect 存取数据库的客户端，系统将默认验证类型为 SERVER，否则，系统将默认验证类型为 SERVER_ENCRYPT 。但是，如果服务器端不支持 SERVER_ENCRYPT 验证类型，客户端将使用一个服务器支持的验证类型重新尝试连接。但当服务器端同时支持多个验证类型时，客户端不会进行上述重新尝试连接的操作，而是报错。之所以这样做的原因是要确保能够为连接选择正确的验证类型，在这种情况下，用户必须使用服务器端支持的一种验证类型对数据库重新编目。

如果在编目数据库的时候指定了验证类型而该验证类型又被服务器所支持，则验证过程可以顺利进行。但如果编目数据库时指定的验证类型不被服务器所支持，DB2 系统会重新尝试验证过程，这会导致客户端和服务器之间的网络负担加大。如果尝试失败，系统将会报错。在客户端与服务器端验证类型不匹配的时候，服务器端验证类型将被假定是正确的。

CATALOG DATABASE 命令可以用来设定客户端的验证类型，并将其保存在系统数据库目录中。可以在执行该命令时不指定验证类型。但是，如果指定了验证类型，则客户端的验证类型一定要与服务器端的验证类型相匹配。如果客户端与服务器端的验证类型不匹配，连接可能会失败。

可以在客户端指定的验证类型包括：

- SERVER
- SERVER_ENCRYPT
- CLIENT
- KERBEROS
- DCE

8.1.2 客户端通过 DB2 CONNECT 连接到主机数据库

在这种环境中，有三种角色参与建立连接：客户端、DB2 Connect 服务器和主机数据库服务器。DB2 Connect 企业版将被安装在 DB2 Connect 服务器上，客户端将通过 DB2 支持的标准网络协议（NetBIOS、TCP/IP 等）连接到 DB2 Connect 服务器，然后 DB2 Connect 服务器将通过 DRDA 支持的网络协议（TCP/IP、APPC）连接到主机服务器，并传递客户端请求。在这种环境中，验证的位置将由在 DB2 Connect 服务器上执行 CATALOG DATABASE 命令时使用 AUTHENTICATION 参数指定。

可以在在客户端指定的验证类型包括：

- SERVER
- SERVER_ENCRYPT
- KERBEROS
- CLIENT
- DCE

下表显示了不同的客户端验证类型和服务器端验证类型组合在一起时，验证过程发生的位置。另外，表中还显示在验证过程中口令是否被加密。在该表中，—N/A 表示口令不在网络中进行传输。

表 在客户端、DB2 Connect 和主机之间的验证选项

客户端验证类型	**DB2 Connect** 服务器端验证类型	验证发生的位置	客户端到 **DB2 Connect** 服务器之间的口令传输是否加密	**DB2 Connect** 服务器到主机之间的口令传输是否加密
SERVER	SERVER_ENCRYPT	主机服务器端	不加密	加密
SERVER_ENCRYPT	SERVER_ENCRYPT	主机服务器端	加密	加密
SERVER_ENCRYPT	SERVER	主机服务器端	加密	不加密
SERVER	SERVER	主机服务器端	不加密	不加密
DCE	DCE	DCE 服务器	加密	加密
CLIENT	CLIENT	客户端	N/A	N/A

8.2 特权

8.2.1 显式特权和隐式特权

在 DB2 中，特权和权限既可以被显式授予，也可以被隐式授予。

如果一个具有 SYSADM 或者 SYSCTRL 权限的用户创建了一个数据库，则该用户将隐式获得该数据库上的 DBADM 权限。假设在创建数据库后，该用户的 SYSADM 或者 SYSCTRL 权限被撤销，该用户仍然拥有该数据库上的 DBADM 权限。如果要将 DBADM 权限从该用户身上撤销，应当使用 REVOKE 命令显式处理。

如果一个用户被授予了数据库上的 DBADM 权限，则该用户还将隐式获得该数据库上的 CONNECT 、CREATETAB 、BINDADD 、IMPLICIT_SCHEMA 和 CREATE_NOT_FENCED 特权。即使从该用户身上撤销了 DBADM 特权，该用户仍然拥有这些隐式授予的特权。如果需要限制该用户对数据库的存取，应当使用 REVOKE 命令显式处理。

如果一个用户被授予表上的 CONTROL 特权，则该用户将隐式获得该表上的其它所有特权 （包括 INSERT 、UPDATE 、DELETE 等），而且该用户还可以将除了 CONTROL 特权之外的其它所有特权授予其它用户。即使从该用户身上撤销了表上的 CONTROL 特权，该用户仍然拥有表上的其它所有特权，而且可以将这些特权赋予其它用户。如果需要限制该用户对表的存取，应当使用 REVOKE ALL 命令显式处理，具体命令如下：

REVOKE ALL ON tablename FROM username/groupname

具有 SYSADM 、DBADM 权限和某个数据库对象上 CONTROL 特权的用户都可以将某些权力授予其它用户，而且在使用 GRANT 命令授权时，如果指定了 WITH GRANT OPTION 选项，被授权的用户还可以将被授予的权力转授给其它用户。但是，如果一个用户给其它用户进行授权后自身的权力被撤销掉，对其他用户授予的特权将不受影响。换句话说， GRANT 和 REVOKE 命令没有层叠性的影响。

8.2.2 组特权

除了将特权授予数据库用户之外，还可以对用户组进行授权。用户组的定义不是在 DB2 中完成，而是在操作系统或其它外部安全设施中进行。

一个用户组可以包含一个或者多个用户。 组中的每个成员将拥有赋予该组的所有权限和特权。 另外， 组中的每个成员可能还会拥有单独赋予该用户的权限和特权。

DB2 有一个特殊的用户组： PUBLIC 用户组。该组包含所有 DB2 用户。如果将一种特权赋予了 PUBLIC 组，就意味着所有的数据库用户都享有该特权。

在创建数据库时，某些特权会被自动赋予 PUBLIC 组。比如，数据库上的 CONNECT 特权、系统编目表上的 SELECT 特权。对于一个安全性要求比较严格的数据库系统，应该注意 PUBLIC 组拥有的特权，并将不恰当的特权从 PUBLIC 组中撤销。

9 数据库对象和数据库设计

9.1 空值（ **Null** ）

空值代表未知的状态。因此，当包含空值的列参与运算时，结果是不可知的。在表定义的过程中，可以指定必须提供一个有效值， 这可通过在列定义中增加一个短语来实现。 CREATE TABLE 语句可以在每个列定义之后添加短语 NOT NULL ，这将保证该列包含确实的数据值，即非空。

在编制 DB2 的应用程序时，往往要对恰当使用空值进行特殊的考虑。 DB2 对空值的处理与对具体值的处理方式是不同的。

注意：关系型数据库允许空值。重要的是要记住，这些空值在你的数据库设计中是否恰当。

如果要定义一个不允许空值的列，可以通过在列定义的最后加上短语 NOT NULL 来实现，例如：

CREATE TABLE t1 (c1 CHAR(3) NOT NULL)

在上面的例子中， DB2 将不允许在 c1 列中存储任何空值。一般来说，除非是数据库设计的需要，应该避免使用允许空值的列。还必须考虑存储空间的额外开销。如果允许空值，每个列都将多占用一个字节。

如果在表中插入一行时省略了一个或多个列的值， 那么这些列或者将或者被置为空值 （如果这个列允许空值），或者被定义成默认值（如果曾经定义过）。如果这个列被定义成不允许空值，除非给该列提供有效的值， 否则插入操作将失败。 DB2 对于每一种 DB2 的数据类型都定义了默认值， 但是你可以为每一列提供默认值。默认值在 CREATE TABLE 语句中定义。通过定义自己的默认值，可以保证数据值被自动置为已知的值。

注意： DB2 的默认值可能不是你想要的！可在 CREATE TABLE 语句中定义默认值。

要保证在进行 INSERT 操作时使用了默认值， 必须保证在 INSERT 语句的 VALUE 部分给出 DEFAULT 关键字

9.2 标识（ **Identity** ）列

如定义标识列，在插入记录时，可由 DB2 自动生成数值序列或其它的值作为部分列值。

在大多数应用程序中， 表中有一个单一的列作为代表每一行的惟一的标识。 这个标识通常是一个数字， 在加入新记录时顺序更新。

DB2 可以自动生成序列值。

如果用 GENERTATED ALWAYS 定义列， INSERT 语句将无法为该列指定值。在默认的情况下，该域的数值将从 1 开始并以 1 作增量递增。范围和增量可以作为列定义的一部分来给定。

此外，默认值可以默认生成，这意味着用户可以选择是否为该域提供值。如果没有提供任何值（使用关键字 DEFAULT ），那么 DB2 将按顺序产生下一个数值。

标识列限定为数值型（整数或十进制数），而且在表中只能有一列。关键字 GENERATE 可以被用于其它列，但是它们不能作为标识列。

9.3 索引

索引是根据指定的一列或多列的内容对行进行排序。 索引主要用于提高查询效率， 但索引也可以用于逻辑数据设计。例如，主键不允许在同一列中输入相同的值，从而保证了没有一行数据是一样的。

创建索引有两个主要目的：

- 确保值的惟一性
- 提高查询性能

一个特定表上可定义多个索引，这有助于改善查询性能。但是索引越多，在更新、删除和插入操作时，数据库管理器就要做越多的工作去保持索引的同步更新。因此那些经常更新的表上创建大量的索引会降低处理速度。

9.3.1 惟一性索引和非惟一性索引

惟一索引保证表中一个或多个列的数据惟一性。在查询过程中使用惟一性索引可加快检索数据的速度。在插入或更新行的 SQL 语句之后将强制进行惟一性检查。在执行 CREATE INDEX 语句时也执行惟一性检查。若表中已有重复行，则不创建索引。

非惟一性索引通过维护数据的排序也能改善查询性能。

根据组成键码的列数，可分以下类型：

● 标量键：单键。

● 组合键：由两个或多个列组成

以下键类型用于实现约束条件：

● 惟一键：用于实现惟一性约束。惟一性约束不允许不同行的键列具有相同的值。

● 主键： 用于实现实体完整性约束。主键是一种特殊的惟一键。每个表只能有一个主键。主键列定义时必须带 NOT NULL 。

● 外键： 用于实现引用完整性约束。引用约束只能引用主键或惟一约束。外键的值只能是它所引用的主键或惟一约束中定义的值，或为 NULL 。（外键不是索引）

9.3.2 引用完整性和索引

主键保证记录的惟一性，使用索引来维护主键。含有主键的索引称为表的主索引，若没有给出约束名，DB2 生成的索引将命名为：

SYSIBM.SQL<timestamp>

主索引或惟一键约束索引不能直接删除，要删除主索引或惟一键约束索引，需使用 ALTER TABLE 语句。要删除主索引使用 DROP PRIMARY KEY 选项；要删除惟一键索引使用 DROP UNIQUE (约束名) 选项。

注意： DB2 使用惟一键索引和 NOT NULL 选项来维护主键和惟一键约束。

9.3.3 空值和索引

理解主键或惟一键约束与惟一性约束之间的区别是很重要。DB2 使用惟一性索引和 NOT NULL 约束这两个元素实现主键和惟一键的关系数据库概念。惟一性索引本身不强制实施主键约束，因为它允许有空值。空值是未知得知，但在索引中，不同行的空值看做是相等的。如果某列是惟一性索引键，不能在该列中插入两个以上的空值，这将违反索引的惟一性规则。

9.3.4 常用索引准则

索引要占用磁盘空间， 其占用的空间由键码列的长度以及被索引行的数目决定。随着更多的数据插入到表，

索引将不断增大。因此，当设计数据库的大小时，应为索引预留空间，关于索引应考虑的一些问题有：

● 对于主键和惟一键约束，系统会自动生成惟一性索引。
● 对外键建立索引常常是有益的。

使用控制中心能估计索引所占用的磁盘空间，若你用控制中心创建一个索引，你将看到 Estimate Size 按钮，点击这个按钮启动 Estimate Size 面板，向面板中输入总行数等信息，就能得知索引大小的估计值。

注意：一个索引键最大不超过 16 列，一个表允许的索引数目不超过 32767 个（或不超过储存器的容量）。索引的总长度不超过 1024 个字节（包括所有的开销）

9.3.5 仅访问（ Index Only Acess）索引（包括惟一索引）

INCLUDE 子句规定可以往索引键码中追加另外的列。对于某些使用只访问索引的查询，这些追加的列将有利于提高性能，但要求表中每行都是唯一的。该选项可以实现下述功能：

● 使更多的查询省去存取数据页的需要。
● 消除冗余的索引。
● 维护索引的惟一性。

看下面的例子。若在下面的索引上执行 Select empno, firstname, job From empno 语句 ,所需数据可通过索引查询得到，而不用读取数据页，这将改善查询性能。

```
CREATE UNIQUE INDEX EMP_IX
ON EMPLOYEE(EMPNO)
INCLUDE (FIRSTNAME , JOB)
```

9.3.6 双向索引

ALLOW REVERSE SCANS 索引创建子句可以实现前后两个方向的扫描。也就是说，既可按创建索引时定义的顺序查看索引，也可按逆序查看索引。这个选项可实现下述功能：

● 更好地使用 MIN 或 MAX 函数。
● 获取先前的键码。
● 免除为优化器反向扫描创建临时表
● 免除多余的逆序索引。

若索引按叶子页从左到右升序排列，则双向索引包含一个指向两个方向的叶子指针，即指向从左到右的相邻叶子页。因此，双向索引能从左到右（升序）或从右到左（降序）扫描。例如：

```
CREATE UNIQUE INDEX EMP_IX
ON EMPLOYEE(EMPNO)
NCLUDE (FIRSTNAME , JOB)
ALLOW REVERSE SCANS;
```

在这个例子中，若对　EMPLOYEE　表执行：

Select * from EMPLOYEE

order by EMPNO desc

因索引已使数据处于正确的顺序，所以　　DB2 不需将结果排序。

9.3.7 修改索引

只有通过重建索引的定义才能改变索引属性。例如，如果不删除先前的定义并重新创建一个新索引，　　就不能往键列表中添加新的列。可以用　　COMMENT ON　语句描述索引的用途。

如果要修改索引，必须先删除它，再重新创建新索引。不存在　　ALTER INDEX　语句。

9.3.8 删除索引

要删除索引，使用下面语句：

DROP INDEX EMP_IX;

注意：若程序包依赖于已删除的索引，它将变成无效。

9.3.9 索引指导

DB2 V8.1 索引指导 （Index Advisor ）是一个实用程序，它向用户自动推荐可创建的索引，索引指导的功能有：

- 得到某一问题的查询的最佳索引
- 为一组查询（工作负载）寻找最佳索引，这取决于给定的资源限制。
- 不用创建索引，就可以测试在一个工作负载上的索引。

可使用以下方法调用索引指导：

- 控制中心
- 命令 db2advis

9.3.10 群集索引

在群集索引下，数据在物理上按顺序在数据页上，重复值也排列在一起，因而在范围查找时，可以先找到这个范围的起末点，且只在这个范围内扫描数据页，避免了大范围扫描，提高了查询速度。

利用群集索引，　在表中插入和更新记录时，　DB2 以索引中主键的顺序在页面中维护着数据的实际顺序。　　群集索引大量提高了大范围查询的性能，这种查询拥有包含一个或多个群集索引主键的预测。在良好的群集索引的帮助下，只需要访问一部分表，当页面连续时，完成预读取的性能将更高。

9.3.11 扩展空间的预留

通常，数据库的数据总是在不断增长。当有一个请求需要插入一行到表格中时，就需要寻找一个有足够的空间来容纳一行的块。

PCTFREE 存储参数告诉 DB2 什么时候应该将数据块从对象的空闲列表中移出。如设置 PCTFREE=10 意味着每个块都保留 10%的空间用作行扩展，也就是说，一旦一个 INSERT 操作使得数据块的 90%被使用，这个数据块就从空闲列表中移出。

PCTFREE 的值较大意味着数据块没有被利用多少就从空闲列表中断开连接，不利于数据块的充分使用。PCTFREE 过小的结果是，在插入记录或更新时可能会出现数据记录迁移 (Migration) 的情况。即 PCTFREE 参数所指定的空间不够扩展，从而记录被强制迁移到新的数据块，DB2 需要频繁分配附加索引页，发生这种情况将较严重的影响数据库的性能。

注：虽然初始 PCTFREE 是在 CREATE INDEX 中指定的，但是 DB2 并没有 ALTER INDEX 语句。在 CREATE INDEX 中创建了初始 PCTFREE，但此后要用 ALTER TABLE 语句来调整它：

ALTER TABLE <tablename> PCTFREE <value1>

9.4 关键字

关键字是列的集合。它们可用来惟一标识一行。关键字可以通过组成它们的列或是它们支持的数据库约束来分类。

9.4.1 定义主关键字

有时候为每一个 DB2 表定义一个主关键字（简称主键）是十分有益的，这将确保每一行的惟一性。通过定义这个列为一个主关键字，DB2 将创建一个系统惟一索引。

主键可是一列或一组列，它具有唯一性和 NOT NULL 组合特性。在主键中，没有列可具有 BLOB 、CLOB 、DBCLOB 、LONG VARCHAR 或 LONG VARGRAPHIC 数据类型。

对一个表的主键非空约束说明与同一组列上建索引的重要差别是：如一个表有主键，它可在一个引用完整性（或称参照完整性）关系中，作为父表来使用。

9.4.2 定义惟一关键字

惟一关键字可以在一组列上强制实现惟一性。一个表中可以定义多于一个的惟一关键字。

注意：通常为主关键字或惟一关键字创建惟一性索引（如果还没有的话）。如果你定义了一个约束名，它将被用来命名该索引；否则，该索引将由系统自动命名。

9.4.3 定义外关键字

外关键字引用其它表中的数据值。一个外键约束（简称外键）说明了 2 个表之间的关系，这两个表分别称

为父表（或双亲表）和子表（或子女表）。

父表的行与子表的行之间 1：n 的关系。这种关系是基于父表中主键列的值与子表中一组列的值相匹配，该子表的一组值就称为外键。

只要外键约束有效， 系统保证： 子表在外键的全部列上具有非空值的每一行， 在父表中对应一行其主键值与它相匹配。

注意：外关键字约束在引用子句中总是和主关键字或惟一约束相关。

外关键字约束有不同的类型：

a） ON DELETE CASCADE: 父表行删除成功，子表中与外键相匹配的全部行也被删除；

b） ON DELETE SET NULL: 父表行删除成功，子表中所有与外键相匹配的列被置为空值；

c） ON DELETE NO ACTION: 父表行删除失败，所有改变将被回退。具有这种选项的约束是在所有瀑布式修改后，删除也已完成时，进行检验。

d） ON DELETE RESTRICT: 同 NO ACTION 选项，父表行删除失败，全部修改被回退。然而，具有 RESTRICT 选项的约束是在瀑布式修改和删除之前进行检验。

在表之间定义父子关系属于声明引用完整性，因为子表总是引用父表。这些约束在表的创建时或在使用 ALTER TABLE 的 SQL 语句时定义。 DB2 对于所有的 INSERT，UPDATE 和 DELETE 行为强制引用约束。

10 数据存储管理

能够高效地存储和快速获取大量数据是任何关系数据库管理系统的主要职责。 数据的物理存储会直接影响查询性能。

10.1 代码页

字符代码页和所有的 DB2 字符数据类型 （CHAR，VARCHAR ，CLOB，DBCLOB ）相关。在执行 CREATE DATABASE 命令期间，在数据库级设置代码页。

代码页是字符 -代码 |和数据库中它的二进制 -编码 ‖（表示）间的映射。一个 DB2 数据库只能够使用单一的代码页。 在执行带 CODESET 和 TERRITORY 选项的 CREATE DATABASE 命令期间， 就创建了代码页。代码页可以使用一个字节（一个字节可以表示 256 个不同的元素）或多个字节来表示一个字母数字。

象英语这样的语言包含相对较少的基本字符； 因此一个字节的代码页存储数据足够了。 像日文这样的语言需要多于 256 个元素来表示所有的不同字符，因此需要使用多字节代码页（通常是双字节代码页）。

代码点是在代码页中字符的位置。 如果应用程序和数据库没有用相同的代码页定义， DB2 将尝试转换代码页。

注意：二进制数据，如 FOR BIT DATA 列和 BLOB 列，和数据库代码页无关，不需要字符转换。

当一个 DB2 应用程序绑定到一个 DB2 数据库时，会比较应用程序和数据库的代码页。如果两者不同，则每条 SQL 语句都会试图转换代码页。 如果用户使用的代码页不同于所访问的数据库的代码页，重要的是确保两者是兼容的，可实现转换。

默认的情况下，数据库的排列顺序（ collating sequence）根据 CREATE DATABASE 命令所使用的代码集定义。如果指定 COLLATE USING SYSTEM 选项，则数据值将按照为数据库指定的 TERRITORY 进行比较。如果使用 COLLATE USING IDENTITY 选项，则所有的值都以二进制的形式逐字节地进行比较。

如果需要以原始格式（二进制）存储数据，要避免使用与代码页相关的数据类型。一般来说，最好使应用程序和数据库使用相同的代码页，以避免代码页的转换过程。

注意： DB2 UDB 版本 8 支持三种新的代码页：

- 阿拉伯代码页 425
- 拉丁 -1 宿主代码页 1047
- Unicode 3.1版本

许多代码页和代码页转换都已扩充以支持欧元符号。

10.2 表空间（ TABLESPACE ）

表空间是在数据库及存储在该数据库中的表之间的逻辑层。表空间在数据库中创建，表在表空间中创建。

由于 DB2 中所有的表都存放在表空间里面。这就意味着可以控制表数据的物理存放地点，可设计数据库使之能很好地适应硬件环境。比如，你可以选择一个比较慢的磁盘去存储那些较少被访问的数据。

备份和恢复可以在表空间级上执行， 这将使管理员能进行更细致的控制， 因为管理员可以单独地备份或恢复每一个表空间。

10.2.1 系统管理空间 SMS 和数据库管理空间 DMS

DB2 支持两种表空间：

- 系统管理空间（ SMS） --操作系统文件系统管理者分配并管理表空间所存储的空间。

在这种存储模式中， 通常包含许多文件， 这些文件表示了存储在文件系统空间中的表对象。 用户决定文件的位置， DB2 控制它们的名字，文件系统负责管理它们。每一个容器是操作系统文件空间中的一个目录。

SMS 是默认的表空间类型。

- 数据库管理空间（ DMS）--数据库管理者控制存储空间。从本质上说，这种表空间是为实现某特定目标而设计的文件系统，能最大程度地满足数据库管理者的需求。

此表空间的定义包括了一个附属于存储数据的表空间的设备或文件的列表。 每一个容器都是一个固定大小的预分配文件或者磁盘等物理设备。

当使用 CREAT TABLE 语句定义表时，可以显式的指定该表存放在哪个表空间。默认情况下，所有表都创建在 USERSPACE1 表空间。表空间为数据库管理员提供控制表、索引和大对象位置的能力。在一个数据库中可以定义任意数目的表空间，在表空间中可以定义任意数目的表。

下面这张表比较了 SMS 和 DMS 的主要特点。

特性	SMS	DMS
能够在表空间中动态地增加容器数目吗？	不能	能
能够把索引数据存放到不同的表空间中吗？	不能	能
能够把长型数据存放到单独的表空间中吗？	不能	能
表可以分散存放到多个表空间中吗？	不能	能
仅在需要时才分配空间？	是	否
表空间可以被定向到不同类型的磁盘空间吗？	是	否
创建之后，区段大小能够改变吗？	不能	不能

10.2.2 列出表空间信息

使用 LIST TABLESPACE 命令列出表空间基本的或者详细的信息。其语法如下：

LIST TABLESPACES [SHOW DETAIL]

用此命令显示的基本信息有：

● 表空间 ID，DB2 给这个表空间使用的内部 ID

● 表空间的名字

● 存储类型（ DMS 或者 SMS）

● 表空间类型，可以是常规（任何数据），长型或者临时

● 状态，一个指出当前表空间状态的十六进制值

如果 SHOW DETAIL 参数被使用了，通常就会显示以下的附加信息：

● 页面的总量

● 可用的页面数量

● 已经使用的页面数量

● 空白的页面数量

● 高水标（用页面作单位）

● 页面大小（用字节作单位）

● 区段大小（用字节作单位）

● 预读数量（用页面作单位）

● 区段大小（用页面作单位）

● 容器的数量

这些信息很重要，使用户知道表空间的使用情况和一些操作是否必要。

10.2.3 列出表空间容器信息

列出在一个指定表空间里的表空间容器基本的和详细的信息，使用 LIST TABLESPACE CONTAINERS 命

令。其语法如下：

LIST TABLESPACE CONTAINERS FOR TABLESPACE_ID [SHOW DETAIL]

使用此命令显示的基本信息有：

● 容器 ID

● 容器名字

● 容器类型

当使用 SHOW DETAIL 这个参数的时候，附加的信息如下：

● 页面总量

● 可用页面的数量

● 可访问（ yes 或者 no）

10.2.4 预取

预取是数据库管理程序的一个高级功能，它能在查询之前把需要的数据页读入缓冲区。这种异步读取可以减少执行时间。

在使用 CREATE TABLESPACE 或 ALTER TABLESPACE 语句的时候，可以改变 PREFETCHSIZE 这个参数来控制预取的大小。默认的情况是，这个值设置为 DFT_PREFETCH_SZ 数据库配置参数。当 DB2 触发预取需求时，这个值决定了多少页面将会被一次性读取。把这个值设置成区段大小的倍数，多个区段就可以被一次性同时读取。这个方法在表空间的容器分布在不同的硬盘上的时候更加有效。

队列里的预取请求的数量和 I/O 服务的数量可以用数据库配置参数 NUM_IOSERVERS 来控制，依靠配置这个参数，多个 I/O 服务可以同时取得数据。

10.2.5 编目（ Catalog）表空间

DB2 系统编目是一个 DB2 数据库存放所有 DB2 对象元数据的地方。编目表空间叫做 SYSCATSPACE 并且它的定义在执行 create database命令的时候产生。下面这些是在规划系统编目的时候应注意的问题：

● 系统编目由许多不同大小的表组成。在使用 DMS 表空间的时候，最少要用两个区段分配给每个表对象。依靠区段大小的选择，可以导致一个很大数量的已分配未使用的空间。如果使用 DMS 表空间，一个小的区段尺寸（ 2－4 页）将被选择，否则，一个 SMS 表空间应该被使用。

● 编目表使用大对象数据类型列，这些列不能放进缓冲池里面，只能在需要的时候从磁盘上读出。通过使用带文件容器的 SMS 表空间或者 DMS 表空间，可利用 LOB 数据类型的缓存。

● 建立越多的 DB2 对象，编目表就越大。如果使用 SMS 表空间 ,不能增加更多的容器，能做的一切就是使用操作系统的系统功能去增加底层文件系统的大小。如果使用 DMS 表空间，就可以增加容器。

10.3 创建数据库

下面是 create database命令的最简单的形式，注意，这是一个命令而不是一个 SQL 语句。

10.3.1 数据库创建

Create database DB2cert

使用此命令需要 SYSADM 或 SYSCTRL 的特权。它创建一组表空间， 表空间的特性可以通过命令 CREATE TABLESPACE 在外部指定，或使用缺省值。

一个数据库在创建时包含 3 个 SMS 表空间，它们是 SYSCATSPACE（供编目表使用），USERSPACE1（供用户数据使用）， TEMPSPACE1（供临时存储使用） 。附加的表空间可以在以后创建。每一个数据库都至少有一个表空间。

10.3.2 创建数据库的示例

下面是一个创建数据库的例子：

CREATE DATABASE DB2CERT DEF_EXTENT_SZ 4 CATALOG TABLESPACE MANAGED BY DATABASE USING (FILE ‘ C: CATALOG.DAT’ 2000,FILE ‘ D: CATALOG.DAT’ 2000) EXTENTSIZE 8 PREFRTCHSIZE 16 TEMPORARY TABLESPACE MANAGED BY SYSTEM USING (?C: TEMPTS’ , ?D: TEMPTS’) USER TABLESPACE MANAGED BY DATABASE USING (FILE ‘ C: USERTS.DAT’ 121) EXTENTSIZE 24 PREFETCHSIZE 48

上述命令建立一个数据库，这个数据库的特点包括：

● 一个默认的 4 个页面的区段大小。

● 一个 DMS 类型的系统编目表空间，有两个文件容器，每一个有 2000 页， 8－页的区段，预读大小是 16 个页面。

● SMS类型的临时表空间 TEMPSPACE1 ， 有两个容器。

● DMS类型的用户表空间 USERSPACE1，有一个容器， 121 个页， 24 页的区段和 48 页面的预读大小。

11 数据库维护

11.1 数据文件的格式和定界

每当数据取出或插入数据库时，都要特别注意对数据的格式进行检查。 DB2 DB2 提供了对以下数据格式的支持：

● DEL -定界 ASCII 文件
● ASC- 定长的 ASCII 文件
● IXF- 集成交换格式文件
● WSF- 工作表格式文件

11.1.1 定界 ASCII 文件

这种文件类型广泛用于关系型数据库管理系统（ RDBMS ）和其他一些软件包中，使用定界符分界。定界符是用来标识数据元素的起始和结束的字符，用在定界 ASCII （DEL ）文件中最主要的一些定界符包括：

● 字符定界符 － 它用来标志字符域的起始。在默认的情况下 ,DB2 用双引号（ ‖）作为字符定界符，数据库管理员 (DBA) 也可不用该默认字符，让 DB2 随意设置其它字符作为字符定界符。

● 列定界符 － 该定界符用来标志列的结束，默认的列定界符为逗号（，），但数据库管理员 (DBA) 可以选择使用其它符号。

● 行定界符－ 用来标识一行或者一个记录的结束。 DB2 用新行符 X?0A‘（通常用于 UNIX 操作系统标识一个新行）作行定界符，在 Windows 中， DB2 用回车 /换行符 X?0D0A‘ 作行定界符。

在 DEL 文件中，数据行是一个接一个存放在文件中，它们用行定界符分开。行中的域由列定界符分开。当选择 DEL 文件的行定界符和行定界符时，注意不要使用数据流中的字符。

11.1.2 非定界 ASCII(ASC) 文件

非定界 ASCII(ASC) 文件有时指的是定长的 ASCII 文件。该文件是由按行列组织的数据值的 ASCII 字符流组成。数据流中的行由回车符 /换行符或者新行符分隔，所有列值都是定长的（如果规定了记录的长度，就不需要行定界符了）。所有变长的字符类型都用空格填充到最大长度。没有列或字符定界符。

注：在非定界 ASCII 文件中列都靠在一起。

11.1.3 IXF 文件

集成交换格式（ IXF ）文件用于在 DB2 数据库中移动数据。比如，可以从宿主数据库输出数据文件，并将它导入到 DB2 服务器上的数据库的一个表。通常， IXF 文件包含连续的变长记录序列。根据数据类型数字值存储为压缩十进制或者是二进制。字符值以它们的 ASCII 表示存储，并且只存储变长字符类型的已使用部分。一个 IXF 文件同时存储表的定义和数据。

注：如果宿主表包含压缩域，在将文件转入 DB2 数据库前应转换这些域。为执行该转换，要为你所需要的所有列在宿主数据库中创建视图。视图自动将压缩域转换成字符域。通过这个视图，就可将所需的数据导出为 IXF 文件。

IXF 文件不能使用普通的文本编辑器来编辑。

使用这种格式类型的优点是表的定义包含在文件中， 这样当从文件中导入数据时表和它的索引都将和文件格式一起重建。

11.1.4 工作表（ WSF ）格式文件

Lotus 1-2-3 和相关产品使用这种文件格式类型导出或导入数据。 虽然不同版本在特定版本文件类型中增加了新的函数，工作表格式只使用为这些产品中的大部分版本所接受的函数子集。不使用这种格式将数据从一个 DB2 表移到另一个表。

11.2 数据移动工具

DB2 为数据的导入和导出提供了一整套的实用工具。使用这些工具可方便地将大量数据移入或移出 DB2 数据库。

这些工具为：

● 导出（ EXPORT ）

● 导入（ IMPORT ）

● 载入（ LOAD ）

11.3 导出命令 **EXPORT**

要使用所有默认选项将数据从表 NAMES 导出到 DEL 类型的文件 NAME.DEL ，可使用以下命令：

```
EXPORT TO name.del OF DEL SELECT * FROM NAMES
```

导出命令的一些关键参数包括如下：

● To 文件名 —— 指定数据导出的文件名。
● OF 文件名 —— 指定输出文件的数据格式： DEL， WSF， IXF 。
● METHOD N 列名 —— 指定用于输出文件的一个或多个列名，只对 WSF 和 IXF 文件有效。
● 选择语句 —— 指定返回导出数据的 SELECT 语句。
● MESSAGES 消息文件 —— 指定写入警告和错误消息的文件，假如不指定，这些消息就写到标准输出。

11.3.1 支持的文件格式

导出工具允许用户将数据导出到以下三个文件类型之一： IXF， DEL 或 WSF。
使用 OF FILETYPE 子句指定文件类型。例如，使用命令选项，可将 PRODUCT 表的数据导出到 IXF 类型的文件 PRODUCT.IXF 中（注：还可指定其它参数）：

```
EXPORT TO product.ixf OF IXF … SELECT… FROM product …
```

虽然导出命令不支持 ASC 文件类型， 用户仍可以通过 SELECT 语句修改 ASC 类型的数据文件。 例如， 有两列分别为定义为 CHAR （10）的 FIRST_NAME 和定义为 CHAR （5）的 EMP_NO，将它们导出到 ASC 类型的文件中，用户可指定 DEL 文件，并修改 SELECT 语句如下：

```
EXPORT… OF DEL … SELECT first_name CONCAT emp_no FROM …
```

命令的输出显示如下：

```
— George 10001 ‖
— Bill 10002 ‖
— Beverly 10003 ‖
```

当将数据导入到表中时，第一列的起始位置为 2，这样可忽略第一个位置的双引号（ → 字符。数字值可投射到字符类型然后连接到其它列。

通过使用 IXF 文件内的表定义， IXF 数据文件可用于在另一个数据库中创建表。 在导出到 IXF 文件时， 通过修改列名，在文件导入目标数据库时后续建立的表可用期望的列名创建。

11.3.2 导出大对象数据

当导出表包含 LOB 数据类型时，用户有两个选项决定 LOB 值如何导出。LOB 数据的头 32KB 可包括在带常规表数据的目标文件中，或 LOB 数据可以分开存储，每个 LOB 数值有自己的一个存储文件。

如果用户决定使用第一种选项，则导出命令中不需指定额外的参数。但是，这种方式只有 32KB 或更少的 LOB 被导出。如果 LOB 值的大小超出 32KB，它就被截断，只有头 32KB 才被导出到文件中。

如果导出工具将每个 LOB 值导出到分别的文件中，用户必须指定文件应放在哪里，文件名是什么。这可以通过指定三个参数来实现：

- LOBSINFILE ——文件类型修饰语，当指定时，告诉导出工具所有的 LOB 值导出到分别的文件中（ DB2 可以连接多个 LOB 在一个文件中）。如果不指定这个参数，导出工具将忽略所有将 LOB 导出到分别的文件的所有其它参数。
- LOBS TO lob-path—— 指定 LOB 值作为单个文件导出的一个或多个路径。如果指定的第一个路径已满，则使用第二个路径，依次类推。当指定路径名时，确保在 Windows 中以 一|字符结束，在 UNIX 中以 一/结束。
- LOBFILE 文件名 —— 告诉导出工具导出 LOB 的文件名。为了确保惟一性，DB2 将产生三个数字符号扩展到文件名中。

从表 EMP_PHOTO 导出 LOB 数据到 EMP_PHOTO.DEL 文件，它的 LOB 值放入目录 D:\LOBDIR1 和 D:\LOBDIR2 ，LOB 文件名为 EMP_PHOTO_LOB ，导出命令如下：

```
EXPORT TO Emp_photp.del OF DEL LOBS TO
D:\LBDIR1\,D:\LOBDIR2
LOBFIEL Emp_photo _lob … MODIFILED BY LOBSINFILE
SELECT * FORM Emp_photo
```

第一个导出的 LOB 文件命名如下：

```
d:\lobdir1\emp_photo_lob.001
d:\lobdir1\emp_photo-lob.002
d:\lobdir1\emp_photo-lob.003
…… 等等 ……
```

如果 D:\LOBDIR1 目录满了，就使用 D:\LOBDIR2 目录。

11.4 导入工具 IMPORT

导入工具将数据从一个输入文件插入到一个表或视图中。如果表或视图中已包含数据，可以将数据替换或添加数据。使用导入工具，可以规定如何对目标表增加或替换数据。

使用该工具前你必须连接到数据库，如果你想使用 CREATE 选项将数据导入新表，你必须有 SYSADM 或 DBADM 权限或者对数据库有 CREATETAB 特权。要替换表或视图中的数据，你必须有 SYSADM 或 DBADM 权限或者对该表或视图有 CONTROL 特权。如果要对已有的表或视图增加数据，你必须有该表或视图上的 SELECT 和 INSERT 特权。

注：如果已有的表包含主键或惟一索引，该主键或惟一索引被另一表中的外键引用，数据不能被替换，只能添加。

11.4.1 使导入命令

使用所有默认选项，将数据从 DEL 类型的 Names.del 文件导入到名为 NAMES 的空表中，可使用以下命令：

```
IMPORT FORM NAMES.DEL OF DEL INSERT INTO NAMES
```

图 10-8 简单导入

导入命令的一些关键参数如下：

- INSERT——不修改当前数据向数据表增加新的行。
- INSERT_UPDATE—— 向目标表增加新的行，或更新匹配主键的已有行。
- REPLACE—— 从表中删除所有已有的数据并插入导入数据。
- REPLACE_CREATE—— 如果表不存在，创建表和索引定义，否则，从表中删除所有已有的数据，并插入导入数据，不修改表或索引定义。
- COMMITCOUNT n—— 每导入 n 条记录就提交一次。
- CREATE—— 创建表定义并插入新行。
- IN 表空间名 —— 指定创建表所在的表空间。
- INDEX IN 表空间名 —— 指定创建表上索引所在的表空间，只适用于 DMS 表空间。
- INTO 表名 —— 指定数据导入的表。
- METHOD L |N ——L 指定导入数据的起始和终止列号（从 1 开始计数，仅适用于 ASC）。 N 指定导入的列名（仅适用于 IXF ）。

11.4.2 支持的文件类型

支持的文件类型有 IXF ， DEL ， WSF 和 ASC。为指定文件类型，使用 OF FILE-TYPE 子句。例如，可使用以下命令导入类型为 IXF 的 PRODUCT.IXF 文件：

```
IMPORT FROM PRODUCT.IXF OF IXF··
```

11.4.3 导入大对象

LOB 可以作为导出文件中表数据的一部分或在分别的文件中被导出。如果 LOB 数据和其它表数据放在同一文件中，它用导入工具导入时不需额外的参数。

如果 LOB 值导出到分别的文件中，导入工具需要：放置 LOB 文件的一个或多个目录以及 LOB 文件名。因为 LOB 文件的文件名存放在导出文件中， 所以用户不需指定它们。 要指定源目录， 使用 LOBS FROM 选项，后面跟着用作源目录的一个或多个目录。然后指定 LOBSINFILE 文件类型修饰语。如果没有 LOBSINFILE 修饰语， LOBS FROM 选项就被忽略。

例如，表 EMP_PHOTO 导出到 DEL 类型的文件 EMP_PHOTO.DEL 中，它的带文件名 EMP_PHOTO_LOB.XXX 的 LOB 文件导出到两个目录中。 EMP_PHOTO.DEL 文件拷贝到另一个数据库服务器上。源数据库服务器上第一个目录中的 EMP_PHOTO_LOB.XXX 文件拷贝到 D:\LOBDIRA 目录中，第二个目录中的 EMP_PHOTO_LOB.XXX 文件拷贝到 D:\LOBDIRB 目录中。要将数据导入到目标数据库中的 EMPLOYEE_PHOTO 表中，在导入命令中应指定以下选项：

```
IMPROT FROM EMP_PHOTO.DEL OF DEL
LOBS FROM D:\LOBDIRA\,D:LOBDIRB\
MODIFIED BY LOBSINFILE …
```

注：在指定路径时， Windows 路径的最后必须包括 ‖\‖字符， UNIX 路径的最后必须包括 ‖/‖字符。

11.5 载入工具 LOAD

载入工具，与导入工具类似，将输入文件中的数据移入目标表中。与导入工具不同的是，在开始载入过程前，目标表必须已经存在在数据库中。

目标表可以是载入前刚创建的， 或是已经存在的表，在该表中数据可以被加入或替换。 该表上的索引可以已存在也可以不存在。载入过程不创建新索引 —— 它只维护表上已定义的索引。

对于 DBA 来说，导入和载入工具的最重要的不同点在于性能。导入工具每次插入一行数据。插入的每行都要被检查是否满足约束条件（如外键约束或表检查约束）。修改也记录在日志文件中。

载入工具向表插入数据的速度要比导入工具快得多， 它不是每次插入一行， 而是使用输入文件读出的行建造页，这些页直接写入数据库。已有的索引（包括主键或惟一索引）可在数据页插入后重建（可全部重建或基于新加的数据部分重建），最后所有不遵从惟一性或主键约束的重复行都从表中删除。在载入期间，单个记录的载入不记录在日志文件中。

因为载入工具修改不记入日志，所以不能使用日志文件前滚。但是，载入工具可以使用源数据的备份（载入备份），因此在需要时可以重新载入。

使用载入工具需要 SYSADM，DBADM 权限，或在数据库上有 LOAD 权限且在表上有 INSERT 特权。如果选择替换模式， 终止模式（终止前一个载入替换操作） ，或重启动模式 （重新启动前一个载入替换操作） ，则在表上还需有 DELETE 特权。

在使用载入工具之前， DBA 必须理解它是如何工作的，探究它的性能优势，并在不正确使用时避免可能发生的任何问题。

注：载入操作现在是表级操作。这以为着不再需要独占整个表空间，并且在载入操作期间，在可以并发存取同一表空间的其它表对象。

11.5.1 插入模式

通过使用带 INSERT 选项的载入命令，输入文件的所有数据都作为新数据插入到已有的表中。例如，将

PRODUCT.IXF 文件中的数据载入到表 PRODUCT 中，用户可指定以下命令：

LOAD FROM Product.ixf OF IXF … INSERT INTO Product …

11.5.2 替换模式

使用 REPLACE 选项，载入工具在载入新数据之前将已有表的所有数据删除。但不重建表。例如，要将 PRODUCT.IXF 文件中的数据替换 PRODUCT 表中的数据，用户可使用以下命令：

LOAD FROM Product.ixf OF IXF … REPLACE INTO Product …

11.5.3 终止模式

有时需终止载入进程； 例如，用户装载了错误的文件或调用载入工具时使用了错误的选项。 要终止被打断的载入操作，用户必须再次调用载入工具，使用的选项与第一次相同，只是将 INSERT 或 REPLACE 选项替换成 TERMINATE 选项。例如，如果载入工具以如下选项调用，然后被打断：

LOAD FROM Product.del OF DEL SAVECOUNT 500

INSERT INTO Product

载入操作可通过将 INSERT 选项替换为 TERMINATE 来终止：

LOAD FROM Product.del OF DEL SAVEPOINT 500

TERMINATE INTO Product

在原始载入用 INSERT 选项调用的情况下， 所有载入到表中的新行都将被删除， 表的状态恢复到调用载入工具之前。

注：在该版本中引入了 LOCK WITH FORCE 选项。这可以强制应用释放它们在表上的锁，使得载入操作获取它所需的锁继续运行下去。

如果原来的载入用 REPLACE 选项调用，那么表中所有的记录，不管旧的还是新的，都将被删除。

11.5.4 重启模式

如果载入操作失败， DBA 应查看消息文件和错误日志来判断失败原因。一旦找到并修复了错误，用户就可以调用带 RESTART 命令选项的载入工具。注意重新调用时必须使用与第一次调用时相同的选项，只是将 INSERT 或 REPLACE 选项替换成 RESTART 命令选项。

如果在载入阶段发生了载入失败，并生成一致性点，可从最后一个一致性点重启载入阶段。
如果在构建或删除阶段发生载入失败，可从有问题阶段的开始重启（例外情况是在线载入或指定了 COPY YES，即使在删除阶段发生失败，重启仍从构建阶段开始。）。

11.5.5 索引模式

在创建阶段，在载入命令执行之前的索引可以被重新创建。用户可以定义如下四种方式：

● Rebuild 这种方式下，不管载入的类型是 INSERT 还是 REPLACE，都会重建索引。

● Incremental 这种方式下，载入命令将使用已存在的索引数据，并在用 INSERT 方式调用载入时将新的数据项加入到现在的索引中。这种方式适用于要载入的数据比现有数据少得多的情况。

● Deferred 这种方式下，载入命令不会重建索引，而是索引将被设置为需要刷新。索引会在接下来的第一次访问或数据库重启时重建。在数据配置文件的 INDEXREC 参数里设定的值决定索引什么时候重建。这种方式不能用于有主键索引或惟一索引的表上，因为必须检查这些表是否有重复的关键字。

● Autoselect 允许载入命令在 REBUILD 和 INCREMENTAL 方式中选择。

例如：一个 PRODUCT 表定义了两个索引，且无惟一键和主键的限制，调用载入命令来替换一些已存在的数据，下列命令行将使索引的重建按 Deferred 方式进行：

```
LOAD FROM … INDEXING MODE DEERRED …
REPLACE INTO Product …
```

如果在载入命令完成后，DBA 需要立刻添加一些数据到 PRODUCT 表，下列命令行可以且用来重建索引：

```
LOAD FROM .. INDEXING MODE REBUILD …
INSERT INTO Product …
```

当连续调用载入命令操作同一个表时，索引的创建将被推迟到最后一个载入命令完成之后。

11.5.6 SET INTEGRITY 语句

一个数据库的数据完整性和正确性是至关重要的。在 DB2 里，约束和规则自动被强制执行。例如，无论何时插入、更新、导入或者载入一个行的时候，将被检查下列内容：

● 数据格式化和长度的正确性

● 主键值

● 惟一约束

无论何时被插入、更新、输入一个行，也将被执行下列检查：

● 外键约束

● 检查约束

● 更新用 IMMEDIATE CHECKED 选项定义的汇总表

当数据载入一个表的时候，不检查或执行上面的第二列。因为没有对它进行检查，所以数据的正确性将受到质疑，直到他们被检查和发现一致为止。由于这个理由，DB2 限制对表的访问，把它置于检查挂起状态。这将防止表里的数据被访问，并且指出需要对数据进行显式检查。

如果在这个表上定义了一个新的约束并且对约束的检查不立即执行，一个表可能被置于这种状态。可以做到这一点，因为不保证表里已经存在的行遵循新的约束。

表中存在的数据的合法性可以用如下语句之一来检验：

● SET INTEGRITY（也就是： SET CONSTRAINTS ）

● REFRESH TABLE

SET INTERGRITY 语句可以用在下列情况下：

● 把表设置成检查挂起状态并且把下列一个或者多个情况标记为不检查：检查约束，外键约束和立即更新汇总表的刷新。把以上任意一种情况标识为不检查情况以后，表就被设置成检查挂起状态。不能关闭主键和惟一键约束检查并且将强迫执行。因为表已经设置成检查挂起状态，对表数据的访问是受限制。

● 检查下面的部分或全部约束把表重置为检查挂起状态：检查约束，外键约束和立即更新汇总表的刷新。

● 通过把下面的约束的任一个标记为已经检查来重新设置检查挂起状态：检查约束，外键约束和立即更新汇总表更新。当前的约束检查将不会执行。这将被当作延期检查查阅。

RERESH TABLE 语句只能在汇总表里使用。它将使用表中的当前的数据更新汇总表里的数据。

注：在版本 8 之前，如果目标表包含生成列，在载入操作之后，目标表保持为检查挂起状态。现在载入工具可以生成列值， 并且不需在载入操作后对包含生成列及没有其他表约束的表发出 SET INTEGRITY 语句。

如果检查存在的数据， 用户可以指定检查应该增量地进行。 这意味着只有那些在检查关闭以后被加入表的数据才会被检查。

11.6 数据维护

表中数据存储的物理分配形式对于使用这个数据库的应用程序的性能有显著影响。 这种影响的形式体现在表的更新，插入和删除操作中。例如，一个删除操作可能留下以后不能再使用的空的数据页面。对各种长度的字段进行更新可能导致新的字段值不适合在同一个数据页面里存放。

数据库管理员，可以使用 DB2 提供的数据管理命令去优化表中数据的存储物理分配形式。有三个有关的程序或命令能帮助组织表中的数据，它们是：

● REORGCHK
● REORG
● RUNSTATS

11.6.1 分析数据的物理组织

REORGCHK 工具用来分析系统编目并且收集你的表和索引的物理组织的信息。用户可使用一些选项，这些选项可以使 REORGCHK 工具在使用时使用当前的编目信息或者在使用前更新它们。

有了从系统编目表里面收集的信息， REORGCHK 工具显示了表和索引的空间分配属性。此工具使用一些规则来帮助你决定你的表和索引是否需要物理重组。

REORGCHK 命令的一些关键参数包括如下：

● UPDATE STATISTICS—— 更新表的统计数据，根据该统计数据判断是否需要重组表。
● CURRENT STATISTICS —— 根据当前表统计数据判断是否需要重组表。

要使用 REORGCHK 工具，必须具有以下权限之一： SYSADM ， DBADM 或表上的 CONTROL 特权。下面是一个使用 REORGCHK 工具的例子。此工具将在表 DB2CERT.CANDIDATE 上执行。

REORGCHK UPDATE STATISTICS ON TABLE DB2CERT.CANDIDATE

REORGCHK 的输出分成两个部分。 第一个部分显示表的统计信息和规则 ,第二部分显示关于表的索引和相关规则的信息。

- SCHEMA-字段指出表所属的模式。
- NAME一字段指出正在被 REORGCHK 工具分析的表的表名。 REORGCHK 能同时分析一批表。
- CARD一指出在基表里的数据行的数量。
- OV(OVERFLOW) -溢出指示器。它指出了溢出的行数。在一个新的字段被加入到一个表或者是一个可变长度的值增加了它的尺寸时可能发生溢出。
- NP(NPAGES)-指出有数据的页面的总数。
- FP(FPAGES)一指出分配给表的页面的总数
- TSIZE-指出表的大小（使用字节为单位）。这个值可用表中的行数乘于行的平均长度得到。
- TABLEPAGESIZE 一指出放置表的表空间的页面大小。
- REORG一字段对前三个规则的每一个规则均有一个指示器。连字符（ -）指示不推荐重组，星号（ *）指示推荐重组。

规则 F1， F2 和 F3 提供表重组的方针，这些规则显示在 REORGCHK 的输出里。

- F1处理溢出行的数量，在有 5%或者更多的行溢出时会建议对表进行重组。
- F2处理空闲和未使用的空间上，如果表大小（ TSIZE ）小于或者等于分配给这个表的总空间的 70%的时候，会建议对表进行重组
- F3处理空白页面，当表里多于 20%的页面是空闲的时候，建议对表进行重组。如果一个页面中没有数据行，就被认为是空白的。

注： REORGCHK 工具是由 DB2 提供的分析工具。要重组表，必须使用 REORG 程序。

可以使用 REORGCHK 工具的 CURRENT STATISTICS 选项来使用那些系统编目表的统计资料。 例如， 要分析当前的表 DB2CERT.TEST_TAKEN 的统计资料，可以使用：

REORGCHK CURRENT STATISTICS ON TABLE DB2CERT.TEST_TAKEN

要回顾当前一个数据库里所有表的统计资料，包括系统编目表和用户表，可以使用：

REORGCHK CURRENT STATISTICS ON TABLE ALL

也可以使用 SYSTEM 选项检查系统编目表的组织。另外，可以指定 USER 关键词来选择在当前用户模式名下所有的表。

REORGCHK CURRENT STATISTICS ON TABLE SYSTEM

如果不指定 CURRENT STATISTICS 参数， REORGCHK 将调用 RUNSTATS 工具。

注： REORGCHK 不能在 -控制中心 |中被运行，可以使用 一命令中心 |来运行 REORGCHK 命令并捕获输出以供分析。

11.6.2 表重组

在使用 REORGCHK 工具之后，如果表需要物理重组，可使用 REORG 命令进行。

注：在使用 REORG 命令的时候，必须有下列权限之一： SYSADM,SYSCTTRL,SYSMAINT 或者是表的控制权限。

REORG 工具将删除所有没有使用的空间并且把表和索引数据写入相互临近的页面中。

REORG 命令的一些关键参数如下：

- ALLOW READ ACCESS — 在重组期间允许对表进行读。
- ALLOW WRITE ACCESS — 在重组期间允许对表进行写。
- RESUME — 重新开始先前暂停的表重组。

REORG TABLE DB2CERT.TEST_TAKEN

注：当使用 REORG 的时候，它强制使用表的全名

注： DB2 可以执行在线 REORG 并支持对整个表的读 /写操作，除了小部分正在被重组的表之外。另外，重组还可 -就地 执行，这样在操作期间就不需要分配大量的临时空间。这种操作只支持类型 2 索引。

11.6.3 生成统计信息

表重组可以帮助应用程序提高性能。 必须接着做一些额外的步骤。 这些操作将使你的现有的应用程序从表重组当中得到好处。

1）在表和索引中使用 RUNSTATS 命令 这将给优化程序提供每个表和索引的物理组织的最新信息。

2）在访问重组表的包上使用 REBIND 程序 这将使应用程序从优化器为提高 SQL 语句的访问策略所作的任何改变中得到好处。

11.6.3.1 RUNSTATS 工具

系统编目表包含关于字段，表和索引的信息。它们包含诸如表中行的数量、表或者索引的空间使用、在字段中不同的值的数量等信息。

无论如何，这些信息不被当即保存，它由 RUNSTATS 工具生成。

RUNSTATS 收集的统计信息能给 DB2 提供信息优化器，以便在执行 SQL 语句的时候选择最佳访问路径。为了获取对数据的有效访问路径， 反映你的表、字段和索引的实际状态的当前统计信息必须存在。 不管何时，执行了一个动态的 SQL 语句， DB2 优化器就读取系统编目表来检验可用的索引，每一个表的大小，字段的属性和其它信息。最后，为执行此查询选择一个最佳的访问路径。

如果统计信息没有体现当前的表状态， 在执行查询的时候， DB2 优化器将无法找到正确的信息来为最佳访问路径做出最好的选择。建议在一个频繁进行大量更新、插入或者删除操作的表上执行 RUNSTATS 。对一个有大量插入删除操作的表，可以决定在一个固定的时间或者在插入和删除操作之后进行统计。

注：建议在重组一个表以后使用 RUNSTATS 工具。

DB2 的一个重要的特征是它允许你重组并在系统编目表里使用 RUNSTATS 工具。DB2 的这个特性能提高查询系统编目的存取。当你执行一个 SQL 语句的时候， DB2 可以访问这些表。因此，得到当前系统编目表的统计信息是很重要的。

要使用 RUNSTATS 工具必须有以下权限之一： SYSADM,SYSCTRL,SYSMAINT ，表上的 DBADMCONTROL 特权或者载入权限。

RUNSTATS 工具没有任何输出显示，只能查询系统编目表来看到它的结果。
例如用 RUNSTATS 更新的数据：

TABSCHEMA TABNAME CARD NPAGES FPAGES OVERFLOW

--------- ------------------ ----------- ----------- ----------- -----------

SYSIBM SYSCOLUMNS 927 32 32 1

SYSIBM SYSTABLES -1 -1 -1 -1

这个输出是在选择了 SYSSTAT.TABLES 编目视图的所有的列后获得的，值 -1（负 1）指出对于这个对象没有可用的统计信息。这些 SYSSTAT.TABLES 视图中的字段和 REORGCHK 工具中具有同样的含义：

- CARD-指出表中数据行的数量
- NPAGES-指出包含数据的页面总量
- FPAGES-指出分配给这个表的页面总量
- OVERFLOW-指出溢出行的数量

11.6.3.2 Rebind 工具

REBIND（重新绑定） 工具支持利用系统编目表里的信息重新建立一个包。 这些允许 SQL 应用程序通过优化器选择去使用不同的访问路线。

建议在运行了 REORG 或者 RUNSTATS 之后运行 REBIND 工具。DB2 SQL 优化器将使用新的组织形式和最近收集的统计信息去产生一个访问路径。此访问路径将更好的适合数据的新的物理组织形式。

例如，设想有一个应用程序名为 DB2CERT，使用的表名为 DB2CERT.TEST_TAKEN ，已经给这个表建立了一个新的索引，并且需要使用 REBIND，使 DB2CERT 程序可以在访问数据的时候使用新的索引。

REBIND DB2CERT

如果对数据库进行了很大修改并且更新了所有的统计信息。可能希望为所有使用数据库的应用程序重新建立包。在这个情形下， 你需要使用 DB2RBIND 工具。DB2RBIND 工具允许对所有当前 SYSCAT.PACKAGES 编目视图中的包作显式的重新绑定。下列命令重新绑定在 DB2CERT 数据库上所有的包。并且在

DB2RBIND.LOG　中记录结果。

```
DB2RBIND DB2CERT /L DB2RBIND.LOG
```

12 数据库恢复

任何一种数据库管理系统的一个基本功能就是可以恢复那些可能使数据库的完整性遭到破坏的故障。在数据库恢复领域中主要的工具有　BACKUP　和 RESTORE。

12.1 恢复的类型

恢复可能是以下类型中的一种：事故恢复　(crash),版本恢复 (version)，或是前滚恢复　(roll-forward) 。

1）事故恢复

事故恢复可以保护数据库不至于陷入一个不一致或者不可用的状态。　数据库的工作单元可能被意外地中断。例如，在一个工作单元中的部分变更被完成和提交之前发生了电源故障，数据库就会停留在一个数据不一致的状态中。

通过撤消未被授权提交的事务处理　,数据库可以再次回到一致状态。　命令 restart database可以实现这个功能。如果数据库配置参数　autorestart 被设置成　on(默认方式)，那么在事故发生后，　首次到数据库的连接将会启动 restart database的过程。

2 ）版本恢复

版本恢复允许数据库恢复到使用　backup 命令建立的数据库的映像或以前版本。　这种恢复机制通过使用一个以前建立的数据库备份恢复出一个完整的数据库。一个数据库的备份允许你把数据库恢复至和这个数据库在备份时完全一样的状态。　而从备份建立时刻到日志文件中的最后记录的所有的工作事务单元将全部丢失。（这些可以利用前滚恢复来修复。）

这种版本恢复方式要求每一个将来你希望恢复的版本都要有一个完整的备份。　在 DB2 中默认支持这种版本恢复方式。

3）前滚恢复

这种技术是版本恢复的一个延伸，它利用完整的数据库备份和日志相结合，可以使一个数据库或者被选择的表空间恢复到一个特定的时间点。

如果从备份的时刻起到现在所有的日志文件都可以获得的话，那么通过使用一个作为基线的完整的数据库备份，　你可以让　DB2 在这个数据库上的任意或者全部的表空间执行所有的工作事务单元，　可以执行到日志上涵盖到的任意时间点。

前滚恢复在　DB2 里被指定在数据库级别中进行，并在配置中要被明确激活才能生效。

注：当编目表空间　SYSCATSPACE 用于表空间的前滚恢复中时，它必须被前滚到日志的末尾。而作为数据库前滚恢复的一部分它可以被恢复到时间上的某一点。

12.2 恢复策略

1) 可恢复和不可恢复数据库

在 DB2 中可恢复的数据库可以利用 logretain 和 userexit 两个数据库配置参数是否被设为激活来辨别。这意味着可恢复的数据库中事故恢复、 版本恢复和前滚恢复技术都是可用的。 不可恢复的数据库不支持前滚恢复。

一个数据库是否需要设置为可恢复的主要取决于以下一些因素：

● 如果该数据库仅仅是用来进行查询，日志中没有工作事务单元，那么该数据库就没有必要被设置为可恢复的。

● 如果该数据库的数据较稳定 (数据很少变更)，而且数据可以很容易地重建，那么也许不必考虑去建立一个可恢复的数据库。

● 含有不易重建的数据的数据库应该配置为可恢复的数据库。

● 如果存在大量数据更新的行为，应该考虑使用一个可恢复的数据库。

2) 联机和脱机访问

当执行各种数据库恢复行为时，数据库有联机（ online ）和脱机（ offline ）两种情况。

联机表明当进行数据库恢复时其他的应用程序仍然可以访问该数据库。 相反地， 脱机表明当进行数据库恢复时其他的应用程序不能访问该数据库。

联机和脱机访问并不是在所有的恢复行为中都可用的。

12.3 使用日志文件

所有的 DB2 数据库都有其关联的日志文件，这些日志文件保存了所有对数据库对象和数据进行更改的记录。所有的数据变更先被写到内存的记录缓冲区里， 在事务提交时再把数据变更写入硬盘上的记录文件中。

例如，在像突然断电之类的意外灾祸发生时，日志文件将被用来使数据库回到一致状态。按照日志文件，所有的工作单元都要被重新执行，还没有提交的工作单元将被回滚。

日志文件有默认的、固定的大小，所以，当一个日志文件写满的时候，将在另一个日志文件上继续记录。

注：为了保持数据库的完整性， DB2 将那些对数据进行更新、 删除、 或者插入的操作首先记录到日志文件里，然后再写入到相关的数据库里。

12.3.1 日志缓冲区

在写入日志文件之前，日志记录会先被写入缓冲区。这些缓冲区的大小范围在 4 个到 512 个 4KB 的页之间，默认的设置是 8 页。在数据库中定义日志缓冲区大小的配置参数被称为 LOGBUFSZ 。日志缓冲区的内存从一个称为数据库堆的内存区域里分配，堆的大小是由数据库中的配置参数 DBHEAP 控制。

当有下列情况发生，日志记录会被写入磁盘：

● 根据数据库配置参数 MINCOMMIT 所定义的一项事务的提交或一组事务的提交时。

● 日志缓冲区已满时。

● 由于其他一些内部数据库管理事件的发生而导致的结果。

对日志记录的缓冲将会使日志文件的 I/O 操作更有效率，因为这样就会减少日志记录写入磁盘的频率，而在每次的 I/O 操作中写入更多的日志记录。如果在系统中存在大量的日志记录行为，可以增大日志缓冲区来提高系统性能。

12.3.2 主日志文件和辅助日志文件

日志文件有两种：主日志文件和辅助日志文件。

● 主日志文件会建立一个分配给恢复日志文件的固定大小的存储空间。这些文件会在第一次连接到数据库的时候被预先分配空间。数据库中的配置参数 LOGPRIMARY 决定主日志文件的数量，参数 LOGFILSIZ 指明文件的大小。

● 辅助日志文件会在主日志文件写满的时候根据需要一次被分配一个 (直到数据库配置参数 LOGSECOND 规定的数目) 。辅助日志文件的大小也由参数 LOGFILSIZ 指定。

如果数据库定期会需要大数量的日志空间, 那么建议使用辅助日志文件。 例如在数据库应用中有一个需要大量的日志空间，超出主日志文件的大小，但每月只运行一次的应用程序。

12.3.3 日志的类型

在 DB2 中可能出现两类日志：循环日志和归档日志。

● 循环日志支持不可恢复的数据库。这种类型的日志只使用活动日志文件（以下会详细介绍）。主日志文件和辅助日志文件的使用如上所述。一个日志文件将它的所有事务都提交或回滚后，就可以被重新使用。使用这种记录日志的方法不能进行前滚恢复, 但可以进行事故恢复和版本恢复。 这是 DB2 数据库创建时的默认日志记录方法。

● 归档日志是一种日志管理技术，其中的日志文件处于非活动状态时便将它们做归档处理。归档日志不是默认的日志方法。它是惟一支持前滚恢复和实现可恢复数据库的日志方法。

日志文件可按如下分类：

● 活动日志－－这些日志文件包含了那些尚未提交 (或是回滚)的事务单元的相关信息。还包含了那些已经提交但改动尚未写入数据库文件的事务的相关信息
活动日志用于事故恢复，以防止导致数据库系统数据不一致的错误。 RESTART DATABASE 和 ROLLFORWORD 命令（用于到时间点的恢复或者到日志结尾的恢复）使用活动日志来进行重新执行或回滚事务，以使得数据库恢复到一个数据一致的点上。

注： DB2 支持数据库级的日志镜像。 DB2 配置参数 MIRRORLOGPATH 允许数据库将日志文件的相同拷贝写到不同的路径。

● 联机存档日志－－当活动日志中记载的所有改动对正常的处理进程来说已经不再需要时，活动日志将关闭，并成为一个存档日志。
这些日志文件中包含了那些重启恢复时不需要的、已经完成的事务的相关信息。 称它们为 ┤联机 ├是因为它们和活动日志存放在相同的子目录下。

● 脱机存档日志－－这些日志文件已经被从活动日志所在的目录中移开了。移动这些文件的方法可以是手

动的，也可以是通过诸如被用户退出这样的过程调用来自动进行。存档的日志可以简单地通过把它们移到另外的目录或是存储到磁带或其他介质上而成为脱机模式。它们也可以通过外挂的存储管理工具来管理，例如 IBM 的 ADSM 产品。

有两个数据库配置参数 LOGRETAIN 和 USEREXIT 允许配置数据库是否使用存档日志模式。

当数据库配置参数 LOGRETAIN 被设置为激活后，日志文件变成非活动时不会被删除。 USEREXIT 参数被设置为激活后， 当一个日志文件可以存档时 DB2 将调用用户退出程序。 数据库名字， 日志文件的路径以及其他参数将被传递到该程序中。

如果用户退出选项被激活，可以选择发出 ARCHIVE LOG 命令强制日志归档。

注：默认状态下， LOGRETAIN 和 USEREXIT 的设置是关闭的，这样循环日志是缺省的日志方式。

前滚恢复可以使用联机存档日志、 脱机存档日志和活动日志来重建一个数据库或表空间， 重建既可以在日志的末尾，也可以在某个特定的时间。前滚功能可以通过重新执行那些在归档和活动日志中发现的已经提交过的工作事务单元来完成数据库的恢复。

12.3.4 日志文件的使用

a) 删除日志

如果一个活动日志被删除，该数据库就不再可用，必须进行恢复才可以再度可用。

如果活动日志被删除了，那么只能对数据库进行前滚 (roll-forward) 操作直到第一个被删除的活动日志前记载的状态。

b) 使用 Not logged Initially 选项

在 CREATE TABLE 和 ALTER TABLE 的语句中都有 NOT LOGGED INITIALLY 选项。使用这个选项后，创建和修改表或是将来激活这个选项的操作的工作单元中的可记录的行为都不会被记入日志。这一点对于在表中第一次插入或更新大量数据时是很有好处的。 对于易于重建数据的工作表这个选项也是合适的。

这个选项的好处是在日志的负载很高的情况下提高性能；缺点是失去了工作单元的可恢复性。

12.4 使用备份和还原进行版本恢复

版本恢复是通过使用 BAKUP 和 RESTORE 命令实现的。

12.4.1 备份数据库

BACKUP 备份命令可以通过命令行处理器、 命令中心 ‖、应用程序接口和 控制中心 ‖来调用。在进行一次备份之前，应考虑以下几点：

● 必须拥有 SYSADM,SYSCTRL 或者 SYSMAINT 的权限才可以执行备份。

● 数据库可以是本地数据库，也可以是远程数据库。它本身的备份保留在数据库服务器上。

● BACKUP 命令可以和外挂存储管理程序接口来直接对备份进行管理，例如 TSM。

● BACKUP 命令可以直接将它的输出通过操作系统传送到磁带上。

● BACKUP 命令可以直接将它的输出传送到所有平台的硬盘上。

● 可以创建多份备份文件，以便包含已经从数据库中备份过的数据。

当调用 BACKUP 命令时，应该意识到以下几点：

● DB2必须被启动。

● 数据库必须处在正常状态活备份挂起状态。

● 当使用 BACKUP 工具时，可以使用别名 (alias)调用数据库名。在大多数情况下，这些别名是和数据库同名的。

● 可以通过 ONLINE 参数以联机模式运行备份，否则以默认的脱机模式运行。联机模式的备份允许在备份工作执行过程中其他应用程序仍保持与数据库的连接并做各种数据处理。对于脱机备份，只用备份工作本身可以和数据库连接。

对于联机模式，数据库必须是可恢复的；也就是说，归档日志模式必须被激活。当运行在联机模式时，DB2 试图获得带有 LOB （大对象）的表上的 S(共享)锁。这可能会导致一些持有不兼容锁的应用程序的操作失败。

12.4.2 使用备份的例子

BACKUP 命令的一些关键参数如下：

● ONLINE——指定联机备份，默认情况是脱机备份。

● INCREMENTAL——指定备份自最后一次全备份以来的所有数据。

● DELTA——指定备份自最近一次备份操作以来的数据。

可以从 DB2 命令行处理器 (DB2 Command Line Processor) 或者命令中心执行 BACKUP 命令。例如：

```
BACKUP DATABASE DB2CERT TO C:\DBBACKUP
```

这条命令将数据库 DB2CERT 备份到 Windows 操作系统的 C:\DBBACKUP 目录。其他参数使用它们的默认值。

```
BACKUP DATABASE DB2CERT TO /dev/rmt0
WITH 2 BUFFERS
BUFFER 512
PARALLELISM 2
```

这个命令把数据库 DB2CERT 备份到一个 AIX 系统上的磁带设备上。一次分配两个 512（4KB）页的缓冲区。两个并行任务用来进行并行的备份。

也可以使用“控制中心”来执行备份。右击要备份的数据库，然后选择备份数据库的选项。

12.4.3 还原数据库

RESTORE 命令可以从 命令行处理器 、命令中心 、应用程序接口或者 控制中心 调用。
在还原一个数据库之前，应考虑以下方面 ：

● 必须拥有 SYSADM, SYSCTRL 或者 SYSMAINT 权限来对一个已经存在的数据库进行 RESTORE 操作，或者拥有 SYSADM 、SYSCTRL 权限，可以 RESTORE 到一个新的数据库。

● 只能对已经用 BACKUP 命令进行备份过的数据库使用该命令。

● RESTORE命令可以和外挂的存储管理程序相连接，例如 TSM，以便直接使用一个由外挂存储管理程序管理的备份镜像。

● RESTORE 要求对数据库的排它连接。当数据库正在还原时，其它的程序不能在这个数据库上运行。一旦开始运行，它将防止其它程序存取当前数据库，直到还原完毕

● 数据库可以是本地的，也可以是远程的。

当使用 RESTORE 这个命令时，必须注意以下几点

● DB2必须已经启动。

● 可以还原到一个新的或是已存在的数据库中。

● 可以通过 ONLINE 参数在联机模式下运行恢复命令 ,但这只对表空间有效

● 参数 TAKEN AT 需要备份的时间戳。如果在同一目录下面有多个备份，这个参数是必须的。时间戳精确地显示了一次成功备份后时间，格式是 yyyymmddhhmmss 。只要不引起混淆，这个时间戳的格式也可以简化。如果在使用 TSM 过程没有指定这个参数，还原将会选择最新的备份

另外，考虑以下方面将提高 RESTORE 的性能：

● 参数 PARALLELISM 。使用这个参数可以减少还原的总时间。 它定义了还原进程或线程数， 默认值是 1。
注：在恢复过程期间，线程 /进程与表空间没有联系。

● 增加还原缓冲区大小和缓冲区数目。通过增加缓冲区大小 (BUFFER 参数)或是增加缓冲区数目 (WITH n BUFFER 参数) 可以提高 RESTORE 的性能。

BUFFER 参数最小值是 8 页；缺省值是 1024 页。还原缓冲区大小必须是备份过程种指定地备份缓冲区大小的正整数倍数。

例如 ,如果备份过程中指定的缓冲区大小是 1024 页，还原过程指定的缓冲区大小是 16 页，则实际的缓冲区大小是 1024 页。如果指定的还原缓冲区大小是 2049，那么实际上还原缓冲区是 2048。

注：如果在数据库还原过程中发生了错误，那么成功的还原过程结束前，这个数据库不能使用。

12.4.4 还原的例子

RESTORE 命令的一些关键参数如下：

● ONLINE—— 只适用于表空间级还原。

● HISTORY FILE ——指定只从备份映像还原历史文件。

● INCREMENTAL—— 不带其它参数，指定手动还原过程。

● INCREMENTAL AUTO/AUTOMATIC— 指定自动增量还原操作。

● REDIRECT—— 指定重定向还原操作。该命令必须后跟一个或多个 SET TABLESPACE CONTAINERS 命

令，然后使用带 CONTINUE 选项的 RESTORE DATABASE 命令。

● WITHOUT ROLLING FORWARD — 在成功还原之后，数据库不置为前滚挂起状态。

可以从 "命令行处理器"或 "命令中心"运行 RESTORE 命令，例如：

```
RESTORE DATABASE DB2CERT FROM C:\DBBACKUP
```

这个命令从 c:\DBBACKUP 目录下还原数据库 DB2CERT 。其它的参数使用默认值。 这可能会使数据库处于前滚挂起状态，所以在使用这个数据库之前，必须运行 ROLLFORWARD 命令 .

```
RESTORE DATABASE DB2CERT FROM C:\DBBACKUP
WITHOUT ROLLING FORWARD
WITHOUT PROMPTING
```

上面这个命令就不必使用 ROLLFORWARD 命令了。 ROLLFORWARD 命令更详细的讨论将会在这一章的 "前滚数据库和表空间"一节中介绍。

参数 WITHOUT PROMPTING 指明还原过程将不需要人为参与，任何需要人为介入的动作将返回一个错误的信息。

```
RESTORE DATABASE DB2CERT FROM C:\DBBACKUP
INTO NEWDB
WITH 2 BUFFERS
BUFFER 512
WITHOUT ROLLING FORWARD
```

上面这个命令把数据库还原到一个名为 NEWDB 的新数据库中，并且指定了两块缓冲区，每块缓冲区大小是 512 页。

也可以从 "控制中心"进行还原。右键点击想还原的数据库，选择 "Restore Database"

12.5 在还原过程中重定义表空间容器

在备份数据库的过程中，被备份的表空间所用到的表空间容器将会被一个记录保存下来。在还原过程中，备份中列出表空间容器将会被检查是否还存在，是否可以存取。如果因为任何原因有一个或多个容器不能使用，还原过程将会失败。为了在这种情况下也可以还原，我们可以在还原过程中重定义容器。重定义容器包括添加，改变，或删除表空间容器。

存在这样的情况， 虽然备份中所列的容器在系统中并不存在，你仍需要在这些容器中重建。 这种情况的一个实例是在进行备份之外的地方对一个系统进行一个灾难恢复。新的系统中可能没有对需要的容器进行定义。为了在这种情况下也可以还原数据库，可以重定向容器。

在上述任一种情况下，还原和重定义容器的过程被称为重定向还原 (redirected restore)

注：重定向还原对于把一个在主要站点的数据库备份到一个指定站点去也是很有用的 ,这样配置起来就只要较少的资源，如容器所需要的磁盘空间。

如上所述，可以通过 "控制中心"中的还原对话框来实现重定向还原，也可以使用 "命令中心"或者 "命令行处理器"。

在重定向过程中 SMS 的表空间和 DMS 的表空间必须与它们各自的存储类型符合。

当进行重定向还原时，应该注意以下几点：

● 目录和文件型容器如果不存在将被自动创建。 除非容器因为某些原因不可访问， 就不需要重定向， 但 DB2 不自动创建设备容器。

● 在任何还原上进行容器重定向的能力为管理表空间容器提供了相当好的灵活性。 例如，即使通过 DB2 也不能直接支持增加 SMS 表空间容器， 你可以通过简单地在一个重定向存储上指定一个附加容器的方法来实现它。类似地，也可以将一个 DMS 表空间从文件容器移动到设备容器。

12.6 前滚恢复

前滚恢复可用于恢复可恢复的数据库。除了能够执行版本恢复之外，还能执行以下操作：

● 在单个表空间级上恢复

● 使用 ROLLFORWARD 指明时间点恢复

● 使用 BACKUP 和 RESTORE 在线操作

如前所述， 使用 LOGRETAIN 和 USEREXIT 这两数据库配置参数中的一个或者两个， 可以启用前滚恢复。当前滚恢复启用时，数据库进入备份挂起状态，必须等到整个数据库备份完后该数据库才能再被使用。

12.6.1 备份一个数据库

在做一个备份之前，应考虑以下几点：

● 使用 BACKUP 的 TABLESPACE 选项，可以备份单个表空间。 通过只备份比较容易丢失的表空间可以减少备份数据库的时间。 PARALLELISM 参数也可用于并行备份多个表空间。

● 一个表空间的备份和恢复不能同时进行，即使备份和恢复的表空间不同。

● 如果你有一个表的索引或者在其它表空间里的大对象部件，那么有关这个表的所有表空间都必须一起备份，以保证能够安全恢复。

当你要调用 BACKUP 命令时，除了要知道在版本恢复讨论里面提到的那些要点外，还有下面的一点：

● 数据库或者表空间都必须在普通或者备份挂起状态下才能使用 BACKUP 。

12.6.2 在表空间级备份的例子

可以在 "命令行处理器"或者 "命令中心"中使用 BACKUP , 也可以在 "控制中心"的图形化界面上操作。 例如，使用 "命令行处理器"：

```
BACKUP DATABASE DB2CERT
TABLESPACE (SYSCATSPACE,FILETS) TO C:\DBBACKUP
```

这个命令备份数据库 DB2CERT 中编目表空间 SYSCATSPACE 和用户表空间 FILETS 到 Windows 系统的 C:\DBBACKUP 目录。其它参数使用默认值。

12.6.3 还原数据库

在执行还原之前，应注意以下几点：

● 可以还原一个完整数据的备份拷贝或者表空间的备份到现有的数据库中。备份的版本可能在别名、数据库名和数据库种子方面与现有的数据库不同。

● 数据库还原只能在脱机模式下进行。

● 表空间还原可以在联机或者脱机模式下进行。在联机模式下，会产生几个数据库的连接。这样的好处是在表空间还原时其它表空间还能同时使用，而还原依然能进行。

● 数据库使用了前滚恢复就必须能在还原后前滚，除非使用了 WITHOUT ROLLING FORWARD 选项。然而，如果备份使用了 ONLINE 选项或者备份拷贝只是表空间，那么这个选项就不能使用。

● 尽管还原和前滚是不同的功能，恢复策略可能是一个数据库的完整前滚恢复的第一阶段。在一次成功的还原之后，配置为前滚恢复的数据库在备份进行时会进入前滚挂起状态，直到 ROLLFORWARD 命令成功运行后才能再使用此数据库。

● 当 ROLLFORWARD 命令执行后，如果数据库在前滚挂起状态，数据库就进行前滚。如果数据库不处在前滚挂起状态，但是表空间处在此状态，当运行 ROLLFORWARD 命令和指定表空间时，就只有表空间被前滚。如果不指定表空间，那么所有处在前滚挂起状态中表空间都会被前滚。

● 另一个数据库 RESTORE 不允许在前滚进程运行时运行。

● 如果表空间当前存在并且是同一表空间，只能还原这个表空间。这意味着进行备份映像和还原时表空间必须有同样的名字并且不能被删除和重新创建。

● 不能以一个表空间备份映像来还原不同数据库中的表空间。

● 如果表的索引或者大对象部件在不同表空间中，有关这个表的所有表空间都必须一起恢复，以保证数据的一致性。

● 可以从一个完整数据库备份映像中还原选定的表空间。所有与表空间有关的从备份时开始的日志文件必须存在。

12.6.4 在表空间级还原的例子

可以在 “命令行处理器”或者“命令中心”执行 RESTORE 命令。当然，也可以通过“控制中心”完成同样的功能。例如在“命令行处理器”中执行如下：

```
RESTORE DATABASE DB2CERT
TABLESPACE (FILETS) ONLINE FROM C:\DBBACKUP
```

这个命令可以在 WINDOWS 平台以联机模式从 C:\DBBACKUP 目录还原数据库 DB2CERT 的表空间 FILETS 。仍然可以进行数据库连接。其它参数使用默认值。这个命令可以让表空间进入前滚挂起状态，只有在使用了 ROLLFORWARD 命令后此表空间才能被使用。

12.6.5 前滚数据库和表空间

ROLLFORWARD 命令能够在“命令行处理器”、应用程序接口（ API ）或者“控制中心”中使用。在执行前滚时注意以下方面：

注：当使用时间点（ PIT）前滚恢复时，可用 ROLLFORWARD 命令的 USING LOCAL TIME 选项指定本地时间而不是 UTC 时间。

- 必须拥有 SYSADM ， SYSCTRL 和 SYSMAINT 权限才能调用 ROLLFORWARD 命令。
- 数据库可以是本地的或者是远程的。
- 数据库必须是可恢复的。
- 数据库在前滚之前必须被成功地还原了（使用 RESOTRE 命令）。
- 除了正在还原的表空间，其它的表空间都能从介质错误或者其它不可预期的事件进入前滚挂起状态。
- 数据库前滚在脱机模式下运行。数据库在前滚完成之前都不可用。联机的表空间可以前滚，SYSCATSPACE 系统表空间除外，这个表空间需要在脱机模式下前滚。其它表空间都可以在联机模式下运行前滚。
- 只要有必要的、能覆盖想前滚的时间段的日志文件，就可以使用任何拥有的备份映象。
- 经常备份可以减少 ROLLFORWARD 的执行时间，前滚只需要在备份和恢复点之间的日志数据。

通常，调用 ROLLFORWARD 命令的次序如下：

- 执行不带 STOP/COMPLETE 选项的 ROLLFORWARD 。

● 执行带 QUERY STATUS 选项的 ROLLFORWARD 。QUERY STATUS 选项表示如果返回点在日志文件结束之前，将会丢失该日志文件。

● 执行带 STOP/COMPLETE 选项的 ROLLFORWARD 。在命令调用之后，不能前滚附加的改变。

当调用 ROLLFORWARD 命令时，必须知道以下几点：

● 如果需要取消一个前滚操作， 可以使用带 CANCEL 选项的 ROLLFORWARD 。这可以使数据库进入还原挂起状态，而不必考虑一个针对数据库的前滚是否在进行中。

● 如果执行带 CANCEL 选项的 ROLLFORWARD ，并且指定一列的在前滚挂起状态下的表空间， 它们就进入还原挂起状态。

● 不能使用带 CANCEL 选项的 ROLLFORWARD 去取消一个正在运行的前滚操作。只能用来取消一个完成的、但是还没有执行 ROLLFORWARD STOP 或者在完成前滚操作之前失败了的前滚操作。

● 日志使用时间戳来联系一个单元工作的完成。在日志中的时间戳使用国际通用时间标准（ UTC）。格式是 yyyy-mm-dd-hh.mm.ss.nnnnnn(年，月，日，时，分，秒，微秒)。想前滚一时间点必须指定本地时间或 UTC 时间。

● 如果正在前滚一个或者多个表空间到一个时间点，必须至少前滚到最小的恢复时间，即表空间的系统目录更新的最后时间。这最小恢复时间可以用 LIST TABLESPACES SHOW DETAIL 命令来显示。

12.6.6 前滚期间的表空间状态

和一个表空间关联的不同状态显示了它的当前状况：

● 表空间在还原或者 I/O 错误之后，会进入前滚挂起状态。表空间必须前滚来消除前滚挂起状态。如果是由 I/O 错误引起的，这种情况必须在做前滚之前改正。

● 正在进行前滚操作的表空间将进入前滚进行状态（ roll-forward-in-progress ）。这种状态一直保持到前滚操作成功完成。如果 STOP/COMPLETE 选项在时间点恢复上没有使用，这表空间也将保持在这种状态。

● 表空间在没有带 CANCEL 的 ROLLFORWARD 或者在 ROLLFORWARD 执行中遇到不可恢复的错误后，会进入还原挂起状态（ resotre-pending）。表空间必须还原和重新前滚。

● 表空间在前滚至时间点或者在 LOAD NO COPY 操作之后， 将会进入备份挂起状态 （backup-pengding）。表空间必须备份后才能使用。

12.6.7 前滚实例

ROLLFORWARD 命令的一些关键参数如下所示：

● isotime ——所有已提交事务前滚的时间点。该值是以国际标准时间（ UTC ）表示的时间戳。 CURRENT TIMEZONE 特殊寄存器指定数据库服务器上 UTC 和本地时间的差别。

● USING LOCAL TIME —— 允许用户前滚到以用户本地时间表示的时间点，而不是 UTC 时间。

● COMPLETE/STOP —— 终止日志记录的前滚，通过回滚任一未完成事务及关闭数据库的前滚挂起状态来完成前滚恢复过程。

● CANCEL —— 取消前滚恢复操作，将数据库或一个或多个表空间置为还原挂起状态。

● ONLINE —— 指定表空间级前滚恢复操作应联机执行。

● NORETRIEVE —— 指定管理员控制前滚备用机器上的哪些日志文件，允许取消归档日志的获取。

可以在 “命令处理器”或者“控制中心”执行 ROLLFORWARD 命令。例如：

```
ROLLFORWARD DATABASE DB2CERT
TO END OF LOGS
OVERFLOW LOG PATH (C:\LOGS)
```

这个命令前滚由 RESTORE 命令导致前滚挂起状态的数据库 DB2CERT 。前滚至日志的结尾，这意味着所有的日志中的工作单元都会重新应用到上次联机归档日志文件的工作单元。这并不表示前滚会执行到当前时间。例如，如果最近的归档日志有些问题，这个命令将依然可以成功。使用 QUERY STATUS 选项能确保最近的工作单元在预期的时间上并且日志文件按照预期执行。

日志文件保存目录 （数据库配置参数 LOGPATH 表示），ROLLFORWARD 命令用此来查找激活和联机归档日志。 OVERFLOW LOG PATH 参数指明脱机归档日志文件保存的目录，因此， ROLLFORWARD 能够用它们来重新应用工作单元。

```
ROLLFORWARD DATABASE DB2CERT
TO END OF LOGS AND COMPLETE
TABLESPACE (FILETS) ONLINE
```

这个命令前滚由 RESOTRE 命令而进入前滚挂起状态的表空间 FILETS 。ONLINE 参数允许并发连接其它表空间。 AND COMPLETE 参数指示 DB2 完成前滚，回滚任何未完成的工作单元，并且通过关闭前滚挂起状态使数据库可以正常使用。

12.7 大对象

如果一个表包含长字段或者大对象（ LOB ）列，应该考虑把这个数据放到一个单独的表空间。因为备份 LOB 数据潜在需要大量的时间和硬盘空间，把它放到一个单独的表空间中，你可以决定以比常规表空间低

的频率对它进行备份。

当创建或修改表的时候，也可以选择不对 LOB 列进行日志记录，这也可以节省日志文件所占的空间。如果 LOB 数据很容易重新创建，这种方法就是切实可行的。

12.8 脱机和联机表空间状态

在数据库启动 /激活的过程中， DB2 进行检查，确保所有的表空间容器都是可访问的。如果一个容器不可访问，那么除了其它诸如备份挂起、恢复挂起和前滚挂起等恢复状态外，这个表空间将会被置为脱机状态。因此一个表空间可以具有这样的状态，例如， BACKUP PENDING + OFFLINE （备份挂起 +脱机）。

在处理完容器问题之后，使表空间重新回到联机状态：

● 与所有应用程序断开连接并且重新连接或者重启数据库。

● 如果还有应用程序和数据库连接，使用带 SWITCH ONLINE 选项的 ALTER TABLESPACE 命令。当处在脱机状态时这个表空间可以被删除。

如果容器的问题不能解决，为了确保没有任何错误地、干净地重新启动数据库，使用带 DROP PENDING TABLESPACES 选项的 RESTART DATABASE 命令。即使提供的表空间列表有错误，它也将指导 DB2 成功地启动数据库。一旦数据库被启动，对于那些不能被恢复的表空间，惟一能执行的操作就是删掉。

12.9 高可用性

DB2 支持高度可用功能，可以在一台运行 DB2 实例的机器出现故障的情况下，由另一台机器自动接管那台出故障的机器的处理过程。

例如，在 AIX 上，这个支持使用 HACMP （高可用性簇集多处理）并且通过这个产品实现。 HACMP 通过处理器簇 （clusters of processors）提供增长的可用性，这些处理器簇共享诸如磁盘或者网络通路等资源。如果一个处理器出现故障，处理器簇中的另一个处理器可以代替这个出故障的处理器。附加的高可用性程序（high-availability programs ）根据操作环境得到。

以下是热备用（ hot standby）和相互替换（ mutual takeover）模式的一个简洁描述。

● 热备用 （hot standby）—— 处于激活状态的处理器被主动地用来运行 DB2 实例并且一个 DB2 服务器处于备用模式，如果第一个处理器发生了操作系统或者硬件故障，就准备接管那个实例。

● 相互替换（ Mutual takeover ） —— 各种各样的处理器都被用来运行单独的 DB2 实例，或者在另一个处理器运行 DB2 应用程序时被用来运行 DB2 实例。如果其中一个处理器出现操作系统或者硬件故障，其它处理器就接管出故障的处理器的任务。一旦故障使处理器彻底停止工作，剩余的那个处理器将完成两个处理器的工作。

注：其它流行的高可用性软件程序包括：微软簇集服务器（ MSCS），Sun 簇集（SC），Steeleye（用于 Linux 环境）和 Veritas 簇集服务器。

13 监控和优化

13.1 健康监视器

健康监视器是一个服务器端工具，甚至无序和用户交互，就可持续监视实例的运行状态。假如健康监视器发现一些事情已超过阈值（如可用日志空间不足）或检测到数据库对象的异常状态（如实例崩溃），健康监视器将引发警报。当引发警报时，将发生两件事情：

● 警报通知将发送到电子邮件信箱或寻呼机，因而可以和负责系统的任何人联系。

● 可以执行预先配置好的措施。例如，可以运行一个脚本或任务（通过 任务中心 执行）。

健康指标 （indicator）是健康监视器核查的系统特性。健康监视器核查这些指标是否在一组预先定义的阈值范围内。健康监视器核查系统的状态，并和健康指标的阈值进行比较，然后确定是否要发出一个警报。使用健康中心、命令或 API，用户可以自定义健康指标的阈值，并且可以定义当发出警报时应通知谁以及应该运行什么脚本或任务。

健康中心提供了图形化的健康监视器界面。用户可用它配置健康监视器，并可以查看发生的实例状态和数据库对象的警报。使用健康监视器的下钻能力，用户可以访问当前警报并得到如何解决该警报的推荐措施列表。

用户可以采用推荐措施之一解决该警报。假如推荐措施需要修改数据库或数据库管理器的配置，新的配置值也将被推荐，用户点击相应的按钮即可执行推荐措施。在其它一些情况下，给出了运行工具如 CLP 或内存可视化器进一步调查的建议。

13.2 数据库监视

有各种各样的方法获取需要的监视信息，这些方法使用下列 DB2 工具：

● 解释工具
● 快照监视器
● 事件监视器。

13.2.1 数据库监视器

数据库监视器是用来收集详细的资源使用情况。监视活动可以从 DB2 客户端或者服务器端执行，监视器界面可以通过 CLP 命令、图形化监视器或者监视 API 调出。DB2 提供了不同的监视类型，它们的不同之处在于收集监视数据的方式。监视的两种方式是：

● 快照监视 —提供特定时刻的数据库活动信息。它是数据库活动当前状态的一个照相。执行快照时，返回

给用户的数据量的大小由监视器开关决定。这些开关在实例一级或应用程序一级设置。

● 事件监视 — 记录数据库特定事件的发生情况。 允许收集有关临时事件的信息， 包括死锁、连接和 SQL 语句。

13.2.2 快照监视

快照可用于确定数据库系统的状态。 使用快照可在常规的时间段内观测趋势和预测潜在的性能瓶颈。 快照监视器以计数器的方式提供累计信息，这些计数器可以重新复位。快照信息以专门的数据结构提供，执行快照的应用程序可以察看那些信息。

快照返回的信息数量取决于开关的设置。监视器的开关以及它们提供的信息在下表中列出：

组	提供的信息	监视器开关	DBM 参数
排序	使用堆的数量、 溢出、排序性能	SORT	DFT_MON_SORT
锁定	持有的锁的数量、死锁的次数	LOCK	DFT_MON_LOCK
表	活动量（读写的行数）	TABLE	DFT_MON_TABLE
缓冲池	读写的数量、花费的时间	BUFFERPOOL	DFT_MON_BUFPOOL
工作单元	开始时间、 结束时间、完成的状态	UOW	DFT_MON_UOW
SQL 语句	开始时间、 停止时间、语句标识	STATEMENT	DFT_MON_STMT
时间邮戳	监视器时间邮戳	TIMESTAMP	DFT_MON_TIMESTAMP

设置实例的有关监视器开关的配置参数会影响实例中的所有数据库，每一个连接到数据库的应用都继承 DBM 配置中的默认开关设置。

注意： DB2 快照监视器只能被具有 SYSADM 、SYSCTRL 或 SYSMAINT 授权的用户使用。事件监视器需要 SYSADM 或 DBADM 授权。

13.2.2 快照监视 （接上页本节）

监视器开关可以在实例 （DBM 配置）一级或应用程序一级使用 UPDATE MONITOR SWITCHES 命令打开或关闭。

例如，为了捕获关于 SQL 语句在 DB2 中执行的详细信息，相应的开关必须打开。使用如下命令：

```
UPDATE DBM CONFIGURATION USING DFT_MON_STMT ON
```

或者使用下面的命令：

UPDATE MONITOR SWITCHES USING STATEMENT ON

UPDATE DBM CONFIGURATION 命令修改数据库管理器的配置。因此，存取此实例的任何数据库的应用程序中的 SQL 信息都被捕获。

注意：修改数据库管理器配置参数的值后，需要停止和启动实例使得修改生效。但是，对默认监视器开关的修改会立即生效，因此，不需要停止和启动实例。

另一方面， UPDATE MONITOR SWITCHES 命令只捕获激活开关的应用程序的 SQL 信息。（在这个例子中，应用程序就是命令行处理器。）

注意：即使一个开关都没有打开，快照监视器仍然会捕获一些基本信息。

13.2.2.1 查看快照监视器数据

当打开一个监视器开关后，开始收集监视器数据。为了查看监视器数据，必须执行一个快照。
注意：当记录数据时，性能稍微有些影响。执行快照也会增加性能开销。

使用 DB2 命令 GET SNAPSHOT 可以请求一个快照。这个命令可以以不同的方式执行：在控制中心图形工具中执行；使用相应的 API 嵌入应用程序中；从命令中心或 CLP 中执行。

在解释快照监视器的输出之前，先讨论如何捕获快照信息。使用命令中心界面去捕获数据库监视器快照。当执行一个快照时，可以定义感兴趣的范围。

快照监视器使用不同层次的监视。这些层次使得用户能专注于分析感兴趣的范围。
可以使用的快照监视器层次如下：

- 数据库管理器 — 捕获活动实例的信息
- 数据库 — 捕获一个数据库或多个数据库的信息
- 应用程序 — 捕获一个应用程序或多个应用程序的信息
- 缓冲池 — 捕获缓冲池的活动信息
- 表空间 — 捕获数据库中表空间的信息
- 表 — 捕获数据库中表的信息
- 锁 — 捕获使用数据库的应用程序持有各种锁的信息
- 动态 SQL — 从数据库的 SQL 语句缓存中捕获某个时间点的语句信息

执行一个快照时， 快照开关和层次结合起来提供不同的监视信息， 它们紧密相关。如果对应的监视器开关没有打开，使用的快照层次不会返回任何数据。

例如，要收集认证数据库 DB2CERT 的锁信息，从命令中心执行下面的 DB2 命令：

GET SNAPSHOT FOR LOCKS ON db2cert

从 APPLICATIONS 快照监视器层次返回的信息，使用下面的命令获得：

GET SNAPSHOT FOR APPLICATIONS ON db2cert

这个快照提供了应用程序活动的详细信息。使用这个快照，能够看到应用程序的活动，比如应用程序读了多少行、更新了多少行或删除了多少行。为了获取有关一个应用程序快照中列出的数据库对象的更多信息，如表或锁，必须使用相应的监视器开关和监视器层次。

要查看一个表层次快照，执行下面的命令：

GET SNAPSHOT FOR TABLES ON db2cert

13.2.2.2 快照监视器开关状态

在任何时候都可以执行下面的命令来确定数据库监视器开关的当前设置：

GET MONITOR SWITCHES

如果想知道监视器开关的实例一级的设置，使用下面的命令：
GET DBM MONITOR SWITCHES

快照监视器返回的数据主要基于计数器，这些计数器与监视器开关关联。
当下面的一种情况发生时，监视器开关被初始化或复位：

- 使用应用程序层次监视时，应用程序连接到数据库
- 使用数据库层次监视时，第一个应用程序连接到数据库
- 使用表层次监视时，表第一次被存取
- 使用表空间层次监视时，表空间第一次被存取
- 执行 RESET MONITOR 命令
- 打开某个监视器开关

在任何时候执行下面的命令，都会复位监视器开关：
RESET MONITOR FOR DATABASE db2cert

复位监视器开关导致所有计数器清零，后续的快照将基于新的计数器值。如果要复位一个实例中所有数据库的监视器开关，则使用 RESET MONITOR ALL 命令。

注意：每一个应用程序都各有一份快照监视器值的拷贝，因此，复位监视器开关仅仅影响执行复位的应用程序的计数器。

13.2.2.3 事件监视

快照监视记录执行快照时数据库活动的状态；事件监视记录当一个事件发生时数据库活动的状态。有些数据库活动需要被监视，但是使用快照监视器却不容易捕获这些活动。这些活动包括死锁过程。当死锁发生时，DB2 对其中一个应用程序执行一条 ROLLBACK 语句来解除死锁。有关死锁事件的信息，用快照监视器不容易捕获，因为在执行快照时，死锁很可能已被解除。

像其他数据库对象一样，事件监视器可以使用 SQL 数据定义语言 (DDL) 创建。事件监视器也像快照开关一样，可以打开或关闭。

注意：创建事件监视器需要 SYSADM 或 DBADM 的权限。

13.2.2.4 事件监视器定义存储

事件监视器的定义存储在下面的系统编目表中：

- SYSCAT.EVENTMONITORS--- 每一个事件监视器有一条记录，记录中包括事件监视器的当前状态以及 SQL 表、文件或管道等目标输出的表识符。
- SYSCAT.EVENTS--- 被监视的每一个事件一条记录。可以定义单个的监视器监视多个事件（如 DEADLOCKS 和 STATEMENTS ）。
- SYSIBM.EVENTTABLES ---将结果写入 SQL 标的事件监视器的每一个目标表对应表中的一行。

下列工具可用来分析事件监视器数据：

- db2eva 一个可以从命令行调用的图形化工具（用来读写入 SQL 表中的数据）。
- db2evmon 一个基于文本的工具，它读出事件记录，生成一个报告。

事件分析器只显示以前收集到的数据，因此，使用事件分析器不象快照监视器那样有额外的开销。事件分析器可能没有你期望的信息，因为它仅仅显示以前已收集到的事件记录。

如前所述，事件监视器是用 SQL DDL 语句创建的数据库对象。在这个例子里，将创建一个死锁事件监视器，并把事件记录存储在 /enentmonitors/deadlock/evmon1 目录中。

注意： 当使用文件输出时， DB2 不会创建事件监视器的输出目录，它必须由数据库管理员创建，并且实例的拥有者对这个目录有写权限。

下面的 SQL 语句创建一个事件监视器，它的名字为 evmon1。

```
CREATE EVENT MONITOR evmon1 FOR DEADLOCKS
WRITE TO FILE ?/eventmonitors/deadlock/evmon1 ‘
MAXFILES 3 MAXFILESIZE 1000
```

这个事件监视器定义为分配给 3 个文件， 每一个大小为 4MB，监视器总共存储空间为 12MB 。事件监视器其它选项包括写缓冲区大小、同步（堵塞）写、异步（不堵塞）写、将事件监视器数据附加到已存在的记录中或者当监视器激活时，替换目录中的数据。

注意：定义事件监视器没有数量上的限制，但是每一个 DB2 实例同时最多能激活 32 个事件监视器。

为了收集监视数据，必须激活一个事件监视器。用 CREATE EVENT MONITOR 语句定义一个事件监视器之后，就可以激活它。下面的语句用来激活一个事件监视器：

```
SET EVENT MONITOR evmon1 STATE=1
```

13.2.2.4 事件监视器定义存储 （接上页本节）

当事件监视器激活后，事件监视器记录被写入指定目录的文件或者管道中（我们的例子是目录/eventmonitors/deadlocks/evmon1）。

在定义时使用 AUTOSTART 选项，使得事件监视器在数据库启动时自动启动。
可再次使用 SET EVENT MONITOR STATE 语句关闭一个事件监视器。

SET EVENT MONITOR evmon1 STATE = 0

关闭一个事件监视器导致将所有的记录写入目标位置。

即使一个事件监视器被关闭了，但是在系统编目表中仍有定义。只不过不记录监视信息了。

为了确定一个事件监视器是处于激活状态还是非激活状态，可使用 SQL 函数 EVENT_MON_STATE 或者在-控制中心 |查看此状态。如果函数 EVENT_MON_STATE 返回值为 1，表示事件监视器处于激活状态；如果返回值为 0，则表示处于非激活状态。

下面是一个 SQL 语句例子，它使用 EVENT_MON_STATE 函数查询事件监视器 evmon1 的状态：

SELECT evmonname, EVENT_MON_STATE(evmonname)
FROM SYSCAT.EVENTMONITORS WH ERE evmonname = ?EVMON1‘

就像其它数据库对象一样， 事件监视器可以从数据库中删除。 下面是一个删除 evmon1 事件监视器的例子：

DROP EVENT MONITOR evmon1

事件监视器文件不能直接用来分析，必须使用程序来分析。

为了保证所有的事件记录都写入磁盘（有些可能被缓冲），简单的做法是关闭事件监视器。也可以使用带 BUFFER 选项的 FLUSH EVENT MONITOR 命令，如下所示：

FLUSH EVENT MONITORS evmon1 BUFFER

这个命令强迫事件监视器的缓冲内容写到磁盘中， 它不会产生不完整的记录， 只有已经在缓冲中的数据才被写入磁盘。

如果一个事件监视器正在监视数据库、 表空间或者表事件， 那么在最后一个使用数据库的应用程序断开连接时，它将写完整的事件记录。可以使用 FLUSH EVENT MONITOR 命令去记录部分事件记录。

为了生成监视器 evmon1 的事件监视器报告，执行下面的命令，指示事件监视器文件放在哪里：

db2evmon –path /eventmonitors/deadlock/evmon1

或者

db2evmon –db db2cert –evm evmon1

db2evmon 命令的 -path 选项用来指示事件监视器文件存放的路径。

注意：用事件监视器的类型（ deadlock）和它的名字 (evmon1)来命名存放事件监视器文件的目录是一个好方法。

13.3 SQL 监视

如果想要了解 DB2 如何执行一个查询， 必须分析它的存取方案。 存取方案是从一个表中检索数据的方法。解释工具可以提供有关 DB2 如何根据 SQL 语句去存取数据的信息。

DB2 分析每一条 SQL 语句，然后决定在一个静态绑定或动态执行中如何处理这条语句。从表中检索数据的方法就叫做存取方案。

DB2 中决定存取方案的部件叫做优化器。 在一条 SQL 语句的静态预处理过程中， 调用 SQL 编译器生成存取方案。存取方案包含数据存取策略， 其中包括索引的使用、 排序方法、 锁定语义和连接方法。 当执行 BIND 命令时（假设使用延缓绑定方法）， SQL 语句的可执行体存储在系统编目表中，称之为程序包 (Package)。

注意：从一个特定的表中检索数据的方法，比如是否使用索引，叫做存取方案。存取方案包含一组存取路径。

有时，在应用开发时不能完全确定一条 SQL 语句。在这种情况下，程序运行过程中需要调用 SQL 编译器生成存取方案，让数据库管理器使用它去存取数据。这样的一条 SQL 语句称为动态 SQL 语句。动态 SQL 语句的存取方案不存储在系统的编目表中，它们临时存储在内存（也叫做全局程序包缓存）中。如果某条动态 SQL 语句的存取方案已经存在于程序包缓存中了，那么就不会再调用 SQL 编译器。

13.3.1 解释表

DB2 使用解释表存储存取方案信息， 这样用户可以看到优化器所作的决策。 在能捕获任何解释信息前， 必须创建解释表。可运行 EXPLAIN.DDL 脚本来创建它们，该脚本可在 SQLLIB 子目录的 misc 子目录中找到。与需要 ―解释‖和―建议‖表的数据库连接。然后发出命令： db2 -tf EXPLAIN.DDL ， 这样就创建了表。如果有必要，这些表还可由 ―索引‖向导自动创建。

注意：第一次使用 ―可视化解释设施‖工具时，也会创建解释表。

13.3.2 收集解释数据

可以收集不同类型的解释数据，不同之处在于写入解释表的列不同。解释数据的选项有：

- EXPLAIN ―捕捉详细的存取方案信息，把信息存储在解释表中。不存储快照信息。
- EXPLAIN SNAPSHOT― 捕捉查询的当前内部表示和相关信息。 快照信息存储在 EXPLAIN_STATEMENT 表的 SNAPSHOT 列中。

不是所有的解释工具都需要相同类型的数据。有些工具使用 EXPLAIN 选项捕捉的信息，而其它如 Visual Explain 需要快照数据。

在创建解释表后，可以开始捕捉解释数据并存储在解释表中。不是所有的 SQL 语句都能被解释。可解释的 SQL 语句包括： SELECT、SELECT INTO 、UPDATE 、INSERT、DELETE 、VALUES 和 VALUES INTO 等语句。

根据 SQL 语句数量或者要解释的应用类型，应采用不同的方法。可用的方法如下：

- EXPLAIN 语句 — 收集 SQL 语句的解释数据。
- CURRENT EXPLAIN MODE 专用寄存器 — 指定收集动态 SQL 语句的解释数据。
- CURRENT EXPLAIN SNAPSHOT 专用寄存器 — 指定收集动态 SQL 语句的解释快照数据。
- BIND 选项 — 指定收集程序包中静态 /或动态嵌入 SQL 语句的解释数据。

下面依次讲述这些方法。

13.3.2.1 EXPLAIN 语句

当要收集一条动态 SQL 语句的解释信息时， EXPLAIN 语句显得非常有用。 EXPLAIN 语句可以从命令行处理器 CLP、命令中心或从程序中调用。

用户可以控制 EXPLAIN 语句把解释信息存储在解释表中的数量。 在默认情况下， 只捕捉常规解释表信息，不捕捉快照信息。如果想改变这种方式，应使用下面的 EXPLAIN 语句选项：

- WITH SNAPSHOT— 捕捉解释数据到解释表中

- FOR SNAPSHOT— 只捕捉解释快照信息。不捕捉其它解释信息，除了在 EXPLAIN_INSTANCE 和 EXPLAIN_STATEMENT 表中发现的信息。

在默认情况下， EXPLAIN 语句只收集解释数据，不收集解释快照数据。如果没有指定其它解释选项，则使用默认情况。

要执行 EXPLAIN 语句，用户必须在解释表上有 INSERT 权限。

注意：用 EXPLAIN 语句解释的 SQL 语句，不会执行，只是收集解释数据而已。

下面给出一个使用 EXPLAIN 语句收集存取方案信息的例子。 EXPLAIN 语句如下所示，它收集所有可用的解释信息，存储到解释表中。

EXPLAIN ALL WITH SNAPSHOT FOR— SELECT * FROM candidate ‖

注意：在上面的例子中，可以用关键字 PLAN 或 PLAN SELECTION 代替关键字 ALL 。

上面的例子向许多解释表中写入数据，包括 EXPLAIN_STATEMENT 表的 SNAPSHOT 列。EXPLAIN_INSTANCE 表中的 SNAPSHOT_TAKEN 列表示对每一条被解释的语句有一个可视化解释快照。

EXPLAIN 语句也可以嵌入一个应用程序中执行。收集到解释数据之后，就可以查询，看是否对一个查询用了理想的存取方案。

13.3.2.2 解释专用寄存器

另一个收集解释信息的方法是使用解释专用寄存器。DB2 用两个解释专用寄存器来收集动态 SQL 语句的解释信息。可以交互式地设置这些寄存器，或者在动态嵌入式 SQL 程序中使用。可用 SET 语句修改解释专用寄存器的值。

这些解释专用寄存器是：

- CURRENT EXPLAIN MODE 只捕捉解释数据，不捕捉快照信息。
- CURRENT EXPLAIN SNAPSHOT 只捕捉解释快照信息

下面的语句用来设置解释专用寄存器的值：

```
SET CURRENT EXPLAIN MODE option
SET CURRENT EXPLAIN SNAPSHOT option
```

解释寄存器的选项有：

- NO—不捕捉动态 SQL 语句的解释信息。
- YES—当执行动态 SQL 语句时，捕捉解释表或快照信息并返回结果。
- EXPLAIN—捕捉动态 SQL 语句的解释表或快照信息，但是不执行 SQL 语句。使用这个值获得解释信息而不需要执行 SQL 语句

注意：一旦用户把寄存器设置成 YES 或 EXPLAIN，将对后续的动态 SQL 语句进行解释，直到寄存器重置为 NO 为止。

13.3.2.3 BIND 的解释选项

有两个 BIND 选项可以指定： EXPLAIN 和 EXPLSNAP 。EXPLSNAP 选项收集解释快照信息，如果要使用 Visual Explain 看存取方案，则使用这个选项。EXPLAIN 选项只收集解释信息，不包括快照。

使用 BIND 选项的解释数据将被捕获：

```
BIND checked.bnd EXPLSNAP ALL
```

在这个例子里，将收集 checkid.bnd 程序包中定义的所有静态 SQL 语句的解释快照信息。因为指定了 ALL 选项，在程序包运行的过程中，也会在运行时收集所执行的动态 SQL 语句的解释快照信息。

这种在绑定过程中获得解释信息的方法，有利于管理员确定程序包执行静态或动态语句的存取方案。

13.3.2.4 使用解释报告工具收集和分析解释数据

还有一些收集解释数据的方法，它们把数据存储在报告中，而不是在解释表中。这些方法是 dynexpln 工具和 db2expln 工具。

Db2expln 工具描述为静态 SQL 语句选择的存取方案，这些 SQL 语句包含在存储在系统编目表中的程序包中。 Dynexpln 工具描述为动态 SQL 语句选择的存取方案，它为这些语句创建一个静态程序包，然后用

db2expln 工具去描述它们。

这两个实用程序的输出存储在一个可读的报告文件中，利用这两个解释报告工具能快速地和易于使用地收集存取方案信息。

13.3.3 检查解释信息

解释数据存储在解释表中后，就可以使用 Visual Explain 或者其它解释工具查询或显示这些数据。Visual Explain 是一个图形用户界面（ GUI），数据库管理员或应用程序开发员可以用它来检查优化器决定的存取方案。 Visual Explain 只能查看用快照选项解释的存取方案。

Visual Explain 可以用来分析以前产生的解释快照或收集解释数据和解释动态 SQL 语句。在启动 Visual Explain 前，如果解释表还没有被创建， 那么就创建它。可以从 "命令中心"或者 "控制中心"调用 Visual Explain 。

从 "控制中心"界面，右键点击数据库图标，将看到有一个称为 Show Explained Statements History 的选项，使用 Explain SQL …选项，可以解释动态 SQL 语句，并以图形方式显示。这是解释单一 SQL 语句的最简单的方法。

打开 "解释语句历史"窗口，将列出所有被解释过的语句。因为可以定制环境，所以显示的信息可能有所不同。

想详细地察看一个存取方案，只需双击解释过的语句，或者选中感兴趣的条目，然后从窗口的菜单中选择 Statement -> Show access plan。

虽然所有解释过的语句都在 "解释过的语句的历史"列表中显示出来，但是只有包含了 EXPLAIN SNAPSHOT 信息的语句才可以用 Visual Explain 查看。

注意：在控制中心或命令中心的 Explain 选项也可以用来解释单个动态 SQL 语句。

Visual Explain 的输出以图形层次结构的方式展示一个 SQL 语句的组成。查询的每一个部分用一个图形对象表示。这些对象也叫做节点，节点有两种基本类型：

● 操作符 节点表示在一组数据上执行的动作。

● 操作数 节点表示数据库对象， 在上面发生了操作动作。 一个操作数就是操作符施加动作的对象。 这些数据库对象通常是一些表和索引。

生成 SQL 语句的解释数据是分析 DB2 优化器所决定的存取方案的唯一途径。在存取方案图中，每一个节点都有详细的信息， 用下面的方法获取： 双击该节点，或者选中该节点后， 从 Node 菜单项中选择 Show details 选项。

13.3.4 索引顾问

"索引顾问"是一个管理工具，它在设计表的索引时提供帮助。 "索引顾问"在下列情况非常有用：

· 为问题查询查找最佳索引。

· 为查询集（工作负荷）查找最佳索引，但受可选择应用的资源限制的支配。

· 测试工作负荷的索引而不必创建索引。

将 CURRENT EXPLAIN MODE 专用寄存器设置为 RECOMMEND INDEXES 。此设置将 导致 SQL 编译程序捕捉解释数据，并将推荐的索引放进 ADVISE_INDEX 表中； 但不执行该 SQL 语句。

当 CURRENT EXPLAIN MODE 专用寄存器设置为 EVALUATE INDEXES 时， ADVISE_INDEX 表将作为解释过程的输入。读取虚拟索引的定义并在解释过程中使用，就好像它们是真正的索引。

13.3.5 配置数据库资源

应该有目的地监视数据活动。 监视的目的可能是要达到更大的并发度或者减少磁盘存取等待时间。 监视数据库活动的另外一个重要目的帮助配置各个 DB2（实例） 和数据库参数， 以求优化内存的使用和提高性能。

注意： 命令 GET DATABASE 和 GET DATABASE MANAGER CONFIGURATION 列出当前值以及下一次实例启动或数据库将用到的值。这两个命令已添加 SHOW DETAILS 选项。

影响数据库性能的最重要因素之一是数据库中每一个缓冲池的大小。 当创建或修改一个缓冲池时， 需要指定它的大小。如果将大小设为 -1，那么表示使用默认的大小，由数据库配置参数 BUFFPAGE 指定。每一个缓冲池都是应用程序和物理磁盘之间的数据缓存。通过为表空间指定单独的缓冲池可以实现将用户的数据放在不同的缓冲池的目的。同样地，多个表空间可以共用一个缓冲池。

如果没有缓冲池， 所有的数据库活动都导致磁盘存取；如果每一个缓冲池的大小都太小，缓冲池的命中率会很低，应用程序执行 SQL 查询时就要等待磁盘存取；如果其中一个或者更多的缓冲池太大，则会浪费服务器的内存；如果所有缓冲池的总空间大于服务器可用的物理内存，则会发生操作系统页面调度（磁盘活动）；如果存取已发生页面调出的缓冲池，效率会很低。

除了默认的缓冲池 IBMDEFAULTBP 之外，准备创建别的缓冲池，必须考虑如何给它们分配空间。事实上不应该给一个包含大量小的、 不经常被存取的表的表空间分配一个很大的缓冲池， 而给一个包含很大的、经常被存取的表的表空间分配一个小的缓冲池。设定缓冲池的大小时，应该考虑表空间中表的大小，以及它们被更新和存取的频率。

DB2 优化器使用不同的缓冲池去实现最好的查询性能。参数 AVG_APPLS 告诉优化器平均会有多少个应用程序处于激活状态，优化器用这个参数判断每一个应用程序会使用每一个缓冲池的多少空间。

在 DB2 中无须停止数据库的活动，就可以增加、改变或者删除缓冲池。因为可以改变缓冲池的分配和在线更新配置参数，所以可以定制具体的任务所需的内存。

注意：在 DB2 正在运行时也可在线修改缓冲池的分配、改变数据库管理器以及配置参数。

数据库级有一块共享内存，叫做数据库堆（ DBHEAP ）。每一个数据库有一个数据库堆。当应用程序连接到数据库时，数据库管理器使用这块内存。它存储有关表、索引、表空间和缓冲池等的控制块信息。

注意： DB2 允许某些 DBM （INSTANCE_MEMORY ）和 DB （DATABASE_MEMORY, MAXAPPLS ）配置参数设置为 AUTOMATIC ，从而允许数据库选择这些参数的值。

13.3.5 配置数据库资源 （接上页本节）

可以配置很多 I/O 缓存，包括一个日志文件缓存（ LOGBUFSZ ），一个系统编目表缓存（CATALOGCACHE_SZ ）。日志缓冲区用作将日志记录写入磁盘的缓冲。每一个事务都会导致写入多条日志，为了优化磁盘写性能，要写入的内容在内存中缓冲，然后周期性地写入磁盘。编目表缓存用来把系统编目表存储在内存中。当某条 SQL 语句被编译或引用时，需要检查数据库对象的信息。如果这些信息存在内存中，就不需要到磁盘中存取数据。程序包缓存 (PCKCACHESZ) 用来减少重新装载程序包中的存取方案（节），当同样的节在一个程序中被多次使用时，缓存能提高性能。

注意：静态和动态 SQL 语句的存取方案都缓存在程序包缓存中。

记录块是一个分布式的缓存技术， 用来在网络上同时传送一组记录， 而不是一个记录。 网络流量的减少会提高应用程序的性能和允许更好的网络吞吐量。 DB2 根据光标的类型和绑定的参数决定是否采用记录块。如果优化器以块的方式返回查询结果，那么每一块的数据量由参数 ALSHEAPSZ 决定。

应用程序堆（ APPLHEAPSZ ）包含几个内存块， DB2 用它们来处理每一个应用程序请求。

排序堆（ SORTHEAP ）定义每一个排序能够使用的最大内存页数量。如果不能实现分区内并行，排序堆空间就分配在代理专用内存中；如果能实现分区内并行，排序操作被并行处理，排序堆空间就分配在代理专用内存中或者分配在数据库全局内存中，取决于执行何种类型的排序（专用排序还是共享排序）。对于一个专用排序，排序堆空间分配在专用代理内存中，独立于每一个并行代理；对于一个共享排序，排序堆空间分配在数据库全局内存中，每个并行代理都共享这个排序堆。优化器用 SORTHEAP 参数确定排序能否在内存中执行还是在磁盘里执行。 DB2 总是试图在内存中执行排序。

参数 sheapthres控制 DB2 服务器分配给排序堆的内存数量。

BUFFPAGE 是一个重要的性能参数，它影响 DB2 使用多少内存作为数据缓存（在数据库级）。下面的命令将默认缓冲池大小修改为 200MB ：

```
UPDATE DB CFG FOR db2cert USING BUFFPAGE 50000
```

对数据库配置的任何更改都不会立即生效，直到数据库下一次激活才能生效。 也就是说， 当前的所有应用都与数据库断开连接之后的第一个数据库连接发生时，这个连接和后续的连接都将使用新的数据库配置参数。如果修改 DBM （实例）配置参数，新的值直到这个实例被停止然后重新启动后才生效。

应用程序支持层堆 （ ASLHEAPSZ ）的内存是用作记录块传送， 下面的命令将记录块大小设置为 200KB(单位为 4KB) ：

```
UPDATE DATABASE MANAGER CONFIGURATION USING ASLHEAPSZ 50
```

除了默认数据库监视器开关参数（例如， DFT_MON_BUFPOOL 、DFT_MON_LOCK 、DFT_MON_SORT 、DFT_MON_TABLE 、 DFT_MON_UOW 和 DFT_MON_STMT ）外，任何对数据库管理器配置参数的修改都不会立即生效，直到这个实例被停止后重新启动。

在上面例子里，实例重新启动后记录将以 200KB 大小块（一般多于一行）的方式跨网络在服务器和应用程序之间传输。如果平均行长度为 1KB ，每块将返回 200 个记录（假设结果表多于 200 个记录）。

注意：记录块应用于远程和本地 DB2 客户应用程序。

13.3.6 配置分区内并行

使用 DBM 配置参数 INTRA_PARALLEL 可以激活 DB2 的分区内并行功能。为了激活 DB2 的分区内并行功能，执行下面的命令：

```
UPDATE DBM CFG USING INTRA_PARALLEL YES
```

注意：在 SMP 机器中，INTRA_PARALLEL 参数的默认值为 YES；在单 CPU 机器中，INTRA_PARALLEL 参数的默认值为 NO。

如果并行度设置为 -1，优化器将决定每一条 SQL 查询的并行度。例如，在一台 4 路的 SMP 机器中，一条 INSERT 语句跨 CPU 以并行方式运行，不会获得任何好处，所以使用 -1 设置，导致这种类型的语句只在一个 CPU 上运行（并行度为 1）。

在单处理器机器环境中设置 DBM 配置参数 MAX_QUERYDEGREE 为 2 能够略微提高 I/O 性能。这是因为单处理机器在处理一个新的查询之前不需要等待完成输入或输出任务，这样就能提高受 I/O 限制的应用的性能。

用 SET RUNTIME DEGREE 命令设置活动应用程序的最大查询并行度。应用程序可以用 SET CURRENT DEGREE 命令设置自己的运行时并行度。实际使用的并行度为下列三者中最低的一个：

- DMB配置参数 MAX_QUERYDEGREE
- 应用程序运行时并行度
- SQL语句编译并行度

13.3.7 性能参数监视和设置

DB2 命令 LIST APPLICATIONS 的输出，实际上就是应用程序的数据库监视器快照。也可以使用 -控制中心 ‖，列出和强迫应用退出。

```
LIST APPLICATIONS FOR DATABASE db2cert
```

命令 LIST APPLICATIONS 将列出所有当前与数据库连接的应用程序。

锁的内存在数据库配置文件中定义。可以用 CLP、-命令中心 ‖或者 -控制中心 ‖修改它的值。获得 db2cert 数据库当前配置设定的 CLP 命令是：

```
GET DATABASE CONFIGURATION FOR db2cert
```

解决锁定问题的第一步是增加 LOCKLIS 的大小和 MAXLOCKS 的值，可执行下列 CLP 命令去更新：

```
UPDATE DB CFG FOR db2cert USING LOCKLIST 250 MAXLOCKS 30
```

用户可以决定改变活动应用程序的平均数量，使与数据库活动相匹配。参数 AVG_APPLS 影响优化器如何使用每一个缓冲池。下面的 CLP 命令将修改参数 AVG_APPLS ，将值设置为 8：

UPDATE DB CFG FOR db2cert USING AVG_APPLS 8

做完这些修改之后，应该重新创建新的应用程序包（为静态 SQL 重新创建存取方案）。使用 REBIDN 命令更新程序包信息，也可以使用 db2rbind 使用工具完成它。

附件常用命令

常用的性能监控和管理命令

下面是一些常用的性能监控和管理命令的列表

性能

get monitor switches	返回会话监控开关的状态
update monitor switches using <monitor> <on|off> 为 <monitor>	设置会话监控开关的状态
reset monitor all	复位性能监控程序值
get snapshot for dbm	返回实例级别的性能信息
get snapshot for all on <dbname>	为数据库 <dbname> 在数据库级别返回所有性能信息
get snapshot for dynamic sql on <dbname>	返回动态 SQL 高速缓存的内容
runstats on table <tbschema>.<tbname>	收集表 <tbname> 的统计信息。表名必须是用 <dbschema> 全限定的
reorgchk on table all	确定是否需要对表进行重组。 这对于对所有表自动执行 runstats 很有用
reorg table <tablename>	通过重构行来消除 碎片 数据并压缩信息， 对表进行重组

管理

export	将数据库数据抽取到一个平面文件中
import	通过使用 IMPORT 实用程序，将数据导入到数据库
load query table <tbname>[to local-message-file][nosummary | summaryonly] [showdelta]	返回 LOAD 实用程序的进度
backup database <dbname> [to <path>]	执行数据库备份
restore database <dbname> [from <path>]	

	执行数据库恢复
get health snapshot for dbm	返回实例的正常快照信息（仅适用于 V8）
get health snapshot for all on <dbname>	返回数据库 <dbname> 的所有正常快照（仅适用于 V8）

管理服务器

get admin cfg	返回管理服务器的配置设置
update admin cfg using <p> <v>	将管理服务器配置参数 <p> 更新为值 <v>