目录

修改一览表

第一部分　　建立数据库文件

第二部分 在程序中处理数据库文件

第三部分 数据库文件管理

修改一览表

• 物理性约束：

系统增加了对唯一键字、主键字和引用约束的支持。有关唯一键字和主键字约束， 在第十五章”物理文件的约束”中描述。

• 引用的一致性：

引用的约束提供给用户一种不需自己写应用程序，可以保护数据库文件完整性的方法。这一重要的部分，将在第十六章”引用的一致性”中描述。

• 触发：

系统增加了对触发操作的支持。当指定的文件通过插入、更新或删除做修改时，触发操作可提供给用户一种初始化方法，或作一些其它的适当动作。有关触发的描述见第十七章“触发”。

第一部分　　建立数据库文件

本部的各章详细描述了如何建立 AS/400 的数据库。它包括数据库文件的描述，在系统中的访问路径的描述及所使用的不同方法。还论述了程序中使用文件描述的方法，使用文件描述数据和程序描述数据的区别。

其中一章是有关描述和生成逻辑文件的指南。它包括有关描述逻辑文件记录格式和使用数据描述规则(DDS)。描述不同类型的字段。还包括用 DDS 描述访问路径和使用系统中现存的访问路径的信息。有关定义逻辑文件成员来把数据分组的内容也在本章中讨论。

描述连接逻辑文件的部分，包括使用连接逻辑文件所应考虑的事项，如何连接物理文件和连接方法的例子以及对连接逻辑文件的性能、完整性和规则的总结。

本部中有一章用于专门讲述数据库的安全性。它包括保护功能的信息，如文件保护、公共权限、修改或删除数据的限制，使用逻辑文件保护数据的方法，还包括对一个数据库文件可授予用户不同类型的权限以及对物理文件的授权类型。

第一章　一般考虑事项

本章论述的主要是建立任何 AS/400 的数据库文件时所要考虑到的问题，以后各章将会讨建立物理和逻辑文件时要特别考虑的问题。

1.1.1 描述数据库文件

对数据库文件中记录有以下的两种描述方法：

- 字段级描述

  记录中字段要做描述，对每个字段的描述包括：字段名、长度、数据类型、有效性检查和说明。用字段级描述建立的数据库文件叫做外部描述的文件。

- 记录级描述

  只描述了文件中记录的长度，但系统并不认识文件中的有关字段，这样的数据库文件叫做程序描述的文件。

无论是记录级描述的文件还是字段级描述的文件，在你想编译一个使用该文件的程序前，必须描述和生成该文件，也就是该文件必须在使用前已在系统中存在了。

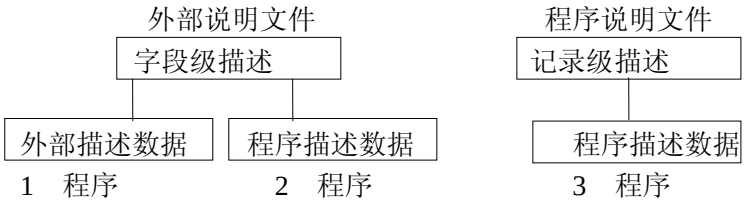
程序使用文件描述有两种方法：

1.程序使用字段级描述，是文件的一部分，由于字段的描述是在程序外，所以叫做外部说明的数据。

2.程序使用其自身描述的字段，文件中的字段仅在记录级描述，并且必须在使用这个文件的程序中进行描述数据叫做程序说明的数据。

程序即能使用外部说明文件也能使用程序说明文件。但是如果你选择了字段级描述的文件，系统将提供更多的功能。例如：当编译程序时，系统将从外部描述文件中提取信息，它自动包括程序中的字段信息。因此，不必对每个使用此文件的程序中的字段信息进行编码。

下图显示了在 AS/400 系统中典型的文件和程序的关系：



1.程序可使用系统定义的字段级描述文件，在编译期间，编译程序将把文件的外部描述复制到程序中。

2.程序可使用对系统定义的字段级描述的文件，但它不能用实际的字段描述。在编译期间编译程序不能把文件的外部描述拷贝到程序中去，文件中的字段将在程序中描述。在这种情况下，在程序中的字段属性和外部描述的字段属性一致。

3.程序使用指定的记录级描述的文件，文件中字段必须在程序中描述。

外部描述文件也可以在程序中描述，这样就可以与先前系统实现兼容。例如：你想在 AS/400 系统上运行一个先前传统文件系统的程序，这些程序使用程序描述数据，文件是记录级描述的。在以后的时间里，你可以将文件作为字段级描述文件(外部描述文件)，以便使用系统中更多的数据库功能。包含程序描述数据的旧程序可以继续使用外部描述文件，而新程序使用文件中部分的字段级描述。然后你可以把一个或多个旧程序转为使用字段级描述的

程序。

1.1.1.1 字典描述数据

一个程序描述文件可以是字典描述的，可以使用交互式数据定义实用程序(IDDU)来描述记录格式。即使文件是程序描述的，AS/400 查询系统、客户访问和数据文件实用程序(DFU)也可使用存放于数据字典中的记录格式描述。

一个外部描述文件也可以是字典描述的，可以使用 IDDU 去描述一个文件，然后用 IDDU 生成，所生成的文件就是外部说明文件。还可以把存在外部描述文件中的文件描述转入到数据字典中，系统总能保证数据字典中的定义和外部说明文件中的描述是一致的。

1.1.1.2 描述数据的方法

如果描述一个记录级文件，可以使用生成物理文件(CRTPF)和生成源物理文件(CRTSRCPF)命令中的记录长度参数(RCDLEN)。

如果描述一个字段级文件，有几种方法：IDDU、SQL/400 命令、数据描述规则(DDS)。

注意：由于 DDS 定义数据有很多选项，在这本书中我们仅限于介绍用 DDS 来描述数据库文件。

1.1.1.2.1 OS/400 交互式数据定义实用程序(IDDU)

用 IDDU 可以描述物理文件，IDDU 是一种菜单驱动、交互式描述数据的方式。你可能很熟悉系统 36 上用 IDDU 描述数据的方法，另外，IDDU 允许描述多格式物理文件，以便用于查询、客户访问/400 和 DFU。

当使用 IDDU 描述文件时，文件定义就成为 OS/400 数据字典的一部分。有关 IDDU 的详细资料将看《IDDU 使用指南》。

1.1.1.2.2 DB2/400 结构查询语言(SQL)

结构查询语言用于 AS/400 数据库文件。SQL 语言提供描述数据库文件字段的语句，并可生成文件。

IBM 系统中的 SQL 语言，是一种基本的标准数据库语言，目前，它用于所有的 IBM 系统 DB2 平台和一些来自不同厂家的数据库工具。

因为 DB2/400 SQL 语言是 IBM 系统应用体系的数据库语言，可以用它来定义和处理数据库数据。用 DB2/400 SQL 语言生成数据库文件时，文件描述将自动加到 SQL 集合的数据字典中。该数据字典(或叫做目录)由系统自动维护。

SQL 语言可处理许多平台上访问数据库文件，它适用于多种系统和分布式数据库。

有关 SQL 的详细资料，请看《DB2/400 SQL 程序设计》和《DB2/400 SQL 参考手册》。

1.1.1.2.3 OS/400 数据描述规则(DDS)

DDS 可以用来描述外部说明文件，用 DDS 可以做字段级、记录级、文件级信息的描述。

因为 DDS 提供许多可选参数，所以程序员在描述数据库数据时大多使用 DDS。例如：只有使用 DDS 才能描述逻辑文件中的键字字段。

DDS 为外部描述数据提供基本格式，DDS 的数据是以列的位置来分类的。这本书中的例子中对列都有编号，并且给出这些列中的可用数据。

《DDS 参考手册》详细介绍了如何使用 DDS 描述物理文件和逻辑文件。

1.1.1.3 描述数据库文件

对系统描述一个数据库文件，主要分为两部分：记录格式和访问路径。

(1)描述记录格式：

记录格式描述了每个记录中字段的顺序，也详细描述每个字段的信息，包括长度、数据类型(例如：压缩十进制或字符型)、有效性检查、说明和其它信息。

下面的例子显示了物理文件中记录格式和记录间的关系：

记录格式	ITMMST：		
字段	描述		
ITEM	区位十进制	5 位	小数位为 0
DESCPR	字符	18 位	
PRICE	区位十进制	5 位	小数位为 2

记录：		
ITEM	DESCPR	PRICE
35406	HAMMER	01486
92201	CREWRIVER	00649

一个物理文件只能有一个记录格式，物理文件中的记录格式描述了数据实际存储的方式。

逻辑文件不包含数据，它用于以不同格式和顺序排列来自于一个或多个物理文件中的数据。例如：逻辑文件可以改变物理文件中字段的顺序，或者选择物理文件中的某些字段提供给程序。

逻辑文件记录格式可以改变物理文件字段的长度和数据类型，系统将对物理文件字段描述和逻辑文件字段描述作出必要的转换。例如：一个物理文件定义字段 FLDA 为 5 位数的压缩十进制字段，使用 FLDA 的逻辑文件可以将其重新定义为 7 位区位十进制字。在这种情况下，当你的程序用逻辑文件去读取记录时，系统能自动把 FLDA 转换成区位十进制格式。

(2)描述访问路径：

访问路径描述了检索记录的顺序，它分为键字顺序访问路径和到达顺序访问路径。有关访问路径的详细描述请见 1.1.1.5”文件访问路径的描述”。

1.1.1.3.1 命名约定

系统规定文件名、记录格式名、字段名可为 10 位字符长，并且必须遵循系统的命名约定，但有些高级语言对命名有比系统更多的限定。例如：RPG/400 语言只允许 6 位字符长的名字，而系统则允许 10 位字符长。在这种情况下你可以临时改名(RENAME)，以满足高级语言的限制。有关在
程序中对数据库字段重定名的详细资料，请看有关的高级语言指南。

另外，名字是唯一的，有如下限定：

- 在一个记录格式中，字段名必须是唯一的
- 在一个文件中记录格式和成员名必须是唯一的
- 库中的文件名必须是唯一的

1.1.1.4 用 DDS 描述数据库文件

用 DDS 描述数据库文件，包括文件、记录格式、连接、字段、键字和选择/省略的信息。

- 文件级 DDS 告诉系统有关整个文件的信息。例如：可以指定文件中所有的键字字段

值必须是唯一的。

- 记录格式级 DDS 告诉系统有关文件中指定的记录格式信息。例如：当描述逻辑文件记录格式时，可以指定它所依据的物理文件。
- 连接级 DDS 告诉系统有关连接逻辑文件所使用的物理文件的信息。例如：指定如何连接两个物理文件。
- 字段级 DDS 告诉系统有关记录格式中每个字段的信息。例如：可以指定每个字段的名字和属性。
- 索引字段级 DDS 告诉系统有关文件中索引字段的信息。例如：可以指定记录格式中哪个字段被用作索引字段。
- 选择/省略字段级 DDS 告诉系统在处理文件时，哪些记录要传给程序。选择/省略字段只用于逻辑文件。

1.1.1.4.1 使用 DDS 描述物理文件的实例

用 DDS 描述物理文件的步骤如下：(图 1-1)

1.文件级(可选):

键字 UNIQUE 常用于说明文件中每个键字字段的值必须是唯一的，文件不允许有重复键字值。

2.记录格式级：

规定记录格式名，也可有可选的说明。

3.字段级：

规定字段名和字段长度，也可有可选的说明。

4 索引字段级(可选)：

这个字段被指定为索引字段。

5.注释(可选)

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER FILE (ORDHDRP)
     A   5
     A                                    1   UNIQUE
     A       2  R ORDHDR                      TEXT('Order header record')
     A         3  CUST           5   0       TEXT('Customer number')
     A            ORDER          5   0       TEXT('Order number')
     A            .
     A            .
     A            .
     A          K CUST
     A     4    K ORDER
```

图 1-1　DDS 描述物理文件(ORDHDRP)

以下显示了物理文件 ORDHDRP，该文件没有指定索引字段，因此是用到达顺序的访问路径，给出了说明这个文件必须的 DDS 语句。

记录格式

顾客号 (CUST)	订货号 (ORDER)	订货日期 (ORDDATE)	购货单号销售 (CUSORD)	订货状态 (SHPVIA)	...	状态 (ORDSTS)	...	(STATE)
压缩十进制(P)	P	P	P	字符		字符		字符
5 位	5 位	6 位	15 位	15 位		1 位		2 位

小数位：0　　0　　0　　0

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER FILE (ORDHDRP)
     A          R ORDHDR                    TEXT('Order header record')
     A            CUST          5   0       TEXT('Customer Number')
     A            ORDER         5   0       TEXT('Order Number')
     A            ORDATE        6   0       TEXT('Order Date')
     A            CUSORD       15   0       TEXT('Customer Order No.')
     A            SHPVIA       15           TEXT('Shipping Instr')
     A            ORDSTS        1           TEXT('Order Status')
     A            OPRNME       10           TEXT('Operator Name')
     A            ORDAMT        9   2       TEXT('Order Amount')
     A            CUTYPE        1           TEXT('Customer Type')
     A            INVNBR        5   0       TEXT('Invoice Number')
     A            PRTDAT        6   0       TEXT('Printed Date')
     A            SEQNBR        5   0       TEXT('Sequence Number')
     A            OPNSTS        1           TEXT('Open Status')
     A            LINES         3   0       TEXT('Order Lines')
     A            ACTMTH        2   0       TEXT('Accounting Month')
     A            ACTYR         2   0       TEXT('Accounting Year')
     A            STATE         2           TEXT('State')
     A
```

第 17 列的 R 说明定义了一个记录格式，记录格名 ORDHDR 在第 19 列到第 28 列指定。

当要描述一字段时，第 17 列应该为空。第 17 列为空时，第 19 列到第 28 列描述的名字即为字段名。

在第 35 列指定数据类型，有效的数据类型有：

字符	含　意
A	字符型
P	压缩十进制
S	区位十进制
B	二进制
F	浮点型
H	十六进制
L	日期型
T	时间型
Z	时间标记

注意:

1.有关双字节字符集数据类型(DBCS)的详细资料，请见附录 B“双字节字符集(DBCS)的考虑”。

2.AS/400 系统做算术操作时，使用压缩十进制比区位十进制更有效。

3.某些高级语言不支持浮点数据。

4.当使用浮点型字段时，要特别注意：

- 浮点字段的精度与该字段是单精度还是双精度及浮点的内部表达有关。它可将浮点型字段能表示的有效的最大值转化为十进制数。
- 当所定义的浮点型字段位数小于系统支持的位数时，字段长度只是一种表示，它对内部运算的精度没有影响。
- 尽管浮点能精确到 7 位(单精度)或 15 位(双精度)十进制数，但也可以指定为 9

到 17 位数。当一个内部格式中浮点数被转换成十进制后，并再次返回到内部格式时，可以使用附加位去唯一的建立内部浮点格式中的内部模式，这样就会获得一致的结果。

如果没有指定数据类型(35 列)，则小数位项通常用于确定数据类型。如果小数位(36 到 37 位)为空，数据类型为字符型(A)；如果该位是 0 到 31 的数，则数据类型为压缩十进制(P)。

在 30 列到 34 列指定字段长度，在第 36 到 37 列指定小数位上的数(数字型字段)。如果一个高级语言程序使用了压缩十进制或区位十进制字段，字段长度必须限定在此高级语言所允许的长度，这个长度并不是存储区中字段的长度，而是在存储器外部指定的字符或数的位数，例如：一个 5 位数的压缩十进制字段在 DDS 中指定的长度为 5 位，但在存储区中只占用了 3 个字节。

由 VARLEN 字段层键字指定的字符或十六进制数据被定义为是具有可变长度的字段。人们通常使用变长的字段。例如：一个数据库中雇员名，名字一般存放在 30 字节长的字段中，但有时你可能需要 100 字节去存放一个很长的名字。如果将字段定义为 100 字节，将浪费存储空间；如果将字段定义为 30 个字节，某些名字将被截断。

可以使用 DDS 中 VARLEN 键字来定义一个字符型字段为变长字段，定义如下：

- 没有分配长度的可变长度
  它允许字段以和数据相等的字节数被存放(每个字段长度增加两个字节，每个记录也附加一些字节)。但是由于所有的数据都存在文件中的可变部分，所以它请求磁盘的两个读操作来检索，这样系统的性能可能会受到影响。
- 分配了长度的可变长度
  该长度相当于数据的最大尺寸。它允许大量的字段数据存放在文件中的固定部分，并将分配给一般定长字段的非使用存储达到最小，当检索字段的数据长度小于分配的字段长度时，仅用一个读操作来检索，当长度大于分配的字段时存放在文件中的可变部分，它需要两个读操作来检索数据。

1.1.1.4.2 用 DDS 描述逻辑文件的实例

用 DDS 描述逻辑文件的规则如下：

1.文件级(可选)

例中，UNIQUE 键字说明了文件中每个记录键值必须是唯一的，它不允许有重复键值。

2.记录格式级

例中，它说明了记录格式名、相关的物理文件和说明(可选)。

3.字段级(可选)

例中，它指定了记录格式中的每个字段名。

4.索引字段级(可选)

例中，Order 字段作为索引字段。

5.选择/省略字段级(可选)

例中，所有 Opnsts 字段的值为 N 的记录都不做为逻辑文件中的记录，也就是说，从此文件中读记录的程序将不会看到 Opnsts 字段为 N 值的记录。

6.注释

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER FILE (ORDHDRP)
     A    6
     A                                    1   UNIQUE
     A      2   R ORDHDR                      PFILE(ORDHDRP)
     A        3   ORDER                      TEXT('Order number')
```

```
A            CUST                      TEXT('Customer number')
A            .
A            .
A            .
A      4   K ORDER
A            O OPNSTS             5    CMP(EQ 'N')
A            S                         ALL
```

图 1-2　简单逻辑文件的 DDS 描述(ORDHDRL)

一个逻辑文件必须在它所依据的所有物理文件生成后才能生成。上述例子中的 PFILE 键字用于指定逻辑文件所依据的物理文件。

逻辑文件的记录格式可以为：

- 由物理文件的各字段组成的新记录格式
- 和预先描述的物理或逻辑文件相同的记录格式(请看 1.1.1.4.6 中"共享已有的记录格式描述")
- 逻辑文件记录格式中的字段必须至少出现在一个物理文件中或者是由物理文件的字段派生出来的。

有关描述逻辑文件的详细资料，请看第 3 章"建立逻辑文件"

1.1.1.4.3 附加的字段定义功能

可用功能键字(DDS 中 45 列到 80 列)来描述物理文件和逻辑文件中字段的附加信息。这些功能包括：

- 有效性检查键字
  它可校验字段数据是否符合标准。例如：可以说明一字段的值在 500—900 范围内。(只有当数据是由键盘敲入并显示时才进行校验)
- 编辑键字
  它用于控制一个字段应该怎样显示或打印。例如，你可以使用 EDTCDE(Y)键字来指定一个日期字段的格式为 MM/DD/YY。EDTCDE 和 EDTWRD 键字均可用于控制编辑。(该编辑只限于显示文件和打印文件)
- 文本标题和名称控制键字
  它用于控制字段的名称和描述。例如：可以使用 TEXT 键字给出每个字段的文本描述。
  该描述包括在编译清单中，使得程序使用的文件有较好的文本说明。TEXT 和 COLHDG 键字是控制文本和列标题定义的。ALIAS 键字可以提供给字段更多的名称，如在程序中使用别名或备用名(如果高级语言支持的话)。
- 内容和缺省值键字
  它可用于控制字段的空值和缺省值数据。ALWNULL 键字指定字段是否允许为空值。若使用了键字 ALWNULL，则说明字段的缺省值为空值。如果在字段级没有指定 ALWNULL 键字，则此字段值不允许为空值。除非使用 DFT(缺省)键字指定了一个值，否则字符型和十六进制字段的缺省为空格，数据型字段缺省值为零。

1.1.1.4.4 使用已有的字段描述和字段引用文件

如果一个字段已经在文件中说明过，在建立新文件时，要想使用该字段描述，可以请求系统将其拷贝到新的文件描述中去。DDS 中键字 REF 和 REFFLD 允许引用已存在文件中的字段描述。这样有助于减少对 DDS 语句的编码工作，也有助于确保字段属性和所引用的文

件中的字段属性相一致。

另外，可建立这样一个物理文件，它只做字段描述，也就是说，该文件不包含数据，它只为其它文件提供要引用的字段描述，这类文件称为字段引用文件。一个字段引用文件是一个不包含数据只包含字段描述的物理文件。

可以使用字段引用文件来简化记录格式的描述，并确保字段描述的一致性。还可将所需的所有字段定义在一个字段引用文件中，用 DDS 和生成物理文件命令(CRTPF)生成一个字段引用文件。

字段引用文件生成后，就可以根据它来建立物理文件记录格式，而不需要描述每个文件中每个字段的特性。当建立一个物理文件时，你所要做的就是指出引用的文件(用 REF 和 REFFL 键字)和相应的修改。在新文件中，字段描述的修改和键字的指定将取代字段引用文件中的描述。

在下面例子中，名为 DSTREFP 的字段引用文件是为各种应用开发建立的。图 1-3 显示了描述文件的 DSPRETP 的 DDS 码：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A            INVNBR         5  0       TEXT('Invoice number')
     A                                       COLHDG('INVOICE' 'NUMBER')
     A            PRTDAT         6  0       EDTCDE(Y)
     A                                       COLHDG('PRINTED' 'DATE')
     A            SEQNBR         5  0       TEXT('Sequence number')
     A                                       COLHDG('SEQ' 'NUMBER')
     A            OPNSTS         1          TEXT('Open status')
     A                                       COLHDG('OPEN' 'STATUS')
     A            LINES          3  0       TEXT('Lines on invoice')
     A                                       COLHDG('TOTAL' 'LINES')
     A            ACTMTH         2  0       TEXT('Accounting month')
     A                                       COLHDG('ACCT' 'MONTH')
     A            ACTYR          2  0       TEXT('Accounting year')
     A                                       COLHDG('ACCT' 'YEAR')

     A
     A* FIELDS DEFINED BY ORDER DETAIL/LINE ITEM RECORD (ORDDTL)
     A            LINE           3  0       TEXT('Line no. this item')
     A                                       COLHDG('LINE' 'NO')
     A            QTYORD         3  0       TEXT('Quantity ordered')
     A                                       COLHDG('QTY' 'ORDERED'
     A                                       CMP(GE 0)
     A            EXTENS         6  2       TEXT('Ext of QTYORD x PRICE')
     A                                       EDTCDE(J)
     A                                       COLHDG('EXTENSION')
     A
     A* FIELDS DEFINED BY ACCOUNTS RECEIVABLE
     A            ARBAL          8  2       TEXT('A/R balance due')
     A                                       EDTCDE(J)
     A
     A* WORK AREAS AND OTHER FIELDS THAT OCCUR IN MULTIPLE PROGRAMS
     A            STATUS        12          TEXT('status description')
     A
```

图 1-3　字段引用文件的 DDS 描述(DSTREFP)

假定图 1-3 中 DDS 编码是录入到源文件 FRSOURCE 中。成员名为 DSTREFP。要生成一个字段引用文件，可使用生成物理文件命令(CRTPF)如下：

```
CRTPF    FILE(DSTPRODLB/DSTREFP)
         SRCFILE(QGPL/FRSOURCE)     MBR(*NONE)
         TEXT('Distribution field referrnce file')
```

参数　MBR(*NONE)告诉系统不对该文件增加成员(因为字段引用文件不包含数据，因此不需加入成员)。

引用文件 DSTREFP 来描述物理文件 ORDHDRP，其 DDS 描述如下(图 1-4)：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER FILE (ORDHDRP) - PHYSICAL FILE RECORD DEFINITION
     A                                      REF(DSTREFP)
     A           R ORDHDR                   TEXT('Order header record')
     A             CUST      R
     A             ORDER     R
     A             ORDATE    R
     A             CUSORD    R
     A             SHPVIA    R
     A             ORDSTS    R
     A             OPRNME    R
     A             ORDAMT    R
     A             CUTYPE    R
     A             INVNBR    R
     A             PRTDAT    R
     A             SEQNBR    R
     A             OPNSTS    R
     A             LINES     R

     A             ACTMTH    R
     A             ACTYR     R
     A             STATE     R
     A
```

图 1-4　用字段引用文件建立的物理文件(ORDHDRP)的 DDS 码

REF 键字(45 列到 80 列)中指定的 DSPREFP 文件来确定在 ORDHDR 记录格式中字段的属性。假定图 1-4 是源文件 QDDSSRC，成员名为 ORDHDRP 的 DDS 描述。为 了生成 ORPHDRP 文件，可以用生成物理文件命令(CRTPF)：

```
CRTPF    FILE(DSTPRODLB/ORDHDRP)
         TEXT('Order Header Physical file')
```

注：本指南中一些例子里的字段都是引用这个字段引用文件中的字段。

1.1.1.4.5 使用数据字典进行字段引用

可以用数据字典和 IDDU 而不用 DDS 字段引用文件。IDDU 允许在数据字典中定义字段，详细请看《IDDU 使用手册》。

1.1.1.4.6 共享已有的记录格式描述

一个记录格式可以在物理文件或逻辑文件(除了连接逻辑文件)中描述，并且可以被多个文件所使用。当要描述一个新文件时，可以指定新文件使用一个已有记录格式，它将有助于减少正常编码所需的 DDS 语句，还可以节省辅助存贮空间。

最初描述记录格式的文件可以删除，但并不影响文件对记录格式的共享。若将使用此记录格式的最后一个文件删除，系统将自动的删除该记录格式描述。

下面是两个文件的 DDS 描述，第一个文件描述了一个记录格式，第二个文件共享第一个文件的记录格式：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R RECORD1                   PFILE(CUSMSTP)
     A             CUST
     A             NAME
     A             ADDR
     A             SEARCH
     A           K CUST
     A
```

图 1-5　逻辑文件 DDS 描述(CUSMSTL)

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R RECORD1                   PFILE(CUSMSTP)
     A                                       FORMAT(CUSMSTL)
     A           K NAME
     A
```

图 1-6　共享一个记录格式的逻辑文件 DDS 描述(CUSTMSTL1)

图 1-5 例子中显示了文件 CUSMSTL，它所包含的字段有 Cust,Name,Addr 和 Search，它们组成了一个记录格式，字段 Cust 被指定为索引字段。

图 1-6 给出 CUSTMSTL1，其中键字 FORMAT 指定文件 CUSMSTL 来提供记录格式。记录格式名必须是 RECORD1，要和图 1-5 中显示的记录格式相同。因为文件共享一个记录格式，所以两个文件的字段都有 Cust,Name,Addr 和 Search。在文件 CUSTMSTL1 中指定了 Name 为索引字段。

下面是有关共享记格式的限制：

- 物理文件不能共享逻辑文件的记录格式
- 连接逻辑文件不能共享其它文件的记录格式，并且其它文件也不能共享连接逻辑文件的记录格式
- 视图不能共享其它文件的记录格式，其它文件也不能共享视图的格式。(在 SQL 中，视图是来自于一个或多个表中的某些数据。一个视图可以包括一个或多个表中的所有列或部分列)

如果通过删除所有相关文件，并且再次生成原始文件和所有相关文件，使得原始记录格式发生相应的改变。如果只是具有原始记录格式有文件被删除并重新建立一个新的记录格式，由所有共享先前记录格式的文件将继续使用原始的格式。

如果定义了一个逻辑文件，但没有指定字段描述，也没有指定 FORMAT 键字，则自动共享第一个物理文件(逻辑文件里 PFILE 键字指定的第一个文件)中的记录格式。逻辑文件中指定的记录格式名必须和物理文件中记录格式名一样。

为了确定一个文件是否和其它文件共享一个记录格式，可在显示数据库关系命令(DSPDBR)中使用 RCDFMT 参数。

1.1.1.5　描述文件的访问路径

访问路径描述了检索记录的顺序。物理文件或逻辑文件中的记录可以使用到达顺序访问路径或键字顺序（索引顺序）访问路径来检索。对于逻辑文件，也可以根据每个记录中一个或多个字段值来选择和省略记录。

1.1.1.5.1 到达顺序访问路径

到达顺序访问路径主要依据记录到达及在文件中存放的顺序。要进行读取或更新，访问记录的方式有：

- 按一定顺序，按记录在文件中的物理位置顺序来取。
- 直接由相关记录号，从文件起始位置来标识记录的顺序。

一个外部说明文件中没有索引字段时，用到达顺序访问路径。

一个到达顺序访问路径只在下列情况下有效：

- 物理文件
- 逻辑文件，并且每个逻辑文件成员都只引用一个物理文件成员
- 连接逻辑文件
- 视图

注意：

1.到达顺序只是一种处理方式，它允许程序使用已删除记录所占用的存储空间，来放另外的记录。要求有相关记录号的记录要有明确的插入。系统用来管理由删除记录而产生的空间的另一种方式是，重用物理文件中删除记录的属性。有关使用重用删除记录属性的详细资料，请看 2.1.2.3”重用删除记录”。有关处理删除记录的详细资料，请看 2.3.5”删除数据库记录”。

2.可以使用高级语言中物理文件成员的命令(DSPPFM)和拷贝文件命令(CPYF)，用到达顺序来处理键字顺序的文件。还可以将此功能用于物理文件，简单逻辑文件或连接逻辑文件。

3.使用高级语言，可以通过相关记录号来处理键字顺序的文件。它适用于物理文件、简单逻辑文件或连接逻辑文件。

4.到达顺序访问路径不占用任何多余的存储空间。它总是随着文件一起保存或恢复。(由于到达顺序访问路径在被存时近似于数据的物理顺序，所以当你保存数据时，也就保存了到达顺序的访问路径)。

1.1.1.5.2 键字顺序访问路径

键字顺序访问路径主要依据 DDS 中定义的索引字段内容，无论是添加、删除记录还是将记录更新，改变索引字段内容，此类访问路径都将被更新。键字顺序访问路径对物理文件和逻辑文件是有效的。文件记录的顺序，在文件生成时，由 DDS 定义，并由系统自动维护。

索引字段为字符型时，依据 EBCDIC 字符定义的顺序来排列。索引字段为数字型时，依据其代数值来排列，除非该字段指定了 UNSIGNED(无符号值)或 ABSVAL(绝对值)DDS 键字。允许索引字字段定义为 DBCS，但只能依据其位表达式按单字节来排序。

- 用替换顺序来安排索引字段

  字符型的索引字段可以依据 EBCDIC 字符定义的顺序或一个替换顺序来排列，考虑下列记录：

记录号	雇员名	单　位	雇员号
1	Jones, Mary	45	23318
2	Smith, Ron	45	41321
3	JOHNSON, JOHN	53	41322
4	Smith, ROBERT	27	56218
5	JONES, MARTIN	53	62213

如果 Empname 字段是索引字段并且是一个字符字段，使用 EBCDIC 字符顺序，记录的排列顺序如下：

记录号	雇员名	单　位	雇员号
1	Jones, Mary	45	23318
3	JOHNSON, JOHN	53	41322
5	JONES, MARTIN	53	62213
2	Smith, Ron	45	41321
4	Smith, ROBERT	27	56218

注：按 EBCDIC 顺序要求小写字符排在大写字符前面，所以它可能会导致意外的分类顺序。因此，Smith.Ron 排在 Smith.ROBERT 前。当记录是以大写和小写混和录入时，一个替换顺序可用于排列记录，顺序如下：

记录号	雇员名	单　位	雇员号
3	JOHNSON, JOHN	53	41322
5	JONES, MARTIN	53	62213
1	Jones, Mary	45	23318
4	Smith, ROBERT	27	56218
2	Smith, Ron	45	41321

要对字符型索引字段使用替换顺序，必须指定 ALTSEQ 键字，还要指定包含替换顺序的表名。建立表时，表中每两个字节对应一个字符。若要改变表中字符排列的顺序，必须将这两位值变成和它所要转换的字符相同的值。

有关 ALTSEQ 键字的详细资料，将由库 QUSRSYS 中的 QCASE256 表提供给你。

- 用 SRTSEQ 参数排列索引字段

  你可以根据几个带 SRTSEQ 参数的有效的排列顺序来排列含有字符数据的索引字段，考虑下列记录：

记录号	雇员名	单　位	雇员号
1	Jones, Marilyn	45	23318
2	Smith, Ron	45	41321
3	JOHNSON, JOHN	53	41322
4	Smith, ROBERT	27	56218
5	JONES, MARTIN	53	62213
6	Jones, Martin	08	29231

如果字段 Empname 是索引字段并且是一个字符型字段，*HEX 顺序(EBCDIC 顺序)将如下排列记录：

记录号	雇员名	单　位	雇员号
1	Jones, Marilyn	45	23318
6	Jones, Martin	08	29231
3	JOHNSON, JOHN	53	41322

5	JONES, MARTIN	53	62213
2	Smith, Ron	45	41321
4	Smith, ROBERT	27	56218

注：*HEX 顺序，所有的小写字符都排在大写字符前。因此，Smith.Ron 在 Smith.ROBERT 前，并且 JOHNSON.JOHN 排在大写和小写的 Jones 之间。当记录中用大写和小写混合键入时，可以用*LANGIDSHR 排列顺序来排列记录。*LANGIDSHR 对大写和小写字符同样处理，结果如下：

记录号	雇员名	单　位	雇员号
3	JOHNSON, JOHN	53	41322
1	Jones, Marilyn	45	23318
5	JONES, MARTIN	53	62213
6	Jones, Martin	08	29231
4	Smith, ROBERT	27	56218
2	Smith, Ron	45	41321

注：*LANGIDSHR 顺序中，小写和大写字符将同样对待。因此，JONES，MARTIN 和 Jones,Martin 是相同的，并且按照它们在库文件中相同的顺序排列。这样排列并非是错误的，但如果让所有的小写 Jones 排在大写 JONES 之前，在报告中看起来会更好些。所以当记录是用大写和小写混合键入时，可使用*LANGIDUNQ 来排列记录。*LANGIDUNQ 排序区分大写和小写，结果如下：

记录号	雇员名	单　位	雇员号
3	JOHNSON, JOHN	53	41322
1	Jones, Marilyn	45	23318
6	Jones, Martin	08	29231
5	JONES, MARTIN	53	62213
4	Smith, ROBERT	27	56218
2	Smith, Ron	45	41321

*LANGIDSHR 和*LANGIDUNQ 排序适用于系统支持的每一种语言，参数 LANGID 决定是使用*LANGIDSHR 排列顺序还是使用*LANGIDUNG 顺序。用 SRTSEQ 参数来指定排列顺序，用 LANGID 参数来指定语言标识。

• 以升序或降序来排列索引字段

索引字段可以以升序或降序来排列，考虑下列记录：

Record	Empnbr	Clsnbr	Clsnam	Cpdate
1	56218	412	Welding I	032188
2	41322	412	Welding I	011388
3	64002	412	Welding I	011388
4	23318	412	Welding I	032188
5	41321	412	Welding I	051888

6	62213	412	Welding I	032188

如果 Empnbr 字段是索引字段，记录的排列有两种可能：

按升序排列，访问路径中记录序号的位置为：

Record	Empnbr	Clsnbr	Clsnam	Cpdate
4	23318	412	Welding I	032188
5	41321	412	Welding I	051888
2	41322	412	Welding I	011388
1	56218	412	Welding I	032188
6	62213	412	Welding I	032188
3	64002	412	Welding I	011388

按降序排列，访问路径中记录序号的位置为：

Record	Empnbr	Clsnbr	Clsnam	Cpdate
3	64002	412	Welding I	011388
6	62213	412	Welding I	032188
1	56218	412	Welding I	032188
2	41322	412	Welding I	011388
5	41321	412	Welding I	051888
4	23318	412	Welding I	032188

当描述一索引字段时，缺省值为升序。但是，也可以使用 DESCEND 键字来指定按降序来排列。

使用多个索引字段：可以用多个索引字段来排列一个文件中的记录，这些字段不必使用同一顺序。例如，当使用两个索引字段时，一个字段用升序排列，另一个字段用降序排列，考虑下列记录：

Record	Order	Ordate	Line	Item	Qtyord	Extens
1	52218	063088	01	88682	425	031875
2	41834	062888	03	42111	30	020550
3	41834	062888	02	61132	4	021700
4	52218	063088	02	40001	62	021700
5	41834	062888	01	00623	50	025000

如果访问路径使用了 Order 字段，并且 Line 字段作为索引字段，两个均按升序排列，则访问路径中记录的顺序为：

Record	Order	Ordate	Line	Item	Qtyord	Extens
5	41834	062888	01	00623	50	025000
3	41834	062888	02	61132	4	021700
2	41834	062888	03	42111	30	020550

1	52218	063088	01	88682	425	031875
4	52218	063088	02	40001	62	021700

如果访问路径中字段 Order 按升序排列，而 Line 字段按降序排列，则访问路径中记录的顺序为：

Record	Order	Ordate	Line	Item	Qtyord	Extens
2	41834	062888	03	42111	30	020550
3	41834	062888	02	61132	4	021700
5	41834	062888	01	00623	50	025000
4	52218	063088	02	40001	62	021700
1	52218	063088	01	88682	425	031875

在同一文件中，当一条记录的索引字段的内容和另一记录索引字段内容相同时，文件被认为含有重复键值的记录。但是，如果它们被叫做重复键值，必须记录中所有索引字段都发生重复。例如：如果一个记录格式有两个索引字段 Order 和 Ordate。当两个或多个记录中 Order 和 Ordate 字段内容是一样时，说明出现了重复键值。有重复键值的记录如下：

Order	Ordate	Line	Item	Qtyord	Extens
41834	062888	03	42111	30	020550
41834	062888	02	61132	04	021700
41834	062888	01	00623	50	025000

用文件的 Line 字段作为第三个键字段，以便不出现重复键字：

(First Key Field) Order	(Second Key Field) Ordate	(Third Key Field) Line	Item	Qtyord	Extens
41834	062888	03	42111	30	020550
41834	062888	02	61132	04	021700
41834	062888	01	00623	50	025000

多格式逻辑文件可能含有重复键值的记录，即使记录格式是建立在不同物理文件基础上。也就是说，尽管键值是来自于不同记录格式，它们也被认为是重复键值。

避免重复键值：AS/400 数据库管理系统允许文件中有带重复键值的记录。但是，也应尽量防止某些文件中出现重复键值。例如：可以建立一个文件，文件中索引字段可定义为顾客号字段。在这样情况下，要系统确保文件中每个记录有唯一的顾客号。

可以通过在 DDS 是指定 UNIQUE 键字，来避免文件出现重复键值。指定了 UNIQUE 键字，如果键值和文件中已存在的记录的键值一样，记录就不能录入或复制到文件中。可以用唯一性约束来加强唯一键字的完整性。有关约束的详细信息，请看第 15 章”物理文件的约束”。

如果物理文件中存在带有重复键值的记录，则相关的逻辑文件不能指定 UNIQUE 键字。如果试图要生成一个指定了 UNIQUE 键字的逻辑文件，并且相关的物理文件中含有重复键值，则此逻辑文件不能生成，系统将返回一条不能生成的信息，并发送信息(信息级别为 20)指出哪些记录重复键值。

当文件中指定了 UNIQUE 键字时，加入文件中的任何记录都不能和文件中现存的记录

有同样键值，尽管文件是用于添加一个新的记录。例如：两个逻辑文件 LF1 和 LF2 都是依据物理文件 PF1 建立的，文件 LF1 中指定了键字 UNIQUE。假设用 LF2 对 PF1 添加记录，若它使得 LF1 中引起了重复键值，则不能增加此记录。

如果某些索引字段允许为空值，那么插入这些字段的空值可不可能引起重复将取决于文件生成时的访问路径是如何定义的。UNIQUE 键字参数*INCNULL 表明确定现存访问路径是否有重复时，包括空值的情况。*EXCNULL 参数表明在确定现存访问路径中是否存在重复值时，不包括空值。有关详细资料，请参考《DDS 参考手册》。

下面显示了要求有唯一键值的逻辑文件的 DDS 描述：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER TRANSACTION LOGICAL FILE (ORDFILL)
     A                                       UNIQUE
     A           R ORDHDR                    PFILE(ORDHDRP)
     A           K ORDER
     A
     A           R ORDDTL                    PFILE(ORDDTLP)
     A           K ORDER
     A           K LINE
     A
```

在这个例子中，不论记录是通过 ORDHDRP 文件、ORDDTLP 文件，还是这里定义的逻辑文件增加的，其索引字段的内容(ORDHDRP 记录格式的 Order 字段和 ORDDTLP 记录格式中 Order 和 Line 字段)必须是唯一的。在 ORDDTL 记录格式中指定 Line 字为第二个索引字段。则两个物理文件中的 Order 字段的值可能相同。因为物理文件 ORDDTLP 有两个索引字段，而物理文件 ORDHDRP 只有一个索引字段，则两个文件中的键值不会发生冲突。

排列重复键字：如果在 DDS 中没有指定 UNIQUE 键字，可以指定系统如何存储带有重复键值的记录。可以用下列方法之一来做。

- 后进先出(LIFO)：当指定了 LIFO 键字(1)时，带重复键值的记录将按记录的物理顺序，以后进先出的方式检索。下面是使用 LIFO 键字的 DDS 例子：

|. . . +. . . . 1. . . . +. . . . 2. . . . +. . . . 3. . . . +. . . . 4. . . . +. . . . 5. . . . +. . . . 6. . . . +. . . . 7. . . . +. . . . 8

```
A* ORDERP2
```

```
A                                     1    LIFO
A           R ORDER2
A             .
A             .
A             .
A           K ORDER
A
```

- 先进先出(FIFO)：如果指定了 FIFO 键字，带有重复键值的记录将按记录的物理顺序先进先出方式被检索。
- 先变先出(FCFO)：如果指定了 FCFO 键字，带有重复键字的记录将按键字的物理顺序以先变先出的方式检索。
- 不指定重复索引字段的顺序(缺省)：当 FIFO、FCFO 或 LIFO 键字均没有指定时，检索重复键字记录的顺序就没有一定。若重复字段没有特定的顺序，则允许多个访问路径共享，这可提高其性能。有关共享访问路径的详细资料，请看 1.3.2.2”使用现存的访问路径”。

当一个简单或多格式逻辑文件以多个物理文件成员为依据时，带有重复键值记录的读取顺序将取决于在生成逻辑文件(CRTLF)或增加逻辑文件成员(ADDLFM)命令中的参数 DTAMBRS 指定的文件和成员的顺序。多格式逻辑件的范例可看《DDS 参考手册》。

带有重复键值的 LIFO 和 FIFO 次序并不是由索引字段内容更新的顺序决定的，而是由文件成员中记录的物理顺序单独决定的。假设物理文件中指定的 FIFO 键字(带有重复键值的记录以先进先出的顺序被检索)，下面显示了记录加入文件的次序：

加入文件中的记录的顺序	键值
1	A
2	B
3	C
4	C
5	D

访问路径的顺序为(FIFO 升序)：

记录号	键值
1	A
2	B
3	C
4	C
5	D

记录 3 和记录 4 带有重复键值，并按 FIFO 次序排列，也就是说，由于记录 3 在记录 4 之前加入文件中，也将在记录 4 之前读出。记录按降序读出，那将会更明显。可以通过在逻辑文件中指定 DESCEND 键字，并生成此逻辑文件来实现这种情况。

访问路径的顺序是(FIFO，降序)：

记录号	键值
5	D
3	C
4	C
2	B
1	A

如果物理记录 1 将键值改为 C，则物理文件中访问路径的顺序为(FIFO，升序)：

记录号	键值
2	D
1	C
3	C
4	C
5	B

最后，若改为降序排列，则逻辑文件访问路径的新次序为(FIFO,降序)：

记录号	键值
5	D
1	C
3	C
4	C
2	B

改变后，记录 1 并不出现在记录 4 之后，尽管其键字段的内容是在记录 4 加入后被更新的。

带有重复键值记录的 FCFO 次序是由键字段内容的更新顺序决定的。在上一例中，记录 1 的键值被改为 C 后，访问路径的顺序为(FCFO，升序)：

记录号	键值
2	B
3	C
4	C
1	C
5	D

对于 FCFO，只有当 FCFO 访问路径被重建或者执行了退回(Roolback)操作时，重复键字次序才能改变。在某些情况下，索引字段可能修改了，但物理键值没有改变。这种情况，尽管索引字段改变了，但 FCFO 次序没有变化。例如：当检索顺序变为索引绝对值为基础时，FCFO 次序也不发生改变。尽管你的键值由负变正，但键字的物理值没有发生变化。由于键字的物理值没变，所以 FCFO 的次序也就不变。

如果物理文件中指定了重用已删除记录的属性，则重复键次序必须为缺省值或是

FCFO。如果文件的键字次序是 FIFO 或 LIFO，或者依据这些物理文件的某个逻辑文件具有 FIFO 或 LIFO 的重复键字次序，则物理文件不允许重用已删除记录的属性。

1.1.1.5.3 使用现存的访问路径规则

可以使用 DDS 的 REFACCPTH 键字来使用别的文件的访问路径。当文件生成时，系统决定共享哪一访问路径。使用 REFACCPTH 键字的文件并不一定要共享 REFACCPTH 键字所指定的文件的访问路径。REFACCPTH 键字一般只用来减少所必需的 DDS 语句数，也就是说，指定了 REFACCPTH 键字，就不必对文件索引字段进行编码了。当生成文件时，系统将由 REFACCPTH 键字所指定的文件的索引字段和选择/省略字段规则拷贝到生成的文件中。

1.1.1.5.4 在访问路径中使用浮点型字段

一个索引文件中记录的排列顺序依赖于键字 SIGNED、UNSIGNED 和 ABSVAL 的说明。对于浮点型字段，最左边一位是符号位，接着是指数位，最后一位是有效位。指定了 UNSIGEND 的排列顺序为：

正实数~正无穷大
负实数~负无穷大

由 SIGNED 键字指定的或缺省的浮点型索引字段，在 DDS 中都有一定的顺序，其排列顺序为：

负无穷大~实数~正无穷大

由 ABSVAL 键字指定的浮点型字段在 DDS 中有一定的绝对值顺序。下面是浮点型字段的排列顺序：

- 0(正或负)的排列和其他正负实数一样
- 在 SIGNED 顺序中，负 0 排在正 0 前面
- 在 ABSVAL 顺序中，负 0 和正 0 一样排列

不能在索引字段使用非数字型(*NAN)值，如果这样做，在文件生成时如检查到索引字段的*NAN 值，则文件不能生成。

1.1.2 保护和监控数据库数据

系统提供了两种方式来改善数据的完整性和一致性

- 引用性约束要求控制(约束)你所定义的具有相互关系的文件中的数据。当约束使文件修改时，引用约束给你规定一些要遵循的原则。约束的详细描述见第 16 章” 引用的一致性”。
- 当文件修改时，触发操作可以让你运行自己的程序，以便采取某些动作或测定修改。要做予定的修改时就运行触发程序。有关触发的详细描述见第 17 章”触发”。

1.1.3 生成数据库文件(简介)

系统提供生成数据库文件的几种方式：

- OS/400 IDDU
- 结构查询语言(SQL)
- OS/400 控制语言(CL)

可以用 IDDU 来生成一个数据库文件。假设用 IDDU 来描述数据库文件，也应该考虑用它来生成文件。

用 SQL 语句来生成一个数据库文件。SQL 语言是 IBM 系统应用体系的相关数据库语言，

它可以在 AS/400 系统中交互式的描述和生成数据库文件。

使用 CL 也可以生成数据库文件。CL 数据库文件生成命令是：生成物理文件(CRTPF)，生成逻辑文件(CRTLF)，生成源物理文件(CRTSRCPF)。

由于附加的系统功能，所以本指南将主要侧重于用 CL 生成文件。

1.1.4 数据库文件和成员的属性(简介)

当生成一个数据库文件时，数据库的属性便随着文件和成员一起被存储，可以用数据库命令参数来指定这些属性。有关指定这些属性和它们选值的详述，请看《CL 参考手册》中的 CRTPF、CRTLF、CRTSRCPF、ADDPFM、ADDLFM、CHGLF、CHGPFM、CHGSRCPF 和 CHGLFM 命令。

1.1.4.1 文件名和成员名(参数 FILE 和 MBR)

可用生成命令的 FILE 参数对一文件命名，也可对文件所属的库来命名。当要生成一个物理文件或逻辑文件时，系统通常是先建立一个和该文件同样名称的成员。但是，也可用生成命令中的 MBR 参数来指定一成员名，也可在生成命令中指定 MBR(*NONE)来选择不生成任何成员。

注意：系统不会自动在地源物理文件中建立一个成员。

1.1.5 控制物理文件成员(参数 DTAMBRS)

使用生成逻辑文件(CRTLF)命令中的 DTAMBRS 参数，可以控制物理文件成员的读出。可以指定：

- 该物理文件中成员的次序
- 使用物理文件成员的个数

有关用此方法使用逻辑文件的详细资料，请看 1.3.3.2”逻辑文件成员”。

1.1.5.1 源文件和源成员(参数 SRCFILE 和 SRCMBR)

参数 SRCFILE 和 SRCMBR 指定了源文件和包含定义要生成文件的 DDS 语句的成员名。如果没有指定一个名称：

- 源文件名的缺省值为 QDDSSRC
- 缺省的成员名就是参数 FILE 指定的名称

1.1.5.2 数据库文件类型(参数 FILETYPE)

一个数据库文件类型即可以是数据型(*DATA)也可以是源类型(*SRC)，生成物理文件(CRTPF)和生成逻辑文件(CRTLF)命令使用的是缺省的数据文件类型(*DATA)。

1.1.5.3 允许成员的最大数(参数 MAXMBRS)

参数 MAXMBRS 指定了文件所能拥有的成员的最大数目。物理文件和逻辑文件成员最大数的缺省值是相同的。源物理文件缺省值为*NOMAX。

1.1.5.4 数据存储地址(参数 UNIT)

系统为文件在辅助存贮器上留有位置。为了指定存储文件的地址，可以使用 UNIT 参数。UNIT 参数可以规定：

- 物理文件中数据记录位置
- 物理文件和逻辑文件的访问路径

下列情况数据将放置在不同的存储单元：

• 在单元中没有足够的空间

• 单元对系统无效

当增加文件成员时，会有一条信息指出文件没有放在所请求的单元中(当文件成员扩充时，没有信息)。

1.1.5.4.1 UNIT 参数的要点

通常，不必指定 UNIT 参数，系统自动选择在磁盘单元中存放文件的地址。这样对性能有好处，也可减轻管理辅存的工作。

如果指定了一个存储单元号和一个辅助存贮池(ASP)，则单元号被忽略。有关辅助存储池的详细资料，请看备份和恢复的有关书目。

1.1.5.5 往辅存上写数据的频率(参数 FRCRATIO)

在使用生成、修改或复盖数据库命令时，可用参数 FRCRATIO 来控制往辅存写入的频率。在正常情况下，系统决定何时把修改的数据从主存写入辅存。关闭文件(除了共享性关闭外)和强行结束数据的操作将强行保留对辅存的更新、删除和增加。如果做了文件的日志，则参数 FRCRATIO 通常为*NONE。

1.1.5.5.1 FRCRATIO 参数的要点

使用 FRCRATIO 参数要考虑到对系统的性能和恢复的影响。要了解这些事项，请看第 13 章”数据库的恢复考虑”。

1.1.5.6 往辅存写访问路径的频率(参数 FRCACCPTH)

强制写访问路径参数(FRCACCPTH)用于控制何时将访问路径写入辅存中，无论何时该存储路径改变，FRCACCPTH(*YES)都强行把访问路径转入辅助存储器。这可以减少重建访问路径可能性，以至系统失败。

1.1.5.6.1 FRCACCPTH 参数的要点

当访问路径发生变化时，指定 FRCACCPTH(*YES)将降低系统性能。可代替的方法是对访问路径做日志。有关强制访问路径和日志访问路径的详细资料，请看第 13 章”数据库的恢复考虑”。

1.1.5.7 检查记录格式描述的修改(参数 LVLCHK)

当打开文件时，系统要检查数据库文件定义是否做过修改。当文件所做的修改使你的程序不能处理此文件时，系统将通知程序。缺省值是做级别检查。下列情况下，可指定是否要做级别检查。

• 生成文件

• 使用修改数据库文件的命令

可以用覆盖数据库文件的命令(OVRDBF)来忽略级别检查。

1.1.5.7.1 级别检查举例

假设在两个月以前对程序进行编译。那时，程序定义的文件中每个记录有三个字段。上星期另一个程序决定在记录格式中增加一个新字段，使得每个记录有四个字段。当程序试图打开文件时，系统会通知程序，在程序上次被编译后，文件定义发生了明显变化，这就叫记

录格式的级别检查。

1.1.5.8　当前访问路径的维护(参数 MAINT)

参数 MAINT 指定了如何维护关闭了的文件的访问路径。当文件打开时，由于文件中数据发生了变化，系统要维护其访问路径，但是，由于同一数据可能多个访问路径，对一个文件中数据的修改可能会引起当前还没打开(使用)的其它文件访问路径的变化。以下是维护关闭文件访问路径的三种办法：

• 访问路径的立即维护

它意味着无论一个文件是否被打开，其访问路径将随着相关数据的变化而维护。被引用约束使用的访问路径总是立即维护的。

• 访问路径的重建维护

它意味着访问路径只有当文件打开才维护，而不是在文件关闭时。当文件下一次打开时，将重建路径。当有重建维护的文件被关闭时，系统将停止维护其访问路径。当文件又一次被打开时，路径将整个被重建。如果一个或多个程序打开了一个具有指定重建维护的特定文件成员，系统将一直维护成员的访问路径，直到最后一个用户关闭该文件成员。

• 访问路径的延迟维护

它意味着当文件成员被再次打开或当它一直打开时，对访问路径可作任何维护。但是，其访问路径并不能象重建维护那样被重建。从关闭成员到它被再次打开，对访问路径的更新将被收集起来。当它打开时，收集其来的这些修改归并到访问路径中去。如果没有指定文件维护的类型，缺省为立即维护。

1.1.5.8.1　MAINT 参数的比较

图 1-7 比较了立即、重建、延迟维护三种类型对打开和处理文件的影响。

图 1 – 7　MAINT 值

功能	立即维护	重建维护	延迟维护
打开	由于是当前的访问路径，所以打开快处比较慢	由于访问路径必须被重建，所以打开慢处比较快	由于访问路径不必重建，但它必须总修改，所以打开为中速。如果有扩充修改时，则打开较慢。中速
处理	要维护这种修改数据的多个访问路径更新和输出操作比较慢（系统必须维护其访问路径）	要维护在修改数据和处理没有打开有多个访问路径时，更新和输出操作较快（系统不必维护其存取路径）	要维护在修改数据和处理没有打开的多个访问路径时，更新和输出操作为中还（系统记录所作的修改，但不维护其访问路径）
注意： 1.带有唯一键字的文件不能指定延迟和重建维护 2.如果文件的访问路径被日志，则不能指定重建维护			

1.1.5.8.2 MAINT 参数的要点

指定访问路径的维护类型取决于文件中记录的个数和文件关闭时对文件的增加、删除和更新的频率。

当文件成员被关闭时，访问路径有少量变化的文件可以使用延迟维护。延迟维护可通过减少立即维护访问路径的个数来减轻系统管理的开销。由于访问路径不必重建，所以可以快速打开。

可以对频繁使用的访问路径指定为立即维护，在文件打开时若不想等待一个访问路径被重建，也可以指定立即维护。如果组成访问路径的记录键字变化不频繁时，可以对不频繁使用的访问路径指定为延迟维护。

通常，对于交互使用的文件，立即维护可有较好的响应时间。对于批处理文件，无论是立即、延迟还是重建维护都是适用的，它取决于成员的大小和变化的频率。

1.1.5.9　访问路径的恢复(参数 RECOVER)

系统发生故障后，那些没有强行写入辅存的访问路径或做了日志的访问路径不能使用，直到它们重建为止。可以用生成物理文件(CRTPF)、生成逻辑文件(CRTLF)和生成源物理文件(CRTSRCPF)命令中的 RECOVER 参数来指定何时重建访问路径。在初始程序装入(IPL)期间或是装入后，文件打开时，均可重建访问路径。

图 1-8 显示了可选的恢复键字和维护类型的几种组合：

图 1 – 8　恢复选项

带有重复键字选项	维护选项	恢复选项
唯一	立即	在 IPL(*IPL)期间重建 在 IPL(*AFTIPL)后重建 在 IPL(*NO)期间不重建，直到首次打开时。
不唯一	立即或延迟	在 IPL(*IPL)期间重建 在 IPL(*AFTIPL)后重建 在 IPL(*NO 缺省)期间不重建，直到首次打开时。
不唯一	重建	在 IPL(*NO 缺省)期间不重建，直到首次打开时。

1.1.5.9.1 恢复参数要点

如果 IPL 为手动方式，在下一次的 IPL 期间，在编辑重建访问路径的显示中给出需要恢复访问路径的文件列表下，可以通过显示中的选项来编辑对文件初始恢复。在 IPL 完成后，可以用访问路径的重建编辑命令(EDTRBDAP)来形成访问路径的重建顺序。如果是正常的 IPL，则不会出现显示，访问路径将按照参数 RECOVER 决定的顺序被重建，你只能看到 *AFTIPL 和*NO(打开)的访问路径。

1.1.5.10　文件共享(参数 SHARE)

数据库系统允许多个用户在同一时间内存取和修改同一个文件，参数 SHARE 允许共享在同一作业中的打开的文件。例如：对一个作业中文件的共享允许作业中的程序共享文件的状态、记录位置和缓冲区。作业中文件的共享还可以通过减少：

- 作业所需的存贮空间
- 打开和关闭文件所需的时间

来提高其性能。

有关同一作业文件共享文件的详细资料，请看 2.1.5 中的”对同一作业或活动组中数据库文件的共享”。

1.1.5.11　锁定文件或记录等待时间(参数 WAITFILE 和 WAITRCD)

当生成文件时，如果另一作业锁定了该文件或记录，可以指定程序等待此文件或记录的时间。如果在文件或记录被解锁前等待时间到，系统将发送一条信息给程序，说明该作业不能使用此文件或不能读记录。有关记录和文件锁定及等待时间的详细资料，请参见 2.1.4.1”记录锁定”和 2.1.4.2”文件锁定”。

1.1.5.12 公共权限(参数 AUT)

当生成文件时，可以指定公共权限。公共权限是指在用户对一个文件没有特别权限或不属于文件特权组的前提下，用户所拥有的文件的权限，有关公共权限详细资料请看 1.4.2 中”公共权限”。

1.1.5.13 指定生成文件的系统(参数 SYSREM)

可以指定文件是生成在当前系统上还是生成在支持分布式数据管理(DDM)的远程系统上。有关 DDM 的详细资料，请参见《分布数据管理手册》。

1.1.5.14 文件和成员说明(参数 TEXT)

可以对你建立的每个文件和成员指定一个说明。该说明在文件和成员的描述信息中是很有用的。

1.1.5.15 编码的字符集标识(参数 CCSID)

可以对物理文件指定一个编码的字符集标识，键字 CCSID 描述了编码方案和文件中字符型字段的字符集。

有关 CCSID 的详细资料，请看《各国语言支持》。

1.1.5.16 分类顺序(参数 SRTSEQ)

可以指定文件的分类顺序。SRTSEQ 参数值总是和参数 CCSID 和 LANGID 一起用，决定文件使用哪个分类顺序表。可以在物理文件和逻辑文件中使用 SRTSEQ 参数，可以指定的有：

- 系统提供带有唯一的或共享排序权的分类顺序表，系统支持的每种语言都有一分类顺序表
- 任何一个用户建立的分类顺序表
- 字符集中字符的十六进制值
- 当前作业中的分类顺序或由 ALTSEQ 参数指定的分类顺序。

除了分类顺序为*HEX 的外，分类序列表将和文件一起存贮。

1.1.5.17 语言标识(参数 LANGID)

当 SRTSEQ 参数值为*LANGIDSHR 或*LANGIDUNQ 时，可以指定系统所用语言标识符。LANGID、CCSID 和 SRTSEQ 的参数值，将决定文件使用哪个分类顺序表，可以对物理文件和逻辑文件使用 LANGID 参数。

可以指定系统支持的任何语言标识符或者当前作业中使用的语言标识符。

第二章　　建立物理文件

1.2　本章讲述有关描述和生成物理文件的问题。

有关描述一个物理文件记录格式的详细资料，请看 1.1.1.4.1 中”用 DDS 描述物理文件举例”。

有关描述物理访问路径的详细资料，请参见 1.1.1.5”描述文件的访问路径”

1.2.1 生成物理文件

生成物理文件的步骤如下：

1.如果使用 DDS，要将物理文件中的 DDS 输入源文件中，这项工作可以使用 AS/400 应用开发工具的源语句录入实用程序(SEU)。有关源语句如何录入到源文件的详细资料，请见 3.5.3”处理源文件”。

2.生成物理文件，可以用生成文件命令(CRTPF)或生成源物理文件命令(CRTSRCPF)。

下面的命令，生成了一个使用 DDS 描述的只有一个成员的文件，并将其放入名为 DSTPRODLB 的库中：

```
CRTPF       FILE(DSTPRODLB/ORDHDRP)
            TEXT('Order herder physical file')
```

如上显示，该命令使用了参数 SRCFILE 和 SRCMBR 的缺省值。系统指定源文件为 QDDSSRC，成员名为 ORDHDRP(和文件名同)，含有一个同名成员的文件 ORDHDRP 将放入库 DSTPRODLB 中。

1.2.2 规定物理文件和成员的属性

可以使用生成物理文件(CRTPF)、生成源文件(CRTSRCPF)、修改物理文件(CHGPF)、修改源物理文件(CHGSRCPF)、增加物理文件成员(ADDPFM)和修改物理文件成员(CHGPFM)的命令来指定物理文件和成员某些属性，它包括以下几节的内容：

1.2.2.1　有效日期

参数 EXPDATE 为文件中每个成员指定了有效日期(ADDPFM、CHGPFM、CRTPF、CHGPF、CRTSRCPF 和 CHGSRCPF 命令)。如果超过有效日期，当文件打开时有信息要通知系统操作员。然后，系统操作员将取消该有效日期，它用于指定成员何时被加入文件(有效日期的检查可以被取代，详见 2.1.3.4”文件有效日期的检查”)。

1.2.2.2　物理文件成员的尺寸

参数 SIZE 指定了每个成员中放入记录的最大数目(CRTPF、CHGPF 和 CHGSRCPF 命令)。下面的公式可用于决定此最大值：

R + (I * N)

这里：R 表示起始记录数

I 表示每次增加的记录数(增量)

N 表示增加的次数

参数 SIZE 的缺省值为：

R：　10,000

I：　1,000

N： 3(CRTPF 命令)
499(CRTSRCPF 命令)

假设文件在生成时 R 为 5000，共分三次每次增加 1000 个记录。系统就会在初始记录数 5000 的基础上，加三次 1000 个记录，最后总数为 8000。当总数达到最大时，系统操作员便停止该作业，或告诉系统应该继续加增量。在增量加好时，会发送一条信息到系统的历史日志中。当文件扩充超出最大尺寸时，最小扩充量为当前尺寸的 10%，这时就不考虑它是否大于所指定的增量。

可以指定*NOMAX 来代替缺省值或规定值，有关文件允许记录最大数的详细资料，请见附录 A“数据库文件尺寸”。

1.2.2.3 存贮分配

当成员加入到文件时，用参数 ALLOCATE 来控制对成员存贮空间的分配(CRTPF、CHGPF、CRTSRCPF 和 CHGSRCPF 命令)。分配的空间足够装下成员的初始记录数。如果增加成员时，没有分配存贮空间，系统将自动扩充所需的存贮空间，只有在 SIZE 参数中指定了最大尺寸，才可使用 ALLOCATE 参数。如果指定 SIZE 为(*NOMAX)，则不能指定 ALLOCATE(*YES)。

1.2.2.4 分配存储区的方法

参数 CONTIG 控制给成员分配物理存储的方式(CRTPF 和 CRTSRCPF 命令)。如果要分配存储空间，可以请求分给一个成员的存储空间是连续的，也就是说，成员中的所有记录都是物理的存放在一起。成员增加时，如果没有足够的连续存储空间，将不能使用连续的存储分配，同时发送一条信息给作业以请求分配。

注：当物理文件第一次生成时，系统总是试图连续的分配它的初始存储空间。使用 CONTIG(*NO)和 CONTIG(*YES)唯一的不同是，当文件生成时，如果不能分配连续的存储空间，系统将发送信息给作业日志。在文件生成后再扩充，不管你在 CONTIG 参数上指定了什么，都不发送任何信息。

1.2.2.5 记录长度

参数 RCDLEN 指定了文件中记录的长度(CRTPF、CRTSRCPF)。如果文件只是记录级描述，当文件生成时就可以指定 RCDLEN 参数。如果文件是用 DDS、IDDU 或 SQL 描述的(系统自动根据文件的字段级描述来确定文件中记录的长度)，则不能指定该参数。

1.2.2.6 删除记录

参数 DLTPCT 规定一个文件中删除了多少百分比的记录时，系统才往历史日志中送信息(CRTPF、CHGPF、CRTSRCPF 和 CHGSRCPF 命令)。当文件被关闭时，系统检查该成员来确定已删除记录的百分比。如果该百分比超出了 DLTPCT 参数所指定的值，将发送信息到历史日志(有关处理历史日志的资料，请见《CL 程序设计》中处理信息的一章)。要知道文件的删除记录达到百分比原因，是想回收已删除记录所占用的空间。在你收到有关删除记录和信息后，可以用重新组织物理文件成员的命令(RGZPFM)来回收空间(有关 RGZPFM 的详细资料，请看第三部分 3.1.2.3 中”重新组织物理文件成员的数据”)。也可以在 DLTPCT 参数中使用*NONE 值来指定不检测删除的记录。DLTPCT 参数中*NONE 为缺省值。

参数 REUSEDLT 指定已删除记录所占用的空间在其后的写操作中是否能重用(CRTPF 和 CHGPF 命令)。在 REUSEDLT 参数中指定了*YES 时，对文件所有的插入请求都试图重用删除记录所用的空间。重用删除记录空间允许你不必使用 RGZPFM 命令即可回收删除记

录所用的空间。使用 CHGPF 命令来修改文件以便重用删除记录时，该命令将运行很长时间，特别是在文件很大并且文件中已存在一些删除记录时，占用的时间会更长。下面是一些重要的注意事项：

- 到达顺序对重用删除记录空间的文件来说已失去本来的意义。当删除记录的空间被重用时，记录已不再总是插入到文件的尾部。
- 如果使用重用删除记录空间的属性建立一个新的物理文件，并且该文件是索引的，则不能对此文件指定 FIFO 或 LIFO 访问路径的属性。对建在此物理文件基础上任何键字的逻辑文件也不能带有 FIFO 或 LIFO 访问路径的属性。
- 如果存在某些逻辑文件对重复键字指定了 FIFO 或 LIFO 顺序，或者是它所依据的物理文件用 FIFO 或 LIFO 的重复键字顺序，则不能为了重用删除记录所用的空间而改变现存的物理文件。
- 对于那些直接处理文件或者相对记录号处理的文件，不能指定重用删除记录空间。

注：有关重用删除记录的详细资料，请见第二部分 2.1.2.3”重用删除记录”。

*NO 为参数 REWSEDLT 的缺省值。

1.2.2.7　物理文件性能

参数 ALWUPD 和 ALWDLT。文件性能是用于控制允许做数据库文件权限之外的哪些 I/O 操作。有关数据库文件性能和权限的详细资料，请看第四章”数据库和安全性”。

1.2.2.8　源文件类型

参数 SRCTYPE 指定了源文件成员的源类型(ADDPFM 和 CHGPFM 命令)。源类型用于决定对成员所使用的语法检查程序、提示和格式化。如果用户指定了唯一的源类型(除了 AS/400 支持的类型，象 COBOL 和 RPG)用户必须提供其程序设计来处理此唯一类型。

如果源类型被改变，对其后打开成员将受到影响，对当前打开的成员没有影响。

第三章　　建立逻辑文件

1.3　本章讨论描述、生成一个逻辑文件要考虑的事项。建立逻辑文件的一些规则适用于逻辑文件的所有种类。在这本书中，只适用于某类逻辑文件的规则都做了说明，适用于所有类型逻辑文件的规则就不特别指出了。

1.3.1 描述逻辑文件记录格式

对于用 DDS 描述的每个逻辑文件记录格式，必须指定一个记录格式名和 PFILE 键字(限于简单和多格式逻辑文件)或 JFILE 键字(限于连接逻辑文件)。用 PFILE 或 JFILE 键字指定的文件名是逻辑文件所依据的物理文件名。一个简单或多格式逻辑文件记录格式可以用 DDS 以下列方法来指定：

1.在简单逻辑文件记录格式中仅指定记录格式名和 PFILE 键字。对于用 PFILE 键字指定的唯一的(或第一个)物理文件的记录格式即为逻辑文件的记录格式。在逻辑文件中指定的记录格式名必须和唯一的(或第一个)物理文件中记录格式名相同。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
     A           R ORDDTL                    PFILE(ORDDTLP)
     A
```

图 3-1 简单逻辑文件

2.在下图中，可以通过列出你想包括的字段名来描述自己的记录格式，可以以不同的顺序指定字段名。用 RENAME 键字对字段重命名，用 CONCAT 键字合并字段，用 SST 键字使用字段的某一部分，还可以通过在逻辑文件中指定不同属性来取代字段的原属性。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
     A           R ORDHDR                    PFILE(ORDHDRP)
     A             ORDER
     A             CUST
     A             SHPVIA
     A
```

图 3-2 指定字段的简单逻辑文件

3.在下面的例子中，用 FORMAT 键字指定的文件名是一个数据库文件名，记录格式通过对逻辑文件的描述来达到共享。文件名可通过库名来限制，如果没有指定库名，就在库列表中查找该文件。要生成所描述的文件时，该文件必须是事先存在的。另外，在逻辑文件中指定的记录格式名必须和 FORMAT 键字指定的文件中的一个记录格式名相同。

|. . . +. . . . 1. . . . +. . . . 2. . . . +. . . . 3. . . . +. . . . 4. . . . +. . . . 5. . . . +. . . . 6. . . . +. . . . 7. . . . +. . . . 8

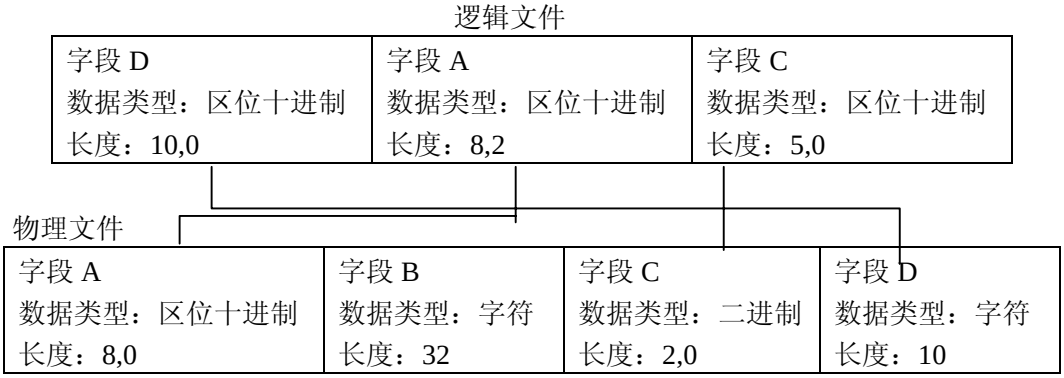
A

```
A            R CUSRCD                    PFILE(CUSMSTP)
A                                        FORMAT(CUSMSTL)
A
```

在下面例子中，程序需要：

- 以不同次序放置字段
- 来自于物理文件字段的一个子集
- 某些字段数据类型做修改
- 某些字段长度做修改

可用逻辑文件做这些修改：



逻辑文件的 DDS 编码为：

|. . . +. . . . 1. . . . +. . . . 2. . . . +. . . . 3. . . . +. . . . 4. . . . +. . . . 5. . . . +. . . . 6. . . . +. . . . 7. . . . +. . . . 8

A

```
A            R LOGREC                    PFILE(PF1)
A              D              10S 0
A              A
A              C               5S 0
A
```

物理文件的 DDS 编码为：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
     A           R PHYREC
     A             A               8S 2
     A             B              32
     A             C               2B 0
     A             D              10
     A
```

从逻辑文件中读记录时，来自于物理文件中的字段为了和逻辑文件中描述的相匹配，要相应的发生变化。如果程序更新或添加了一个记录，则字段不发生改变。要用逻辑文件进行添加或更新操作，程序必须提供和逻辑文件使用的格式相一致的数据。

下表显示了物理文件和逻辑文件中有效的数据类型的转换。

逻辑文件中数据类型

数据类型	字符或十六进制	区位十进制	压缩十进制	二进制	浮点	日期	时间	时间标志
字符或十六进制	有效	见注 1	无效	无效	无效	无效	无效	无效
区位十进制	见注 1	有效	有效	见注 2	有效	无效	无效	无效
压缩十进制	无效	有效	有效	见注 2	有效	无效	无效	无效
二进制	无效	见注 2	见注 2	见注 3	见注 2	无效	无效	无效
浮点	无效	有效	有效	见注 2	有效	无效	无效	无效
日期	无效	有效	无效	无效	无效	有效	无效	无效
时间	无效	有效	无效	无效	无效	无效	有效	无效
时间标记	无效	无效	无效	无效	无效	有效	有效	有效
注：1.只有当字符或字节数和数字位相等时有铲。 2.只有当二进制字段小数位为 0 时有效。 3.只有当两个二进制字段有同样的小数位时有效。								

注释：

有关转换 DBCS 字段的详细资料，请见附录 B”双字节字符集(DBCS)的考虑”。

1.3.1.1 描述逻辑文件的字段

可以指定数据库中的字段是只输入的、输入/输出的或即不输入也不输出的。在第 38 列做如下说明：

项	意义
空格	对简单或多格式逻辑文件，缺省值为 B(输入/输出)
	对于连接逻辑文件，缺省值为 I(只输入)
B	即输入又输出，对连接逻辑文件无效
I	只输入(只读)
N	即不输入也不输出，只对连接逻辑文件有效

注：在引用功能上不用 38 列(用途)。一个文件引用(用 REF 或 REFFLD)该逻辑文件中一个字段，这列的值不复制到那个文件去。

1.3.1.1.1 B (输入/输出)

一个 Both 字段可以进行输入/输出操作，程序可以从字段中读数据，也可以往字段中写数据。由于连接逻辑文件是个只读文件，所以 Both 字段对连接逻辑文件是无效的。

1.3.1.1.2　I (只输入)

只输入字段只能用于读操作。程序可以从字段中读数据，但不能更新文件的该字段。典型的只输入字段有索引字段(为了降低对访问路径维护所用的开销，要避免对索引字段值的修改)，用户只能看而不能更新的敏感字段(例如：薪水)，以及指定了转换表(TRNTBL)键字或子串(SST)键字的字段。

如果程序更新了一个记录而该记录中只包含输入字段，则该字段不发生变化。如果程序添加一个包含只输入字段的记录，则该字段取缺省值(DFT 键字)。

1.3.1.1.3　N（既不输入也不输出）

Neither 字段既不能用于输入也不能用于输出，它只对逻辑文件有效。Neither 字段可用非连接逻辑文件中的连接字段，但你的程序不能读也不能更新 Neither 字段。

当所用物理文件做连接的字段属性不匹配时可使用 Neither 字段。在这种情况下，必须重新定义一个或两个连接字段，但在记录格式中不能包括那些重定义的字段。(在应用程序中看不到重定义字段)。因此，重定义字段将编码为 N，这样它们就不会在记录格式中出现。

第 38 列为 N 的字段不会出现在程序的缓冲区中，但其字段描述能通过显示文件字段描述命令（DSPFFD）被显示。

Neither 字段不能用作选择/省略或索引字段。

有关 Neither 字段的例子，请看 1.3.4.6 “不在记录格式中出现的字段描述（例 5）”。

1.3.1.2　从已有字段中派生新字段

逻辑文件中的字段可从它所依据的物理文件中的字段或同一逻辑文件中的字段派生出来。例如，可以使用 CONCAT 键字合并物理文件中的一个或多个字段，使它们作为一个字段在逻辑文件中出现，反之，也可以用 SST 键字把物理文件中的一个字段分解成多个字段出现在逻辑文件中。

1.3.1.2.1 合并字段

使用 CONCAT 键字，可以将物理文件记录格式中的两个或多个字段合并成逻辑文件记录格式中的一个字段。例如，物理文件记录格式中包含字段 Month（月）、Day（日）、Year（年），对于逻辑文件，可把这些字段合并为一个字段，Date（日期）。

合并后字段的长度是它所包括各个字段的长度和，(除非物理文件中的字段是二进制、压缩十进制，这种情况下，它们将被转换为区位十进制)，结果字段的长度将由系统自动计算，合并字段具有：

- 列标题
- 有效性检查
- 文本描述
- 编辑码和编辑字（仅限于数字连接字段）

注：这些编辑和有效性检查并不是数据库管理系统使用的，而是当显示文件或打印文件中引用数据库文件的字段描述时被检索。

当字段被合并时，数据类型将发生改变，(合并后的数据类型由系统自动决定)，以下是一些规则和限定：

- OS/400 程序可在合并字段的数据类型的基础上指定数据类型。
- 合并字段的最大长度取决于合并字段的数据类型和组合的各个字段的长度。如果合并字段是区位十进制（S），它的总长就不会超过 31 个字节；如果是字符型（A），总长将不超过 32766 个字节。
- 在连接逻辑文件中，用于合并的字段必须来自于同一物理文件。在 CONCAT 键字给

出的第一个字段指出所用的物理文件。因此，构成逻辑文件的各个物理文件中这一个字段必须是唯一的，否则将要使用 JREF 键字来指定是使用的哪个物理文件中的字段。

- 如果该合并字段是变长字段，其用法必须为 I（只输入），否则可为 B（既输入又输出）。
- 数据类型为 O 或 J 的合并字段不能指定 REFSHIFT 键字。
- 如果一些字段为空值，则它们的合并字段也为空值。

注：

有关合并 DBCS 字段的详细资料，请参见附录 B，“双字节字符集的考虑”。

当合并的只是数字字段时，该组中最后一个字段的符号即为该合并字段的符号。

注：

1.所有小数位非零的数字型字段都不能组合成合并字段。

2.日期型、时间型、时间标记和浮点型字段也不能组成合并字段。

下面是使用 DDS 进行合并的字段描述（键字 CONCAT 用于指定合并的字段。）

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
00101A             MONTH
00102A             DAY
00103A             YEAR
00104A             DATE                      CONCAT(MONTH DAY YEAR)
     A
```

例中，逻辑文件的记录格式包括月、日、年三个字段和一个合并字段日期，这样可用以下格式：

- 月、日、年格式
- 仅用日期格式
- 月、日、年格式和日期格式

当记录格式中既有分开字段又有合并字段时，任何对字段的更新都将按 DDS 中指定的顺序处理，在上述例中，如果日期字段为 103188，而月字段改为 12，当记录被更新时，日期字段中的月不变，更新后的记录仍为 103188。如果该日期字段是第一个指定的，则更新后的记录为 123188。

合并字段也能用作索引字段和选择/省略字段。

1.3.1.2.2 子串字段

可以用 SST 键字指定从哪个字段中（字符、十六进制或区位十进制）取子串，也可通过指定 S（区位+进制），将物理文件中压缩字段的子串作为逻辑文件数据类型，例如，物理文件 PF1 中日期字段定义为 6 个字符长，可以在逻辑文件中描述 3 个字段，每个字段 2 个字符长。用 SST 键字规定字段 DATE，MM 字段是从 DATE 第 1 位开始，长度为 2，DD 从 DATE 有第 3 位开始，长度为 2，YY 从第 5 位开始，长度为 2。

下面是用 DDS 描述子串字段，键字 SST 用于指定取子串的字段。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R REC1                        PFILE(PF1)
     A
     A            MM                  I       SST(DATE 1 2)
     A            DD                  I       SST(DATE 3 2)
     A            YY                  I       SST(DATE 5 2)
     A
```

注：子串的起始位是根据它在字段中的位置，而不是根据在文件中的位置，用法为 I 表示只输入。

子串字段可用作索引字段和选择/省略字段。

1.3.1.2.3 重命名字段

用 RENAME 键字，可以在逻辑文件中取和物理文件字段不同的名字。由于程序使用不同的字段名，或者原始字段名和你所使用的高级语言命名限定不一致，在逻辑文件中需要对字段重命名。

1.3.1.2.4 转换字段

用 TRNTBL 键字可以指定一个字段的转换表。当读逻辑文件记录，其中一个或多个字段规定了转换表，系统将把物理文件中字段值转换成转换表所确定的字段值。

1.3.1.3 描述逻辑文件的浮点字段

在逻辑文件中可以用浮点字段人为变换字段，一个单精度或双精度的浮点字段可以变换为区位、压缩、零精度二进制或另一浮点字段，但不能把浮点字段和非零精度二进制字段、字符十六进制和 DBCS 字段之间变换。

不同精度的浮点字段（单精度和双精度）之间，浮点字段和其它数字字段间的变换可能导致舍入或精度的丢失。双精度浮点数到单精度浮点数的变换可导致的舍入情况，取决于它的内部表达和所涉及的特殊数值，舍入的是最近的那位（偶数位），结果总是尽可能精确。如果降低了精度位数，就可能在两个十进制数之间发生精度的丢失。

若程序没有明显的修改动作，可以忽略该字段值的变化。对于浮点字段，物理文件中双精度字段变换成逻辑文件中单精度字段，通过逻辑文件进行记录更新，就是这种情况，当它变换到逻辑文件时，若浮点数的内部表达使其被舍入，那么逻辑文件记录的更新将使其涉及的物理文件造成永久性精度丢失。若舍入的是物理文件记录的索引字段，则物理文件中记录的顺序也将发生变化。

若逻辑文件中字段精度降低，则忽略对应的定点数字字段的更新。

1.3.2 描述逻辑文件的访问路径

指定逻辑文件记录格式的访问路径有以下三种方法：

1.索引顺序访问路径

在最后一个记录或字段级说明后指定索引字段，索引字段名必须是在记录格式中的，对于连接逻辑文件，索引段必须来自于另一个或主物理文件。

|. . . +. . . . 1. . . . +. . . . 2. . . . +. . . . 3. . . . +. . . . 4. . . . +. . . . 5. . . . +. . . . 6. . . . +. . . . 7. . . . +. . . . 8

```
A           R CUSRCD                    PFILE(CUSMSTP)
A           K ARBAL
A           K CRDLMT
A
```

2.到达顺序访问路径

它不需指定索引字段，可在 PFILE 键字指定唯一的物理文件（当增加逻辑文成员时，也只指物理文件成员中的一个）。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R CUSRCD                      PFILE(CUSMSTP)
```

3.预先定义的索引顺序访问路径（只限于简单和多格式逻辑文件）

在文件级指定 REFACCPTH 键字，相当于将一个预先生成的数据库文件的访问路径和选择/省略规范复制到逻辑文件中，不能用 REFACCPTH 键字指定分开的索引或选择/省略字段。

注：尽管使用了指定的文件访问路径，最终还是由系统决定使用哪个文件的访问路径，不论是否使用 REFACCPTH 键字，系统总是试图共享访问路径。

|. . . +. . . . 1. . . . +. . . . 2. . . . +. . . . 3. . . . +. . . . 4. . . . +. . . . 5. . . . +. . . . 6. . . . +. . . . 7. . . . +. . . . 8

REFACCPTH(DSTPRODLIB/ORDHDRL)

```
A            R CUSRCD                    PFILE(CUSMSTP)
```

当定义一逻辑文件中的记录格式时，共享另一文件访问路径的索引字段说明，（使用键字 REFACCPTH），可以使用相关物理文件记录格式的任何一些字段，这些字段不必须出现在描述访问路径的文件中，但文件中所用的任何索引字段和选择/省略字段必须在新的记录格式中使用。

1.3.2.1 使用逻辑文件中的选择和省略记录

使用逻辑文件时系统可以选择和省略记录，这有助于我们处理方便或出于安全考虑而排除一些记录。选择和省略记录的处理是在逻辑文件的 DDS 格式的第 17 列指定，它和高级语言中一系列的编码比较相类似。例如，在逻辑文件中包含了订货的详细记录，可以指定订货量大于销售量的记录，所有其它的记录都从访问路径中省略。省略的记录保留在物理文件中而不被逻辑文件检索。如果要在物理文件上添加记录，是所有的记录被增加，但如果用选择/省略路径，则只有适合选择/省略标准的那些记录才能被添加。

在 DDS 中，第 17 列指定点 S（选择）或 O（省略），然后给出将在选择和省略处理中使用的字段名（第 19 列到 28 列），第 45 列到 80 列指定比较条件。

注：选择/省略说明在索引说明之后（如果指定索引），记录可以通过几种类型的比较完成选择和省略操作；

• VALUES

将字段的内容和表中不超过 100 个值相比较，若发现与它相匹配的，则选择或省略记录。下面例子中，如果 VALUES 指定的值中有一个和 Itmnbr 字段相同，则记录被选择。

|. . . +. . . . 1. . . . +. . . . 2. . . . +. . . . 3. . . . +. . . . 4. . . . +. . . . 5. . . . +. . . . 6. . . . +. . . . 7. . . . +. . . . 8

```
A           S ITMNBR                    VALUES(301542 306902 382101 422109 +
A                                       431652 486592 502356 556608 590307)
A
```

• RANGE

将字段的内容和 RANGE 中上限和下限作比较，如果字段值大于或等于下限并且小于或等于上限，则记录被选择或省略，下例中，Itmnbr 字段值在 301000 到 599999 范围内的所有记录将被选择。

|. . . +. . . . 1. . . . +. . . . 2. . . . +. . . . 3. . . . +. . . . 4. . . . +. . . . 5. . . . +. . . . 6. . . . +. . . . 7. . . . +. . . . 8

```
A             S ITMNBR                    RANGE(301000 599999)
```

A

• CMP

将字段内容和一个值或另一字段内容相比较，有效的比较码是 EQ、NE、LT、NL、GT、NG、LE 和 GE。如果满足比较条件，记录将被选择或忽略，下例中，如果字段 Itmnbr 值小于或等于 599999，记录就被选择。

|. . . +. . . . 1. . . . +. . . . 2. . . . +. . . . 3. . . . +. . . . 4. . . . +. . . . 5. . . . +. . . . 6. . . . +. . . . 7. . . . +. . . . 8

A S ITMNBR CMP(LE 599999)

A

在 CMP、VALUES 和 RANGE 中指定的，数字字段值要以小数位为基准，必要时可以添零，如果字段没有指定小数位，则小数点被放在该值最右边一位的右侧 ，例如，一数字字段长度为 5，小数值为 2，值 1.2 被解释为 001.20，值 100 被解释成为 100.00。

记录的状态是由指定的选择和省略语句的求值顺序决定的。如果一条记录满足了选择或省略条件，则随后的语句就被省略。

通常选择和省略比较之间是相互独立的，该比较是用“或”连接的，也就是，如果满足选择或者省略的条件，记录就可被选择或省略。如果不满足条件，系统将处理下一个比较。为了将各个比较连接起来，可以在 DDS 格式的第 17 列留出一个空格，用这种方式连接起来的比较，在记录被选择或省略前，必须全部满足条件，也就是各比较是用“与”连接的。

比较越少，工作效率越高，因此，当有几个选择/省略比较要处理时，应首先指定选择或忽略记录最多的比较。

在下面例子中，Rep 字段为 JSMITH 的记录很少。这些例子给出用 DDS 去选择在 1988 年以前在纽约的、名为 JSMITH 的销售商，使用不同的方法，有不同的效率（例中 3 是最好的）。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          S ST                        CMP(EQ 'NY')         1
     A            REP                       CMP(EQ 'JSMITH')
     A            YEAR                      CMP(LT 88)
     A

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          O YEAR                      CMP(GE 88)           2
     A          S ST                        CMP(EQ 'NY')
     A            REP                       CMP(EQ 'JSMITH')
     A

|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8

     A          O REP                       CMP(NE 'JSMITH')     3
     A          O ST                        CMP(NE 'NY')
     A          S YEAR                      CMP(LT 88)
     A
```

图 3-3　选择/省略功能的三种编码方法

1.记录在被选择和省略前必须与选择字段 St、Rep 和 Year 相比较。

2.所有的记录必须先和字段 Year 相比较，然后再把 1988 年以前的记录和 St、Rep 相比较。

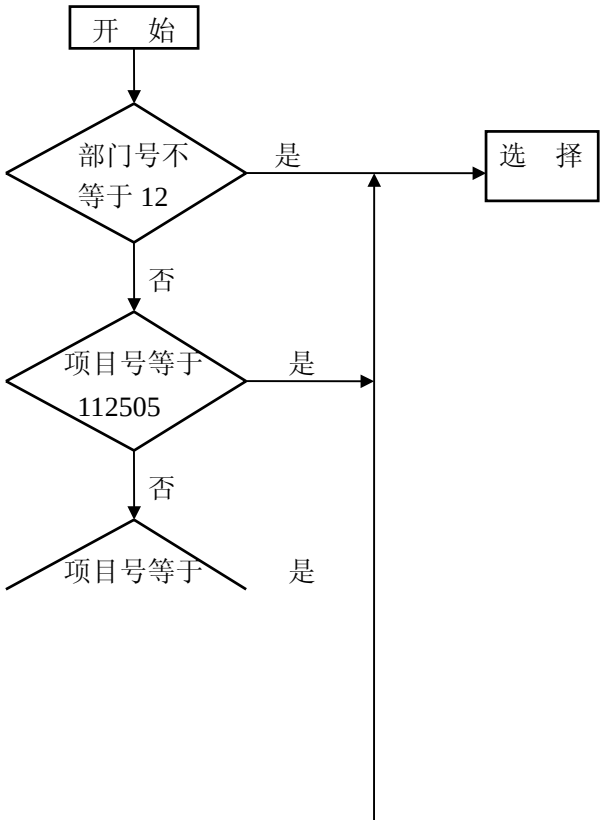
3.所有的记录先和字段 Rep 相比较，几乎很少符合 JSMITH 记录和 St 比较，最后将剩下的和 Year 比较。

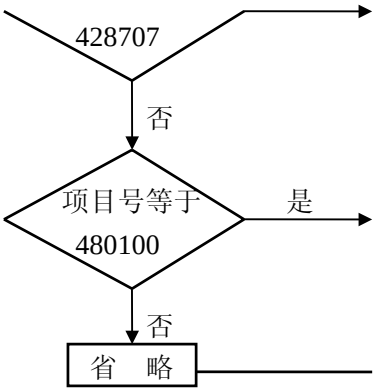
再举个例子，假设要做如下选择：

- 除了部门 12 外所有部门。
- 部门 12 中的项目号为 112505、428707、480100 的记录，部门 12 中其它记录不被选择。

如果用一个分类顺序表做上述例子，在比较前要根据分类表转换选择/省略字段。例如，对于大写和小写用共享分类表，NY 和 ny 同等的，详细的请看《DDS 参考手册》。

下面的流程图显示了例中的逻辑关系：





下面显示了例子中有关选择和省略功能的 DDS 编码：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          S DPTNBR                    CMP(NE 12)
     A          S ITMNBR                    VALUES(112505 428707 480100)
     A
```

可能有选择/省略值的访问路径并以到达顺序处理文件，例如，使用高级语言程序可以指定省略键字访问路径。在这种情况下，以到达顺序从文件中读取记录。但只有满足文件指定的选择/省略值的记录被返回到高级语言程序中。

带有索引字段和指定的选择/省略值的逻辑文件可以用到达顺序或随机的相关记录号来处理，由选择/省略值，忽略的记录不被处理，也就是，如果用相关记录号请求省略的记录，该记录不能返回到高级语言程序中。

系统不能保证，通过逻辑文件添加或改变的记录再次在同一逻辑文件中存取，例如，如果逻辑文件选择字段 FLD1 为 A 的记录，而程序将字段 FLD1 更新为 B，则程序将不能用逻辑文件检索到这条记录。

注：不能根据浮点字段值为做选择或省略条件。

有两种选择/省略操作：访问路径的选择/省略和动态的选择/省略，缺省值为访问路径的选择/省略，两种操作的选择/省略说明本身是一样的，不同的是系统将在不同的时间内进行选择和省略操作。

1.3.2.1.1 访问路径的选择/省略

用访问路径的选择/省略，访问路径只能包含符合逻辑文件指定的选择/省略值的键字。当文件指定了索引字段，文件也就有了访问路径，并且当你要增加或更新形成逻辑文件的物理文件的记录时，该访问路径将由系统维护，访问路径中唯一的索引入口是符合选择/省略值的。

1.3.2.1.2 动态的选择/省略

用动态的选择/省略，当程序从文件中读记录时，系统只能返回符合选择/省略值的记录，也就是，实际上的选择/省略操作是在程序读记录的时候，而不是记录增加或改变的时候，而索引顺序访问路径则包含所有键字 ，不只是选择记录的键字，使用动态的选择/省略访问路径允许更多的访问路径共享，它可以提高系统的性能，有关共享访问路径的详细资料，请见 1.3.2.2 中“使用现存的访问路径”。

为了使用动态的选择/省略，可以规定动态的选择键字（DYNSLT），动态的选择/省略时，不需要索引字段。

如果你的文件要经常修改而不经常读，那么在程序读文件以前可以不必更新选择/省略的访问路径，在这种情况下，可用动态的选择/省略。下面例子有助于了解这些内容。

在选择/省略记录中，用一个不经常修改的编码字段（A=活动的，I=不活动的），程序处理活动的记录，并且记录大部分是活动的（超过 80%），用 DYNSLT 在处理期间做动态选择，比在编码字段发生改变时进行访问路径的维护更有效。

1.3.2.1.3 使用 OPNQRYF 命令选择/省略记录

选择记录的另一方法是使用 OPNQRYF 的 QRYSLT 参数，用 OPNQRYF 命令生成的打开数据路径，就象一个临时逻辑文件，也就是说，当它被关闭时，也就自动被删除，而一个逻辑文件将一直保存到你删除它为止。有关 OPNRYF 命令的详细资料，请看第二部 2.2.3“使用打开的查询文件命令（OPNQRYF）”。

1.3.2.2 使用现存的访问路径

当两个或多文件是依据于同一物理文件并且有同样的顺序和相同的索引字段时，它们将自动地共享同一键字顺序访问路径。当访问路径被共享时，用于维护访问路径的系统活动量和文件所用的辅存量都减少。

当生成一个带有键字顺序访问路径的逻辑文件，系统总是试图共享现存的访问路径，若要共享访问路径，该访问路径必须已存在于系统中并满足下列条件：

- 要增加的逻辑文件成员必须依据具有现存访问路径的同一物理文件。
- 在新文件和原有文件中的每个索引字段的长度、数据类型和小数位数都要一致。
- 如果没有指定键字 FIFO、LIFO 和 FCFO，新文件可能比现存路径的索引字段少，也就是说，如果索引字段的起始部分是一致的，一个新的逻辑文件就可共享已有的访问路径，但是，当文件共享已有路径的部分索引字段时，一些记录对共享访问路径索引字段某些部分修改将会改变访问路径中记录的位置，请看 1.3.2.2.1 中“默认的共享访问路径例子”。
- 访问路径的属性（象 UNIQUE、LIFO、FIFO 或 FCFO）和索引字段的属性（象 DESCEND、ABSVAL、UNSIGNED 和 SIGNED）必须是一致的。

特殊情况：

1.如果满足了共享访问路径的其它请求，则 FIFO 访问路径可以共享指定了 UNIQUE 键字的访问路径。

2.如果满足了共享访问路径的其它请求，一个 UNIQUE 访问路径可以共享一个需要重建的 FIFO 访问路径（例如，指定了*REBLD 维护）。

- 如果一个新的逻辑文件含有选择/省略说明，则它们必须和已有的访问路径的选择/省略说明相一致。但是，如果这个新逻辑文件指定了 DYNSLT，它就可以共享现有的访问路径，该访问路径具有：

  --指定的动态选择键字（DYNSLT）

  --没有选择/省略键字
- 新逻辑文件成员的替换排列顺序（键字 ALTSEQ）和转换表（键字 TRNTBL）必须和已有的访问路径的替换排列顺序和转换表相一致。

注：包含有合并或子串字段的逻辑文件不能和物理文件共享访问路径。

一个访问路径的主人是最初建立访问路径的逻辑文件成员。对于共享的访问路径，如果拥有访问路径的逻辑文件成员被删除，则共享访问路径的第一个成员将成为新的主人。对于共享的访问路径，CRTLF 命令中的参数 FRCACCPTH、MAINT、RECOVER 和 UNIT 不必和共享的访问路径中的这些参数相匹配。当几个逻辑文件成员共享一个访问路径时，并且

FRCACCPTH、MAINT、RECOVER 和 UNIT 参数是不一致的，系统将通过共享成员指定的每个参数的最大限定值维护访问路径，如下说明：

MBRA 所指定的：	MBRB 所指定的：	系统所做的：
FRCACCPTH(*NO) MAINT(*IMMED) RECOVER(*AFTIPL)	FRCACCPTH(*YES) MAINT(*DLY) RECOVER(*NO)	FRCACCPTH(*YES) MAINT(*IMMED) RECOVER(*AFTIPL)

访问路径的 UNIT 参数一般来自于主逻辑文件成员。

访问路径共享并不依赖于成员间的共享，因此，它不限制成员删除的顺序。

显示文件描述（DSPFD）和显示数据库关系（DSPDBR）的命令显示了访问路径的共享关系。

1.3.2.2.1 默认的共享访问路径举例

这个例子的目的是帮助你全面的理解默认的访问路径的共享。

两个逻辑文件 LFILE1 和 LFILE2，是建立在物理文件 PFILE 基础上的。LFILE1 先生成，有两个索引字段 KFD1 和 KFD2。LFILE2 有三个索引字段 KFD1、KFD2 和 KFD3。这两个逻辑文件使用了其中两个相同的索引字段，但并不共享访问路径，因为有三个索引字段的逻辑文件在两个索引字段的文件之后生成。

图 3 – 4 保存和重存前的物理文件和逻辑文件

	物理文件（PFILE）	逻辑文件 1（LFILE1）	逻辑文件 2（LFILE2）
访问路径		KFD1、KFD2	KFD1、KFD2、KFD3
字　　段	KFD1、KFD2、KFD3、A、B、C、D、E、F、G	KFD1、KFD2、KFD3、F、C、A	KFD1、KFD2、KFD3、D、G、F、E

有一个应用的例子，用文件 LFILE1 来访问记录，并改变字段 KFD3，如果它的值为 C 则变为空格，如果值为空格则变为 C。由于访问路径没有共享，所以能给出用户预期的结果。但是，在物理文件和两个逻辑文件保存和重存后，程序不做什么事情，但要用很长时间去处理。

除非做一些事情来修改重存操作，AS/400 系统将：

- 首先重存具有最大键字数的逻辑文件。
- 不建立不必要的访问路径。

有关改变条件的资料请看第三部分 3.4.3.2.2 “当访问路径被重建时的控制”。

因为 LFILE2 有三个索引字段，所以它首先被重存，之后重存 LFILE1，它默认共享 LFILE2 的访问路径，不理解默认共享访问路径的用户不会认识到，当他们在重存后使用 LFILE1 时，实际上使用的是 LFILE2 的索引字段。

图 3 – 5 保存和重存后的物理文件和逻辑文件，注意与保存和重存前的唯一不同是逻辑文件共享了同一个访问路径

	物理文件（PFILE）	逻辑文件 1（LFILE1）	逻辑文件 2（LFILE2）
访问路径		KFD1、KFD2、KFD3	KFD1、KFD2、KFD3
字　　段	KFD1、KFD2、KFD3、A、	KFD1、KFD2、KFD3、F、	KFD1、KFD2、KFD3、D、G、

	B、C、D、E、F、G	C、A	F、E

检测到和修改了的记录是：

相关记录	KFD1	KFD2	KFD3
001	01	01	<空格>
002	01	01	<空格>
003	01	01	<空格>
004	01	01	<空格>

由第一个索引值读到的记录 0101<空格>，变成 0101C，则记录如下：

相关记录	KFD1	KFD2	KFD3
001	01	01	C
002	01	01	<空格>
003	01	01	<空格>
004	01	01	<空格>

当应用要取下一个索引值时，在 0101<空格>上的值是 0101C，这是刚被改过的那条记录，这次操作将把字段 KFD3 从 C 改为空格。

由于用户不理解默认访问路径的共享，应用对每个记录存取和修改要做两次，结果是应用程序运行了较长的时间，而看起来记录并未改变。

1.3.3 生成逻辑文件

在逻辑文件生成以前，逻辑文件所依据的物理文件必须已经存在。

生成逻辑文件的步骤如下：

1.用 SEU 或其它方式，将 DDS 源码录入源文件中，以建立逻辑文件。有关在源文件中如何进入源语句，请见 3.5.3“处理源文件”，下面是逻辑文件 OR DHDRL 的 DDS 描述（一个定货标题文件）。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER LOGICAL FILE (ORDHDRL)
     A           R ORDHDR                    PFILE(ORDHDRP)
     A           K ORDER
```

该文件使用字段 Order（序号）来定义访问路径，记录格式与物理文件 ORDHDRP 记录格式相同，因为没有给出字段描述，所以逻辑文件记录格式名必须和物理文件记录格式名相同。

2.生成逻辑文件，可以用 CRTLF 命令。

下面是如何使用 CRTLF 命令：

```
CRTLF    FILE(DSTPRODLB/ORDHDRL)
           TEXT('Order header logical rile')
```

上述命令使用了缺省值。例如：由于没有指定 SRCFILE 和 SRCMBR 参数，系统 将使用 IBM 支持的源文件 QDDSSRC，源成员为 ORDHDRL（和 CRTLF 定的文件名相同），具有一个同名成员的文件 ORDHDRL 被放入库 DSTPRODLB 中。

1.3.3.1 生成多格式逻辑文件

一个多格式逻辑文件是用一个逻辑文件使用来自两 个或多个物理文件的相关记录，每个记录格式和一个或多个物理文件相联系。也可以在多个记录格式中使用同一物理文件。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER DETAIL FILE (ORDDTLP) - PHYSICAL FILE RECORD DEFINITION
     A                                     REF(DSTREF)
     A           R ORDDTL                  TEXT('Order detail record')
     A             CUST      R
     A             ORDER     R
     A             LINE      R
     A             ITEM      R
     A             QTYORD    R
     A             DESCRP    R
     A             PRICE     R
     A             EXTENS    R
     A             WHSLOC    R
     A             ORDATE    R
     A             CUTYPE    R
     A             STATE     R
     A             ACTMTH    R
     A             ACTYR     R
     A
```

图 3-6　建立在字段引用文件上的物理文件（ORDDTLP）的 DDS 描述

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER HEADER FILE (ORDHDRP) - PHYSICAL FILE RECORD DEFINITION
     A                                     REF(DSTREFP)
     A           R ORDHDR                  TEXT('Order header record')
     A             CUST      R
     A             ORDER     R
     A             ORDATE    R
     A             CUSORD    R
     A             SHPVIA    R
```

```
A            ORDSTS    R
A            OPRNME    R
A            ORDMNT    R
A            CUTYPE    R
A            INVNBR    R
A            PRTDAT    R
A            SEQNBR    R
A            OPNSTS    R
A            LINES     R
A            ACTMTH    R
A            ACTYR     R
A            STATE     R
A
```

图 3-7　建立在字段引用文件上的物理文件（ORDHDRP）的 DDS 描述

下面例子，显示了如何建立有两个记录格式的逻辑文件 ORDFILL，一个记录格式定义为来自物理文件 ORDHDRP 的定货标题记录，另一个为来自物理文件 ORDDTLP 的订货明细记录。（图 3-6 是物理文件 ORDDTLP 的 DDS，图 3-7 为物理文件 RDHDRP 的 DDS，图 3-8 为逻辑文件 ORDFILL 的 DDS）。

逻辑文件的记录格式 CRDHDR 使用一个索引字段 Order 并以 Order 来排序。逻辑文件记录格式 SRDDTL 使用两个索引字段 Order 和 Line，并用其排序。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* ORDER TRANSACTION LOGICAL FILE (ORDFILL)
     A          R ORDHDR                    PFILE(ORDHDRP)
     A          K ORDER
     A
     A          R ORDDTL                    PFILE(ORDDTLP)
     A          K ORDER
     A          K LINE
     A
```

图 3-8　逻辑文件 ORDFILL 的 DDS

要生成具有两个相连物理文件的逻辑文件 ORDFILL，可使用 CRTLF 命令，如下：

```
CRTLF    FILE(DSTPRODLB/ORDFILL)
              TEXT('Order transaction logical file')
```

DDS 源语句在文件 QDDSSRC 的成员 ORDFILL 中，具有同名成员的文件 ORDFILL 被放入库 DSTPRODLB 中，逻辑文件成员 ORDFILL 的访问路径，将对来自 ORDHDRP 和

ORDDTLP 两个文件中的记录进行排序，两个物理文件的记录格式是以公共字段 Order 为索引的。由于在逻辑文件的描述中指定了 Order，则它们将以 Order 来排序。先检索文件 ORDHDRP，再检索 ORDDTLP，将两个文件中重复的部分合并起来，因为没有指定 FIFO、LIFO 和 FCFO，在同一文件中的重复键值的检索顺序就不能得到保证。

注：在一定环境中，使用多个逻辑文件要好于使用多格式逻辑文件，例如：多格式逻辑文件使用索引字段访问时，若其中一个文件的记录非常少，就可能使性能非常不好，尽管有多个记录格式，但逻辑文件只有一个索引，从这里处理每一个物理文件，用这种处理方式的应用程序（例如：用 RPG 中 SETLL 和 READE 来处理小文件），系统必须搜索所有的索引入口，找各个小文件的项。如果该索引有许多项，检索索引将用很长时间，这个时间取决于每个文件的索引字段数和它们的顺序。（如果一个小文件中没有记录，对性能没有影响。这是因为系统可以采用一个快速路径，避免检索索引）。

1.3.3.1.1 控制如何在多格式文件中检索记录

在多格式的逻辑文件中，需要定义索引字段。每个记录格式都有自己的键字定义，并且定义记录格式的索引字段只是为了合并不同格式中的记录，每个记录格式不非要包含键了中的每个索引字段，考虑下面记录：

Header 记录格式：

Record	Order	Cust	Ordate
1	41882	41394	050688
2	32133	28674	060288

Detail 记录格式：

Record	Order	Line	Item	Qtyord	Extens
A	32133	01	46412	25	125000
B	32133	03	12481	4	001000
C	41882	02	46412	10	050000
D	32133	02	14201	110	454500
E	41882	01	08265	40	008000

在 DDS 描述中，Header 记录格式定义在 Detail 之前，如果访问路径使用了 Order 字段作为两个记录格式中第一个索引字段，LINE 字段只作为第二个记录格式的第二个索引字段，并且两个都是升序，则访问路径中的记录是：

记录 2
记录 A
记录 D
记录 B
记录 1
记录 E
记录 C

注：带有重复键值的记录应按物理文件指定的顺序排列。然后，如果一个记录格式中仍有重复键值记录，则按 FIFO、LIFO 或 FCFO 指定的顺序进行排列。例如，如果逻辑文件指

定了 DDS 键字 FIFO，那么记录格式中的重复记录将按先进先出的顺序出现。

对于多格式逻辑文件，可以对索引字段使用*NONE 功能，这样，就能把从同一访问路径中取出的一个记录格式的记录与另一个记录格式中的记录分开。通常，从所有记录格式中产生的记录都是按索引值来合并的，但如果对一索引字段指定了*NONE，那么只有在*NONE 之前，出现在所有记录格式中的有索引字段的记录才被合并。

下例中的逻辑文件有三个记录格式，每个都和不同的物理文件相关联。

记录格式	物理文件	索引字段
EMPMSTR	EMPMSTR	Empnbr(雇员号)1
EMPHIST	EMPHIST	Empnbr,Empdat(受雇日期)2
EMPEDUC	EMPEDUC	Empnbr,Clsnbr(级别号)3

注：所有的记录格式都有一个公共的索引字段，Empnbr，例中的 DDS 为：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
     A          K EMPNBR    1
     A
     A          K EMPNBR    2
     A          K EMPDAT
     A
     A          K EMPNBR    3
     A          K *NONE
     A          K CLSNBR
     A
```

对文件 EMPMSTR 的第二和第三个索引字段以及 EMPHIST 中第三个索引字段指定为*NONE，因为在这些键字段位置上没有索引字段。

下面是记录的排列顺序：

Empnbr	Empdat	Clsnbr	记录格式名
426			EMPMSTR
426	6/15/74		EMPHIST
426		412	EMPEDUC
426		520	EMPEDUC
427			EMPMSTR
427	9/30/75		EMPHIST
427		412	EMPEDUC

*NONE 作为记录格式 EMPHIST 和 EMPEDUC 的分离标识，EMPHIST 中带有相同字

段 Empnbr 的所有记录分成一组，并按 Empdat 字段分类。EMPEDUC 中有相同字段 Empnbr 的所有记录分成一组，并按 Clsnbr 字段分类。

注：为了保证上述的排列次序，将附加索引字段值放入键字顺序访问路径中，这样就不能预知重复键值。

请看 DDS 参考手册中有关*NONE DDS 功能的附加举例。

1.3.3.1.2 控制如何在多格式文件中增加记录

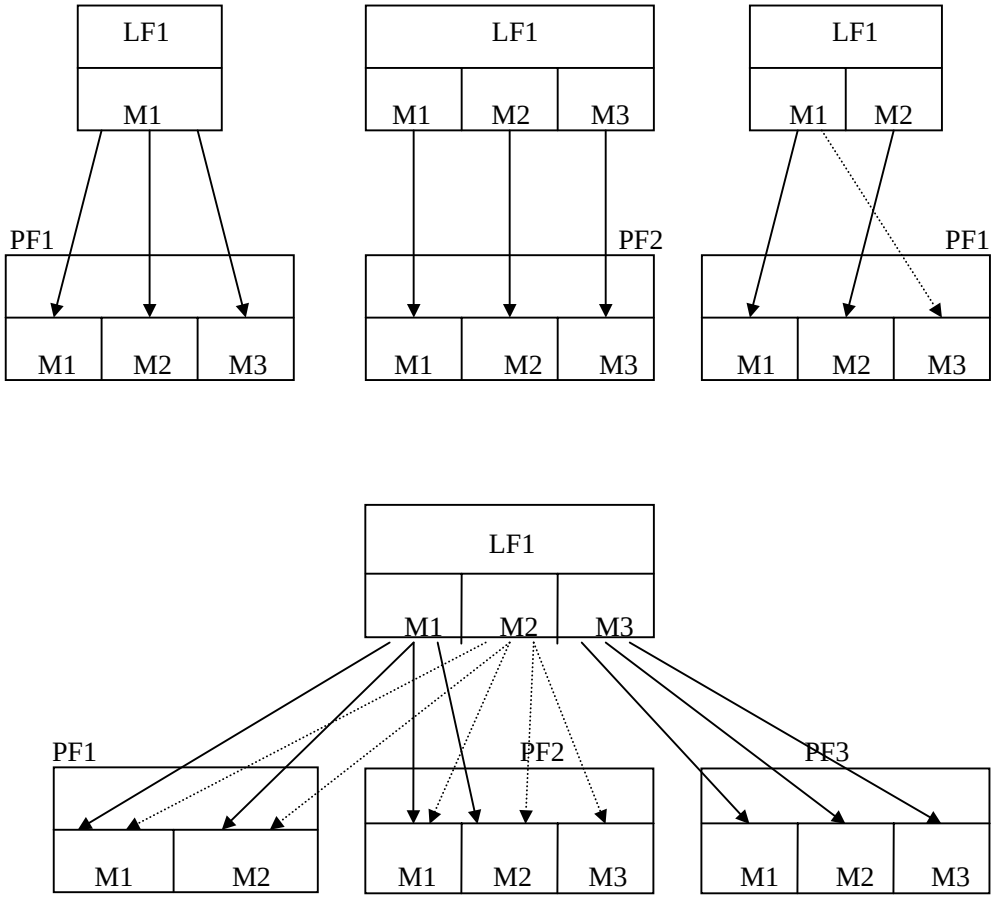
为了在多格式逻辑文件中添加记录，要指出往哪个物理文件成员中写记录。如果应用程序不允许在格式中指定某个成员，那么逻辑文件中的每个格式都必须和一个物理文件成员相联系，如果物理文件包含不只一个成员，就需要指定参数 DTAMBRS（在 1.3.3.2 “逻辑文件成员”中描述）来确定每个格式的单一成员。最后，在多格式逻辑文件中给每个记录格式唯一的名称。如果以这种方式定义的多格式逻辑文件，那么当在增加操作指定了格式名时，就可以确定哪个物理文件成员中增加了记录。

当在多格式逻辑文件中增加记录，并且应用程序使用的是一个文件名而不是记录格式名时，就需编写一个格式选择程序。有关格式选择程序的详细资料，请看第二部分 2.3.4.1 “用多格式文件标识增加记录的记录格式。”

1.3.3.2 逻辑文件成员

可以定义逻辑文件成员，以便将数据分成各个逻辑组，该逻辑文件成员可以和一个或几个物理文件成员相关联。

下面说明了此概念：



当文件生成时，逻辑文件中所有成员的记录格式都由 DDS 定义，如果需要一个新的记录格式，必须建立另一个逻辑文件或记录格式。

逻辑文件访问路径的属性是由 DDS 定义，以及在生成逻辑文件的命令中规定的，用 CRTLF 和 ADDLFM（增加逻辑文件成员）命令中的参数 DTAMBRS 来指定数据成员的选择。

定义逻辑文件时，逻辑文件所用到的物理文件是通过 DDS 中记录级键字 PFILE 和 JFILE 指定的。如果在 DDS 中定义了多个记录格式，必须对每个记录格式都指定 PFILE 键字，每个 PFILE 键字都可以指定一个或多个物理文件。

在建立逻辑文件或在文件中增加成员时，可以用 CRTLF 和 ADDLFM 命令中 DTAMBRS 参数来指定哪个物理文件的成员做为逻辑文件的数据来源。如果物理文件中没有规定成员，指定*NONE 作为物理文件成员名。

下例中，逻辑文件定义了两个记录格式：

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A
00010A          R LOGRCD2                   PFILE(PF1 PF2)
     A            .
     A            .
     A            .
00020A          R LOGRCD3                   PFILE(PF1 PF2 PF3)
     A            .
     A            .
     A            .
     A
```

用 CRTLF 和 ADDLFM 命令中 DTAMBRS 参数格式如下：

DTAMBRS((PF1 M1) (PF2(M1 M2))　(PF1 M1)　(PF2(*NONE)) (PF3 M3))

记录格式 LOGRCD2 和物理文件 PF1 中成员 M1 及 PF2 中成员 M1 和 M2 相关，记录格式 LOGRCD3 与物理文件 PF1 中的 M1 及 PF2 中的 M2 相关。在 PF2 中没有成员和 LOGRCD3 相关。如果多个 PFILE 键字指定了同一物理文件，则每个键字中出现的物理文件名要作为不同的物理文件来处理。

如果 PFILE 键字中的文件没有指定库名，当逻辑文件生成时，将从库列表中去找物理文件。然后物理文件名和库名将成为逻辑文件描述的一部分，由 DTAMBRS 指定的物理文件名和库名，必须和逻辑文件描述的相一致。

如果没有在 DTAMBRS 参数中指定文件所在的库名，则库名的缺省值为*CURRENT，系统使用存在有关逻辑文件描述中的相关物理文件中的库名。该库名或者是由 DDS 中 PFILE 键字指定的库名，或者是当逻辑文件生成时，用库列表找到文件所在的库名。

要在逻辑文件中增加成员时，可如下指定数据成员：

- 不指定相关物理文件成员（DTAMBRS 缺省为(*ALL)），逻辑文件的成员与 DDS 描述中所有 PFILE 中规定的所有物理文件的所有成员相关联。
- 指定相关物理文件成员（DTAMBRS）。如果不指定库名，用逻辑文件确定使用哪个库，当一个物理文件中指定了多个成员，如果引用的成员之间出现了重复键值，那么要按记录被检索的顺序来指定成员名。如果不包括物理文件中的成员，就不指定

物理文件名，或指定物理文件名但成员名为*NONE，这种方法可用来定义一个逻辑文件成员。它包含这个逻辑文件记录格式中的某些子集。

要生成一个逻辑文件，可以用 CRTLF 命令去生成另一个成员，然后用 ADDLFM 命令去增加成员。但是，如果要增加多个成员，必须在 CRTLF 命令的 MAXMBRS 参数中指定大于 1 的值。下例是用 CRTLF 命令，往 1.3.3 节中生成的逻辑文件增加一个成员：

```
CRTLF FILE(DSTPRODLB/ORDHDRL)
        MBR(*FILE)    DTAMBRS(*ALL)
        TEXT('Order header logical file')
```

参数 MBR 的缺省值为*FILE，意思是成员名和文件名一样，物理文件（ORDHDRP）中的所有成员都用做逻辑文件成员（ORDHDRL），其文本描述即成员的文本描述。

1.3.4 连接逻辑文件考虑事项

本节所包含的内容有：

- 连接两个物理文件的基本概念（例 1）
- 建立一个连接逻辑文件
- 用多个字段来连接文件（例 2）
- 用 JDUPSEQ 来处理次文件中的重复记录（例 3）
- 处理属性不匹配的连接字段
- 使用没在连接逻辑文件记录格式中出现的字段 neither 字段（例 5）
- 在连接逻辑文件中指定索引字段（例 6）
- 在连接逻辑文件中指定选择/省略语句
- 连接三个或三个以上物理文件（例 7）
- 连接一个物理文件和它本身（例 8）
- 对次文件中没有记录的使用缺省数据--键字 JDFTVAL（例 9）
- 描述一复杂的连接逻辑文件（例 10）
- 性能考虑
- 连接逻辑文件规则的总结

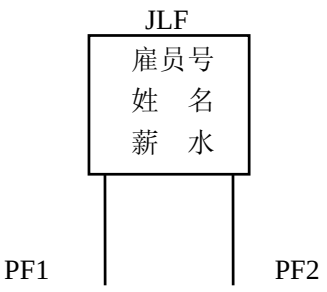
通常，本节中的例子包括文件表，文件的 DDS 语句和样本数据，例 1 中，给出几种情况说明在不同情况下如何连接文件（当物理文件中数据变化时）。

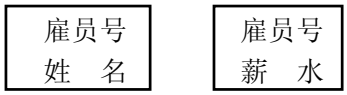
在这些例子中，为了简单和便于识别，连接逻辑文件用标号 JLF 表示，物理文件用标号 PF1、PF2、PF3 等来说明。

1.3.4.1 连接两个物理文件的基本概念（例 1）

一个连接逻辑文件是组合两个或多个物理文件字段的逻辑文件（组合在一个记录格式中），在这个记录格式中，并非所有的字段都需要来自于各个物理文件。

下例是说明连接两个物理文件的一个连接逻辑文件。该例用于例 1 中的五种情况。





例中，雇员号是两个物理文件（PF1 和 PF2）的公共部分，而姓名只存在于 PF1 中，薪水只存在于 PF2 中。

使用连接逻辑文件，应用程序只做一个读操作(对连接逻辑文件的记录格式）就可得到两个物理文件中所有数据，若不使用连接规范，该逻辑文件将包含两个记录格式，一个依据 PF1，另一个依据 PF2，并且应用程序将不得不做两个读操作来得到所需的两个物理文件的所有数据。这样，在设计数据库时，连接提供了较好的灵活性。

但是，连接逻辑文件也有几个限制：

- 不能用 DFU 去显示一个连接逻辑文件
- 在连接逻辑文件中只能指定一个记录格式
- 连接逻辑文件不能共享其它文件的记录格式
- 索引字段必须是在连接记录格式定义的字段，并且必须是 JFILE 键字指定的第一个物理文件（也叫主文件）中的字段
- 选择/省略字段必须是在连接逻辑文件记录格式中定义的字段，但可以来自于任何一个物理文件。
- 落实控制不能用于连接逻辑文件

下面是例 1 中的 DDS 语句：

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R JOINREC                  JFILE(PF1 PF2)
     A          J                          JOIN(PF1 PF2)
     A                                     JFLD(NBR NBR)
     A            NBR                      JREF(PF1)
     A            NAME
     A            SALARY
     A          K NBR
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R REC1
     A            NBR           10
     A            NAME          20
     A          K NBR
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R REC2
```

```
A            NBR            10
A            SALARY          7  2
A          K NBR
A
```

图 3-9　连接两个物理文件的 DDS 举例

下面描述例 1 中连接逻辑文件的 DDS（有关键字的详细内容请看 DDS 参考手册）。

• 记录级规定指出了连接逻辑文件中的记录格式名。

R：说明为记录格式，在连接逻辑文件中只能有一个记录格式。

JFILE：代替在简单和多格式逻辑文件中的 PFILE 键字，必须指定至少两个物理文件，由 JFILE 指定的第一个文件为主文件，另一个文件则为次文件。

连接规则描述了一对物理文件被连接的方法，这一对中的第二个文件永远是次文件，并且对每个次文件都必须有一个连接规则。

J：标明连接规则的开始。在一个连接逻辑文件中必须至少指定一个连接说明，在 19 列到 28 列，指定了第一个字段名或在 17 列指定了下一个 J 都说明此项连接已结束。

JOIN：通过连接说明指出哪两个文件被连接。如果只有两个物理文件被连接成一个连接逻辑文件，则键字 JOIN 是可选的。有关如何使用该键字举例，请看 1.3.4.9“连接三个以上物理文件（例 7）”。

JFLD:指出连接字段。连接由 JOIN 指定的物理文件中的记录。在每个连接说明中 JFLD 至少被指定一次。连接字段是物理文件的公共字段，第一个连接字段来自于 JOIN 指定的第一个文件，第二个连接字段来自于 JOIN 指定的第二个文件。

除了字符型字段，连接字段必须有相同的属性（数据类型、长度和小数位）。如果字段是字符型字段，它们就不必有相同的长度。如果要连接的物理文件中没有相同属性的字段，为了在连接逻辑文件中使用，你可以重新定义它们，其详细的描述和举例，请看 1.3.4.5 中“使用不同属性的连接字段”。

• 字段级规定指出了连接逻辑文件所包括的字段。

字段名：指明应用程序用哪几个字段（例中为 Nbr、Name 和 Salary），至少要规定一个字段名，可以指定逻辑文件所用的物理文件中任何一个字段名，也可以象在简单逻辑文件和多格式逻辑文件中那样使用键字 RENAME、CONCAT 或 SST。

JREF：在记录格式中（它紧接连接说明，如果有索引字段，要放在索引字段前）给出的字段名指出它是来自于哪个物理文件。在上例中，字段 Nbr 出现在两个文件 PF1 和 PF2 中。因此，键字 JREF 就需要指出 Nbr 字段描述来自于哪个文件。

• 索引字段级说明是可选的。它包括连接逻辑文件的索引字段名。

K：标识了索引字段说明。K 在第 17 列出现，索引字段说明是可选的。

索引字段名：索引字段名是可选的（例中 Nbr 是唯一的索引字段），并且使连接逻辑文件成为一个索引文件（键字顺序），在连接逻辑文件中，索引字段必须来自于主文件。其字段名必须是在逻辑文件记录格式中第 19 到 28 列指定的。

• 选择/省略字段级是可选的。它包括逻辑文件中的选择/省略字段名。

S 或 O：标识选择或省略说明。S 或 O 出现在第 17 列选择/省略规范是可选的。

选择/省略字段名：只有那些满足选择/省略值的记录可以返回到使用逻辑文件的程序中，选择/省略字段必须在逻辑文件记录格式的第 19 到 28 位给出。

1.3.4.1.1 读一个连接逻辑文件

下面解释 1.3.4.1 中图 3-9 的连接逻辑文件如何把记录传给应用程序。

由 JFILE 键字指定的第一个文件 PF1，它是主文件，当应用程序请求一个记录时，系统做如下工作：

1.使用主文件中第一个连接字段的值（PF1 中字段 Nbr）。

2.用相匹配的连接字段找出次文件中第一个记录。（PF2 中 Nbr 字段和 PF1 中 Nbr 字段相匹配）

3.对每一个匹配，把物理文件中的字段连接成一条记录，并把此记录提供给程序。根据物理文件中有多少个记录，可能发生下列情况：

a.对于主文件中所有的记录，在次文件中只能找到一个匹配记录，那么连接逻辑文件中只包含主文件中各个记录中的一个记录。详细请看 1.3.4.1.2“主文件和次文件中的匹配记录（情况 1）”。

b.对于主文件中的某些记录，在次文件中找不到匹配记录。

如果指定了 JDFTVAL 键字：

对于主文件中那些可以在次文件中找的匹配记录，系统连接一个次文件或多个次文件，结果是对主文件中的每个记录会有一个或多个记录产生。

对于主文件中那些不能在次文件中找到匹配的记录，系统将在次文件中增加缺省值字段，并且继续做连接操作，可以用物理文件中的 DFT 键字来定义用哪个缺省值，请看 1.3.4.1.3 和 1.3.4.1.4 的内容。

注：如果在次文件中指定了 DFT 键字，那么由 DFT 指定的值可在连接中使用，结果是对每个主文件记录至少有一个连接记录。

如果在次文件中存在记录，但主文件没有与之相匹配的值，则没有记录返回给程序，可以用第二个连接逻辑文件来变换主文件和次文件的顺序。用来决定在这种情况下返回哪些记录。

如果没有指定 JDFTVAV 键字：

如果次文件中存在匹配记录，系统将连接次文件或多个次文件。结果是对主文件的每个记录可能在逻辑文件中产生一个或多个记录。

如果次文件中不存在匹配记录，系统将不返回记录。

注：没有指定 JDFTVAL，对主文件中的一个记录在每个次文件都能找到匹配时，系统仅返回一个记录。

在以下例子中，第 1-第 4 种情况描述顺序读操作，第 5 种情况描述索引读操作。

1.3.4.1.2 主文件和次文件有匹配记录（情况 1）

假定用 1.3.4.1 中图 3-9 规定的连接逻辑文件。物理文件 PF1 和 PF2 有 4 个记录，如下：

PF1

```
235  Anne
440  Doug
500  Mark
729  Sue
```

PF2

```
235  1700.00
440   950.50
500  2100.00
729  1400.90
```

程序要作四个读操作得到下列记录：

JLF

235	Anne	1700.00
440	Doug	950.50
500	Mark	2100.00
729	Sue	1400.90

1.3.4.1.3　次文件没有匹配记录且没指定 JDFTVAL 键字（情况 2A）

用 1.3.4.1 中图 3-9 规定的连接逻辑文件。在文件 PF1 中有四个记录，PF2 中有三个记录。如下：

PF1

235	Anne
440	Doug
500	Mark
729	Sue

PF2

235	1700.00
440	950.50
729	1400.90

在 PF2 中对于 nubmer=500 没有找到记录。

使用例 1 中的连接逻辑文件，程序读到下列记录：

JLF

235	Anne	1700.00
440	Doug	950.50
729	Sue	1400.90

在 PF2 中没找到序号为 500 的记录。

如果没有指定 JDFTVAL 键字，次文件中连接字段没有匹配记录，则在逻辑文件中也不包括此记录。

1.3.4.1.4 次文件中没有匹配记录，但指定了 JDFTVAL 键字（情况 2B）

使用 1.3.4.1 中图 3-9 的逻辑文件，但指定了 JDFTVAL 键字，DDS 编码如下：

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
```

```
A                                       JDFTVAL
A           R JOINREC                   JFILE(PF1 PF2)
A           J                           JOIN(PF1 PF2)
A                                       JFLD(NBR NBR)
A             NBR                       JREF(PF1)
A             NAME
A             SALARY
A           K NBR
A
```

程序读连接逻辑文件并得到如下记录：

JLF

235	Anne	1700.00
440	Doug	950.50
500	Mark	0000.00
729	Sue	1400.90

如果指定了 JDFTVAL 键字，程序也读到序号为 500 的记录（但 SALARY 的值为 0）。

指定了 JDFTVAL，尽管在 PF2 中没有序号为 500 的记录，系统也将返回此记录，没有此记录，连接记录中的某些字段值就读不到，（在这种情况下，就没有字段 Salary）。指定了 JDFTVAL 键字，字符字段通常用空格做缺省值，数字字段是用零代替，但在物理文件中指定了 DFT 键字，则使用由 DFT 指定的缺省值。

1.3.4.1.5　对主文件中的一个记录次文件有多个匹配记录（情况 3）

假定用 1.4.3.1 中图 3-9 中的连接逻辑文件，其 PF1 中有 4 个记录，PF2 中有五个记录，如下：

PF1

235	Anne
440	Doug
500	Mark
729	Sue

PF2

235	1700.00
235	1500.00
440	950.50
500	2100.00
729	1400.90

在 PF2 中有二个序号为 235 的记录。

程序将得到 5 个记录：

JLF

235	Anne	1700.00
235	Anne	1500.00
440	Doug	950.50
500	Mark	2100.00
729	Sue	1400.90

除非指定 JDUPSEQ 键字，读到的序号为 235 的二个记录的顺序是随机的。

详细资料请看 1.3.4.4 “读次文件中的重复记录（例 3）”。

1.3.4.1.6 次文件中的多余记录（情况 4）

假定用 3.4.1 中图 3-9 的逻辑文件，PF1 中有四个记录，PF2 中有五个记录。

PF1

235	Anne
440	Doug
500	Mark
729	Sue

PF2

235	1700.00
301	1500.00
440	950.50
500	2100.00
729	1400.90

记录 301 只在于 PF2 中出现

程序读连接逻辑文件并得到四个记录，这同指定了 JDFTVAL 键字一样。（因为要得到连接记录，此记录必须是在主文件中的）。

JLF

235	Anne	1700.00
440	Doug	950.50
500	Mark	2100.00
729	Sue	1400.90

1.3.4.1.7 随机存取（情况 5）

假定使用 1.3.4.1 中图 3-9 的连接逻辑文件，注意此连接逻辑文件定义了索引字段。这种情况表明用这种逻辑文件对于随机的读操作能返回哪些记录。

假定 PF1 和 PF2 有下列记录：

PF1

235	Anne
440	Doug
500	Mark
729	Sue
997	Tim

PF2

235	1700.00
440	950.50
729	1400.90
984	878.25
997	331.00
997	555.00

PF2 中 没有序号为 500 的记录。

序号为 984 的记录只存在于 PF2 中。

PF2 中序号为 997 的有重复记录。

程序可以得到如下记录：

若 Nbr 字段的值为 235，系统将提供下面的记录：

```
235  Anne  1700.00
```

若 Nbr 字段值为 500 并指定了 JDFTVAL 键字，系统提供下面记录：

```
500  Mark     0.00
```

注：如果在逻辑文件中没有指定 JDFTVAL 键字，由于次文件中没有匹配记录，斜不返回值为 500 的记录。

若 Nbr 字段值为 984，系统将不提供记录，这是因为 984 号记录不在主文件中。

若 Nbr 字段的值为 997，系统将返回下列记录之一：

```
997  Tim    331.00
```

或

```
997  Tim    555.00
```

不能预测哪个记录返回给程序，若要确定哪个记录被返回，应该在逻辑文件中指定 JDUPSEQ 键字，请看 1.3.4.4 中“读次文件中的重复记录（例 3）”。

注：1.使用随机存取，应用程序必须注意 PF2 中可能存在重复记录，并确保程序对带有重复键字的记录要进行多次读操作。如果程序使用顺序存取，则第二次读操作取到的是第二个记录。

2.如果指定了 JDUPSEQ 键字，系统将对逻辑文件生成一个分开的访问路径（这是因为系统没有机会找到一个现存的可以共享的访问路径），如果没有指定 JDUPSEQ 键字，系统将共享另一文件的访问路径。（在这种情况下，系统共享 PF2 的访问路径）。

1.3.4.2 建立连接逻辑文件

要建立连接逻辑文件，步骤如下：

1.找出想在逻辑文件记录格式中用到的所有物理文件字段的字段名。（可以用显示文件字段描述命令（DSPFFD）来显示文件中的字段）。

2.在记录格式中描述字段，以竖向列表的方式写出字段名，这是连接逻辑文件记录格式的起始部分。

注：可以用任何顺序指定字段名，如果在不同的物理文件中出现同一字段名，要用 JREF 键字说明物理文件名，也可以用 RENAME 来将字段重命名，并且用 CONCAT 键字合并来自于同一物理文件的字段，用 SST 来定义字符、十六进制或区位十进制字段的子串，且子串的类型不变，但可以重新定义它们的类型，长度或小数位。

3.在 JFILE 的参数值上指定物理文件名，指定的第一个文件为主文件，其余的均为次文

件。要使性能好一些，应该在主文件之后指定记录最少的一个次文件。

4.对每个次文件，规定连接规范。在每个连接规范中，指出连接哪两个文件（用 JOIN 键字）。如果只有一个次文件时，JOIN 是可选的，并且指出是用哪个字段连接这时文用 JFLD 键字，（每个连接规范中至少有一个 JFLD 键字）。

5.做为可选项，指定下列键字：

a.JDFTVAL　如果在次文件中没有匹配记录，对主文件中的每个记录都返回一个记录，使用此键字。

b.JDUPSEQ　对于那些在次文件中有重复值的字段可以使用此键字。JDUPSEQ 指定了用哪个字段（除了连接字段）对重复进行排序。

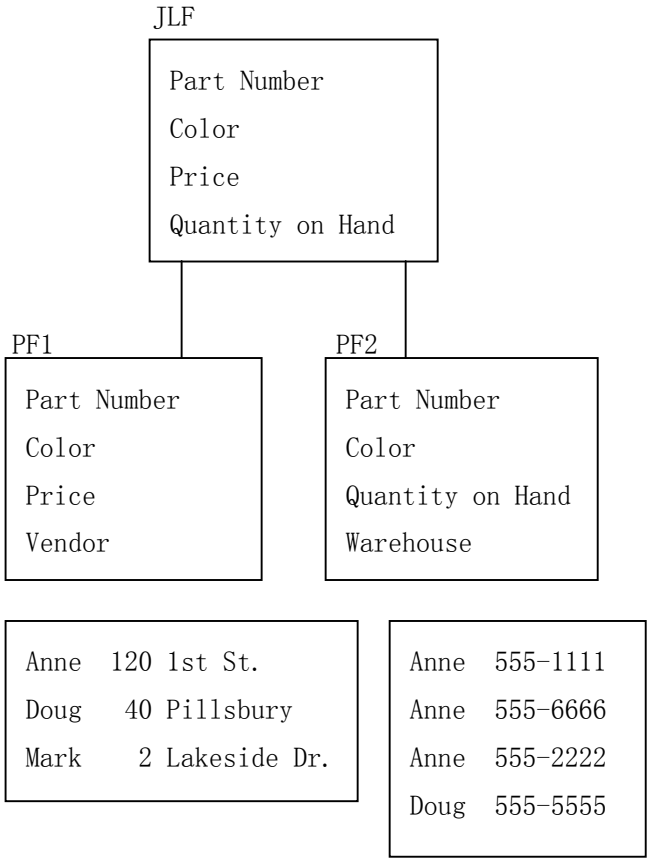
c.索引字段　索引字段不能来自于次文件，如果不用索引字段，记录将以在主文件中出现的顺序被返回。

d.选择/省略字段　在某些情况下，必须在文件级指定动态选择键字（DYNSLT）。

e.Neither 字段　有关描述，请看 1.3.4.6 中“描述在记录格式中不出现的字段（例 5）”。

1.3.4.3 使用多个字段来连接文件（例 2）

可以指定多个连接字段来连接一对文件，下面是逻辑文件和两个物理文件的字段：



文件的 DDS 描述如下：

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
```

```
      A           R JOINREC                   JFILE(PF1 PF2)
      A           J                           JOIN(PF1 PF2)
      A                                       JFLD(PTNBR PTNBR)
      A                                       JFLD(COLOR COLOR)
      A             PTNBR                     JREF(PF1)
      A             COLOR                     JREF(PF1)
      A             PRICE
      A             QUANTOH
      A
```

PF1

```
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
      A           R REC1
      A             PTNBR          4
      A             COLOR         20
      A             PRICE          7  2
      A             VENDOR        40
      A
```

PF2

```
      |...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
      A           R REC2
      A             PTNBR          4
      A             COLOR         20
      A             QUANTOH        5  0
      A             WAREHSE       30
```

假设物理文件中有如下记录：

PF1

```
100 Black    22.50  ABC Corp.
100 White    20.00  Ajax Inc.
120 Yellow    3.75  ABC Corp.
187 Green   110.95  ABC Corp.
187 Red     110.50  ABC Corp.
190 Blue     40.00  Ajax Inc.
```

PF2

```
100 Black    23  ABC Corp.
100 White    15  Ajax Inc.
120 Yellow  120  ABC Corp.
187 Green     0  ABC Corp.
187 Red       2  ABC Corp.
190 White     2  Ajax Inc.
```

如果文件按顺序处理，程序接收到下列记录：

JLF

```
100 Black    22.50   23
100 White    20.00   15
120 Yellow    3.75  102
187 Green   110.95    0
187 Red     110.50    2
```

注：由于次文件中对于零件号为 190、蓝颜色这两个字段没有匹配记录，所以程序中无记录返回。由于没有指定键字 JDFTVAL，则无记录返回。

1.3.4.4 读次文件中的重复记录（例 3）

有时对一个次文件的连接会从次文件中产生多个记录，这时可以在对次文件的连接规范中规定 JDUPSEQ 键字，告诉系统要依据的次文件中有重复的记录某个字段做顺序。

物理文件和逻辑文件的 DDS 如下：

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R JREC                      JFILE(PF1 PF2)
     A           J                           JOIN(PF1 PF2)
     A                                       JFLD(NAME1 NAME2)
     A                                       JDUPSEQ(TELEPHONE)
     A             NAME1
     A             ADDR
     A             TELEPHONE
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R REC1
     A             NAME1          10
     A             ADDR           20
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R REC2
     A             NAME2          10
     A             TELEPHONE       8
     A
```

图 3-10　使用 JDUPSEQ 的 DDS 举例

物理文件所包含的记录有：

PF1

```
Anne  120 1st St.
Doug   40 Pillsbury
Mark    2 Lakeside Dr.
```

PF2

```
Anne  555-1111
Anne  555-6666
Anne  555-2222
Doug  555-5555
```

连接逻辑文件返回的记录有：

JLF

```
Anne  120 1st St.     555-1111
Anne  120 1st St.     555-2222
Anne  120 1st St.     555-6666
Doug   40 Pillsbury   555-5555
```

Anne 的电话号码按升序排列。

程序读所有有效的记录，先是 Anne，然后是 Doug、Mark，Anne 有一个地址，但有三个电话号，因此 Anne 将返回三个记录。

Anne 的记录是通过电话号码按升序进行分类的，这是因为在 JDUPSEQ 键字中指定按升序分类，除非在键字参数中指定为*DESCEND。下列显示了 DDS 中*DESCEND 的用法：

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R JREC                    JFILE(PF1 PF2)
     A          J                         JOIN(PF1 PF2)
     A                                    JFLD(NAME1 NAME2)
     A                                    JDUPSEQ(TELEPHONE *DESCEND)
     A            NAME1
     A            ADDR
     A            TELEPHONE
     A
```

当在 JDUPSEQ 中指定了*DESCEND 时，返回的记录如下：

JLF

```
Anne  120 1st St.     555-6666
Anne  120 1st St.     555-2222
Anne  120 1st St.     555-1111
Doug   40 Pillsbury   555-5555
```

Anne 的电话号码按降序排列。

注：JDUPSEQ 适用于仅规定了一个连接规范。有多个连接规范的连接逻辑文件中的 JDUPSEQ 键字的例子，请看 1.3.4.12 中“一个复杂的连接逻辑文件（例 10）”。

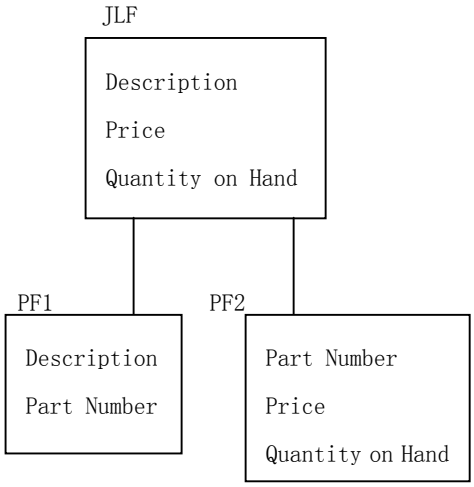
1.3.4.5 使用属性不同的连接字段（例 4）

用作连接的物理文件中的字段通常都具有相同属性，(长度、数据类型和小数位)。例如，1.3.4.4 的图 3-10，在物理文件 PF1 中字段 Name1 是一个 10 个长的字符型字段，它可以和物理文件 PF2 中的字段 Name2 相连接。Name2 也是一个 10 个字符长的字符型字段。Name1 和 Name2 有相同特征，所以它们很容易用作连接字段。

也可以将不同长度的字符型字段用作连接字段，而不需任何对字段的重定义。例如，如果 PF1 的 Name1 是 10 个字符长，PF2 中的 Name2 是 15 个字符长，这两个字段可用作连接字段而不需对其重定义。

下面是一个连接字段没有相同属性的例子，物理文 PF1 中的 Nbr 字段和物理文件 PF2 中的 Nbr 字段都在 34 列指定了 3 位长度，但 PF1 中的字段是区位的（35 列为 S），PF2 中字段为压缩的（35 列为 P）。要用这个连接字段来连接文件，必须将这两个字段之一重新定义以便它们有相同属性。

下面是物理文件和逻辑文件中的字段：



文件的 DDS 码如下：

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R JOINREC                    JFILE(PF1 PF2)
     A          J                            JOIN(PF1 PF2)
     A                                       JFLD(PRTNBR PRTNBR)
     A            PRTNBR          S  N       JREF(1)
     A            DESC
     A            PRICE
     A            QUANT
```

```
     A          K DESC
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R REC1
     A            DESC          30
     A            PRTNBR         6P 0
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R REC2
     A            PRTNBR         6S 0
     A            PRICE          7  2
     A            QUANT          8  0
     A
```

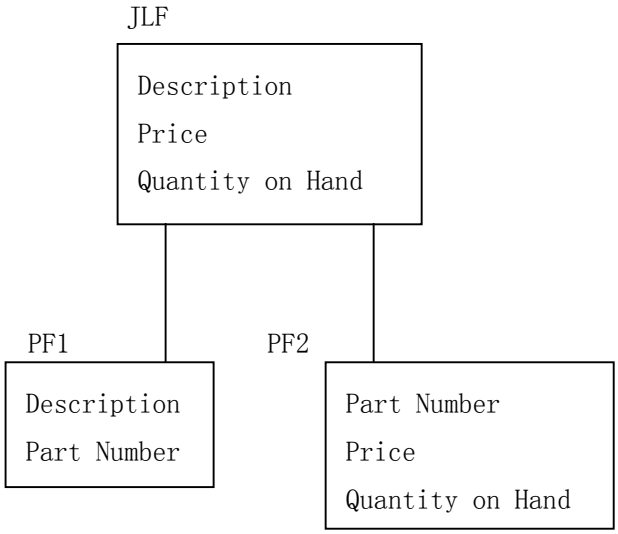
注：例中，逻辑文件中字段 Nbr 来自于 PF2，这是因为指定了 JREF（2），可以在 JREF 中指定相关的文件号，代替指定物理文件名，例中，2 即表示 PF2。

因为用 Nbr 字段用作连接字段，它们必须具有相同属性，而例中它们没有相同属性，因此，必须重新定义其中之一以使其具有相同属性。本例中为了解决这两个雇员号字段属性的不同，在 JLF 中的 Nbr 字段（来自于 PF2）被重新定义为区位的（JLF 中 35 列为 S）。

1.3.4.6 描述在记录格式中不出现的字段（例 5）

Neither 字段（第 38 列为 N）用于连接逻辑文件中表示即不输入也不输出，使用此连接逻辑文件的程序即不能看也不能读 Neither 字段。Neither 字段不包括在记录格式中。Neither 字段不能用作索引字段或连接文件的选择/省略语句。可用 Neither 字段做为连接字段（在连接规范中用 JFLD 键字），它可在记录层重新定义，但程序并不需要它。

下例中，程序读了库存中的该零件的描述、价格和库存的数量，零件号本身并不排除将零件记录放在一起，但是，由于零件号具有不同的属性，所以至少有一个要重新定义。



文件的 DDS 码如下：

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R JOINREC                   JFILE(PF1 PF2)
     A           J                           JOIN(PF1 PF2)
     A                                       JFLD(PRTNBR PRTNBR)
     A             PRTNBR           S   N    JREF(1)
     A             DESC
     A             PRICE
     A             QUANT
     A           K DESC
     A

PF1
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R REC1
     A             DESC           30
     A             PRTNBR          6P 0
     A


PF2
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R REC2
     A             PRTNBR          6S 0
     A             PRICE           7  2
     A             QUANT           8  0
     A
```

在 PF1 中，字段 Prtnbr 为压缩十进制，PF2 中字段 Prtnbr 为区位十进制，在连接逻辑文件中，它们被用作连接字段，并且 PF1 中的 Prtnbr 字段通过在字段级第 35 列指定 S，将其重新定义为区位十进制，键字 JREF 指出了字段来自于哪个物理文件，然而该字段并不包括在记录格式中，因此，在第 38 列指定为 N，使其成为一个 Neither 字段，使用此文件的程序也看不到该字段。

此例中，售货员可以打出零件的描述，程序为了得到一匹配的记录去读连接逻辑文件，并且显示一个或多个零件给用户去检查，它包括描述、价格和数量，该应用程序认为零件号对于完成一顾客订货单或为仓库订多种零件不是必须的。

1.3.4.7 在连接逻辑文件中指定索引字段（例 6）

如果在逻辑文件中指定了索引字段，要遵守下列规则：

- 索引字段必须存在于主物理文件中。
- 索引字段必须在逻辑文件的连接记录格式的 19 到 28 列中被命名。
- 索引字段不能是在逻辑文件中定义为 Neither 的字段。

  （在字段的 38 列指定为 N）

下面说明了索引字段的规则：

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R JOINREC                   JFILE(PF1 PF2)
     A           J                           JOIN(PF1 PF2)
     A                                       JFLD(NBR NUMBER)
     A                                       JFLD(FLD3 FLD31)
     A             FLD1                      RENAME(F1)
     A             FLD2                      JREF(2)
     A             FLD3          35    N
     A             NAME
     A             TELEPHONE                 CONCAT(AREA LOCAL)
     A           K FLD1
     A           K NAME
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R REC1
     A             NBR            4
     A             F1            20
     A             FLD2           7   2
     A             FLD3          40
     A             NAME          20
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R REC2
     A             NUMBER         4
     A             FLD2           7   2
     A             FLD31         35
     A             AREA           3
     A             LOCAL          7
     A
```

下面是不能作为索引字段的字段：

- Nbr（在第 19 列到 28 列没有命名）
- Number（在第 19 到 28 列没有命名）
- F1（在第 19 到 28 列没有命名）

- F1d31（来自于次文件）
- F1d2（来自于次文件）
- F1d3（是 Neither 字段）
- Area 和 Local（在第 19 到 38 列没有命名）
- Telephone(它所依据的字段来自于次文件）

1.3.4.8 在连接逻辑文件中指定选择/省略语句

如果在一连接逻辑文件中指定了选择/省略语句，规则如下：

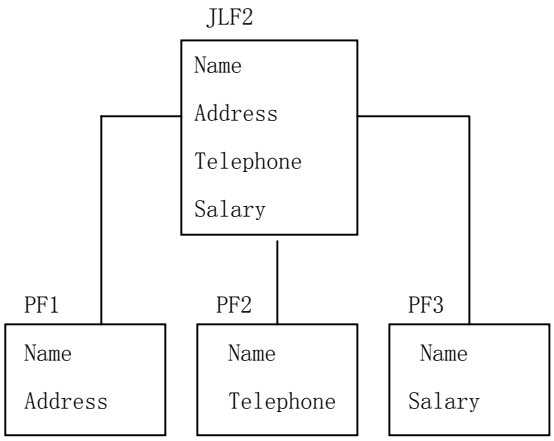
- 字段来自于逻辑文件使用的物理文件（指定 JFILE 键字)
- 指定了选择/省略语句的字段不是定义为 Neither 的字段(在字段的 38 列指定为 N)
- 在某些情况下，当在连接逻辑文件中指定了选择/省略语句时，必须指定键字 DYNSLT。有关详细资料和范例，请看 1.3.4.12 中“一个复杂的连接逻辑文件(例 10)”。

1.3.4.9 连接三个以上物理文件(例 7)

可以用一个连接逻辑文件连接多达 32 个物理文件。这些文件必须是由 JFILE 键字指定的。JFILE 指定的第一个文件是主文件，其它的文件均为次文件。

物理文件必须是成对连接的，每一对都是通过一个连接规范来描述的，每个连接规范都必须指定一个或多个连接字段。

下面显示了文件中的字段，其中有一个字段对逻辑文件中相关的物理文件来说是公共的。



在这个例子中，字段 Name 对所有的物理文件(PF1、PF2、PF3)是公共的，并且将其作为连接字段。

下面是物理文件和逻辑文件的 DDS 描述:

```
JLF
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R JOINREC                   JFILE(PF1 PF2 P3)
     A          J                           JOIN(PF1 PF2)
     A                                      JFLD(NAME NAME)
     A          J                           JOIN(PF2 PF3)
     A                                      JFLD(NAME NAME)
```

```
      A              NAME                          JREF(PF1)
      A              ADDR
      A              TELEPHONE
      A              SALARY
      A            K NAME
      A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
      A            R REC1
      A              NAME           10
      A              ADDR           20

      A            K NAME
      A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
      A            R REC2
      A              NAME           10
      A              TELEPHONE       7
      A            K NAME
      A
```

PF3

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
      A            R REC3
      A              NAME           10
      A              SALARY          9  2
      A            K NAME
      A
```

假定物理文件有如下记录:

PF1

```
Anne  120 1st St.
Doug   40 Pillsbury
Mark    2 Lakeside Dr.
Tom   335 Elm St.
```

PF2

```
Anne  555-1111
Doug  555-5555
Mark  555-0000
Sue   555-3210
```

PF3

```
Anne  1700.00
Doug   950.00
Mark  2100.00
```

程序读出逻辑文件记录如下:

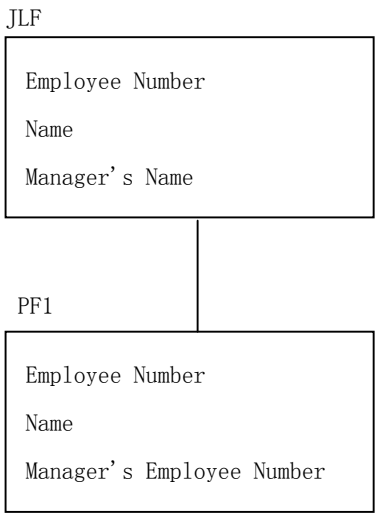
JLF

```
Anne  120 1st St.        555-1111  1700.00
Doug   40 Pillsbury      555-5555   950.00
Mark    2 Lakeside Dr.   555-0000  2100.00
```

由于在 PF2 和 PF3 中没有发现记录 Tom，并且没有指定 JDFTVAL 键字，所以记录 Tom 没返回，由于文件中没有记录 Sue，所以记录 Sue 也没返回，由于主文件没记录 Sue，所以记录 Sue 也不返回。

1.3.4.10 连接物理文件和它本身（例 8）

为了读由物理文件自身的两个或多个记录合并起来所形成的记录，可以将该物理文件与其自身连接起来。下面显示了此例子：



以下是这些文件的 DDS 描述：

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                    JDFTVAL
     A          R JOINREC                 JFILE(PF1 PF1)
     A          J                         JOIN(1 2)
     A                                    JFLD(MGRNBR NBR)
     A            NBR                     JREF(1)
     A            NAME                    JREF(1)
     A            MGRNAME                 RENAME(NAME)
     A                                    JREF(2)
```

```
      A

PF1
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
      A            R RCD1
      A              NBR              3
      A              NAME            10            DFT('none')
      A              MGRNBR           3
      A
```

注：

1.由于在 JFILE 键字上同一文件名被指定了两次，所以相关的文件号也必须在 JOIN 键字中指定。相关文件号 1 是指由 JFILE 键字指定的第一个物理文件。2 则表示第二个文件，以此类推。

2.由于在 JFILE 键字中指定了同一物理文件，所以对字段级的每个字段都必须有 JREF 键字。

假定 PF1 包含的记录如下：

PF1

235	Anne	440
440	Doug	729
500	Mark	440
729	Sue	888

程序读到的逻辑文件记录如下：

PF1

235	Anne	Doug
440	Doug	Sue
500	Mark	Doug
729	Sue	none

注：由于指定了 JDFTVAL 键字，管理员 Sue 有一个记录被返回。还要注意到物理文件 PF1 的字段 Name 使用了 DFT 键字，所以返回了一个 None 值。

1.3.4.11 使用次文件中无记录的缺省值（例 9）

如果连接两个以上的文件，可以指定 JDFTVAL 键字，对于一个次文件中没有的并用于连接其它次文件的连接字段，将由系统提供缺省值，如果在次文件中指定了 DFT 键字，则逻辑文件中用 DFT 指定的值。

文件的 DDS 如下：

JLF

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A                                     JDFTVAL
     A           R JRCD                    JFILE(PF1 PF2 PF3)
     A           J                         JOIN(PF1 PF2)
     A                                     JFLD(NAME NAME)
     A           J                         JOIN(PF2 PF3)
     A                                     JFLD(TELEPHONE TELEPHONE)
     A             NAME                    JREF(PF1)
     A             ADDR
     A             TELEPHONE               JREF(PF2)
     A             LOC
     A
```

PF1

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R RCD1
     A             NAME           20
     A             ADDR           40
     A             COUNTRY        40
     A
```

PF2

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R RCD2
     A             NAME           20
     A             TELEPHONE       8          DFT('999-9999')
     A
```

PF3

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A           R RCD3
     A             TELEPHONE       8
     A             LOC            30          DFT('No location assigned')
     A
```

假定 PF1、PF2 和 PF3 有如下记录：

PF1

```
Anne  120 1st St.
Doug   40 Pillsbury
Mark    2 Lakeside Dr.
Sue   120 Broadway
```

PF2

```
Anne  555-1234
Doug  555-2222
Sue   555-1144
```

PF2 中没有 Mark 的电话号码

PF3

```
555-1234  Room 312
555-2222  Main Lobby
999-9999  No telephone number
```

PF3 中没有电话号码为 555-1144 的记录

在连接逻辑文件中规定了 JDFTVAL，程序读到下列逻辑文件记录：

```
Anne  120 1st St.       555-1234   Room 312
Doug   40 Pillsbury     555-2222   Main Lobby
Mark    2 Lakeside Dr.  999-9999   No telephone number
Sue   120 Broadway      555-1144   No location assigned
```

此例中，Anne 和 Doug 有完整的数据，但是 Mark 和 Sue 则丢失了部分数据。

- 由于 Mark 没有电话号码,所以 PF2 丢失了一个记录,用键字 DFT 将 PF2 中 Telephone 字段的缺省值定义为 999-9999，因此本例中，当没有电话号码时，返回 999-9999 的电话号码，由于在连接逻辑文件中指定了 JDFTVAL，它使得 PF2 中 Telephone 字段的缺省值（为 999-9999）可以和 PF3 中的一个记录相匹配。（PF3 中包括一个记录描述，其电话号码为 999-9999）。若不使用 JDFTVAL，则 Mark 将没有记录返回。
- Sue 的电话号码也没有相应的分配定位单元，因此，电话号 555-1144 的记录在 PF3 中丢失了，若不指定 JDFTVAL，则 Sue 将没有记录返回，指定了 JDFTVAL，系统将提供一个由 PF3 中 Loc 字段指定的缺省值（是 No location assignd）。

1.3.4.12 复杂的连接逻辑文件（例 10）

下例显示了一个较为复杂的连接逻辑文件，假定三个物理文件的数据如下：

连接逻辑文件的记录格式包含如下字段：

- Vdrnam（卖主名）
- Street,City,State 和 Zipcode（卖主地址）
- Jobnbr（工作号）
- Prtnbr（零件号）

• Descr（零件描述）
• Qorder（订货数量）
• Untprc（单价）
• Whsnbr（仓库号）
• Prtloc（零件位置）

连接逻辑文件 DDS 如下：

```
Vendor Master File (PF1)
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD1                      TEXT('VENDOR INFORMATION')
     A            VDRNBR         5          TEXT('VENDOR NUMBER')
     A            VDRNAM        25          TEXT('VENDOR NAME')
     A            STREET        15          TEXT('STREET ADDRESS')
     A            CITY          15          TEXT('CITY')
     A            STATE          2          TEXT('STATE')
     A            ZIPCODE        5          TEXT('ZIP CODE')
     A                                       DFT('00000')
     A            PAY            1          TEXT('PAY TERMS')
     A

Order File (PF2)
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD2                      TEXT('VENDORS ORDER')
     A            VDRNUM         5S 0       TEXT('VENDOR NUMBER')
     A            JOBNBR         6          TEXT('JOB NUMBER')
     A            PRTNBR         5S 0       TEXT('PART NUMBER')
     A                                       DFT(99999)
     A            QORDER         3S 0       TEXT('QUANTITY ORDERED')
     A            UNTPRC         6S 2       TEXT('PRICE')
     A

Part File (PF3)
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A          R RCD3                      TEXT('DESCRIPTION OF PARTS')
     A            PRTNBR         5S 0       TEXT('PART NUMBER')
     A                                       DFT(99999)
     A            DESCR         25          TEXT('DESCRIPTION')
     A            UNITPRICE      6S 2       TEXT('UNIT PRICE')
     A            WHSNBR         3          TEXT('WAREHOUSE NUMBER')
```

```
A            PRTLOC         4          TEXT('LOCATION OF PART')
A            QOHAND         5          TEXT('QUANTITY ON HAND')
A
```

Join Logical File (JLF)

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
A                                    1   DYNSLT
A                                    2   JDFTVAL
A          R RECORD1                     JFILE(PF1 PF2 PF3)
A     3    J                             JOIN(1 2)
A                                        JFLD(VDRNBR VDRNUM)
A                                    4   JDUPSEQ(JOBNBR)
A     5    J                             JOIN(2 3)
A                                    6   JFLD(PRTNBR PRTNBR)
A                                        JFLD(UNTPRC UNITPRICE)
A       7    VDRNUM         5A  N        TEXT('CHANGED ZONED TO CHAR')
A            VDRNAM
A            ADDRESS                 8   CONCAT(STREET CITY STATE +
A                                         ZIPCODE)
A            JOBNBR
A            PRTNBR                  9   JREF(2)
A            DESCR
A            QORDER
A            UNTPRC
A            WHSNBR
A            PRTLOC
A    10    S VDRNAM                       COMP(EQ 'SEWING COMPANY')
A          S QORDER                       COMP(GT 5)
A
```

1.由于指定了 JDFTVAL 键字和选择字段，所以必须指定 DYNSLT 键字。
2.规定 JDFTVAL 是为了采用物理文件的缺省值。
3.第一个连接规范。
4.由于在 PF2 中有重复的卖主号，所以要指定 JDUPSEQ 键字。
5.第二个连接规范。
6.指定了两个 JFLD 键字是为了确保从 PF2 和 PF3 中连接到正确的记录。
7.字段 Vdrnum 由区位十进制重新定义为字符型。
（这是因为该字段被用作连接字段，并且它在 PF1 和 PF2 中属性不同）。
8.键字 CONCAT 将来自于同一物理文件的四个字段合并成一个字段。
9.由于 Prtnbr 字段存在于两个物理文件中，并且你想用的是 PF2 中的该字段，所以必须

指定 JREF 键字。

10.Vdrnam 和 Qorder 是选择/省略字段。（注意它们来自于不同的物理文件）。

1.3.4.13 性能考虑

可以用下列方法去提高连接逻辑文件的性能：

- 如果要连接的物理文件有不同数量的记录，应该先说明记录最少的文件（JOIN 键字后的第一个参数）。
- 考虑使用 DYNSLT 键字，详细信息请看 1.3.2.1.2 中“动态的选择/省略”。
- 考虑描述一连接逻辑文件,使它能自动的共享一现存的访问路径,详细资料请看 1.3.2.2 “使用现存的访问路径”。

注：连接逻辑文件总是用 JFLD 键字指定的一对字段中的第二个字段作为访问路径。该字段就象简单逻辑文件中的索引字段。如果没有现存的访问路径，则在立即维护的访问路径中生成隐含的访问路径。

1.3.4.14 数据完整性考虑

除非连接逻辑文件中用到的物理文件加锁，否则可能发生如下情况：

- 程序读取的一个记录可能是在次文件中有两个或多个记录相对应，系统将提供一个记录给程序。
- 如果有另一个程序要更新你正在读的主文件中的记录，可改变连接字段。
- 程序发出另一个读操作请求，系统将在主文件连接字段的当前值（新值）的基础上提供下一个记录。

这些考虑也可以应用于次文件。

1.3.4.15 连接逻辑文件的规则总结

1.3.4.15.1 要求

连接逻辑文件的基本要求是：

- 每个连接逻辑文件都必须有：
  - —只有一个记录格式，并且是用 JFILE 键字指定的。
  - —由 JFILE 键字指定的至少有两个物理文件名（在 JFILE 中的物理文件名不必是不同的文件）。
  - —至少有一个连接规范（第 17 列为 J，并且使用了 JFLD 键字）。
  - —最多可有 31 个次文件。
  - —在字段级中至少有一个字段名，其字段不能用 N（Neither).
- 如果 JFILE 只指定了两个物理文件，则不需要 JOIN 键字，它只包括一个连接规范。并且连接了两个物理文件。
- 如果 JFILE 指定了多于两个物理文件，要遵守如下规则：
  - —主文件必须是第一个 JOIN 键字指定的一对文件中的第一个文件（主文件也可以是其它 JOIN 指定一对文件中的第一个文件）。
    注：当在 JFILE 键字中同一文件名被指定了两次时，JOIN 和 JREF 键字中指定相关的文件号。
  - —次文件作为 JOIN 键字指定的一对文件中的第二个文件，每个次文件必须只被指定一次，这意味着对 JFILE 中的每个次文件来说，都必须包括一个连接规范（两个次文件意味着两个连接规范，三个次文件意味着三个连接规范）。

—在连接规范中次文件出现的顺序必须和它们在 JFILE 中的顺序匹配。

1.3.4.15.2 连接字段

有关连接字段的规则如下：

- 对于每个要连接的物理文件至少要有一个连接字段，连接字段是 JFLD 键字中一个参数。
- 连接字段（JFLD 指定的）必须有相同的属性（长度、 数据类型和小数位），否则要在连接逻辑文件的记录格式中重新定义以便其具有相同的属性。如果连接字段是字符型，则字段长度可以不同。
- 连接字段不须在连接逻辑文件的记录格式中规定。（除非重新定义以便使其具有相同属性）。
- 如果重新定义了一个连接字段，可以在第 38 列指定为 N（使其成为 Neither 字段），以防止使用此连接逻辑文件的程序使用这个重定义字段。
- 用在连接的物理文件中字段的最大长度等于物理文件和逻辑文件键字的最大尺寸，（请看附录 A“数据库文件尺寸”）。

1.3.4.15.3 连接逻辑文件中的字段

有关连接逻辑文件中的字段规则如下：

- 连接逻辑文件记录格式中的字段必须是连接逻辑文件所用到的物理文件中的字段，或者是，如果指定了 CONCAT、RENAME、TRNTBL 或 SST，则该字段为其中一个物理文件字段的结果字段。
- 作为 CONCAT 参数值的字段必须来自于同一物理文件，如果 CONCAT 指定的第一个字段名在物理文件中不是唯一的，必须对该字段指定 JREF 键字，以指出用的字段来自于哪个物理文件。
- 如果一个连接逻辑文件记录格式中的字段名在多个物理文件中都出现过，必须用 JREF 键字唯一的指定该字段来自于哪个物理文件。
- 如果指定了索引字段，它必须来自于主文件，连接逻辑文件中的索引字段不必是主文件中的索引字段。
- 选择/省略字段可以来自于连接逻辑文件所用到的任何一个物理文件，但在某情况下需要指定 DYNSLT 键字。
- 如果指定了索引字段和选择/省略字段，则它们必须在记录格式中被定义。
- 如果 JFILE 键字指定了多个物理文件名，则必须用 JOIN 和 JREF 来说明相关的记录号。

1.3.4.15.4 其它情况

使用连接逻辑文件时所要记住的其它规则有：

- 连接逻辑文件是一个只读文件。
- 连接记录格式不能被共享，也不能共享其它记录格式。
- 下面是连接逻辑文件所不允许的：
  —REFACCPTH 和 FORMAT 键字。
  —BOTH 字段（38 列为 B）。

第四章　数据库安全

这个章节描述了一些数据库安全功能，包括有数据文件安全性、公共权限、对文件中数据修改和删除的约束以及用逻辑文件保护数据。

关于详细 AS/400 中安全功能的资料，请参阅《安全参考手册》。

1.4.1 文件和数据权限

下面介绍能分配给用户的对数据库的各种权限。

1.4.1.1 目标操作权限

对下列操作，必须具有目标操作权限：

- 打开文件用来处理（必须至少有一种数据权限）
- 编译使用文件描述的程序
- 显示一个文件中活动成员信息
- 打开文件用来查询处理，例如，命令 OPNQRYF 打开一个文件用来查询处理。

注：对某些打开操作也要有适当的数据操作权限。

1.4.1.2　目标存在权限

对下列操作必须有目标存在权：

- 删除文件
- 保存、重存和释放文件存贮空间，如果目标存在权限没有明确授予用户，*SAVSYS 的特权允许用户保存、重存、释放文件空间。*SAVSYS 权和目标存在权限不相同。
- 从文件中移出成员
- 转移文件的主人

注：除保存/重存文件，所有这些功能都需要目标操作权。

1.4.1.3　目标管理权限

对下列操作，必须具有目标管理权：

- 生成一个只有索引顺序访问路径的逻辑文件，（由逻辑文件对物理文件的引用需要目标管理权限）。
- 分配和撤消授权，只能分配和撤消已有的权限（同时有对文件操作权）。
- 修改文件
- 增加文件成员（文件的主人变为新成员的主人）。
- 修改文件成员
- 移动文件
- 重命名文件
- 重命名文件中的成员
- 消除文件中一个成员（也要有删除数据权）
- 初始化文件中成员（初始化缺省记录也应有增加数据权，初始化被删除的记录要有删除数据权）

1.4.1.4 目标更改权限

目标更改权用于下面情况：

许多操作与目标管理权相同（见前面一节），目标更改权是对目标管理权替换的权力。

1.4.1.5 目标引用权限

目标引用权提供下列功能：

从一个目标引用另一目标的权力。这样，引用目标的操作将受到被引用目标的约束。

增加一个物理文件引用限制要对其文件检查目标的管理权和引用权。物理文件限制将在第 15 章中“物理文件限制”和 16 章“引用完整性”中描述。

1.4.1.6 数据权限

请求分配给物理文件和逻辑文件的数据权限是：

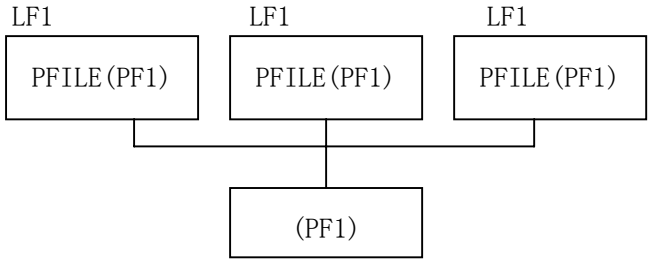
- 读：可以读取文件中的记录
- 添加：可以添加新的记录
- 更新：可以更新已存在记录（为了做更新读一个记录，必须同时有读取权）
- 删除：可以用来删除文件中记录（为了做删除读一个记录，必须同时有读取权）
- 执行：主要用在库和程序中。例如，你正在更改一个与触发器有关文件，必须对触发器程序的有执行权。如果没有，系统不会启动触发器程序。关于触发器详细内容参阅 17 章“触发器”。

通常，在文件中你所具有的对数据的权限是在实际执行了输入/输出的操作时检验的，用打开查询文件和数据库文件命令打开文件时，要检查数据权限。

如果用户没有对一个文件的操作权，那么就无法打开文件。

下面例子显示了分配给逻辑文件和它所涉及的物理文件权限之间的关系。逻辑文件 LF1、LF2、LF3 都基于物理文件 PF1，USERA 对 PF1 有数据的读取（*READ）、添加（*ADD）的权限，对 LF2、LF1 有目标操作（*OBJOPR）、读取（*READ）、添加（*ADD）的权限，这意味着，USERA 不能对 PF1 打开以及直接使用数据。因为他没有对 PF1 的操作权（*OBJOPR），但他可以打开 LF1、LF2 并通过 LF1、LF2 对 PF1 的记录读取和添加。注意，用户没有授予对 LF3 的权限，因而不能使用它。

```
GRTOBJAUT OBJ(LF1) USER(USERA) AUT(*OBJOPR *READ *ADD)...
GRTOBJAUT OBJ(LF2) USER(USERA) AUT(*OBJOPR *READ *ADD)...
```




```
GRTOBJAUT OBJ(PF1) USER(USERA) AUT(*READ *ADD)...
```

1.4.2 公共权限

生成文件时，可以通过生成命令中的 AUT 参数确定公共权限，那些对文件没有特权或不是成组授权的成员，他们使用公共权限。

公共权限是权限检验的最后一级，也就是说，用户如果有特定权限或是成组授权中的一员，那么不会检验公共权限。

公共权限可以如下设定：

*LIBCRTAUT	当要在一个库中生成文件时，要检查库来确定文件的公共权限。每个库都有一个权限，这个权限是在库生成时指定的。如果用 CRTLF、CRTPF 和 CRTSRCPF 命令生成文件时，其中的参数 AUT 规定为 *LIBCRTAUT，那么所有生成在这个库的文件都被分配这个库的公共权限，*LIBCRTAUT 的值是缺省的公共权限。
*CHANGE	所有没有特别权限或成组权限的用户都有文件数据的修改权。
*USE	所有没有特别权限或没有成组权限的用户都对文件数据有读取的权。
*EXCLUDE	只有所有者、安全员、有特别权限或是有特别权限组中的一员才能使用此文件。
*ALL	没有特别权限或不是特别权限组的所有用户有数据所有者的权限以及目标操作、目标管理、目标存在权。
授权表名	授权表是用户和他所具有权限的列表，这个列表允许用户和他们的不同权限组合在一起。

注：当生成一个逻辑文件时，不分配数据权限。因此，*CHANGE 和*USE 相同，并且不能分配*ALL 的权限。

可以用编辑目标权限命令（EDTOBJAUT），分配目标权限命令（GRTOBJAUT）或撤消目标权限的命令（RVKOBJAUT）来处理目标的权限。

1.4.3 数据库文件性能

一个文件的性能用来控制不受文件权限约束的输入/输出操作。

生成一个物理文件时，可以用 CRTPF 和 CRTSRCPF 命令中的 ALWUPD 和 ALWDLT 的参数指定物理文件更新或删除的属性，通过生成一个不允许更新或删除的文件，可以进一步强调数据一经录入，就不能从文件中修改或删除。

逻辑文件不能明确设定它的属性。文件的性能要由所基于的物理文件性能来决定。

在文件生成后，不能修改文件性能，必须删除它，然后重建立某些性能的文件，显示文件描述的命令（DSPFD）可给出文件的性能。

1.4.4 用逻辑文件保护数据

可以用逻辑文件使物理文件中的某个字段不能读出，即在逻辑文件的格式中不包括不希望被阅读出的字段，详细内容参阅 1.3.1 节中“描述逻辑文件记录格式”的内容。

可以通过在 DDS 第 38 列指定 I（仅输入）来防止修改物理文件中一个或多个字段，详细内容参阅 1.3.1.1“描述逻辑文件的字段”的内容。

可以通过逻辑文件保护那些基于的物理文件记录中一个或多个字段内容。为了能保护这些记录，在描述逻辑文件时使用选择/忽略键字。详细内容参阅 1.3.2.1“使用逻辑文件的选择/省略字段”的内容。

第二部分　在程序中处理数据库文件

- 这部分包括在程序中如何处理数据文件信息。其中包括在程序或作业中如何处理文件以及在程序中如何提高性能。这章包括文件执行参数的描述和性能规定，能更有效处理文件的一些运行时选项。
- 介绍作业间数据库文件的共享，这样可以在同一时刻多个用户同时存取文件，另外也介绍文件、记录、成员加锁的办法，这样可以防止在交叉作业中的同时读取。介绍打开查询文件命令（OPNQRYF）和打开数据库文件命令（OPNDBF）在程序中打开数据文件成员。
  例如，性能考虑，在编写高级语言程序的要遵循的规则，同时也介绍可能发生的典型错误。
- 最后介绍基本数据库操作，其中包括在数据库中定位、读、更新、添加、删除记录，多种读取记录的方法，讨论在逻辑文件或物理文件中修改已存在记录，向物理文件成员中写进新记录，这节还讨论完成对数据文件处理后关闭文件的方法，以使程序与文件脱钩，还要介绍在程序中始何监控数据库文件错误的信息。

第五章　运行时的考虑

- 在打开文件进行处理前，应考虑在程序或作业中怎样使用文件，对运行时文件处理参数的理解有助于避免非期望结果的产生，因此可以改善程序性能。
- 当一个文件打开时，数据文件中的属性说明与程序参数联系起来。通常，打开和处理文件时所需的大部分系统信息都在文件属性和应用程序本身中找到。
- 有时也要忽略在程序或文件中找到的处理参数。例如，如果不处理文件中第一个成员，就要有办法告诉系统要处理哪个成员，可用遮盖数据库文件命令（OVRDBF）来完成这个操作，这个命令也允许设定作业中参数以改进作业性能，但不能在文件属性或程序中规定。OVRDBF 命令参数优先于文件程序的属性，详细的信息见《ILE Concepts》。
- 这章描述了文件处理的参数，这些参数的值由高级语言文件属性和程序调用前处理的打开或遮盖的命令来决定。关于这些参数及何处使用它们可在 2.1.7 节“运行时问题总结”中找到。关于命令的处理参数，请参考 CL 参考手册中下面命令的说明：

  生成物理文件（CRTPF）
  生成逻辑文件（CRTLF）
  生成源物理文件（CRTSRCPF）
  增加物理文件成员（ADDPFM）
  增加逻辑文件成员（ADDLFM）
  修改物理文件（CHGPF）
  修改物理文件成员（CHGPFM）
  修改逻辑文件（CHGLF）
  修改逻辑文件成员（CHGLFM）
  修改源物理文件（CHGSRCPF）
  遮盖数据库文件（OVRDBF）
  打开数据库文件（OPNDBF）
  打开查询文件（OPNQRYF ）
  关闭文件（CLOF）

2.1.1 文件和成员名

FILE 和 MBR 参数：

- 在处理数据库文件的数据之前，必须指出所要用的文件和成员。通常是在高级语言程序中指定文件名，如果必要还要指出成员名。系统就会在程序请求打开文件时使用这个名字，为遮盖程序中已有的文件名，打开另一个文件，可以用（OVRDBF）中的 TOFILE 参数。如果程序中没有指定成员名，那么就处理文件中第一个成员（生成时定义）。
- 如果在高级语言程序中没办法指定成员名（有些高级语言程序不允许成员名）或是要处理的不是第一个成员，可以用 OVRDBF 或打开文件命令（OPNDBF 或 OPNQRYF）中的参数 FILE 和 MBR 指出要处理的文件和成员名。
- 要处理文件中的所有成员用遮盖文件命令(OVRDBF)的 MBR(*ALL)参数。例如，FILEX 有三个成员，要处理所有的成员，可以指定

  OVRDBF　FILE(FILEX)　　MBR(*ALL)

- 如果指定 MBR(*ALL)，程序以成员生成顺序读取对每个成员，程序要看文件是索引的或顺序的来确定按键字或到达顺序读取记录。

2.1.2 文件处理选项

下面说明几个运行时处理的选项，包括程序中所用的文件操作的标识、指定文件开始位置、重用删除的记录、忽略索引访问路径、规定如何控制文件结束的操作以及标识文件中记录长度。

2.1.2.1 规定处理类型

OPTION 参数。

当在程序中用到一个文件，系统需要知道你对文件处理的类型。例如，系统需要知道你是仅从文件读数据还是既要读又要更新数据。有效的操作选项是：输入、输出、更新和删除，系统通过在程序中或是在打开数据库文件命令（OPNDBF 和 OPNQRYF）中指定 OPTION 参数来决定其处理属性。

系统通过这些选项决定在程序中允许哪些操作。例如，如果仅为输入打开文件，而程序企图做输出，那么会发生错误。

通常，系统在程序的输入输出时确认所需数据的权限。然而，当你用 OPNQRYF 或 OPNDBF 命令时，系统在文件打开时就确认执行 OPTION 参数所需权限。关于数据文件权限的详情请参阅 1.4.1.6 节“数据权限”。

系统也用这些选项确定加锁功能来保护文件和记录中数据完整性。详情见 2.1.4“在交叉作业间共享数据库文件”。

2.1.2.2 规定文件初始位置

OPSITION 参数。

系统需知道在文件打开后，在何处开始执行操作。缺省情况是从文件的第一个记录开始（第一个顺序读操作从第一个记录读起），但可以用 OVRDBF 命令告诉系统从最后一个记录读起或从文件中间的某个记录开始，也可以在程序中动态地指定位置。关于在程序中设置文件位置的详细情况，参阅 2.3.1 节“在文件中指定位置”。

2.1.2.3 重用已删除的记录

REUSEDLT 参数。

在生成物理文件（CRTPF）或修改物理文件（CHGPF）命令中指定 REUSEDLT(*YES) 参数，下面的操作可能用不同的方式处理：

- 重用删除记录空间的文件，使到达顺序毫无意义，记录可能不会加到文件末尾。
- 不会发生文件结束的延迟。
- 使用 DDM 从前个版本系统转到当前版本系统的应用程序，使用重用删除记录的文件时可能得到不同结果。
- 不能保证百分之百回收已删除记录原来占有的空间，即使文件已达到全满条件或已超出，在文件中也可能存在删除记录的空间。

由于系统重用删除记录空间的方式不同，在生成或修改文件使之重用删除记录空间之前要考虑以下几点：

用相关记录号处理的文件和由一个应用程序来确定某个文件的相关记录号做为另个文件的键字时，不能重用删除记录的空间。

用作队列的文件不能使用重用记录空间。

应用程序涉及的文件要把新记录加到这个文件末尾不能重用删除记录空间。

如果决定修改已存在的物理文件为重用删除记录空间，而且引用这个物理文件的一些逻辑文件用 LIFO、FIFO 来处理重复键值，可以用下面的方法重新生成没有 LIFO、FIFO 属性的逻辑文件，并避免重建已有的访问路径。

1.重命名有 LIFO、FIFO 属性的逻辑文件；

2.生成第二个逻辑文件和没改名的逻辑文件同名，不同的是不规定重复键字的顺序，新文件共享改名文件的访问路径；

3.删除改名文件。

2.1.2.4 忽略索引顺序访问路径

ACCPTH 参数。

当你处理一个带索引访问路径的文件时，一般都用这个路径获得数据。如果在文件中定义了索引字段，系统会自动用索引访问路径，然而，有时可以忽略索引顺序而用到达顺序来改善性能。

可以在高级语言程序或打开文件（OPNDBF）命令中告诉系统忽略索引访问路径。当忽略索引访问路径时，不允许用索引读数据的操作。操作按到达顺序访问路径进行，(如果这一功能用在选择/省略值定义的逻辑文件)，就使用到达顺序，并只有那些符合选择/省略值的记录返回。这个操作类似于文件中规定了 DYNSLT 索引一样。

注：不能忽略那些基于多个物理文件成员的逻辑文件成员中的索引顺序访问路径。

2.1.2.5 延迟结束文件操作的处理

EOFDLY 参数。

当正在读数据库文件时，程序检索到数据的末尾。正常的，系统会告诉程序已没有数据可读，有时可能想用另外的方式，不是通知程序无数据可读，而让系统挂起程序，直到有数据到达文件中。当更多的数据到达文件后，程序可以读入新到达的数据。如果你要求这种处理，可以在 OVRDBF 命令中使用 EOFDLY 参数，关于这个参数更详细内容请参阅 2.3.2.3 节“当文件结束时等待新记录”。

注：文件延迟结束操作不可使用重用删除记录空间的文件。

2.1.2.6 指定记录长度

系统应知道程序要处理的记录长度，但不非在程序中指定。系统会从程序中给出的文件名中指出的属性和描述自动确定这些信息，但有一个选项可以在高级程序中指定记录长度。

如果打开的文件中包括的记录比程序中指定的长，系统会分配一个存储区适合文件成员的记录长，这个选项被忽略，这时整个记录传给程序（但有些高级语言只让你存取程序中规定的记录长度部分），当文件记录小于指定长度，系统分配一个指定长度存储区，程序可以使用其余部分，但只有文件中定义的记录长可以用做输入/输出操作。

2.1.2.7 忽略记录格式

当使用多格式逻辑文件，系统假定你要用文件中的所有格式，但你并不要用所有的格式，那么可以指定哪些要用、哪些不用。如果不用这个选项来忽略某些格式，程序要能处理文件中定义的所有格式。详细说明参阅高级语言指南。

2.1.2.8 确定是否存在重复键字

DUPKEYCHK 参数。

对于索引访问路径用它来检测其键值是否重复，从而确定不同的输入/输出操作。

对输入操作（读取），索引访问路径是文件打开的路径，其它任何存在于物理文件的索引路径都不考虑，而且，任何在索引路径中因选择/省略项而被省略的记录在决定是否为重复键值时不予考虑。

对输出和更新操作，要查找所有基于物理文件的用*IMMED 维护的非唯一索引访问路径来决定对这个输出或修改操作的键值是否重复。如果访问路径在回送时仍是打开的，那么只考虑有*RBLD 和*DLY 维护的索引路径。

如果处理有索引的 COBOL 程序，可以指定重复键值通过 COBOG 语言回送给程序或用 OPNDBF 或 OPNQRYF 命令，但是在 COBOL 中用重复键值回送可导致性能下降。

2.1.3 数据的恢复和完整性

下面将解释运行时数据完整性考虑。

2.1.3.1 用日志和落实控制保护文件

COMMIT 参数。

日志和落实控制是在 AS/400 上对数据和交易恢复使用的方法。数据库文件的日志功能用启动物理文件日志命令（STRJRNPF）来执行，对访问路径的日志是对文件用启动访问路径日志命令（STRJRNAP）或使用系统管理访问路径保护（SMAPP）来实现。可以通过启动落实控制命令（STRCMTCTL）或在通过高级语言告诉系统文件将在落实控制运行，也可以在打开文件（OPNDBF）和 OPNQRYF 中规定 COMMIT 参数。详细信息请见第十三章“数据库恢复的考虑”及《备份恢复高级手册》。如果要对与用约束与删除规则、更新规则或两者都有但不是 RESTRICT 相关的文件进行插入、更新或删除时，则必须用日志，详情请参阅第十六章“引用的一致性”。

2.1.3.2 往辅存中写数据和访问路径

FRCRATIO 和 FRCACCPTH 参数。

通常 AS/400 集成数据管理系统决定何时从主存向辅存写入修改的数据。如果你要来控制，可以在生成、修改和覆盖数据库文件命令中用强制写入频率（FRCRTIO）参数，在生成修改数据库文件的命令中用强制访问路径参数（FRCACCPTH）。使用 FRCRTIO 和 FRCACCPTH 参数应考虑对系统性能和恢复的影响。了解这些内容见第十三章“数据库恢复的考虑”。

2.1.3.3 检查对记录格式描述的修改

LVLCHK 参数。

在打开文件时，系统检验所用的文件记录格式自编译后是否改过，以至于程序不能处理文件。正常的系统会将上述情况通知程序，这种情况叫做级别检验。当使用生成或修改文件命令时，可以指定要做这个级别检验，也可以在 OVRDBF 命令中用 LVLCHK 参数覆盖文件级别检查。有关这个参数的详情，请参阅 3.2.1 节“在文件描述中修改字段”。

2.1.3.4 检验文件的有效日期

EXPDATE 和 EXPCHK 参数。

系统会校验在文件中指定的日期是否仍是当前的，你可以在生成和修改文件的命令中用 EXPDATE 指定终止日期，而且可以在覆盖文件命令（OVRDBF）中用 EXPCHK 参数指定系统是否检验日期。如果指定了终止日期，并当天日期大于终止日期，在文件打开时，会给

系统操作员发送一条消息。

2.1.3.5 防止作业修改文件中的数据

INHWRT 参数。

如果要测试程序，但又不想在测试中实际修改文件中的数据，那么可以告诉系统防止程序对文件的任何修改。为了防止修改文件，可以在覆盖文件命令（OVRDBF）中指定INHWRT(*YES)参数。

2.1.4 在交叉作业间共享数据库文件

通过定义，所有数据库文件可以在同一时间被许多用户使用，但某些操作可以锁住文件、成员、成员中数据记录，以防止它们在交叉作业中被共享。

每个数据库命令或高级语言程序都能分配它使用的文件、成员和数据记录，根据操作要求，其它用户就不能使用已分配的文件、成员和记录。对逻辑文件或其成员的操作，能根据逻辑文件依据的数据或访问路径分配文件或成员。

例如，打开逻辑文件可以分配该文件所基于的物理文件成员的数据。如果程序更新逻辑文件成员，在同一时间，另外的用户就无法请求得到使用这个逻辑文件清除数据所用的物理文件成员。

有关数据库功能及它们对文件的加锁类型的情况，请参阅附录 C“数据加锁考虑”。

2.1.4.1 记录锁定

WAITRCD 参数。

AS/400 具有记录内部的完整性机制。例如，如果 PGMA 读并更新一个记录，它对该记录加锁，另一个程序就不能读同一记录来更新，直到 PGMA 释放该记录。但另一个程序可以为了查询而读该记录，通过这种办法，系统确保数据库完整性。

系统根据程序中文件处理类型和操作要求来决定加锁条件。例如，如果打开文件要更新和删除，每读一条记录都加锁，这样任何一个其它用户可以同时读，但仅有一个用户可以更新记录。

系统通常会来锁住记录，等待一定的时间，时间过了之后，向程序发出无法获得所请求记录的消息，缺省的记录等待时间为 60 秒；但可以在生成和修改文件命令或覆盖文件命令中指定 WAITRCD 参数来自定时间，如果程序被告知所要求记录被另一操作锁住，可以让程序做相应的动作（如，可以向操作者发送一条当前无法获得记录的消息）。

系统在记录更新或删除后自动解锁，但也可以没有更新就解锁，关于如何解锁，请参阅高级语言手册。

注：使用落实控制可修改记录加锁规则，详情参阅《备份和恢复高级手册》。

可以用显示记录锁命令（DSPRCDLCK）来显示物理文件成员当前记录加锁状态（等待或挂起），这个命令同时也指出当前加锁类型。（详情参阅“备份和恢复高级手册”）。根据所指定的参数，这个命令可以显示指定记录的加锁状态或成员中所有记录的加锁状态，也可以通过处理作业（WRKJOB）屏来显示记录锁。

可以用检查记录加锁的命令（CHKRCDLCK）来判定当前作业是否有记录锁，这条命令返回一条是否有记录加锁的信息（可以监控）。当在使用成组作业时，这条命令很有用。例如：在往另一组作业做传输出前，要检查是否有加锁记录，当判定确定有锁程序要解锁。

2.1.4.2 文件锁定

WAITFILE 参数。

有些文件操作是独占式的分配文件，在文件被独占分配期间，其余任何企图打开文件的程序都要等到文件被释放后再进行。可以通过在生成、修改文件和覆盖数据文件命令中的WAITFILE 参数来指定文件允许使用前要等待的时间。如果没有指定等待时间，系统的缺省值是零秒。

一个文件在修改属性的操作时被独占分配，这些操作（如移动、改名、分配或撤消权限、修改所有者或删除），不允许另一个对同一文件或成员在同一时间做操作，其它文件操作（如显示、打开、DUMP 或检验）仅使用文件定义，这样对文件很少独占性，它们可以在同一时期对一个成员做各自操作或输入输出操作。

2.1.4.3　成员锁定

对成员的操作（如增加和移出）自动地独占分配文件，这样可以防止对同一成员的其它操作，对同一成员输入/输出操作不能同时进行，但对同一文件中的其它成员允许做输入/输出操作。

2.1.4.4　记录格式数据的锁定

RCDFMTLCK 参数。

如果要锁住一个记录格式中的所有记录，（例如一个物理文件中所有记录）可以在OVRDBF 命令中用 RCDFMTLCK 参数。

2.1.5　在同一作业或活动组中共享数据库文件

SHARE 参数。

由缺省规定数据库管理系统允许一个文件同时可由多个用户读或修改，也能在打开数据库文件时做到：

在同一程序中使用多次

在同一作业或活动组中由不同程序使用

注：关于在集成语言环境中共享的详情，请参阅应用开发手册。

在生成、修改、覆盖文件命令中的 SHARE 参数允许在同一作业或活动组中共享，包括共享文件、属性、位置和存储区，共享文件由于减少主存要求和减少打开、关闭文件的时间，从而改善程序的性能。

用参数 SHARE(*YES)使同一作业或活动组中多个程序共享打开数据路径（ODP）。打开数据路径是指可以通它完成对文件的所有输入/输出操作，就象程序和这个文件联结起来一样。如果没有指定 SHARE(*YES)参数，每次文件打开时，就生成一个打开数据路径。如果在同一作业或活动组中文件打开多次，可使用当前打开文件的 ODP，而不必新建打开数据路径。这样就减少了第一次打开文件后再次打开的时间，并减少对主存储区的要求。在第一次打开文件时必须指定 SHARE(*YES)，以后对同一文件的打开就共享路径，性能好的程序设计通常都在同一作业或活动组中共享在多个程序中打开文件的路径。

指定 SHARE(*NO)告知系统对文件不共享打开的路径。

通常这个规定只用在对那些很少使用或某个程序唯一处理的文件中。

注：在高级语言程序中处理打开关闭就象文件是共享的一样，不能在高级语言程序中指定文件共享。要通过 SHARE 参数指定，而 SHARE 参数只能在生成、修改和覆盖文件命令中指定。

2.1.5.1　在同一作业或活动组中打开共享文件的考虑

当在作业或活动组中打开共享文件时考虑以下几项：

请确认在作业或活动组中第一次打开共享文件时，所有后续文件打开操作的功能是否符合第一次打开共享文件的规定，如果不符合就会向程序发送错误信息。（可以用 OPNDBF 或 OPNQRYF 命令参数或修改程序，去掉不匹配项来改正这个错误）。

例如，PGMA 是打开 FILE1 的第一个程序，且只读这个文件，PGMA 调用了 PGMB、PGMB 中要对共享文件进行删除记录，因此，PGMA 为让 PGMB 进行删除，它必须象自己要进行删除一样打开文件，那么在高级语言中要做正确地规定，(为做到这点在某些高级语言中可以使用文件操作语句，但不运行它，详情见高级语言手册)，也可以在 OPNDBF 和 OPNQRTF 命令的 OPTION 参数中指定。

有时在同一作业或活动组中使用共享文件并不合适。例如，一个程序要用到达顺序从文件读记录，而另一个需用索引顺序，这种情况下，不能共享打开数据路径，要在覆盖文件命令（OVRDBF）中指定 SHARE(*NO)，以保证文件不能共享。

如果在第一次打开产品库中一个共享文件后，进入 UPDPROD(*NO)的调试模式，那么后续共享的对此文件的打开，共享原始的打开数据路径，并允许修改文件。为防止这些，在打开文件调试前，用 OVRDBF 命令中的 SHARE(*NO)参数。

对第一次打开共享文件使用落实控制，要求后续打开都要用落实控制。

键字反馈（Key feedback），插入键字反馈或复制键字反馈，如果它们类型需要有后续共享打开文件，那它们必须规定为完全打开。

如果在程序中或在 OVRDBF 中未指定库名，则用*LIBL 参数，且系统假定在上次用 *LIBL 打开共享文件后，库列表未改动过。如果库列表修改过，要用 OVRDBF 命令去指定库名，确保打开正确的文件。

在完全打开中指定的记录长度是后续共享打开记录的长度，即使在共享打开文件中规定的是较长的记录长度时也是这样。

要处理的是共享文件第一次打开时指定的覆盖和程序规范，在后续打开指定的覆盖和程序规范，除了修改文件名或 OVRDBF 命令中 SHARE 或 LVLCHIC 参数的值，它们都被忽略。

用 OPNQRYF 命令对第一次打开文件规定的覆盖，可以用来修改将用打开查询文件命令处理的文件、库、成员的名字，在 OVRDBF 中规定参数值，除去 TOFILE、MBR、LVLCHK 和 SEQONLY 外都被 OPNQRYF 忽略。

打开数据库文件（OPNDBF）和打开查询文件（OPNQRYF）命令，可以依据以下几种情况把 ODP 的作用域改变成打开作用域（OPNSCOPE）参数中设定的级别：

—系统先在活动组中检索共享路径，然后再到作业中检索。

—改变作用域为一个活动组范围的共享打开不一定能在不同活动组间共享。

—改变作用域为在作业中共享的打开可以在同一时间被任何数目的活动组共享，并贯穿作业始终。

CPF4123 诊断信息给出发生在完全打开和后续共享打开间的不匹配。这些不匹配不会使共享打开文件失败。

注：OPNQRYF 命令不共享在作业或活动组中已存在的打开数据路径。如果在作业或活动组中已经存在一个与 OPNQRYF 中指定的相同文件名、库名、成员名的打开数据路径，系统会发出一条错误信息，且不打开查询文件。

2.1.5.2　在同一作业或活动组中共享文件的输入/输出

在同一作业或活动组中处理一个共享的数据库文件应考虑以下几点：

- 因为一个共享文件只有一个 ODP，所以在作业或活动组中共享的文件对所有程序来讲，只管理一个记录位置。如果一个程序对一个记录用读或读且更新操作确定位置，这时又调用了共享此文件的另一个程序，当被调用程序返回到调用程序的记录位置

可能已经移动或释放了记录锁，由于位置移动或不定的加锁状态，在调用程序时可能出错。在共享文时，应当重新确定记录位置及锁状态，以管理记录位置和记录锁。

- 当共享文件第一次为更新而打开时，没必要为每个后续程序申请记录加锁。系统会根据每个程序使用的文件来决定加锁类型。系统在保证数据完整的情况下努力使加锁争用为最小。

例如，PGMA 是作业或活动组中第一个打开共享文件的程序。PGMA 要更新文件中记录，当程序为更新而读出记录时，它要锁住该记录。PGMA 这时调用了 PGMB，PGMB 也共享这个文件。但它不做更新，只是读数据，即使 PGMA 是做为更新来打开的共享文件。但 PGMB 是只读处理，对记录不加锁，这样系统能确保数据完整，使记录锁争夺为最小。

2.1.5.3　在同一作业或活组中共享文件的关闭

当在作业或活组中对共享文件关闭时考虑以下几点：

- 只有当使用共享打开数据路径的最后一个程序关闭文件时，才执行完整的关闭操作。(包括释放文件、成员、记录锁，强行修改辅存，撤消 ODP)。
- 用 CLOF 命令可以关闭由 OPNDBF 和 OPNQRYF 打开的文件。如果满足以下条件之一，可用回收资源命令（RCLRSC）来关闭由 OPNQRYF 打开的文件：
  —OPNSCOPE(*ACTGRPDFN)，并且只在缺省的活动组中要求打开。
  —规定 TYPE(*NORMAL)。
- 如果指定下面中的一项，即使用 RCLRSC 命令，文件仍保持打开：
  —OPNSCOPE(*ACTGRPDFN)，并且不是缺省活动组要求的打开路径。
  —OPNSCOPE(*JOB)
  —TYPE(*PERM)

2.1.5.3.1 关闭共享文件的例子

下面几个例子给出在同一作业中关闭共享文件要考虑的问题。

例 1　用类似的处理选项使用一组文件

在这个例子中，用户注册且大多数程序处理相同文件集。PGMA（CL 程序）是第一个程序（用来建立应用程序，包括覆盖和打开共享文件）。PGMA 然后把控制传给 PGMB，它显示程序菜单。假定在这个例子中，用到 A、B、C 三个文件，A、B 是共享文件，在建立 A、B 时用的是 SHARE(*NO)，因此，要在每个 OPNDBF 命令前用 OVRDBF 命令规定 SHARE 为(*YES)，C 生成时规定 SHARE(*NO)，程序中不共享它。

```
PGMA:   PGM       /* PGMA - Initial program  */
        OVRDBF    FILE(A) SHARE(*YES)
        OVRDBF    FILE(B) SHARE(*YES)
        OPNDBF    FILE(A) OPTION(*ALL) ....
        OPNDBF    FILE(B) OPTION(*INP) ...
        TFRCTL    PGMB
        ENDPGM



PGMB:   PGM       /* PGMB - Menu program */
```

```
        DCLF      FILE(DISPLAY)
BEGIN:  SNDRCVF   RCDFMT(MENU)
        IF        (&RESPONSE *EQ '1') CALL PGM11
        IF        (&RESPONSE *EQ '2') CALL PGM12
        .
        .
        IF        (&RESPONSE *EQ '90') SIGNOFF
        GOTO      BEGIN
        ENDPGM
```

在 PGMA 中打开的文件或者作用域限定在作业中，或者在属于同一个活动组的 PGMA、PGM11、PGM12 中，且文件打开路径的作用域限定在这个活动组中。

在这个例子中，假定：

PGM11 打开文件 A 和 B，因为这些文件是在 PGMA 中用 OPNDBF 共享打开的，因此减少了打开时间，当共享文件关闭时，关闭文件的时间也减少，OVRDBF 命令在控制传给 PGMB 时（PGMA 中的 TFRCTL 命令）也起作用。

PGM12 打开文件 A、B 和 C，A 和 B 已经做为共享文件打开了，因此打开时间减少，因为文件 C 只用在这个程序中，因此它不做为共享文件打开。

此例中，因为仅需要一组文件，因此不使用 CLOF 命令，当用户注锁时，文件自动关闭，它假定 PGMA（初始程序）仅在作业开始时调用。有关在 ILE 环境中怎样回收资源的内容，请看 ILE 概念的书。

注：PGMB 的 DISPLAY 也可定义共享文件，那么在后来用到它的程序中，当做为显示文件打开也可以改善性能。

在例 1 中，OPNDBF 命令是放在 PGMA 中的，这样，作业中的其它处理程序，能尽可能的有效运行，即作业中其它程序用到的重要文件都在 PGMA 中打开。打开之后，其余的处理程序（PGMB、PGM11、PGM12）都能共享这个文件，打开和关闭它们都能快一些，另外，把 OPNDBF 放在 PGMA 中而不放在 PGMB 中，PGMB 需要的主存也减少了。

在初始程序（PGMA）中可以使用复盖和打开命令，然后其它程序可以从作业中撤消它，但程序生成时打开数据路径都一直保持存在且作业中的其它程序都可以使用它。

注意 OVRDBF 命令的处理是与 OPNDBF 有关的。复盖命令必须在文件打开之前规定。OVRDBF 命令的某些参数在 OPNDBF 命令中也存在。如果有冲突，用 OVRDBF 的值。有关 ILE 环境中复盖作用的详细内容，请看 ILE 概念的书。

例2　用类似的处理选项使用多组文件

假定一个菜单要操作员指定某个应用程序来打开需要的文件，当应用程序结束时，用 CLOF 命令关闭文件，CLOF 也减少作业需要的主存空间。在这个例子中，每个应用程序使用不同的文件，用户在选择一个新的应用程序之前用常规时间正常工作。

帐号接收程序如下：

```
PGMC:   PGM       /* PGMC PROGRAM */
        DCLF      FILE(DISPLAY)
BEGIN:  SNDRCVF   RCDFMT(TOPMENU)
```

```
        IF      (&RESPONSE *EQ '1') CALL ACCRECV
        IF      (&RESPONSE *EQ '2') CALL ACCPAY
        .
```


Examples of Closing Shared Files

```
        .
        IF      (&RESPONSE *EQ '90') SIGNOFF
        GOTO    BEGIN
        ENDPGM



ACCREC: PGM     /* ACCREC PROGRAM */
        DCLF    FILE(DISPLAY)
        OVRDBF  FILE(A) SHARE(*YES)
        OVRDBF  FILE(B) SHARE(*YES)
        OPNDBF  FILE(A) OPTION(*ALL) ....
        OPNDBF  FILE(B) OPTIONS(*INP) ...
BEGIN:  SNDRCVF RCDFMT(ACCRMENU)
        IF      (&RESPONSE *EQ '1') CALL PGM21
        IF      (&RESPONSE *EQ '2') CALL PGM22
        .
        .
        IF      (&RESPONSE *EQ '88') DO /* Return */
                CLOF FILE(A)
                CLOF FILE(B)
                RETURN
                ENDDO
        GOTO    BEGIN
        ENDPGM


   TYPE(*PERM).
```

支付帐号菜单的程序很类似，但使用不同的一组 OPNDBF 和 CLOF 命令。此例中，文

件 A 和 B 用 SHARE(*NO)生成。这样，OVRDBF 必须放在 OPNDBF 命令之前。在例 1 中，为了减少作业所占用的主存，把 OPNDBF 命令放在另外的程序中再调用它。这个另外的程序也要有 CLOF 命令，OPNDBF 命令放在从菜单中调用的启动程序中，它把控制传给规定的应用程序菜单(在这个启动程序中要有复盖命令)。但这些功能调用的其它程序也使用系统资源，那么在每个应用程序菜单中包括 OPNDBF 和 CLOF 命令要好一些，这个例子就是这么做的。

另外的做法是在 PGMC 中用 RCLRSC 命令来代替 CLOF 命令，RCLRSC 命令关闭一些文件且释放文件和程序占用的存储空间，但它不关闭在 OPNDBF 或 OPNQRYF 命令中用下列参数打开的文件：

OPNSCOPE(*ACTGRPDFN)，且是在非缺省活动组中打开的。

OPNSCOPE(*ACTGRP)，如果 RCLRSC 命令是从一个活动组数低于打开的活动组数发出的，可以释放存储。

OPNSCOPE(JOB)

TYPE(*PERM)

下例给出用 RCLRSC 命令关闭文件的例子：

```
.
.
IF          (&RESPONSE *EQ '1') DO
            CALL ACCRECV
            RCLRSC
            ENDDO
IF          (&RESPONSE *EQ '2') DO
            CALL ACCPAY
            RCLRSC
            ENDDO
.
.
```

例 3　用不同处理选项使用一组文件

如果某些程序需只读处理，而其它程序需一些或全部的文件处理方式（输入/输出/更新/添加/删除），就要用下面的方法之一；如果文件在一些程序中使用特定参数而在另一些程序中不用，也可用这些办法。（如有时用到落实控制）。

一个打开数据文件命令（OPNDBF）可以规定 OPTION(*ALL)参数，ODP 将被共享地打开（如前一个 OVRDBF 命令用 SHARE(*YES)参数）。每个程序需要的打开类型都在程序中规定。在某些情况下，并不须特别考虑这点，因为一个规定只为输入打开的程序就象没有共享打开一样地处理（如在读记录时，不用额外的锁住记录）。

但 OPNDBF 命令的某些功能可以影响程序的操作，比如在程序的打开文件命令中指定了 SEQONLY(*NO)的选项。如果 OPNDBF 指定 SEQONLY(*YES)选项并且程序企图进行非顺序处理，就会发生错误。

ACCPTH 参数也必须与程序使用访问路径(到达或键字顺序)的方式相一致。

如果 OPNDBF 命令命令中规定了 COMMIT(*YES)并且 STRCMTCTL 命令规定 LCKLUL(*ALL)或 LCKLVL(*CS)参数，那么每个读记录操作都对记录加锁（落实控制的记

录锁定规则），这可能导致不期望，并出错。

两个 OPNDBF 命令可对同一数据使用（例如，一个规定 OPTION(*ALL)，另一个是 OPTION(*INP)参数），那么第二个使用的必须是针对同一物理文件的逻辑文件，然后，这个逻辑文件可以用 SHARE(*YES)打开,并为同一作业中的多个用户使用。

2.1.6 顺序处理

SEQONLY 和 NBRRCDS 参数。

如果程序对一个数据文件只作顺序的只输入或只输出操作，可以在 OPNDBF 中或 OVRDBF 中规定 SEQONLY 参数来改进性能。要使用 SEQONLY 操作，文件必须以只输入或只输出方式打开，NBRRCDS 可以和任意打开选项连用。（只要可能，[OPNDBF]都使用顺序处理），依据高级语言的规定，可使用顺序处理为缺省方式。例如，如果用只输入打开文件，并在程序中规定顺序读操作，那么高级语言就会自动请求顺序操作。

注：文件定位操作不认为是顺序读操作。因此，包括定位操作的高级语言程序就不会自动的做为仅顺序处理。（RPG/400 中的 SETLL 和 COBOL/400 中的 START 操作就是文件定位的例子）。虽然高级语言程序不能自动申请顺序操作，但你可以在 OVRDBF 命令中指定 SEQONLY 参数来请求。

如果使用仅顺序处理，同时可指定移动的记录的数目做为系统数据库主存区和作业内部主存区间传递的一个单位。如果没有指定这个数目，那么系统就以适合 4096 字节缓冲的记录数来计算。

系统同时提供控制在主辅存间作为一个单位传递的记录数。如果用记录物理存贮的顺序来读取数据，可以用 OVRDBF 中的 NBRRCDS 来改进作业性能。

注：顺序处理不能用在键字顺序文件中，除非物理数据和访问路径的顺序是一致的。在物理数据重新组织成访问路径顺序之前，SEQONLY(*YES)将导致低下的应用性能。

2.1.6.1 顺序处理的打开

下面是用顺序处理打开文件的考虑。如果系统确定不能用仅顺序处理，就会发出一个程序信息，指出不接受顺序处理，但文件仍被打开。

- 如果程序打开只输出成员，并且如果已指定 SEQONLY(*YES)（记录数没指定），并且打开的成员或者是一个逻辑成员，或是一个唯一键字的物理成员，或者物理成员已有访问路径，SEQONLY(*YES)被改成 SEQONLY(*NO)，这样程序可以处理在输出时的任何错误（如重复键字，映象转换和选择/忽略的错误）。如果你要系统运行仅顺序处理，在 SEQONLY 参数中规定*YES 和记录数两项。
- 仅顺序处理只能为只输入（读）或只输出文件指定，如程序有更新或删除操作，系统不允许仅顺序处理。
- 如果文件为输出打开，它必须是物理文件或基于一个物理成员的逻辑文件。
- 只有成员是为只输出打开时,落实控制可一起用于文件打开,如作业中有返回(rollback)操作，那个在返回操作时恰在作业存储区中的记录不会到写入系统存储中，也不会出现在落实控制的日志中。如果作业中没有返回操作，那么整个落实控制的交易在日志中没有反映。
- 对于只输出操作，作为一个单位移出的记录数和强制比率要做比较，且在必要时调整。如果记录数大于强制比率，记录数降到强制比率，若相反情况，强制比率降至记录数。
- 如果程序为只输出打开成员，并指定 SEQONLY(*YES)（记录数未指定），而且又需要重复或插入键字反馈操作，则 SEQONLY(*YES)变为 SEQONLY(*NO)，那么可在

有记录插入文件时，对按记录顺序操作提供一个反馈。

• 如果下面各项都符合，块中的记录数将变为 1：

－成员为只输出打开

－对指定仅顺序处理没做有效的操作

－由于记录增加数为零，使打开的文件不能增大

－文件中可用的字节数小于记录块所要求的字节数

若没有指定仅顺序，并且用 OPNQRYF 打开文件时，要考虑下面的情况：

如果下面条件都满足，就会发送一个信息，指出将使用仅顺序处理，查询文件打开：

• 如果 OPNQRYF 在 GRPFLD 参数中指定一个或多个字段,或 OPNQRYF 需成组处理。

• 如果 OPNQRYF 指定一个或多个字段，或 UNIQUEKEY 参数中指定*ALL。

• 如果在 SQL 的 SELECT 语句中指定 DISTINCT 选项使用窗口，则自动使用 SEQONLY(*YES)。

关于 OPNQRYF 命令详细内容参阅 2.2.3。

2.1.6.2 顺序处理的输入/输出

在文件使用仅顺序处理进行输入/输出操作时，考虑下面情形：

• 输入时，程序每次从输入缓冲中接收一个记录，输入缓冲中所有记录处理后，系统自动读下一组记录。

注：在记录读入输入缓冲后的改变不会在输入缓冲区中反映出来。

• 输出时，程序必须向输出缓冲中一次写一个记录，当输入缓冲区写满后，系统自动把记录加到数据库中。

注：如果使用日志，整个缓冲一次一起写入日志，就象整个是一个逻辑整体一样，在记录加到数据库之前处理日志。

• 如果用仅顺序处理输出，可能看不到发生的所有改变。例如，对 PGMA 使用的文件规定仅顺序处理，PGMA 往文件中加记录，且 SEQONLY 参数指定缓冲区的记录数为 5，只有当缓冲被新加的记录填满，才写到数据库中。在此例中，只有第五个记录进到缓冲区，另一组的五个记录才传到数据库，才能被系统中的其它作业使用。

• 用仅顺序处理输出，如果不能处理从缓冲区向文件写记录中发生的错误，某些附加数据可能不能写入数据库。例如，假定缓冲区有五个记录，第三个记录有与文件中另一记录相同的键字，而文件指定为唯一键字，这种情况下，当系统向数据库传送缓冲区时，只写入前两个记录，第三个则发生重复键字错误，由此，缓冲区中的第三、四、五个记录都不能写入数据库。

• 对输出操作可用强制结束数据功能，强制缓冲区所有记录写入数据库（除了上面说引起重复键字的记录），这一功能只在某些高级语言可以使用。

若下面条件符合，块中的记录数将改为 1：

－成员为只输出打开或仅顺序处理

－仅顺序操作中覆盖命令无效

－由于记录增加数为零，没有增大打开的文件

－文件中可用字节数小于块中记录的字节数

2.1.6.3 顺序处理的关闭

当指定为仅顺序处理的文件被关闭时，所有输出缓冲中的记录写入数据库，但如果一个记录有错误，其后的记录都将不能写入数据库。

如果同一个作业中有多个文件共享一个仅顺序输出文件，一直到最后一个关闭操后，输

出缓冲区才清空。因而一个关闭操作（不是作业中最后一个关闭操作），不会使仍在缓冲中的记录写入数据库。

2.1.7 运行时问题总结

下表列出了控制程序使用数据库成员的参数，并指出参数使用的地方。对那些可以多处使用的参数，系统连接起这些值。OVRDBF 命令参数优于程序参数。OPNDBF 和 OPNQRYF 命令的参数优于生成或修改文件参数。

注：任何覆盖参数（不包括 TOFILE、MBR、LVLCHK、SEQONLY、SHARE、WAITRCD、INHWRT）都被 OPNQRYF 忽略。

表 5-1　在 CL 命令中规定的数据库处理选项

说　明	参 数	CRTPF 或 CRTLF 命令	CHGPF 或 CHGLF 命令	OPNDBF 命令	OPNQRYF 命令	OVRDBF 命令
文件名	FILE	X	X(1)	X	X	X
库名		X	X(2)	X	X	X
成员名	MBR	X		X	X	X
成员处理选项	OPTION			X	X	
记录格式锁状态	RCDFMTLCK					X
打开后起始文件位置	POSITION					X
程序作仅顺序处理	SEQONLY			X	X	X
忽略索引顺序访问路径	ACCPTH			X		
等待文件锁时间	WAITFILE	X	X			X
等待记录锁时间	WAITRCD	X	X			X
保护覆盖	SECURE					X
从辅存往主存传送的记录数	NBRRCDS					X
与其他程序共享打开数据路径	SHARE	X	X			X
格式选择	FMTSLR	X(3)	X(3)			X
强制频率	FRCRATIO	X	X			X
禁止写	INHWRT					X
记录格式级别检查	LVLCHK	X	X			X
超期检查	EXPCHK					X
有效日期	EXPDATE	X(4)	X(4)			X
强制访问路径	FRCACCPTH	X	X			
落实控制	COMMIT			X	X	
延迟文件结束	EOFDLY					X
重复键字检查	DUPKEYCHK			X	X	
重用删除的记录空间	REUSEDLT	X(4)	X(4)			

编码字符级标识	CCSID	X(4)	X(4)			
分类顺序	SRTSEQ	X	X		X	
语言标识	LANGID	X	X		X	

（1）文件名：CHGPF 和 CHGLF 命令仅用文件名作标识，你不能修改文件名。

（2）库　名：CHGPF 和 CHGLF 命令仅用库名作标识，你不能修改库名。

（3）格式选择：仅在 CRTLF 和 CHGLF 命令中使用。

（4）有效日期，重用删除的记录和编码字符集标识：仅在 CRTPF 和 CHGPF 命令中使用。

5-2　在程序中规定的数据库处理选项

说明	RPG/400 语言	COBOL/400 语言	AS/400 BASIC	AS/400 PL/I	AS/400 Pascal
文件名	X	X	X	X	X
库名			X	X	X
成员名			X	X	X
程序记录长度	X	X	X	X	X
成员处理选项	X	X	X	X	X
记录格式锁状态			X	X	
程序要使用的记录格式	X		X		
清除物理文件成员记录		X	X		X
程序作仅顺序处理	X	X		X	X
忽略索引顺序访问路径	X	X	X	X	X
与其他程序共享打开数据路径				X	X
记录格式级别检查	X	X	X	X	
落实控制	X	X		X	
重复键字检查		X			

注：控制语言参数也能规定上述这些参数，详细信息看图 5-1。

2.1.8 存储池分页选项对数据库性能的影响

共享池的分页对读及修改数据库文件的性能有显著影响。

*FIXED 的分页选项可使程序对内存的要求较小：

－用更小的块从辅存向主存中传送数据

－频繁的向辅存中写入文件的修改（更新已存在记录或添加新记录）

这个选项允许系统更多的象未规定分页选项时一样处理。

*CALC 分页选项改进程序读取或更新数据的性能，在共享池中，有足够内存可以使用时，程序可以：

－从辅存中传送更大的数据块

－向辅存中写入修改的数据

对数据文件的分页操作，根据使用的文件和可用内存是很随机的。经常被引用的文件比不常访问的要驻留更长时间。内存有时象常用数据的缓冲，利用*CALC 的分页选项可以降低 I/O 操作的总数。

分页选项的详细内容，参阅工作管理手册。

第六章　打开数据库文件

这一章将讨论打开数据库文件的操作，还介绍 OPNDBF 和 OPNQRYF 命令。

2.2.1 打开数据库文件成员

在程序中使用数据文件，程序中必须有打开文件操作。在一些语言中，如果不能规定打开操作，它会自动打开，如果在程序或 OVRDBF 中不能指明成员名，将使用文件中的一个成员（按生成日期和时间定义）。

如果指定成员名，文件名正确，但没有成员名，文件被忽略。如果在不同的库中有多个名为 FILEA 的文件，那么打开的是在库列表中第一个符合的库中成员。例如，在库列表中有三个库 LIB1、LIB2、LIB3 都包含 FILEA，但只有 LIB3 中的 FILEA 含有要求的 MBRA，那么打开 LIB3 的 FILEA 中的 MBEA，而忽略 LIB2、LIB3。

在找到成员后，系统把数据库文件与程序相连，此后程序就可以对文件进行输入/输出操作。有关在高级语言程序中打开文件的详细内容，请参阅相应的语言指南。

可以用高级语言的语句打开文件，也可以用 CL 命令 OPNDBF 和 OPNQRYF。OPNDBF 在作业初始程序打开共享文件很有用，OPNQRYF 命令提供的信息处理需要的数据。

2.2.2 使用打开数据库文件命令（OPNDBF）

通常在使用 OPNDBF 时，可以使用参数的缺省值。对下面的参数，某些情况下可以不用缺省值而自己指定：

OPTION 参数：如果程序用到仅输入处理（不用更新的只读操作），指定*INP 选项，这允许系统可以不用对每个读记录加锁。如程序用仅输出时（向文件中写记录而不用读或更新），指定*OUT 选项。

注：如程序不对活动记录做直接输出操作（用相关记录号更新），必须用*ALL 代替*OUT。如程序仅对删除记录做直接输出操作，必须指定*OUT。

MBR 参数：如要打开的不是文件的第一个成员，就必须指定成员名，或在 OPNDBF 前使用 OVRDBF 命令。

注：在随后的程序中必须在 OVRDBF 中指定要打开的成员名。

OPNID：如果用标识符而不是文件名，必须指定这个参数。其它处理文件的 CL 命令中可用打开标识符，如 CLOF 用标识符指定要关闭的文件。

ACCPTH：如文件中用键字访问路径，并且打开选项是*OUT 或打开选项是*INP 或*ALL，但程序不用键字访问，就可在 OPNDBF 中指定 ACCPTH(*ARRIVAL)。忽略键字访问顺序，可改进作业性能。

SEQONLY：如后续程序按顺序处理文件，指定*YES,这个参数可指定在系统和程序数据缓冲传递的记录数。在 OPNDBF 中不能规定 SEQONLY(*YES),除非同时规定 OPTION(*INP)或 OPTION(*OUT)。除非物理数据按访问路径存储，否则对键字访问路径不允许用仅顺序处理。

COMMIT：如程序中用落实控制，要规定*YES。如果指定*YES，程序一定要在落实环境中运行（执行 STRCMTCTL），否则 OPNDBF 失败。如果程序不用落实控制要用*NO 的缺省值。

OPNSCOPE：这个参数指定 ODP 作用域。如果 ODP 请求是来自缺省的活动组，就规定*ACTGRPDFN，这时，ODP 的作用域是发出命令的程序的调用级。如果请求来自其他活

动组，ODP 的作用域是活动组级。指定*ACTGRP 使 ODP 可以用在相应活动组级。如要 ODP 限定在作业级指定*JOB，如果同时指定这个参数和 TYPE，就会收到错误信息。

DUPKEYCHK：指定是否要重复键字反馈。如指定*YES，重复键字反馈就在 I/O 操作中回送，如指定*NO，就不回送。如果程序不是用 COBOL/400 或 C/400 写的或 COBOL 和 C 程序中不用 重复键字反馈信息，就用缺省值(*NO)。

TYPE：指定在应用程序中如果发生没监控的例外情况，要采取的动作，如指定为 *NORMAL 时，可能发生如下情况：

你的程序可以使用 RCLRSC 命令来关闭在调用堆栈中比发出 RCLSRC 命令的程序更高的级别打开的文件。

你用的高级语言也可完成关闭操作。

如果想不用再次打开文件来继续运行程序，指定*PERM，如果发生下面情况，TYPE(*NORMAL)使文件关闭：

程序接收到错误信息

文件在调用堆栈中的更高级上打开

TYPE(*PERM)允许在收到错误信息后，文件仍然打开，如已指定 OPNSCOPE，不要指定这个参数。

2.2.3 使用打开查询文件命令（OPNQRYF）

OPNQRYF 允许对数据库文件进行多种数据处理。本质上讲，OPNQRYF 是做为处理程序与数据库记录之间的过滤器，数据库文件可以是物理文件或逻辑文件。

不象 CRTPF 和 CRTLF 命令生成的数据库文件，OPNQRYF 命令只生成处理数据的临时文件而不是永久文件。

要理解 OPNQRYF，首先要了解物理文件、逻辑文件、键字段、记录格式及连接逻辑文件的概念。

OPNQRYF 有与 DDS，CRTPF 和 CRTLF 命令中相似的功能，DDS 要求源语句和生成文件的独立步骤。OPNQRYF 允许不用 DDS 动态定义，OPNQRYF 不支持所有 DDS 功能，但有一些明显超过 DDS 功能。

另外，Query/400 可以完成某些 OPNQRYF 功能，但 OPNQRYF 命令更有用。

OPNQRYF 命令的参数也有类似 SQL 中 SELECT 语句的功能。如，FILE 类似于 SQL 中 FROM、QRYSLT 类似 SQL 中的 WHERE，GRPRLD 类似于 SQL 中的 GROUPBY，GRPSLT 类似于 HAVING。有关 SQL 详情参阅 DB2/400 SQL 编程指导。

下面列出 OPNQRYF 的主要功能，每个功能在以后都要详述：

动态记录选择
动态索引顺序访问路径
在连接中的动态索引顺序访问路径
动态连接
处理在次连接文件中没有的记录
唯一键字处理
映象字段定义
分组处理
仅处理最终总计
改进性能
打开查询标识
顺序执行排序

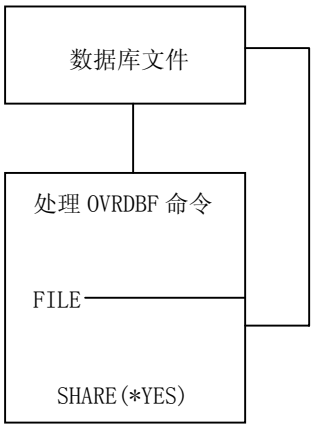
要理解 OPNQRYF 命令还要熟悉它的两种处理方式：使用文件中的格式和使用与文件中不同的格式。OPNQRYF 的典型用法是选择、安排和格式化数据，以使高级语言程序可以连续读取数据。

OPNQRYF 命令的句法和参数，参看 CL 参考手册。

2.2.3.1　使用文件中已存在的记录格式

假设程序只处理 code 字段等于 D 的记录，生成程序时如同在 code 字段仅有 D 的记录。也就是说，在程序中不需要任何选择操作，然后执行 OPNQRYF 命令，规定只有 code 字段为 D 的记录返回给程序。OPNQRYF 做记录选择，程序只处理符合选择值的记录，可以用这种方法选择一组记录，并以相同或不同于记录存储的顺序返回给程序。

下面是用 OPNQRYF 命令进行选择和排序记录的例子：



处　理
OPNQRYF
命　令

1.生成高级语言程序来处理数据库文件，就象用外部描述数据的一般生成过程一样，并且必须已存在于文件中。

2.运行 OVRDBF 命令，指定要处理的文件和成员名，且 SHARE(*YES)(如果成员永久修改为 SHARE(*YES)，并且只使用第一个或唯一一个成员的话，可不做这步。）

OVRDBF 可在 OPNQRYF 之后运行，除非想覆盖 OPNQRYF 中指定的文件名。在这个例子中，OVRDBF 先出现，同时使用 OPNQRYF 和 OVRDBF 时有一些限制和 MBR(*ALL) 导致错误信息，并且不能打开文件，详细信息参考 2.2.3.30。

3.运行 OPNQRYF，指定数据文件、成员、格式名、选择选项、顺序选项及打开文件的作用域。

4.调用步骤 1 中生成的程序，除用高级语言外，也可用 CPYFRMQRYF 来处理由 OPNQRYF 生成的文件，其他 CL 命令（如 CPYF 和 DSPPFM）和工具（如 QUERY）不能使用 OPNQRYF 生成的文件。

5.除非想让步骤 3 中打开的文件保持打开，否则关闭上述文件，可用 CLOF 命令。

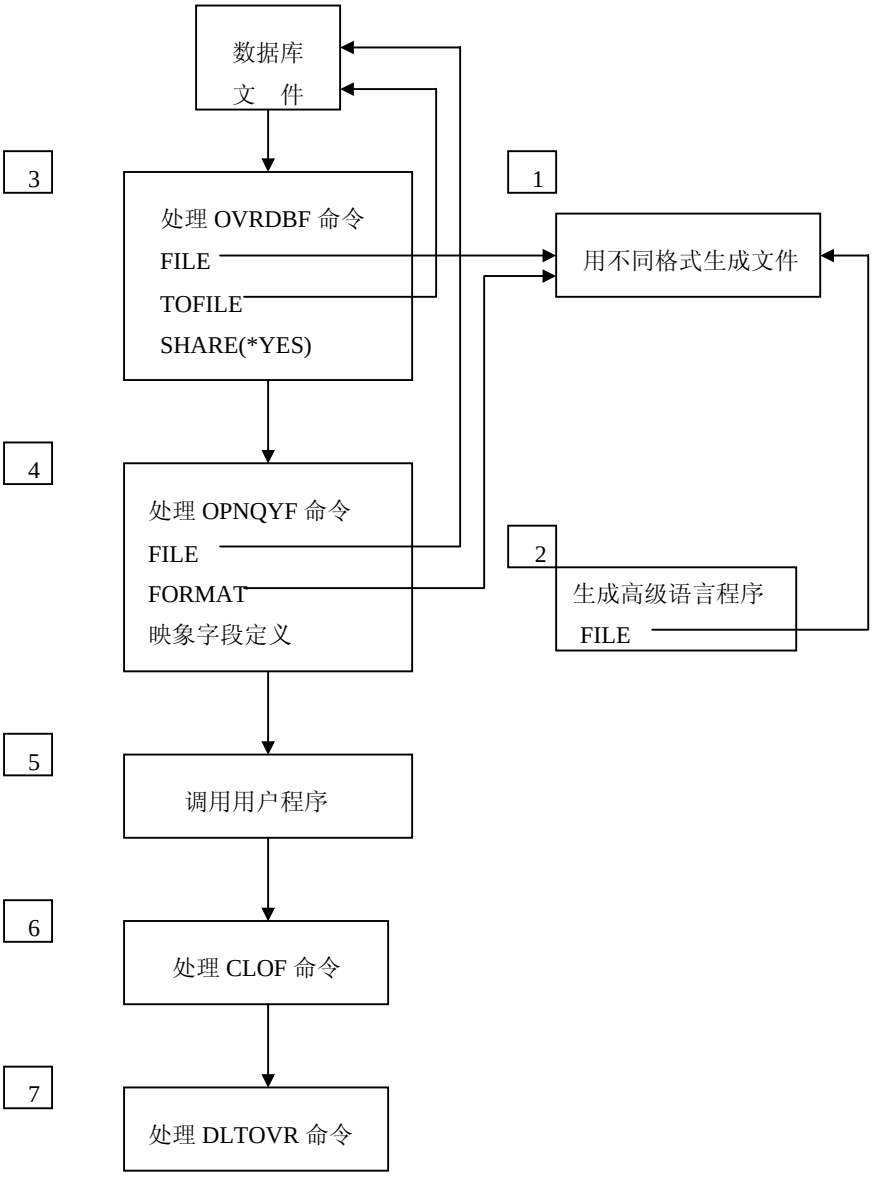
6.用 DLTOVR 命令来删除步骤 2 中的规定的覆盖，这步不一定都需要，但为了保持一致，在所有例子中都给出这步。

2.2.3.2　使用不同的记录格式的文件

使用 OPNQRYF 的高级功能（如，从不同文件中动态连接记录），必须定义一个有不同

记录格式的文件，这个新文件与要处理的文件相独立。它应有要用 OPNQRYF 生成的字段，这个功能还允许你定义当前不在数据库中，但可用它们构成的字段。当编制高级语言程序时，要规定不同记录格式的文件名，这样程序才能处理已有的外部描述字段和构成字段。

在调用高级程序前，一定要先用 OVRDBF 告诉程序打开查询的文件名，在 OPNQRYF 命令中，要规定数据库文件和程序中用到的有特别格式的新文件。如果你查询的文件没有 SHARE(*YES)的规定，要在 OVRDBF 中规定 SAHRE(*YES)。



1.用不同记录格式指定 DDS，并生成文件。这个文件包含程序中要用到的字段，通常数据不在这个文件中，它不需要成员，可以正常的生成不带键字的物理文件，可用一个字段引用文件来说明字段。记录格式名可以不同于数据库文件中的记录格式名，可以用任何数据库或 DDM 文件实现这个功能。这个文件可以是逻辑文件，并可是索引文件，可以有一个或多个成员，有或没有数据都行。

2.生成能处理步骤 1 中文件的高级语言程序，在这个程序不用包含数据的文件名。

3.执行 OVRDBF 命令，在 FILE 参数中指定带有新格式的文件名。

在 TOFILE 中指定要查询的文件名，可以在 MBR 中指定成员名。如要查询的成员不是

SHARE(*YES)，必须在 OVRDBF 中指定 SHARE(*YES)。

4.执行 OPNQRYF 命令，在 FILE 参数中指定要查询的数据库文件，在 FORMAT 中指定的步骤 1 中生成的新格式的文件名，在 OPNQRYF 中也可用映象字段定义，用来指出数据如何从数据库文件映象到步骤 1 生成的新格式中，同时还可以指定选择选项、顺序选项以及打开文件的作用域。

5.调用步骤 2 中生成的程序。

6.用 OPNQRYF SHARE(*YES)打开的步骤 4 中 FILE 参数的第一个文件现仍打开，必须用 CLOF 关闭。

7.删除步骤 3 中指定的覆盖。

上面的步骤给出用外部描述数据的一般流程,不一定要为每个 OPNQRYF 生成唯一 DDS 和记录格式名，可以重用已存在记录格式。但记录格式中的所有的字段必须都是数据库文件中的实际字段或用映象字段定义的。如果用程序描述的数据，可在任何时候生成程序。

可以用步骤 1 中生成的文件保存用 OPNQRYF 生成的数据。例如，可以用一个能把数据拷贝到不同记录格式的高级语言程序或用 CPYFRMQRYF 命令来代替步骤 5，不能用 CPYF。可以在步骤 5 后用 CPYF 和 Query 来做。

2.2.3.3 OPNQRYF 例子

下面几节介绍为实现上面提到的功能如何指定 OPNQRYF 参数以及在高级语言程序中使用 OPNQRYF 命令的方法。

注：

1.如果在命令行运行有 OPNSCOPE(*ACTGRPDFN)或 TYPE(*NORMAL)的 OPNQRYF 命令，在 OPNQRYF 成功运行后发生的错误信息使文件不能关闭。对 V2R3 版本的文件在使用 TYPE(*NORMAL)将关闭，除了 CPF0001 错误信息以外，当发生错误时系统会自动执行 RCLRSC 命令，但 RCLRSC 只关闭在缺省活动组中在调用堆栈级别中比它高的文件。

2.在运行了用 OPNQRYF 命令的做顺序处理后，文件位置通常是在文件末，如果你想运行同一个程序或不同程序但用相同文件，必须定位文件或关闭文件后用相同的 OPNQRYF 命令打开，也可用 POSDBF 命令来定位文件，在某些情况下，可以用高级语言程序的语句。

2.2.3.4 使用 OPNQRYF 命令的 CL 程序

OPNQRYF 有三个基本规则可避免编码错误：

1.不用(&)从文件中选择字段，字段在 CL 程序中用 DCL 或 DCLF 说明时要有&。

2.在 DCL 或 DCLF 中用单引号括上字段定义。（如，’&testfld’）

3.在比较字符字段时用双引号括上所有的参数，数字字段时用单引号。

下面的例子中，字段 INVCUS 和 INVPRD 定义为字符数据：

```
QRYSLT('INVCUS *EQ ″' *CAT &K1CUST *CAT '″ *AND +
        INVPRD *GE ″' *CAT &LPRD *CAT '″ *AND +
        INVPRD *LE ″' *CAT &HPRD *CAT '″')
```

如果字段定义为数字型，QRYSLT 参数如下：

```
QRYSLT('INVCUS *EQ ' *CAT &K1CUST *CAT ' *AND +
        INVPRD *GE ' *CAT &LPRD *CAT ' *AND +
```

```
|           INVPRD *LE ' *CAT &HPRD *CAT ' ')
```

2.2.3.5　零长度文字和包含(*CT)功能

零长度文字的概念曾在版本 2.1.1 中介绍过，在 OPNQRYF 命令中，零长度字符是在两个双引号间的空串，但还不是空格。

零长度字符使包含(*CT)功能做自变量的比较操作时，改变比较的结果，考虑语句：

QRYSLT('field *CT　" " ')

用零长度文字，它返回包含没有任何内容的记录，在本质上，它是任意个字符之间的比较，它等于：

'Field = %WLDCRD("**")'

在没有零长度文字支持之前(版本 2.1.1 之前)，自变量（" "）解释为一个单字节空格，这条语句返回包括一个空格的字段。在本质上它是任意个字符，接一个空格，再接任意个字符的比较，它等于：

'Field = %WLDCRD("* *")'

2.2.3.5.1　有零长度字符支持的原则

为了取得版本 2.1.1 之前具有包含功能的结果，必须象下面那样明显地包括空格：

QRYSLT('field　*CT "　"')

2.2.3.6　不用 DDS 选择记录

动态记录选择允许可以在不用 DDS 来请求一个记录子集。如可以选择有特定值或一定范围的记录（如，所有客户号在 1000 和 1050 之间的记录）。OPNQRYF 允许可以把这些和其它功能联合使用，产生很强的记录选择功能。

2.2.3.6.1　用 OPNQRYF 选择记录的例子

在下面所有例子中，假定处理单格式数据文件（物理或逻辑），OPNQRYF 的 FILE 参数允许指定多格式逻辑文件中的一个记录格式名。

参看 CL 参考手册，了解 OPNQRYF 命令中详细的 QRYSLT 能数格式及所用表达式。

例1　用特定值送记录

假定要从 FILEA 中选择所有 CODE 字段为 D 的记录，处理程序为 PGMB，PGMB 仅看到符合选择条件的记录（在程序中不必检测）。

注：可以先使用 OPNQRYF 命令的提示功能来找到所用参数。

象下面那样指定：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) QRYSLT('CODE *EQ "D" ')
CALL        PGM(PGMB)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

注：1.QRYSLT 参数全部表达式在要括在引号中。

2.在 OPNQRYF 命令中指定字段名时，记录中的名字不在引号中。

3.字符必须放在双引号或两个单引号中（例子中用双引号）。

在引号中字符大小写一定要符合要选择的数据库中选择的值（例子中都是大写）。

4.要请求一个数字型选择：

```
OPNQRYF     FILE(FILEA) QRYSLT('AMT *GT 1000.00')
```

注意：数字常数不用放在双引号内。

5.当字段的值与 CL 变量比较时，引号使用如下(只能用 CL 的字符变量)：

如选择一个字符、时间、日期或时间标记字段、规定：

```
OPNQRYF     FILE(FILEA) QRYSLT(''' *CAT &CHAR *CAT ''' *EQ FIELDA')
```

或用相反顺序：

```
OPNQRYF     FILE(FILEA) QRYSLT('FIELDA *EQ ''' *CAT &CHAR *CAT ''' ')
```

注意用引号或双引号括起 CL 变量和*CAT 操作符。

如选择数字字段：规定

```
OPNQRYF     FILE(FILEA) QRYSLT(&CHARNUM *CAT ' *EQ NUM')
```

或用相反顺序：

```
OPNQRYF     FILE(FILEA) QRYSLT('NUM *EQ ' *CAT &CHARNUM)
```

注意引号只括上字段和操作符。

当比较两个字段或常数时，数据类型必须匹配，下表列出有效的比较。

表 6-1　OPNQRYF 命令有效的数据类型比较

	任意数字	字符	日期（1）	时间（1）	时间标记（1）
任意数字	有效	无效	无效	无效	无效
字符	无效	有效	有效（2）	有效（2）	有效（2）
日期（1）	无效	有效（2）	有效	无效	无效
时间（1）	无效	有效（2）	无效	有效	无效
时间标记（1）	无效	有效（2）	无效	无效	有效

（1）　日期、时间和时间标记数据类型可以由字段或表达式来表示，不能用常量表示，但字符常量可以表示日期、时间或时间标记值。

（2）　字符字段或常量在与日期、时间、时间标记类型比较时，必须表示成有效的日期、时间、时间标记值。

注：有关 DBCS 信息参阅附录 B。

如果系统中一些文件使用键字顺序访问路径来选择字段，可以大大提高选择记录的性能，这样就使系统能快速访问仅符合条件的记录。如果没有这样的访问路径，系统不得不读每一条记录来判断是否符合选择值。

即使文件中存在访问路径，系统也不一定使用它。例如，如果系统用到达顺序访问路径处理更快些，就会这样做，详细内容，参见 2.2.3.26。

例 2. 用指定日期选择记录

假定要选择 Date 字段与当日期相同的所有记录，并且日期格式与系统格式相同，在 CL 程序中，可以指定：

```
DCL         VAR(&CURDAT)  TYPE(*CHAR)  LEN(6)
RTVSYSVAL   SYSVAL(QDATE) RTNVAR(&CURDAT)
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) QRYSLT('″' *CAT &CURDAT *CAT '″ *EQ DATE')
CALL        PGM(PGMB)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

CL 变量前加&不放在引号中，所有表达式括在引号中，操作符 CAT 和 CL 变量括在单双引号中。

了解数据库中的数据是定义为字符、日期、时间、时间标记或数字是很重要的。在上面例子中，日期字段被定义为字符型。

如果 DATE 字段定义为日期类型，上面例子应这样：

```
OVRDBF FILE(FILEA) SHARE(*YES)
OPNQRYF FILE(FILEA) QRYSLT('%CURDATE *EQ DATE')
CALL PGM(PGMB)
CLOF OPENID(FILEA)
DLTOVR FILE(FILEA)
```

注：日期字段不一定要和系统格式相同。

也可以象下面那样：

```
DCL VAR(&CVTDAT) TYPE(*CHAR) LEN(6)
DCL VAR(&CURDAT) TYPE(*CHAR) LEN(8)
RTVSYSVAL SYSVAL(QDATE) RTNVAR(&CVTDAT)
CVTDAT DATE(&CVTDAT) TOVAR(&CURDAT) TOSEP(/)
OVRDBF FILE(FILEA) SHARE(*YES)
OPNQRYF FILE(FILEA)
       QRYSLT('″' *CAT &CURDAT *CAT '″ *EQ DATE')
```

这里 DATE 是日期格式，作业缺省的日期格式为 MMDDYY，缺省的分隔符为(/)。

注：任何日期字符的表示都为：MMDDYY、DDMMYY、YYMMDD、或 JuLian 之一。

作业缺省的日期格式和分隔符都要一致。

如果使用常数，QRYSLT 应如下：

```
QRYSLT('″12/31/87″ *EQ DATE')
```

作业的缺省的格式必须是 MMDDYY，并以‘/’分隔。

如果想用已在库中数字字段与变量做比较，仅可使用字符变量。例如，想选择所有压缩日期字段大于某个变量的所有记录，就必须保证变量是字符格式，即通常在用 OPNQRYR 命令前，要用 CHGVAR 命令把变量从十进数字字段变为字

符字段，CHGVAR 命令如下：

```
CHGVAR VAR(&CHARVAR)  VALUE('123188')
```

QRYSLY 参数规定如下：

```
QRYSLT(&CHARVAR  *CAT ' *GT DATE')
```

如果用常数，QRYSLT 规定如下：

```
QRYSLT('123187 *GT DATE')
```

例 3：用一定范围的值选择记录

假定日期字段的格式为 YYMMDD 并以"."为分隔符，要处理所有 1988 年的记录，应如下定义：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) QRYSLT('DATE *EQ %RANGE(″88.01.01″ +
                ″88.12.31″) ')
CALL        PGM(PGMC)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

这个例子在 DATE 字段为日期格式，作业缺省值为 YYMMDD，分隔符为'.'时也可运行。

如果范围定义为字符变量，DATE 是字符类型，QRYSLT 规定如下：

```
QRYSLT('DATE *EQ %RANGE(″' *CAT &LORNG *CAT '″' *BCAT '″'  +
   *CAT &HIRNG *CAT '″)')
```

但如果 DATE 定义为数字字段，QRYSLT 规定：

```
QRYSLT('DATE *EQ %RANGE(' *CAT &LORNG *BCAT &HIRNG *CAT ')')
```

注：如果 QRYSLT 参数用在 CL 程序中，也可用*BCAT，但它不能用在交互命令行。

例 4：用包含功能选择记录

假定要处理所有 Addr 字段中有 BROADWAY 街名的所有记录，包含功能(*CT)确定字符串是否在字段中出现。语句如下：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) QRYSLT('ADDR *CT ″BROADWAY″ ')
CALL        PGM(PGMC)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

在这个例子中，数据是用大写，如果数据是小写或混合大小写，在比较前要有把它们变为大写的转换操作，系统提供的表 QSYSTRNTBL 可以把字母 a 到 z 变为大写（也可用其它表来完成转换），规定如下：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) QRYSLT('%XLATE(ADDR QSYSTRNTBL) *CT +
                  "BROADWAY" ')
CALL        PGM(PGMC)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

在 QRYSLT 句中用%XLATE 功能时，传到高级语言程序的字段值与数据库中一样，可以用 MAPFLD 参数中的%XLATE 功能强制字段变大写。

例 5：用多个字段选择记录

假如要处理那些 Amt 字段为零的，或是 Lstdat 字段（YYMMDD 格式的字符型）小于等于 88-12-31 的所有记录，可以规定：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) QRYSLT('AMT *EQ 0 *OR LSTDAT +
                  *LE "88-12-31" ')
CALL        PGM(PGMC)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

本例在 LSTDAT 字段为日期型时一样可用，LSTDAT 可以为各种合法日期格式，但作业缺省应为 YYMMDD，分隔符为‘-’。

如果使用变量 QRYSLT 规定如下：

```
QRYSLT('AMT *EQ ' *CAT &VARAMT *CAT ' *OR +
   LSTDAT *LE "' *CAT &VARDAT *CAT '"')
```

或用相反顺序：

```
QRYSLT('"' *CAT &VARDAT *CAT '" *GT LSTDAT *OR ' +
   *CAT &VARAMT *CAT ' *EQ AMT')
```

注：VARAMT 必须为字符类型，如果数字变量型传到 CL 程序中，必须把它转换成字符型，可用 CHGVAR 来做这个转换。

例 6：在一个程序多次使用 OPNQRYF 命令

可以高级语言程序中多次使用 OPNQRYF 命令。例如，要向用户提示选择项，并显示几页记录，在完成第一个请求后，用户可能要做其它选择值并显示记录，可以这样做：

1. 在调用程序前，用 OVRDBF 命令指定 SHARE(*YES)
2. 在高级程序中提示用户选择某个值。
3. 把选择的值传给运行 OPNQRYF 的 CL 程序（或调用程序 QCMDEXC 运行这条命令）处理 OPNQRYF 前，文件必须关闭，可以用 CLOF 命令关闭，

并监控文件是否关闭。

4. 返回到高级语言程序。
5. 在高级语言程序中打开文件。
6. 处理记录。
7. 在程序中关闭文件。
8. 返回步骤 2。

在程序结束后，用 CLOF 或 RCLRSC 命令关闭文件，并删除在步骤 1 中规定的覆盖。

注：在被调用的 CL 程序中使用的覆盖命令，不会影响主程序的打开，在程序结束时，所有的覆盖都被删除。（然而如果必要，可以从高级语言程序调用 QCMDEXC 来指定覆盖）。

例 7：压缩数字字段的映象字段

假定有 MMDDYY 格式的压缩十进日期字段，并要选择所有 1988 年的记录，不能直接从压缩十进字段选择出来，但可以用 OPNQRYF 命令的 MAPFLD 参数生成一个新字段，然后用它来选择字段的一部分。

映象字段的定义格式如下：

（结果字段　‘表达式’　属性）

注：

结果字段 = 结果字段名

表达式 = 结果字段如何导出表达式，可包括子串，其它内部函数或数学语句

属性 = 结果字段的可选属性，如果没有给出（或文件中没有定义这样的字段），OPNQRYF 以字段的表达式来确定字段属性。

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) QRYSLT('YEAR *EQ "88" ') +
              MAPFLD((CHAR6 '%DIGITS(DATE)')  +
                (YEAR '%SST(CHAR6 5 2)' *CHAR 2))
CALL        PGM(PGMC)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

在本例中，如果 DATE 是日期数据类型，可以规定如下：

```
OPNQRYF FILE(FILEA) +
QRYSLT ('YEAR *EQ 88') +
MAPFLD((YEAR '%YEAR(DATE)'))
```

第一个映象字段定义为 CHAR6 字段是由压缩十进日期字段生成，%DIGITS 功能把数字转换字符并忽略小数点位置(如 1234.56 转换为'123456',因为 CHAR6 没有定义，系统分配它为 6 位长，第二个映象字段 Year 为字符型 2 位长，这个表达式用子串功能把 CHAR6 中的最后二个字符映象到 Year 字段中。

注：映象字段定义是按规定它们时的顺序进行处理的。本例中，日期字段转

换为字符并分配给 CHAR6 字段，然后 CHAR6 后两个数字（年）分配给 YEAR 字段，对这一顺序的任何改动导致不正确的结果。

注：映象字段定义总在 QRYSLT 参数求值之前处理。

```
OPNQRYF     FILE(FILEA) +
              QRYSLT('%SST(CHAR6 5 2) *EQ "88" ') +
              MAPFLD((CHAR6 '%DIGITS(DATE)'))
```

例 8：使用“通配符”功能

假定要处理格式为 MMDDYY 的压缩十进日期字段，要选择 1988 年 3 月的记录可以这样做：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) +
              QRYSLT('%DIGITS(DATE) *EQ %WLDCRD("03__88")')
CALL        PGM(PGMC)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

注：只有当结果使用一个字段名（不是函数或表达式）做为参数的函数时，需要定义数据库字段来做为函数%DIGITS 的结果。%WLDCRD 操作对在*EQ 之前出现操作符没有这样的限制。

虽然数据库中的字段是数字型，仍要用双引号括起，使其定义和 CHAR6 相同，通配符功能不支持 DATE、TIME 或 TIMESTAMP 数据类型。

%WLDCRD 让你选择所有符合选择值的记录，下划线（_）对应一个字符值，在例 8 中两个下划线符允许选择三月中任何一天。%WLDCRD 函数还允许给通配符命名（缺省通配符为下划线）。

通配符功能支持两种不同的格式：

在上例中给出的固定位置的通配符形式，一个下划线对应一个字符，例如：

```
QRYSLT('FLDA *EQ %WLDCRD("A_C")')
```

这个比较可以选择到 ABC、ACC、ADC、AXC 等，这个例子中，只分析比较 3 个长度字符字段，如长度长于 3，要用第二种格式，可变位置的通配符将对应无字符或多个字符，OPNQRYF 命令用星号（*）作为通配符，或自行指定通配符，用法如下：

```
QRYSLT('FLDB *EQ %WLDCRD("A*C*") ')
```

它可以选择到 AC、BC、AXC、ABCD、A*****C 等等，星号使命令忽略任何内部字符，在本例中的星号放在要出现的字符的前边和后面，在检索自变量的末尾没有星号，就选择出以 C 结尾的字段，这样要注意星号在选择字符串中的位置：

例如：

```
QRYSLT('FLDB *EQ %WLDCRD("*ABC*DEF*") ')
```

可以检索到 ABCDEF　ABC*DEF　ABCXDEFX　ABCXXXXDEF　ABCXXXDEFXXX　XABCDEF　XABCXDEFX 等。

可以象下例中那样组合两种形式的通配符：

```
QRYSLT('FLDB *EQ %WLDCRD("ABC_*DEF*") ')
```

可以检索到 ABCXDEF　ABCXXXXDEF　ABCXXXDEFXXX 等。

下划线强制在 ABC 和 DEF 中之间至少出现一个字符，因此不会选择出 ABCDEF。

假如有字符名字字段

JOHNS　　JOHNS SMITH　　JOHNSSON　　JOHNSTON

如果象下指定，只能 检索到第一个记录：

```
QRYSLT('NAME *EQ "JOHNS"')
```

由于比较在条件式中有空格，所以无法检索到其他记录。

用下列办法可以选择到四个名字：

```
QRYSLT('NAME *EQ %WLDCRD("JOHNS*")')
```

注：关于%WLDCRD 函数中使用 DBCS 的内容，参阅附录 B“双字节字符集考虑”。

例 9：使用复合选择语句

可以使复合的选择语句，如下：

```
QRYSLT('DATE *EQ "880101" *AND AMT *GT 5000.00')
```

```
QRYSLT('DATE *EQ "880101" *OR AMT *GT 5000.00')
```

也可以指定：

```
QRYSLT('CODE *EQ "A" *AND TYPE *EQ "X" *OR CODE *EQ "B")
```

操作符的优先权规则在 CL 参考手册中讲述，其中一些规则如下：

先执行*AND，因此，记录按如下选择：

Code 字段 = “A”　且 Type 字段 = “X”

或　　Code 字段 = “B”

括号可以用来控制如何处理表达式，如下例：

```
QRYSLT('(CODE *EQ "A" *OR CODE *EQ "B") *AND TYPE *EQ "X" +
        *OR CODE *EQ "C"')
```

Code 字段 = “A”　且 Type 字段 = “X”

或　　Code 字段 = “B”

或　　Code 字段 = “C”

可以用 CL 参考手册中给出的符号代替关系符（如用=代替*EQ），如下例：

```
QRYSLT('CODE = "A" & TYPE = "X" | AMT > 5000.00')
```

这个命令选择如下的记录：

Cdde 字段 = “A”　且 Type 字段 = “X”

或　　Amt 字段 > 5000.00

复合选择式可以下例一样规定：

```
QRYSLT('CUSNBR = %RANGE("60000" "69999") & TYPE = "B" +
```

```
        & SALES>0 & ACCRCV / SALES>.3')
```

这个命令选择如下记录：

范围在 60000~69999 之间的 Cusnbr 字段，且

Type 字段 = “B”且

Sales 字段大于 0，并且

Accrcv 被 Sales 除后大于 30%。

例 10：使用编码字符集标识符（CCSIDs）

关于 CCSIDs 的一般信息，参阅国际语言支持手册。

在数据文件中的每个字符和 DBCS 字段都用 CCSID 标记。CCSID 允许进一步定义在文件中的数据存储，以便可以进行有意义的比较联结和显示字段。如：要比较 FILE1 中的 FILD1（它的 CCSID 为 37(USA)）和 FILE2 中的 FIELD2（它的 CCSID 为 273(Aastria　Germany)，发生相对的映象，以保证有意义的比较。

```
OPNQRYF FILE(FILEA FILEB) FORMAT(RESULTF) +
        JFLD((FILEA/NAME FILEB/CUSTOMER))
```

如果字段 NAME 的 CCSID 为 37，CUSTOMER 字段的 CCSID 为 273。在处理 OPNQRYF 时要执行 NAME，或 CUSTOMER 的映象，以使两个字段联结成有意义的结果。

通常，在 MAPFLD、QRYSLT、GRPSLT 参数中定义的常数都用所在作业中的 CCSID 标记，当两个有不同作业 CCSID 的用户都运行一个 OPNQRYF 命令（或包含 OPNQRYF 的程序），并且在 OPNQRYF 中有常数定义，用户可能得到不同结果。这是由于标记常数的 CCSID 引起常数做不同的解释。这时，可用 MAPFLD 参数规定 CCSID 来标记一个常数，在 MAPFLD 中定义常数，然后对它指定 CCSID，那么常数就用在 MAPFLD 参数中定义的 CCSID 标记。如：

```
OPNQRYF FILE(FILEA) FORMAT(RESULTF) QRYSLT('NAME *EQ MAP1') +
        MAPFLD((MAP1 '"Smith"' *CHAR 5 *N 37))
```

常数“Smith”标记为 CCSID 37，而不管发出 OPNQRYF 命令的作业的 CCSID。在这个例子中，所有用户可以得到相同结果。（虽然结果记录会被映象成用户作业的 CCSID）。相反，如果按以下查问：

```
OPNQRYF FILE(FILEA) FORMAT(RESULTF) QRYSLT('NAME *EQ "Smith"')
```

根据用户执行命令的作业的 CCSID 不同，结果可能会不同。

例 11：使用分类顺序和语言标识：

要了解如何使用分类顺序，请运行本节中例子，来处理表 6-2 中的文件 STAFF。

表 6－2　STAFF 文件

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	Sanders	20	Mgr	7	18357.50	0
20	Pernal	20	Sales	8	18171.25	612.45
30	Merenghi	38	MGR	5	17506.75	0
40	OBrien	38	Sales	6	18006.00	846.55

50	Hanes	15	Mgr	10	20659.80	0
60	Quigley	38	SALES	00	16808.30	650.25
70	Rothman	15	Sales	7	16502.83	1152.00
80	James	20	Clerk	0	13504.60	128.20
90	Koonitz	42	sales	6	18001.75	1386.70
100	Plotz	42	mgr	6	18352.80	0

在这个例子中，给出各种方式分类，用下面语句得出的结果：

*HEX 分类顺序

语句标识 ENU 共享权排序

语句标识 ENU 唯一权排序

注：在 OPNQRYF 中规定 SRTSEQ（*LANGIDUNQ）或 SRTSEQ（*LANGIDSHR）以及 LANGID（ENU），那么就选择了 ENU 作语言标识。

下面的命令选择 JOB 字段中有 MGR 值的记录：

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"')
```

表 6-3 显示了用*HEX 分类排序选择的记录，被选择的记录要确切地与 QRYSLT 语句中描述的那样，只选择大写的 MGR。

图 6-3　使用*HEX 分类排序

```
FILE(STAFF) QRYSLT('JOB *EQ "MGR"')
SRTSEQ(*HEX)
```

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
30	Merenghi	38	MGR	5	17506.75	0

表 6-4 显示了用共享排序选择的记录，选择记录忽略大小写用这种办法，mgr、Mgr 、MGR 都被选择。

表 6-4　使用共享权分类顺序

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"')
SRTSEQ(LANGIDSHR) LANGID(ENU)
```

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
10	Sanders	20	Mgr	7	18357.50	0
30	Merenghi	38	MGR	5	17506.75	0
50	Hanes	15	Mgr	10	20659.80	0
100	Plotz	42	mgr	6	18352.80	0

表 6-5 给出了用唯一权排序选择的记录，选择记录时忽略大小写用唯一的方式选择，这样 mgr、Mgr 和 MGR 都认为是不同，只选择 MGR。

表 6-5　使用唯一权分类顺序

OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"')

SRTSEQ(LANGIDUNQ) LANGID(ENU)

ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
30	Merenghi	38	MGR	5	17506.75	0

2.2.3.7　不用 DDS 指定键字顺序访问路径

动态访问路径功能允许你可以指定键字访问路径来处理数据。

如果访问路径已存在并可共享，系统就会共享，如果需要一个新路径，会在记录传到程序前建立它。

例 1：用一个键字字段排序记录

假如想要处理用程序 PGMD 中 Cust 字段排序的 FILEA 中的记录，可以指定如下：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) KEYFLD(CUST)
CALL        PGM(PGMD)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

注：由于 PGMD 生成时指定 FILEA 为处理文件，因此，OPNQRYF 命令中不需要 FORMAT 参数。FILEA 可以是到达顺序或键字顺序文件。如果 FILEA 为键字文件，它的键字字段可以是 Cust 字段或完全不同的字段。

例 2：用多键字排序记录

如果想用 Cust 顺序处理，然后再用 Cust 中的 Date，如下规定：

```
OPNQRYF     FILE(FILEA) KEYFLD(CUST DATE)
```

如果用 Date 的降序排序，可指定：

```
OPNQRYF     FILE(FILEA) KEYFLD((CUST) (DATE *DESCEND))
```

在这两个例子中，不用 FORMAT 参数（如定义不同的格式，所有键字段都要在其中）。

例 3：用唯一权分类顺序排序记录

为了能用 JOB 字段的值，按唯一权分类顺序处理表 6-2 中的 STAFF 文件中的数据规定。

```
OPNQRYF FILE(STAFF) KEYFLD(JOB) SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

在 JOB 字段排序结果如下：

```
Clerk
mgr
```

```
Mgr
Mgr
MGR
sales
Sales
Sales
Sales
SALES
```

例 4：用共享权分类顺序排序记录

用这种方法对 STAFF 文件的记录用 JOB 字段的值进行排序，指定如下：

```
OPNQRYF FILE(STAFF) KEYFLD(JOB) SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

这个查询结果类似例 3，mgr 和 Sales 项可能以任何形式，因为认为大、小写字母是相同的，也就是说，共享权排序把 mgr　Mgr 和 MGR 看作相等值。同样，sales　Sales 和 SALES 也等值。

2.2.3.8 从不同的文件中指定键字字段

逻辑文件的动态键字访问路径，允许指定一个不同于物理文件中的键字处理顺序。（DDS 限制这些键字要放在主文件中）。这个规定和前面的方法相同，访问路径是用所需要的什么样的键字来定义的，且不限制键字是否在物理文件中，但键字段在连接规范中的次文件中，必须做一个联结记录的临时拷贝，在查询文件打开之前，系统还必须在拷贝的记录上建立键字顺序访问路径，键字字段必须在 FORMAT 参数确定的格式中。

例 1：用次文件中的字段做键字字段

假设有一个连接逻辑文件 JOINLF，FILEX 是主文件和 FILEY 连接，要通过 FILEY 中的 Descrp 字段处理 JOINLF 中的记录。

假设文件记录格式包含下列字段：

FILEX	FILEY	JOINLF
Item	Item	Item
Qty	Descrp	Qty
		Descrp

可以指定

```
OVRDBF    FILE(JOINLF) SHARE(*YES)
OPNQRYF   FILE(JOINLF) KEYFLD(DESCRP)
CALL      PGM(PGMC)
CLOF      OPNID(JOINLF)
DLTOVR    FILE(JOINLF)
```

如想用 Descrp 中的 Qty 来安排记录（Descrp 是主键字段，Qty 是次键字段），可以指定：

```
OPNQRYF    FILE(JOINLF) KEYFLD(DESCRP QTY)
```

2.2.3.9　不用 DDS 动态连接数据文件

动态连接功能允许不用先规定 DDS 再生成连接逻辑文件来连接文件，可以使用 OPNQRYF 命令中的 FORMAT 参数来指定连接的记录格式，可以连接任何物理文件、逻辑文件（包括连接逻辑文件和窗口，DDS 不允许联结逻辑文件），可以规定到达或键字访问路径，如果指定键字，键字段可以来自任何文件（包括连接文件，DDS 只允许用主文件的键字）。

在下面例子中，假定在 FORMAT 中指定的文件已生成，通常要在生成程序之前文件要生成好，这样可以使用外部描述的数据定义。下面几例中，都用 JORDER 参数的缺省值*ANY，告诉系统它可以确定连接文件的顺序，也就是系统决定哪个为主文件，哪个为次文件，这样就可以让系统来改善连接性能。

联结规范和记录选择规范一样受分类顺序（SRTSEQ）和语言标识（LANG1D）的影响。（看 2.2.3.6.1 的例 11）

例 1：动态连接文件

假设想连接 FILEA 和 FILEB，各自包含下列字段：

FILEA	FILEB	JOINAB
Cust	Cust	Cust
Name	Amt	Name
Addr		Amt

连接字段是两个文件都有的 Cust 字段，可以在 OPNQRYF 中指定要连接的文件，文件不须有成员，记录不须以键字顺序。可以指定如下：

```
OVRDBF     FILE(JOINAB) TOFILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA FILEB) FORMAT(JOINAB) +
             JFLD((FILEA/CUST FILEB/CUST)) +
             MAPFLD((CUST 'FILEA/CUST'))
CALL       PGM(PGME) /* Created using file JOINAB as input */


CLOF       OPNID(FILEA)
DLTOVR     FILE(JOINAB)
```

JOINAB 是一个无数据的物理文件，它包含 OPNQRYF 命令中 FORMAT 参数指定的记录格式包含的文件。

注：OVRDBF 中的 TOFILE 参数指定了连接操作中的主文件是在 OPNQRYF 的 FILE 参数中规定的第一个文件。在本例中，OPNQRYF 命令的 FILE 参数指定了文件的连接顺序（A 到 B），文件的格式用 JOINAB，JFLD 给出用 FILEA 的 Cust 字段连接 FILEB 中的 Cust 字段。因 Cust 不是唯一的在连接记录格式中的都有的字段，因此在 JFLD 中必须说明。有时，在 OPNQRYF 中不指定 JFLD，系统会判定最有效的值。使用前边的例子，如果规定：

```
OPNQRYF     FILE(FILEA FILEB) FORMAT(JOINAB) +
                QRYSLT('FILEA/CUST *EQ FILEB/CUST') +
                MAPFLD((CUST 'FILEA/CUST'))
```

由于 QRYSLT 参数中规定了值，系统使用 Cust 字段连接 FILEA 和 FILEB。

注：在本例中，没有指定 JFLD，然而，如果在 OPNQRYF 中规定了 JDFTVAL(*ONLYDFT)或者 JDFTVAL(*YES)，就必须指定 JFLD。

在 OPNQRYF 要用 MAPFLD 参数，以此来描述在 JOINAB 中 Cust 使用哪个文件的字段，如果在 MAPFLD 中指定了字段，它的非限定名（在本例中，Cust 字段不带文件名）可用在 OPNQRYF 中任何地方。由于 Cust 在 MAPFLD 中指定，JFLD 的第一个值不须限定。，例如，下面的语句可取得相同结果：

```
JFLD((CUST FILEB/CUST)) +
MAPFLD((CUST 'FILEA/CUST'))
```

在 OPNQRYF 中用到的不是由 MAPFLD 参数定义的文件中取得的字段都必须加上文件名。

因为没有指定 KEYFLD，所以记录将根据 OPNQRYF 选择的顺序出现。可以强制系统以主文件的顺序安排，要做到这点，在 KEYFLD 中指定*FILE，即使主文件为到达顺序时，也能这样做。

JDFTVAL 参数（类似 DDS 中 JDFTVAL 键字）可以在 OPNQRYF 中指定，以此描述当次要文件没有记录时系统应如何做。在这个例子中，没有指定 JDFTVAL，所以只选择两个文件中都有的记录。

如果告诉系统改进查询结果（通过 OPNQRYF 的参数），一般选择最少的记录数的文件做主文件，但系统仍避免建立临时文件。

可以在 OPNQRYF 中指定 JORDER（*FILE）来强制系统遵循命令中 FILE 参数规定的文件做顺序，如指定 JDFVAL(*YES)或 JDFVAL(*ONLYDRT)系统不会改变连接文件顺序，因不同顺序可能导致不同结果。

例 2：只读取次文件中有匹配的记录

假设要连接文件 FILEAB、FILECD 和 FILEEF 来选择有次文件匹配的记录，定义一个文件 JOINF，并描述要使用的记录格式：

假设记录格式中包含下列记录格式。

FILEAB	FILECD	FILEEF	JOINF
Abitm	Cditm	Efitm	Abitm
Abord	Cddscp	Efcolr	Abord
Abdat	Cdcolr	Efqty	Cddscp
			Cdcolr

			Efqty

这个例子中，做成连接文件的所有字段名字都有 2 个字母的前缀（标识字段所在文件）和指出所有文件都有的一个后缀（如 xxitm）这使得所有的字段名唯一且不用文件名做限定。

xxitm 字段把 FILEAB 到 FILECD 进行连接。两个字段 xxitm　xxcolr 允许从 FILECD 到 FILEEF 连接。

在这些文件中不一定要有键字顺序访问路径，但如果有，将明显改善性能。因为系统总试图在可能时用已存在访问路径，如果没有，系统会自动生成，且保持到文件关闭。

```
OVRDBF      FILE(JOINF) TOFILE(FILEAB) SHARE(*YES)
OPNQRYF     FILE(FILEAB FILECD FILEEF) +
              FORMAT(JOINF) +
               JFLD((ABITM CDITM)(CDITM EFITM) +
              (CDCOLR EFCOLR))
CALL        PGM(PGME) /* Created using file JOINF as input */
CLOF        OPNID(FILEAB)
DLTOVR      FILE(JOINF)
```

连接字段对不一定要按上面的顺序指定，用 JFLD 参数也可起到同样效果：

```
JFLD((CDCOLR EFCOLR)(ABITM CDITM) (CDITM EFITM))
```

不必标识每个连接字段对的属性，通常对字符字段的填充和数字字段的小数点定位会自动完成。

因没有指定 JDFTVAL，所以假定为*NO。在组织连接字段时，没有缺省值可用。如指定了 JDFTVAL(*YES)，但在 FILECD 中没有与 FILEAB 中相同连接字段值的记录，就用 Cddscp 和 Cdcolr 字段的缺省值来连接 FILEEF。用这些缺省值，在文件 FILEEF 中找到匹配的记录（根据缺省值是否与次文件中的记录匹配）否则，这些主文件和 Efqty 字段用缺省值表示。

例 3：用映象字段作为连接字段

可以用在 MAPFLD 参数中定义的字段作为连接字段对中的一个，这种方式在次文件中字段被定义为单一字段（例如，6 字符日期字段），而在主文件中是可以分开的字段（如月、日、年）时很有用。假设 FILEA 有年、月、日字符字段，要连接 FILEB、FILEB 中有 YYMMDD 格式的日期字段，假定已在 JOINAB 中定义了要求的格式，可以规定：

```
OVRDBF      FILE(JOINAB) TOFILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA FILEB) FORMAT(JOINAB) +
              JFLD((YYMMDD FILEB/DATE)) +
              MAPFLD((YYMMDD 'YEAR *CAT MONTH *CAT DAY'))
```

```
CALL       PGM(PGME) /* Created using file JOINAB as input */
CLOF       OPNID(FILEA)
DLTOVR     FILE(JOINAB)
```

MAPFLD 参数定义了 YYMMDD 字段做为从 FILEA 中几个字段的连接。你不须在 MAPFLD 中指定字段属性（如长度、类型），系统根据表达式能自动计算。

2.2.3.10 处理连接文件中次文件里没有的记录

系统允许你来控制在次要文件中找不到记录时是否使用缺省值。（类似 DDS 中的 JDFTVAL 键字）。也可以指定只处理有缺省值的记录，这就允许你选择那些在次文件中没有匹配字段的记录。

例 1：在主文件中有而在次文件中没有的记录

在 2.2.3.9 中的例 1 中，没有规定 JDFTVAL，所以仅是读到从 FILEA 到 FILEB 成功连接的记录，如果想得到在 FILEA 中有但在 FILEB 中没有的，应象下面那样，在 JDFTVAL 参数中规定*ONLYDFT

```
OVRDBF     FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA FILEB) FORMAT(FILEA) +
             JFLD((CUST FILEB/CUST)) +
             MAPFLD((CUST 'FILEA/CUST')) +
             JDFTVAL(*ONLYDFT)
CALL       PGM(PGME) /* Created using file FILEA as input */
CLOF       OPNID(FILEA)
DLTOVR     FILE(FILEA)
```

JDFTVAL(*ONLYDFT)使得仅在次文件中没有匹配字段的记录返回给程序。由于连接操作返回的 FILEB 中字段都用缺省值，通常都使用 FILEA 中的格式，所选的记录都是在 FILEB 中没有匹配字段的记录，即使 FILE 参数中描述了不只一个文件时，FORMAT 参数也是必须的，但文件名可以是 FILE 参数中的文件之一，程序用 FILEA 生成。

反过来，也可以得到在 FILEB 中有但在 FILEA 中没有匹配字段的记录。

可以在所有规定中把次文件定义为主文件来生成。指定如下：

```
OVRDBF     FILE(FILEB) SHARE(*YES)
OPNQRYF    FILE(FILEB FILEA) FORMAT(FILEB) JFLD((CUST FILEA/CUST)) +
             MAPFLD((CUST 'FILEB/CUST')) JDFTVAL(*ONLYDFT)
CALL       PGM(PGMF) /* Created using file FILEB as input */
CLOF       OPNID(FILEB)
DLTOVR     FILE(FILEB)
```

注：本例中 OVRDBF 参数使用 FILE（FILEB）是因为在 OPNQRYF　FILE

参数中必须规定第一个文件，CLOF 命令也用 FILEB，JFLD 和 MAPFLD 参数也改变，程序用 FILEB 生成。

2.2.3.11 唯一键字处理

唯一键字允许只处理一组中的第一个记录，这个组是由具有相同键字值的一个或多个记录组成，仅处理第一个记录就含有你接收到的记录是有唯一键字的。

当用唯一键字时，仅可读顺序文件，键字字段的顺序由 SRTSEQ 和 LANGID 决定。（见 2.2.3.7 的例 3 和例 4）。

例 1：只读唯一键字记录

假定要处理 FILEA 中 Cust 字段的重复键字，程序 PGMF 仅处理 Cust 字段唯一键字值的每个记录，规定如下：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) KEYFLD(CUST) UNIQUEKEY(*ALL)
CALL        PGM(PGMF)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

例 2：只用某些键字段读记录

假定要按 Slsman、Cust 和 Date 的顺序处理同一文件，且每个 Cust，Slsman 只要一个记录，假定文件记录如下：

Slsman	Cust	Date	Record #
01	5000	880109	1
01	5000	880115	2
01	4025	880103	3
01	4025	880101	4
02	3000	880101	5

可以指定唯一键字的数目，从第一个键字段开始：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) KEYFLD(SLSMAN CUST DATE) UNIQUEKEY(2)
CALL        PGM(PGMD)
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEA)
```

下面是程序接收到的记录：

Slsman	Cust	Date	Record #

01	4025	880101	4
01	5000	880109	1
02	3000	880101	5

注：NULL 值认为是相等的，所以返回第一个 NULL 值。

2.2.3.12 定义由现有字段派生的字段

映象字段定义：

允许生成有选择值的内部字段（见 2.2.3.6 的例 7）。

允许在多个文件中有相同字段名时，避免混淆（见 2.2.3.9 中例 1）。

允许生成仅在处理格式中而不在数据库中的字段。这就允许完成转换、子串、连接和复杂的数字操作。下面的例子给出了这些功能。

例 1：使用派生的字段

假设在记录格式中有 Price 和 Qty 字段，可以用 OPNQRYF 用一个字段乘另一个字段生成一个导出的 exten 字段来完成，假设要处理 FILEA，并已生成 FILEAA，假定文件的记录格式包含如下字段：

FILEA	FILEAA
Order	Order
Item	Item
Qty	Exten
Price	Brfdsc
Descrp	

exten 字段是映象字段，值为 Qty 乘 Price ，在新记录格式中没必要有 Qty 和 Price，但如需要也可存在于格式中。

Brfdsc 字段是 Descrp 字段的部分（用前 10 个字符），假设用 PGMF 处理新格式，要生成这个程序，读 FILEAA 文件，可以指定：

```
OVRDBF      FILE(FILEAA) TOFILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) FORMAT(FILEAA) +
               MAPFLD((EXTEN 'PRICE * QTY') +
               (BRFDSC 'DESCRP'))
CALL        PGM(PGMF) /* Created using file FILEAA as input */
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEAA)
```

注：Exten 的属性是 FILEAA 中的记录格式定义的，如果字段的计算结果太大，会向程序发出一个异常信息。

如果只想要一个字段开始的字符串，就没有必要使用子串功能来得到 Brfdsc

字段，可以在 FILEAA 的记录格式中定义 Brfdsc 的长度。

存在 FORMAT 参数中定义的字段，都要在 OPNQRYF 中描述。也就是说，所有输出格式中的字段必须要在 FILE 指定文件的某个记录格式中或在 MAPFLD 中规定。如果在 FORMAT 中的一些字段程序没有使用，可用 MAPFLD 参数把字段置为零或空格。假定在输出格式中 FLdc 字段是字符字段，FLdn 是数字字段，在程序中不用这些值，可以在 OPNQRYF 命令中规定：

```
MAPFLD((FLDC ' ″ ″ ')(FLDN 0))
```

象下面这样规定来避免错误：

注：双引号中包括空格值，对不使用的字段定义常数时，可以避免必须生成唯一格式给每修正 OPNQRYF 命令使用。

例 2：使用内部函数

假定要计算 FILEA 中 FLdn 字段的正弦值，首先用下面字段生成文件（假定为 FILEAA）。

FILEA	FILEAA
Code	Code
Fldm	Fldm
	Sinm

然后用 FILEAA 作为输入生成程序（PGMF）。

```
OVRDBF      FILE(FILEAA) TOFILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) FORMAT(FILEAA) +
                MAPFLD((SINM '%SIN(FLDM)'))
CALL        PGM(PGMF) /* Created using file FILEAA as input */
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEAA)
```

内部函数%SIN 计算做为参数字段的正弦值。因为 sinm 字段在 FORMAT 指定的格式中定义，OPNQRYF 把正弦值的定义（浮点表示）转换成 Sinm 字段的定义，这一技术可以避免某些高级语言对浮点字段的限制。例如，定义 Sinm 为压缩十进制字段，PGMF 可用任何高级语言编写，即使它的值是用浮点字段表示也没有问题。

例 3：使用派生字段和内部函数

假定在上例中，Fldx 在 FILEA 中，并有存储 Fldm 字段正弦值的相应属性，假定你不用 Fldx 中的内容，可以用 MAPFLD 在把它传到程序前改变它的内容，可以规定：

```
OVRDBF      FILE(FILEA) SHARE(*YES)
```

```
OPNQRYF    FILE(FILEA) MAPFLD((FLDX '%SIN(FLDM)'))
CALL       PGM(PGMF) /* Created using file FILEA as input */
CLOF       OPNID(FILEA)
DLTOVR     FILE(FILEA)
```

这里，不需在 FORMAT 中指定不同的记录格式(默认是使用 FILE 参数中第一个文件的格式），这样程序用 FILEA 生成。用这个技术时，必须确认主定义的字段属性能保证执行正确。最简单的方法是用要处理的字段生成一个单独的文件。

也可把这个技术与映射字段定义和%XLATE 功能一起使用来转换字段。这样，可以与在数据文件中不同的方式出现在程序中。例如，可以转换小写字段，使程序只用大写。

排序顺序和语言标识也可影响%MIN 和%MAX 的结果。例如，依据排序顺序和语言标识的不同，大小写字母可以相等，也可以不等。注意转换的字母用于决定最大最小，而未转换的值用结果记录返回。

例子中 FIELA 做输入文件，也可以用 OPNQRYF 更新数据，但如果用映象字段定义来修改，将忽略字段的更新。

2.2.3.13 处理零做除数

零做除数在 OPNQRYF 中认为是错误的。

记录的选择正常是在字段映象错误前进选择（例如，字段映象引起除法错误）。因此，能导致被零除的记录是否丢失取决于 QRYSLT 参数和记录的有效数据。这种情况下，记录被丢失，而 OPNQRYF 命令的处理继续进行。

如果要零结果，下面给出的是典型的商务数据的处理方法：

假设要用 A 除以 B 结果是 C，（A/B=C）下表中 B 可以为零：

字段	数字位	小数位
A	6	2
B	3	0
C	6	2

可以用下面表达式：

```
(A * B) / %MAX((B * B) .nnnn1)
```

%MAX 返回 B*B 或一个小值的最大数，这个小值要有足够的前置 0，以使它小于任何 B*B 的值（除非 B=0）。在这个例子中，B 小数点位为零，所以可以用 0.1。小数点后的零应是 B 中的小数位的两倍，如 B 有 2 位小数，要用 0.00001。

定义 MAPFLD 如下：

```
MAPFLD((C '(A * B) / %MAX((B * B) .1)'))
```

用第一个乘号的目的是当 B 为零时，让被除数为零，并且保证结果为零。当 B 为零时，由于用 1 做除数，而不会发生用零做除数的问题。

2.2.3.14 从数据库记录中汇总数据（成组处理）

成组处理功能可以用存在的数据库记录汇总数据，指定如下：

• 分组字段
• 选择成组前后的值
• 对新记录的键字顺序访问路径
• 映象字段定义允许进行下列操作：求和、求平均值、标准偏差数、统计方差，以及
统计每组中的记录数
• SRTSEQ 和 LANGID：由成组字段值提供权因子

通常用包含下列类型字段的记录格式来生成文件：
• 组合字段：用 GRPFLD 说明组合定义，每组中包含所有字段的常数集合，组合字段
不必出现在 FORMAT 规定的记录格式中。
• 集合字段：用 MAPFLD 参数与下列一个或多个内部函数来定义：

%COUNT	统计组中记录数
%SUM	组中字段值求和
%AUG	组中字段值的算术平均值
%MAX	组中字段值最大值
%MIN	组中字段值最小值
%STDDEV	组中字段值标准偏差
%VAR	组中字段值统计方差

• 常数字段：允许字段值为常数，OPNQRYF 命令对输出格式中所有字段的限制，对
组合功能同样有效。
当使用组合功能时，只能顺序读文件。

例 1：使用组合处理
假定要用顾客号组合数据，并分析总计字段，数据文件为 FILEA，生成一个有下列字段的文件 FILEAA：

FILEA	FILEAA
Cust	Cust
Type	Count (count of records per customer)
Amt	Amtsum (summation of the amount field)
	Amtavg (average of the amount field)
	Amtmax (maximum value of the amount field)

当定义新文件字段时，必须保证它可以容纳下结果，例如，Amt 字段，定义为 5 位数字，Amtsum 字段必须定义为 7 位数字，任何数字溢出都导致程序非正常结束。
假定 FILEA 中有下列值：

Cust	Type	Amt
001	A	500.00
001	B	700.00
004	A	100.00
002	A	1200.00
003	B	900.00
001	A	300.00
004	A	300.00
003	B	600.00

用 FILEAA 做输入来打印记录生成程序（PGMG）。

```
OVRDBF      FILE(FILEAA) TOFILE(FILEA) SHARE(*YES)
OPNQRYF     FILE(FILEA) FORMAT(FILEAA) KEYFLD(CUST) +
              GRPFLD(CUST) MAPFLD((COUNT '%COUNT') +
                (AMTSUM '%SUM(AMT)') +
                (AMTAVG '%AVG(AMT)') +
                (AMTMAX '%MAX(AMT)'))
CALL        PGM(PGMG) /* Created using file FILEAA as input */
CLOF        OPNID(FILEA)
DLTOVR      FILE(FILEAA)
```

由程序中读到的记录如下：

Cust	Count	Amtsum	Amtavg	Amtmax
001	3	1500.00	500.00	700.00
002	1	1200.00	1200.00	1200.00
003	2	1500.00	750.00	900.00
004	2	400.00	200.00	300.00

注：如果指定 GRPFLD 参数，组合不会以升序排列。要保证用指定的顺序，必须规定 KEYFLD 参数。

假如只打印 Amtsum 字段值大于 700.00 的记录总计，由于 Amtsum 是给定顾客的集合字段，用 GRPSLT 参数规定在组合后选择，加上 GRPSLT 参数：

```
GRPSLT('AMTSUM *GT 700.00')
```

程序读出的记录为：

Cust	Count	Amtsum	Amtavg	Amtmax

001	3	1500.00	500.00	700.00
002	1	1200.00	1200.00	1200.00
003	2	1500.00	750.00	900.00

OPNQRYF 命令支持成组前选择（QRYSLT）和成组后选择（GRPSLT）。

假定也要选择字段类型为 A 的顾客记录，因 TYPE 是 FIEA 中记录格式的字段，并且不是集合字段，就要在组合前加 QRYSLT 语句：

```
QRYSLT('TYPE *EQ "A" ')
```

注：用来选择的字段不必须出现在程序处理的格式中。

程序取到如下记录：

Cust	Count	Amtsum	Amtavg	Amtmax
001	2	800.00	400.00	500.00
002	1	1200.00	1200.00	1200.00

注：CUST　001 的值改变了，是因为选择发生在组合之前。

假定要根据 Amtavg 字段降序来做输出，要加上前边的 QRYSLT 参数，可以把 OPNQRYF 命令的 KEYFLD 改成：

```
KEYFLD((AMTAVG *DESCEND))
```

程序得到的记录为：

Cust	Count	Amtsum	Amtavg	Amtmax
002	1	1200.00	1200.00	1200.00
001	2	800.00	400.00	500.00

2.2.3.15 最终总计处理

最终总计处理是不说明组合字段来进行组合的一种特殊形式。它是输出一个记录，所有组合用的内部函数都可以使用。也可规定选择组或最终总计的记录。

例 1：简单总计处理

假定有数据文件 FILEA，并为总计记录生成文件 FINTOT。

FILEA	FINTOT
Code	Count (count of all the selected records)
Amt	Totamt (total of the amount field)
	Maxamt (maximum value in the amount field)

FINTOT 中包含生成最终总计的一个记录，可以规定：

```
OVRDBF      FILE(FINTOT) TOFILE(FILEA) SHARE(*YES)
```

```
OPNQRYF    FILE(FILEA) FORMAT(FINTOT) +
             MAPFLD((COUNT '%COUNT') +
               (TOTAMT '%SUM(AMT)') (MAXAMT '%MAX(AMT)'))
CALL       PGM(PGMG) /* Created using file FINTOT as input */
CLOF       OPNID(FILEA)
DLTOVR     FILE(FINTOT)
```

例 2：用记录选择做最终总计

假定要把上面例子改为以 Code 为 B 的记录做总计，可以加上 QRYSLT 参数：

```
OVRDBF     FILE(FINTOT) TOFILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) FORMAT(FINTOT) +
             QRYSLT('CODE *EQ "B" ') MAPFLD((COUNT '%COUNT') +
               (TOTAMT '%SUM(AMT)') (MAXAMT '%MAX(AMT)'))
CALL       PGM(PGMG) /* Created using file FINTOT as input */
CLOF       OPNID(FILEA)
DLTOVR     FILE(FINTOT)
```

可在最终总计中使用 GRPSLT 键字，你规定的 GRPSLT 选择的值决定能否得到最终总计记录。

例 3：用新记录格式进行最终总计

假定要用 CL 程序处理新的文件/格式，要读取文件并用最终总计发送消息，可以指定：。

```
DCLF       FILE(FINTOT)
DCL        &COUNTA *CHAR LEN(7)
DCL        &TOTAMTA *CHAR LEN(9)
OVRDBF     FILE(FINTOT) TOFILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) FORMAT(FINTOT) MAPFLD((COUNT '%COUNT') +
               (TOTAMT '%SUM(AMT)'))
RCVF
CLOF       OPNID(FILEA)
CHGVAR     &COUNTA &COUNT
CHGVAR     &TOTAMTA &TOTAMT
SNDPGMMSG  MSG('COUNT=' *CAT &COUNTA *CAT +
             ' Total amount=' *CAT &TOTAMTA)
DLTOVR     FILE(FINTOT)
```

必须把信息中的数字字段转换为字符字段。

2.2.3.16 控制系统如何运行 OPNQRYF 命令

优化功能允许规定如何使用查询结果。

当使用 OPNQRYF 时，有两步要考虑性能，第一步是实际执行 OPNQRYF 命令本身，这一步要决定是用已存在路径还是另建一个新路径，第二步在于当程序使用 OPNQRYF 结果来处理数据时。（见附录 D）

对多数批处理，可能只关心上面二步的总计时间，因此，OPNQRYF 的缺省值是 OPTIMIZE(*ALLIO)，这表明，OPNQRYF 将考虑使用二步的总时间。

如果在交互环境中，并不关心处理整个文件，只想要尽快显示第一个屏幕，由于这个原因，要在第一步尽可能避免重建访问路径，那么可以指定 OPTIMIZE(*FIRSTIO)。

如果多个程序使用 OPNQRYF 的同一结果，就需在第一步建立高效的 ODP。这样，就要在 OPNQRYF 中指定 OPTIMIZE(*MINWAIT)，来使程序在第二步中读入的记录数最少。

当没有路径共享，而 OPNQRYF 命令中的 KEYFLD 和 GRPFLD 参数需要访问路径，就会不管 OPTIMIZE 而建立整个访问路径，优化功能主要影响选择处理。

例 1：对第一组记录优化处理

假定有一个交互作业需要 Code 字段为 R 的所有记录，程序的子文件规定每屏有 15 个记录，要使第一屏尽快出现，指定如下：

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF   FILE(FILEA) QRYSLT('CODE = "B" ') +
            SEQONLY(*YES 15) OPTIMIZE(*FIRSTIO)
CALL      PGM(PGMA)
CLOF      OPNID(FILEA)
DLTOVR    FILE(FILEA)
```

系统不管在 Code 字段上是否有访问路径，它会优化管理查询，并在完成整个查询前填满第一个缓冲区。

例 2：最少读记录数的优化

假定有多个程序访问同一个由 OPNQRYF 打开的文件，这时可以优化性能，使程序只读取有用的记录，这就意味着要求 OPNQRYF 尽可能高效的选择，可以规定：

```
OVRDBF    FILE(FILEA) SHARE(*YES)
OPNQRYF   FILE(FILEA) QRYSLT('CODE *EQ "B"') +
          KEYFLD(CUST) OPTIMIZE(*MINWAIT)
CALL      PGM(PGMA)
POSDBF    OPNID(FILEA) POSITION(*START)
```

```
CALL     PGM(PGMB)
CLOF     OPNID(FILEA)
DLTOVR   FILE(FILEA)
```

2.2.3.17 生成文件和使用 FORMAT 参数的考虑

当在 FILE 中指定多个文件并要合并处理时，必须在 FORMAT 中指定记录格式名，（也即不能规定 FORMAT(*FILE)）。在指定组合功能或在 MAPPLD 中指定一个完整表达式来定义派生字段时，都要指定记录格式名，这时应考虑下面情况：

• 记录格式可以定义为任何选择的名，也可以不同于要查询的数据库的格式名。

• 字段名可以任选，如果在查询文件中，字段名唯一，系统就会用同样的名把它映象

到 FILE 参数和 FORMAT 参数中，参见 2.2.3.9 中例 1。

• 如果字段名唯一，属性与在 FILE 和 FORMAT 中定义的不同，数据隐含着映象操作。

• 当用 MAPFLD 定义派生字段时，必须用正确的属性。例如，使用%SUM 函数，就

必须指定足够大的字段来容纳总计结果，否则就发生溢出，并给程序发送异常信息。

• 对所有字段有十进定位时要映象到 FORMAT 中的记录格式中，假设在查询结果字

段中有一个 5 位长，0 位小数点的字段，要做计算或映象成 0.12345 字段，因为小数

点右边的数字被截断，得到的结果为零。

2.2.3.18 排列记录的考虑

OPNQRYF 命令的缺省处理是用能改善性能的顺序并不与 KEYFLD 上的顺序冲突，因此，除非在 KEYFLD 参数中指定键字段名或 KEYFLD(*FILE)，每次执行 OPNQRYF 命令程序都会得到不同顺序的记录。

在 OPNQRYF 中指定 KEYFLD(*FILE)，而且不是用*HEX 而用作业缺省值或 OPNQRYF SRTSEQ 排序，可能得到并不反映实际记录顺序的结果，如果文件是键字的，查询顺序就体现文件键字的顺序，并发送 CPI431F 信息。

如果文件分类顺序和交替分配顺序表都被忽略，这就允许用户不必指出所有字段名，而仅指定哪个字段做分类顺序。如果没有指定查询分类顺序（如*HEX）就会按 V2.3 版中的规定排序。

2.2.3.19 DDM 文件考虑

OPNQRYF 可以处理 DDM 文件，所有在 FILE 参数中指定的文件必须在 AS/400 中存在。一个指定操作并用 DDM 文件的 OPNQRYF 需要源文件和目标文件在同类型系统中。

2.2.3.20 编写高级语言程序的考虑

象在 2.2.3.1 中讲述的那样，高级语言程序能直接访问数据文件。在程序外选

择或排序，程序以你指定顺序接收选择的记录，程序得不到选择时忽略的记录。通过逻辑文件用选择/忽略值处理时，具有同样功能。

如果使用 FORMAT 参数，程序使用在 FORMAT 中用的文件名，写程序时就象这个文件包含实际的数据一样。

如果顺序读文件，高级程序自动忽略键字顺序，你可以象读到达顺序记录一样写程序。如果在 OPNQRYF 命令中用 KEYFLD 参数，要收到一个能忽略的诊断信息。如果用随机的键字处理文件，高级语言要有相应的键字规定，如果有选择的值，它可以防止程序访问在数据库中的记录。或者用 OPNQRYF 或用 DDS 中用选择/忽略条件的逻辑文件。在随机读中都可能发生记录未找到的情况。

在某些情况下，可以监控诸如数字溢出之类的映象错误引起的异常，但最好是对所有字段定义正确的属性。

2.2.3.21 OPNQRYF 命令运行时发送信息

当 OPNQRYF 运行时，会给交互用户发送 OPNQRYF 的状态信息。例如，当键字访问顺序建立用来满足 OPNQRYF 需要时，会发送一个信息。在运行 OPNQRYF 时可能发送下面信息：

信息标识	说明
CPI4301	Query 运行
CPI4302	Query 运行，建立访问路径……
CPI4303	Query 运行，生成文件备份……
CPI4304	Query 运行，完成选择……
CPI4305	Query 运行，备份文件排序……
CPI4306	Query 运行，建立文件访问路径……
CPI4011	Query 运行，处理的记录数……

如不让这些信息出现，参考 CL 手册中关于信息控制的内容。

当作业在 Debug 下运行（用 STRDBG），信息发往作业日志中。描述了要处理 OPNQRYF 的实现方法，这些信息提供优化处理建议。也可以用来作为改善 OPNQRYF 运行性能的工具，这些信息如下：

CPI4321	建立文件访问路径
CPI4322	从键字文件建立访问路径
CPI4324	从文件生成临时文件
CPI4325	为 Query 建立临时文件
CPI4326	用 JOIN 位置处理文件
CPI4327	用 JOIN 位置 1 处理文件
CPI4328	使用文件访问路径
CPI4329	使用文件的到达顺序
CPI432A	Query 优化超时
CPI432C	考虑文件的所有访问路径

大部分信息都提供了使用当前功能的原因，每个信息的二级文本给出了扩展的说明，某些信息提供了改进 OPNQRYF 性能的建议。

2.2.3.22 使用 OPNQRYF 的其它可选功能

OPNQRYF 命令支持 OPTION 参数来决定处理的类型，默认值为

OPTION(*INP)，所以文件为仅输入打开，也可在 OPNQRYF 命令中使用 OPTION 的其它值和一个高级语言程序通过打开文件，来添加、更新、删除记录。

但如果指定 UNIQUEKEY、GRPFLD 或 GRPSLT 参数，用了一个组合功能或是在 FILE 中指定多个文件，对文件的使用就仅限于输入。

连接逻辑文件是只输入处理，如果在视图的定义中用了组合，连接合并，或分类处理时，视图也限于只输入。

如果要对文件中某些记录改变当前值，可把 OPNQRYF 命令和高级语言程序联合一起用，例如：要把文件中 Flda 字段的值 ABC 改为 XYZ，可以指定如下：

```
OVRDBF     FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) OPTION(*ALL) QRYSLT('FLDA *EQ "ABC" ')
CALL       PGM(PGMA)
CLOF       OPNID(FILEA)
DLTOVR     FILE(FILEA)
```

PGMA 处理读到的所有记录，但选择条件限制只处理 Flda 为 ABC 的记录，程序把它 改为 XYZ，并更新。

可以用 OPNQRYF 删除数据文件的记录，例如，要删除字段值为 X 的记录，可以编写下面程序并用 OPNQRYF 值选择要删除的记录。

```
OVRDBF     FILE(FILEA) SHARE(*YES)
OPNQRYF    FILE(FILEA) OPTION(*ALL) QRYSLT('DLTCOD *EQ "X" ')
CALL       PGM(PGMB)
CLOF       OPNID(FILEA)
DLTOVR     FILE(FILEA)
```

也可以用 OPNQRYF 添加记录，但如果查询条件中有选择值，程序可能由于这种选择被禁止读添加过的记录。

2.2.3.23　用 OPNQRYF 做日期、时间和时间标记的比较

日期、时间或时间标记可以和同一类型的其它值或表达式串进行比较。所有比较都是按年代分的，即这个时间距离 0001 年 1 月 1 日多长时间。

对包含时间值和表达式的比较经常包括秒，如果表达式串中忽略秒，用零代替。

对包含时间标记的比较是按年代分，而不管实际上可能相等的情况。

如下面表达式为真：

```
TIMESTAMP('1990-02-23-00.00.00') > '1990-02-22-24.00.00'
```

当字符 DBCS-open 或 DBCS-either 字段或常数字段用于表示日期、时间时间标记时，可用下列规则：

日期：如果日期格式为*ISO　*USA　*EUR　*JIS　*YMD　*MDY 或 *DMY 时，字段或文字长度至少为 8，当格式为 JUL(yyddd)时，变量的长度至少为 6，（包括 yy 和 ddd 间的分隔符），可用空格填充。

时间：对所有格式（*USA *ISO *EUR *JIS *HMS），字段或文字长度至少为 4，可用空格填充。

时间标记：对时间标记格式（yyyy-mm-dd-hh.mm.ss.uuuuuu）字段或文字长度到少为 16，可用空格填充。

2.2.3.24 用 OPNQRYF 命令对日期、时间、时间标记做运算

日期、时间、时间标记的值可以进行增加、减少，做减法运算。这些操作包括一个叫时间间隔十进制数。下面是时间间隔的定义和对日期、时间、时间标记做运算的规则。

2.2.3.24.1 时间间隔

时间间隔是一个数字，代表时间的间隔，四种类型的时间间隔是：

标记间隔：

是用一个数字代表的时间单位（可以是代表式的结束），可以做为七个时间间隔内部函数的操作数：%DURYEAR %DURMONTH %DURDAY %DUROUR %DURMINUTE %DURSEC %DURMICSEC

分别对应着年、月、日、小时、分钟、秒、微秒的间隔。

如果是 DECIMAL(15,0)的格式要做数字转换。标记间隔只能用在另一操作数为*DATE *TIME *TIMESTP 类型的运算中，因此，表达式

HIREDATE + %DURMONTH(2) + %DURDAY(14)是有效的

HIREDATE + (%DURMONTH(2) + %DURMONTH(14)是无效的。

在上面两个表达式中，标记间隔为%DURMONTH(2)和%DURDAY(14)。

日期间隔：

日期间隔是用 DECIMAL(8,0)格式表示的年、月、日的数字。

为能准确解释，数字必为 yyyymmdd 格式，yyyy 代表年，mm 代表月，dd 代表日，一个日期值减另一个日期的结果，就为日期间隔，例如：HIREDATE-BRTHDATE。

时间间隔：

时间间隔是用 DECIMAL(6,0)格式表示的小时，分钟，秒，微秒的数字，为了准确解释，必为 hhmmss 格式，hh 代表小时，mm 代表分钟，ss 代表秒，两个时间相减的结果为时间间隔。

时间标记间隔：

时间标记间隔是由 DICIMAL(20,6) 的格式表示的年、月、日、小时、分钟、秒、微秒的数字，为准确解释，数字必为 yyyymmddhhmmsszzzzzz 格式，其中 yyyy,mm,dd,hh,mm,ss,zzzzzz 分别代表年、月、日、小时、分钟、秒、微秒，两个时间标记相减的结果为时间标记间隔。

2.2.3.24.2 日期、时间、时间标记运算的规则

对日期和时间值所做的算术操作为加或减，如果加法的一个操作数是日期或时间值，另一个操作数必为间隔。

加法操作的规则是：

如果一个操作数是日期，另一个必为日期间隔或年、月、日的标记间隔。

如果一个操作数是时间，另一个必为时间间隔或小时、分秒的标记间隔。

如果一个操作数是时间标记，另一个必为时间间隔，可为任意类型间隔。

因为不能从间隔减去日期或时间，并且两个时间或日期相减与从日期或时间中减去间隔不同，所以对日期和时间的加减运算规则是不同的。下面规定了对日期和时间做减法的规则：

如果第一个操作数为日期，第二个必为日期、日期间隔，表示日期的串或是年、月、日的标记间隔。

如果第二个操作数是日期，第一个必为日期或表示日期的串。

如果第一个操作数是时间，第二个必为时间，时间间隔，表示时间的串或小时、分钟、秒的标记间隔。

如果第二个操作数是时间，第一个必为时间或表示时间的串。

如果第一个操作数是时间标记，第二个必为时间标记，表示时间标记的串或间隔。

如果第二个操作数是时间标记，第一个必为时间标记或表示时间标记的串。

2.2.3.24.3　日期运算

日期可以相减、增加、减少

减日期：一个日期（DATE1）减去另一个日期（DATE2）的结果是两个日期之间的年、月、日间隔，结果的数据类型为 DECIMAL(8,0)，如果 DATE1 大于或等于 DATE2，就用 DATE1 减 DATE2，如果 DATE1 小于 DATE2，则从 DATE2 减去 DATE1，结果符号为负，下面给出完成表达式 RESULT = DATE1—DATE2 的步骤。

```
If %DAY(DATE2) <= %DAY(DATE1)
;     then %DAY(RESULT) = %DAY(DATE1) - %DAY(DATE2).

If %DAY(DATE2) > %DAY(DATE1)
;    then %DAY(RESULT) = N + %DAY(DATE1) - %DAY(DATE2)
;      where N = the last day of %MONTH(DATE2).
;    %MONTH(DATE2) is then incremented by 1.

If %MONTH(DATE2) <= %MONTH(DATE1)
;    then %MONTH(RESULT) = %MONTH(DATE1) - %MONTH(DATE2).

If %MONTH(DATE2) > %MONTH(DATE1)
;    then %MONTH(RESULT) = 12 + %MONTH(DATE1) - %MONTH(DATE2).
;    %YEAR(DATE2) is then incremented by 1.

%YEAR(RESULT) = %YEAR(DATE1) - %YEAR(DATE2).
```

增加和减少日期：把日期加上间隔或从日期减去间隔，仍是日期，（为了做到这点，一个月代表日历中一页，把日期中加上月份，就象掀日历的页一样，从日期所在的月份那页开始）。结果必需要在 0001 年 1 月 1 日和 9999 年 12 月 31 日之间，如果是年的间隔加或减，只有年的部分有变化，月、日不变，除非是闰年的 2 月 29 日变为 28 日，同样，如果是月的间隔加减，只有月变，除非必需年有变化，日也不变，除非日无效（如 9 月 31 日），这时日被设为月的最后一天。

对日期的加减也只影响日，不影响年、月。无论是正的或负的日期间隔都可与日期做加、减，与标记间隔运算，结果是有效日期。当正的日期间隔加上日期，

或从日期减去负的日期间隔，日期就会依次按年、月、日的数增加。因而，DATE1 + X，如果 X 是正的 DECIMAL(8,0)数字，就相当于表达式

```
DATE1 + %DURYEAR(%YEAR(X)) + %DURMONTH(%MONTH(X)) +%DURDAY(%DAY(X))
```

当从日期减去正的日期间隔或负的日期间隔加一个日期，日期就会依次按年、月、日的数减少，因而，DATE1 – X，如果 X 是正的 DECIMAL(8,0)，就相当于表达式

```
DATE1 - %DURDAY(%DAY(X)) - %DURMONTH(%MONTH(X)) - %DURYEAR(%YEAR(X))
```

把日期中加上间隔，对某个日期加上一个月就出下月的相同日期，除非下月没有这个日期，这时，就设成下月的最后一天，如果 1 月 28 日加上一个月为 2 月 28 日，1 月 29、30、31 日加一个月在闰年为 2 月 29 日，否则为 2 月 28 日。

注：如果把日期加上一个或多个月，然后从结果再减去相同月份，最后日期和原始日期相同。

2.2.3.24.4　　时间运算

时间可以相减、增加或减少。

时间减法：一个时间（TIME1）减去另一个时间（TIME2）的结果是两者间的小时、分钟、秒的时间间隔，结果的数据类型为 DECIMAL(6,0)，如 TIME1 大于等于 TIME2，TIME1 减去 TIME2，否则，TIME2 减 TIME1，结果为负，下面给出了完成 RESULT + TIME1 - TIME2 的步骤：

```
If %SECOND(TIME2) <= %SECOND(TIME1)
;    then %SECOND(RESULT) = %SECOND(TIME1) - %SECOND(TIME2).

If %SECOND(TIME2) > %SECOND(TIME1)
;    then %SECOND(RESULT) = 60 + %SECOND(TIME1) - %SECOND(TIME2).
;    %MINUTE(TIME2) is then incremented by 1.

If %MINUTE(TIME2) <= %MINUTE(TIME1)
;    then %MINUTE(RESULT) = %MINUTE(TIME1) - %MINUTE(TIME2).

If %MINUTE(TIME2) > %MINUTE(TIME1)
;    then %MINUTE(RESULT) = 60 + %MINUTE(TIME1) - %MINUTE(TIME2).
;    %HOUR(TIME2) is then incremented by 1.

%HOUR(RESULT) = %HOUR(TIME1) - %HOUR(TIME2).
```

时间的增加和减少：从时间加或减一个间隔，结果仍是时间，任何上溢及下溢的小时数被丢弃，这样保证结果总为时间，如果是小时的间隔的加或减，只有小时部分改变，分钟和秒不变。

类似，对分钟的间隔加或减，只有分钟改变，如必要，小时受影响，秒不变。

加或减秒只改变秒部分，分钟或小时有潜在影响。

时间间隔不管正负都可以加上或减去时间，结果以小时、分钟、秒的顺序增

加或减少，因此，TIME + X，如 X 定 DECIMAL(6,0)，它相当于表达式

```
TIME1 + %DURHOUR(%HOUR(X)) + %DURMINUTE(%MINUTE(X)) +%DURSEC(%SECOND(X))
```

2.2.3.24.5　　时间标记运算

时间标记可以相减，增加或减少。

时间标记相减：一个标记 TS1 减去另一个标记 TS2，是一个在两个时间之间的代表年、月、日、小时、分钟、秒和微秒的间隔，数据类型为 DECIMAL(20,6)，如果 TS2 大于或等于 TS2，用 TS1 减去 TS2，否则用 TS2 减 TS1，结果为负。下面描述完成表达式 RESULT = TS1−TS2 的步骤：

```
 If %MICSEC(TS2) <= %MICSEC(TS1)
;    then %MICSEC(RESULT) = %MICSEC(TS1) -
;    %MICSEC(TS2).

If %MICSEC(TS2) > %MICSEC(TS1)
;    then %MICSEC(RESULT) = 1000000 +
;    %MICSEC(TS1) - %MICSEC(TS2)
;    and %SECOND(TS2) is incremented by 1.
```

时间标记的秒和分部分，用截取时间的原则来截取：

```
 If %HOUR(TS2) <= %HOUR(TS1)
;    then %HOUR(RESULT) = %HOUR(TS1) - %HOUR(TS2).

If %HOUR(TS2) > %HOUR(TS1)
;    then %HOUR(RESULT) = 24 + %HOUR(TS1) - %HOUR(TS2)
;    and %DAY(TS2) is incremented by 1.
```

时间标记的日期部分减法规则与日期减法相同。

时间标记的增加和减少：从时间标记加或减一个间隔，结果仍是时间标记。时间和日期的运算规则如前定义，不同的是小时的下溢和上溢要代入结果的日期部分，微秒的上溢代入秒部分。

2.2.3.25 用 OPNQRYF 进行随机处理

前面讲的多数例子都是顺序处理中用 OPNQRYF。在大多数情况下，也可用随机处理（如 RPG/400 中 CHAIN 和 COBOL/400 中 READ），然而，如果用组合或唯一键字功能就不能随机处理文件。

2.2.3.26 性能考虑

参看附录 D，那里有设计指导和性能优化的技术。

当 OPNQRYF 命令使用已存在的键字顺序访问路径就会获得最佳性能。例如，如果要选择 Code 字段为 B 的所有记录，并且已有一个路径存在，系统会利用这个路径进行选择（键字位置选择），而不是在运行时读和选择（动态选择）。

当有下面任何条件时，OPNQRYF 不会利用已存在的索引：

路径中的键字字段是从子串功能导出的。

路径中的键字字段是从连接功能导出的。

与查询有关的分类排序表中，（在 STRSEQ 参数规定），下面两项都为真时：

——是共享权顺序表

——访问路径与顺序表不符（分类顺序表或交替分配顺序表）

与查询有关的分类排序表中下面两项都为真时：

——是唯一权顺序表

——访问路径与顺序表不符（分类顺序或交替分配顺序表），并且下列之一

为真时：

——指定了顺序（KEYFLD 参数）

——有不用*EQ，*NE，*CT，%WLDCRD 或%UALUES 的记录选择

（QRYSLT 参数）

——有不用*EQ 和*NE 的连接选择（JFLD 参数）

OPNQRYF 处理的一部分是用来决定能满足你要求的最快方法。如果使用的文件很大，而且，大多数记录 Code 字段都为 B，用到达顺序比已有的键字访问要快，程序会得到相同记录，OPNQRYF 只有在已存在对 Code 的访问路径时才进行这种判断。通常，如果要得到文件中 20%或更多的记录，OPNQRYF 会忽略已存在路径而用到达顺序来读文件。

如果不存在对 Code 的访问路径，系统会读所有记录并把选择的记录传给程序，这时用到达顺序。

系统做选择要比程序更快，如果没有适当的访问路径存在，程序或系统对要处理的记录进行选择，让系统做选择，一般认为比把所有记录传给程序要快。当打开文件做更新更是这样，这是因为每个记录都要传到程序并对每个记录加锁（在更新情况下），让系统选择记录，就只有符合选择值的记录传到程序并加锁。

如果用 KEYFLD 参数为读记录请求指定的顺序，如果它有一个与键字规定的访问路径存在或类似的键字顺序访问路径存在，（如键字是包含你所规定的所有字段，在键字末尾加上另外字段），就会有较好性能。这对 GRPFLD 参数和 JFLD 中 TOFIELD 也同样有效。如果没有这样的访问路径，系统会建立一个路径，并在文件打开时维护它。处理不存在访问路径的文件中所有记录时，如果被处理记录数（不是文件的记录总数）超 1000 时，一般不如用全记录排序有效，尽管通常建立键字访问路径比排序快，但使用到达顺序的处理从总的作业时间上看可以帮助排序数据。如果存在一个有用的访问路径，使用路径比排序要快，假如这是处理记录最快的方法，可以用 OPNQRYF 中的 ALWCPYDTA(*OPTIMIZE)参数，允许系统进行全部记录排序。

如果不要读全部记录，并且 OPTIMIZE 参数是*FIRSTIO 或*MINWAIT，可以指定要取回的记录数，如果这个数字小于要返回的记录数，系统会选择较快的访问方法。

如果使用组合功能，在组合前选择(QRYSLT)比在组合后选择（GRPSLT）要有好性能，对包括合计功能的比较操作只能用 GRPSLT 参数。

对多数 OPNQRYF 命令，都使用新的或已存在的访问路径访问数据并传给程序。在某些情况下，系统必须建立临时文件，生成临时文件的规则是复杂的，下面是些典型情况：

当指定动态连接，并且 KEYFLD 从不同文件中描述键字段。

当指定动态连接，并且 GRPFLD 从不同的物理文件中描述键字段。

当同时指定 GRPFLD 和 REYFLD 参数，但两者不同时。

当在 KEYFLD 参数中规定的字段总数超过 2000 字节长。

当指定动态连接，并且 OPTIMIZE 的参数定*MINWAIT。

当指定用连接逻辑文件指定动态连接，且 JDFTVAL 不符合动态连接类型。

当指定逻辑文件，并且逻辑文件的格式引用了多个物理文件。

当指定了一个 SQL 视图，系统用临时文件保存视图的结果。

当指定了 ALWCPYDTA(*OPTIMIZE)参数，使用临时结果会改进查询性能。

当发生动态连接（JDFTVAL(*NO)），OPNQRYF 会重排文件，并把带有最小记录数的文件和最大记录数的文件连接。为避免 OPNQRYF 重排文件，规定了 JORDER(*FILE)，它强制用在 OPNQRYF 命令中规定的顺序连接文件。

2.2.3.27　分类排序表的性能考虑

2.2.3.27.1　组合、连接和选择

当使用已存在的索引，优化能保证选择、连接和组合的属性符合已存在的索引中的键字属性。同时与查询相关的分类排序表必须符合存在的索引有关的排序表（分类排序表或交替分配排序表），如果顺序表不匹配，就不能用已存在的索引。

然而，如果与查询有关的分类排序表是一个唯一权顺序表（包括*HEY），就可能有另外的优化。即使没有规定带有任何有下列功能的组合字段、选择或连接的顺序表，也要进行优化：*EQ　*NE　*CT　%WLDCRD　%VALUES。

这样做的优点是优化可以自由使用键字匹配字段且访问路径如下之一的任何已存在访问路径：

没有包含顺序表；

包含一个唯一权顺序表（这个表不一定符合与查询有关的唯一权顺序表）。

2.2.3.27.2　排序

为了排序字段，优化不能自由地使用已存在路径，除非优化做一个符合要求的排序，否则查询与索引的排序表必须匹配。当使用一个顺序时，在排序时转换，而不管优化对符合选择条件的已存在路径的自由使用权。

2.2.3.27.3　范例

下面例子中，假定对 JOB 字段已存三条路径，这三条路径用下面三个排序表：

1. SRTSEQ(*HEX)
2. SRTSEQ(*LANGIDUNQ) LANGID(ENU)
3. SRTSEQ(*LANGIDSHR) LANGID(ENU)

例1　不用顺序表的 EQ 选择。

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"') SRTSEQ(*HEX)
```

优化可以使用索引 1(*HEX)或 2(*LANGEDUNQ)

例2　用唯一权顺序表的 EQ 选择。

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"') SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

优化可用索引 1(*HEX）或 2(*LANGIDUNQ)

例3　用共享权顺序表的 EQ 选择。

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"') SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

优化只能用索引 3(*LANGIDSHR)

例4　用唯一权的 GT 选择。

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *GT "MGR"') SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

优化只能用索引 2(*LANGIDUNQ）

例5　用唯一权顺序表的连接选择。

```
OPNQRYF FILE((STAFF1)(STAFF2)) JFLD(1/JOB 2/JOB *EQ)
        SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

优化可用 1(*HEX)或 2(*LANGIDUNQ)

例6　用共享权顺序表连接选择。

```
OPNQRYF FILE((STAFF1)(STAFF2)) JFLD(1/JOB 2/JOB *EQ)
        SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

优化只能用索引 3(*LANGIDSHR)

例7　不用顺序表的排序

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"')
        SRTSEQ(*HEX) KEYFLD(JOB)
```

优化只能用索引 1(*HEX）

例8　用唯一权顺序的排序

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"')
        SRTSEQ(*LANGIDUNQ) LANGID(ENU) KEYFLD(JOB)
```

优化仅用索引 2(*LANGIDUNQ)

例9　用共享权顺序表排序

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"')
        SRTSEQ(*LANGIDSHR) LANGID(ENU) KEYFLD(JOB)
```

优化只能用索引 3(*LANGIDSHR）

例10 用 ALWCPYDTA 和唯一权顺序表排序

```
OPNQRYF FILE(STAFF) QRYSLT('JOB *EQ "MGR"')
        SRTSEQ(*LANGIDUNQ) LANGID(ENU) KEYFLD(JOB)
        ALWCPYDTA(*OPTIMIZE)
```

优化可以用索引 1(*HEX)或 2(*LANGIDUNQ)做选择。

用*LANGIDUNQ 顺序表做排序。

例11 不用顺序表组合

```
OPNQRYF FILE(STAFF) GRPFLD(JOB) SRTSEQ(*HEX)
```

优化可用索引 1(*HEX)或 2(*LANGIDUNQ)

例12 用唯一权顺序表组合

```
OPNQRYF FILE(STAFF) GRPFLD(JOB)
        SRTSEQ(*LANGIDUNQ) LANGID(ENU)
```

优化可用索引 1(*HEX）或 2(*LANGIDUNQ)

例13 用共享权顺序表组合

```
OPNQRYF FILE(STAFF) GRPFLD(JOB)
```

```
SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

优化可用索引 3 (*LANGIDSHR)

2.2.3.27.4 更多范例

下面例子中，前例 1-例 13 中在 JOB 字段访问路径（1，2，3）都还存在。

在例 14-例 20 中，在 JOB 和 SALARY 字段上是有访问路径（4，5，6），这些访问路径用下列排序表：

1. SRTSEQ(*HEX)
2. SRTSEQ(*LANGIDUNQ) LANGID(ENU)
3. SRTSEQ(*LANGIDSHR) LANGID(ENU)
4. SRTSEQ(*HEX)
5. SRTSEQ(*LANGIDUNQ) LANGID(ENU)
6. SRTSEQ(*LANGIDSHR) LANGID(ENU)

例14 使用唯一权顺序表对相同字段排序和组合。

```
OPNQRYF FILE(STAFF)
        SRTSEQ(*LANGIDUNQ) LANGID(ENU)
        GRPFLD(JOB SALARY) KEYFLD((JOB)(SALARY))
```

优化使用索引 5 (*LANGIDUNQ)来满足组合和排序的要求，如果索引 5 不存在，优化就用*LANGIDUNQ 建立一个顺序表索引。

例15 用 ALWCPYDTA 和唯一权顺序表对相同字段排序和组合。

```
OPNQRYF FILE(STAFF) ALWCPYDTA(*OPTIMIZE)
        SRTSEQ(*LANGIDUNQ) LANGID(ENU)
        GRPFLD(JOB SALARY) KEYFLD((JOB)(SALARY))
```

优化用索引 5 来满足排序和组合的要求。如果索引 5 不存在，优化会做下面之一：

—建立一个用*LANGIDUNQ 顺序表的索引

—用索引 4 (*HEX)完成满足条件的排序。

例16 用共享权顺序表对同一字段排序和组合。

```
OPNQRYF FILE(STAFF)
        SRTSEQ(*LANGIDSHR) LANGID(ENU)
        GRPFLD(JOB SALARY) KEYFLD((JOB)(SALARY))
```

优化用索引 6(*LANGIDSHR)满足组合和排序的要求，如果它不存在，优化就建立一个用*LANGIDSHR 顺序表的索引。

例17 用 ALWCPYDTA 和共享权表排序表对相同字段做组合和排序。

```
OPNQRYF FILE(STAFF) ALWCPYDTA(*OPTIMIZE)
        SRTSEQ(*LANGIDSHR) LANGID(ENU)
```

```
GRPFLD(JOB SALARY) KEYFLD((JOB)(SALARY))
```

优化用索引 6(*LANGIDSHR)满足组合和排序的要求，若它不存在，就建立用*LANGIDSHR 顺序表的索引。

例 18　用唯一权顺序表对不同字段排序和组合。

```
OPNQRYF FILE(STAFF)
        SRTSEQ(*LANGIDUNQ) LANGID(ENU)
        GRPFLD(JOB SALARY) KEYFLD((SALARY)(JOB))
```

优化用 4(*HEX)和 5(*LANGIDUNQ)满足组合需要。组合的临时文件满足排序要求。

例19 用 ALWCPYDTA 和唯一权顺序表对不同字段组合和排序。

```
OPNQRYF FILE(STAFF) ALWCPYDTA(*OPTIMIZE)
        SRTSEQ(*LANGIDUNQ) LANGID(ENU)
        GRPFLD(JOB SALARY) KEYFLD((SALARY)(JOB))
```

优化用索引 4(*HEX)和 5(*LANGIDUNQ)满足组合要求，然后做分类来满足排序要求。

例20 用 ALWCPYDTA 和共享权顺序表对不同字段组合和排序。

```
OPNQRYF FILE(STAFF) ALWCPYDTA(*OPTIMIZE)
        SRTSEQ(*LANGIDSHR) LANGID(ENU)
        GRPFLD(JOB SALARY) KEYFLD((SALARY)(JOB))
```

优化用索引 6(*LANGIDSHR)满足组合要求，然后做分类满足排序要求。

2.2.3.28　与其它数据库功能的性能比较

OPNQRYF 使用与逻辑文件和连接逻辑文件同样的数据库支持，因此，象建立键字访问路径或做连接操作的性能都相同。

用 OPNQRYF 命令做选择（用 QRYSLT 和 GRPSLT）与逻辑文件中选择/省略也很类似，主要区别在于 OPNQRYF 是系统决定是用路径选择还是用动态选择（类似 DDS 中逻辑文件指定省略或 DYNSLT 键字）来做为系统可用的访问路径以及为 OPTIMIZE 参数指定哪个值。

2.2.3.29　字段使用的考虑

当使用组合功能时，打开查询文件记录格式（FORMAT 中指定）的所有字段和所有键字字段（KEYFLD 指定），只能用组合字段、常量或聚合功能定义的组合字段（用 GRPFLD 指定）或是映象字段（在 MAPFLD 指定）。

聚合功能是：%AVG、%COUNT、%MAX（仅用一个操作数）、%MIN（仅用一个操作数）、%STDDEU、%SUM 和%VAR。

下面情况需要组合处理：

当在 GRPFLD 参数中指定组合字段名。

当在 GRPSLT 参数中指定组合选择值。

当在 MAPFLD 中指定映象字段用到定义中的聚合功能。

不能在 OPNQRYF 命令中指定那些包含在 FILE 规定格式中的。在用 DDS 生成文件时，用 N（不是输入也不输出）值定义的字段。只能用那些定义为 I（仅输入）或 B（既输入又输出）的字段。OPNQRYF 忽略指定了 N 的字段。

一般情况下，在打开查询文件记录中的字段都有与 FORMAT 中指定的字段相同属性。假如文件为包括输出、更新的选项（OPTION）打开时，在在 FORMAT 中指定 B 字段（既输入又输出）。

在打开查询文件中被改为 I（仅输入）时，OPNQRYF 就会发出一个信息。

如果请求连接处理或组合处理或者规定了 UNIQUEKEY 处理，所有查询记录中的字段都定义为只输入，从正在处理的文件中的只输入字段映象出的字段都认为只输入。用 MAPFLD 参数定义的字段在查询文件中通常被用作只输入。如果下面的情况为真，在 MAPFLD 中定义的字段被赋值成一个符合它的构成字段的值：

由于在这节中前面说明的情况中不需要只输入属性。

在 MAPFLD 中指定的字段定义表达式为一个字段名（不是操作符或内部函数）。

用在字段定义表达式中的字段存在于在 FILE 参数中定义的文件、成员或记录格式中
（不是 MAPFLD 定义的另外字段）。

基本字段和映象字段是兼容类型（映象字段不是数字和字符混合字段，除非是在相同长度和字符字段之间映象）。

如果基本字段是有小数位的二进制，那么映象字段也必须是二进制并有相同小数位。

2.2.3.30　在作业中文件共享的考虑

为了让程序能使用由 OPNQRYF 命令建立的 ODP，程序必须共享查询文件，如果程序不是用共享打开文件，实际上是以原始编译时的方式打开文件（不是用 OPNQRYF 建立的查询路径），根据下面情况，程序将共享查询 ODP：

应用程序必须共享打开文件，当查询到的第一个或唯一的成员（在 FILE 参数中指定）时有 SHARE(*YES)属性，程序就会共享它。如果第一个或唯一的成员有 SHARE(*NO)属性，就必须在调用程序之前用 OVRDBF 命令中的 SHARE(*YES)来规定。

由程序打开的文件名必须与 OPNQRYF 中的文件名相同，在程序中指定的文件名和成员名与查询的第一个或唯一的成员名相同时（用 FILE 参数），就是这种情况。如果第一个或唯一成员有不同的名字，就要用 OVRDBF 命令重编译程序，把文件名换成第一个或唯一的成员名。

程序必须在与查询 ODP 范围内的活动组中运行。如果查询 ODP 的范围是作业，就可以在这个作业中的任何活动组中运行。OPNQRYF 命令从不共享在作业或活动组中已存在 ODP。如果在打开文件要求中有相同库名、文件名和成员名，打开查询文件就会失败并有一个错误信息，当下面情况为真时，也是这样：

- 在 OPNQRYF 中指定 OPNSCOPE(*ACTGRPDFN)或 OPNSCOPE(*ACTGRP)，并且
  ODP 的范围在 OPNQRYF 运行的同一个活动组或作业中。
- 在 OPNQRYF 中指定 OPNSCOPE(JOB)，并且 ODP 的范围在与 OPNQRYF 运行时

的同一作业中。

后继的共享打开都保持 OPNQRYF 运行时的相同功能（例如 SEQONLY）。详细信息参阅 2.1.5 的内容。

2.2.3.31　检查记录格式描述是否修改过

如果指出要做记录格式级别检查，就要去检查打开查询文件记录格式的级别。（用 FORMAT 参数指定），而不管程序编译时记录格式的级别。当程序共享以前打开过的查询文件时就会发生这样情况。如果遇到下面情况，程序会检查共享打开的记录格式级别。

- 第一个或唯一查询文件必须有 LVLCHK(*YES)属性（在 FILE 中指定）。
- 没有对第一个或唯一的查询文件做 LVLCHK(*NO)的覆盖。

2.2.3.32　其他运行时考虑

覆盖可以改变由 OPNQRYF 处理的文件名、库名、成员名（但，除了 TOFILE、MBR、LVLCHK、INHWRT 或 SEQONLY 以外的其它 OVRDBF 参数值可都被 OPNQRYF 忽略）。

如果对第一个或唯一的查询成员做覆盖名字的修改，那么其余覆盖的内容将针对这个新名字，而不是在 OPNQRYF 的 FILE 中指定的名字。

2.2.3.33　拷贝打开的查询文件

CPYFRMQRYF 命令可以从打开的查询文件拷贝到一个文件或打印记录清单。任何打开查询文件（除 DDM 文件），用 OPNQRYF 的 FILE 参数指定的输入、更新或所有操作值都可以用 CPYFRMQRYF 拷贝。CPYFRMQRYF 不能拷贝逻辑文件，详细内容请看数据管理手册。

虽然 CPYFRMQRYF 使用打开查询文件的 ODP，但它不打开文件，这样就不必为要拷贝的文件指定 SHARE(*YES)。

下面是使用 OPNQRYF 和 CPYFRMQRYF 的例子。

例1　用记录子集建立文件

假如要用 CUSTOMER/ADDRESS 文件中 STATE 字段为的值为 Texas 的记录生成一个文件，可以指定：

```
OPNQRYF FILE(CUSTOMER/ADDRESS) QRYSLT('STATE *EQ "TEXAS"')
CPYFRMQRYF FROMOPNID(ADDRESS) TOFILE(TEXAS/ADDRESS) CRTFILE(*YES)
```

例2　打印选择的记录

假定要从 FILEA 中打印 CTTY 字段值为 Chicago 的所有字段。可以指定：

```
OPNQRYF FILE(FILEA) QRYSLT('CITY *EQ "CHICAGO"')
CPYFRMQRYF FROMOPNID(FILEA) TOFILE(*PRINT)
```

例3　把记录子集拷贝到软盘上

假定要把 FILEB 中所有 FILEDB 值为 10 的记录拷贝到软盘上，可以指定：

```
OPNQRYF FILE(FILEB) QRYSLT('FIELDB *EQ "10"') OPNID(MYID)
CPYFRMQRYF FROMOPNID(MYID) TOFILE(DISK1)
```

例4　生成一个动态连接输出的拷贝

假定要生成一个由 FILEA 和 FILEB 的格式和数据的物理文件，文件包含下列字段：

FILEA	FILEB	JOINAB
Cust	Cust	Cust
Name	Amt	Name
Addr		Amt

连接字段是两个文件中都有的字段 Cust，为了连接文件，并把结果放在新的物理文件 MYLIB/FILEC 中，可以指定：

```
OPNQRYF FILE(FILEA FILEB) FORMAT(JOINAB) +
  JFLD((FILEA/CUST FILEB/CUST)) +
  MAPFLD((CUST 'FILEA/CUST')) OPNID(QRYFILE)
CPYFRMQRYF  FROMOPNID(QRYFILE) TOFILE(MYLIB/FILEC) CRTFILE(*YES)
```

文件 MYLIB/FILEC 由 CPYFRMQRYF 命令生成，虽然一些属性可能改变，但这个文件仍有 FILEA 中的属性，文件格式象 JOINAB，文件会包含 FILEA 和 FILEB 中用 Cust 字段连接的数据，库 MYLIB 中的 FILEC 可以象其它物理文件一样用 CL 命令处理，如 DSPPFM。也可用工具处理，如 QUERY。关于 CPYFRMQRYF 和其它拷贝命令的其它内容，请看数据管理手册。

2.2.3.34　使用 OPNQRYF 命令的典型错误

必须正确规定 OPNQRYF 命令的几个功能，程序才能得到正确的结果。如果出现问题，DSPJOB 是最有用的，这个命令支持打开文件选项和覆盖文件选项。如有问题应两者都查看。

下面是最常见的问题以及如何改正：

• 共享 ODP。OPNQRYF 通过共享 ODP 来操作，为了正确处理文件，成员必须对共

享 ODP 打开。如果有问题，用 DSPJOB 中打开文件选项来判定成员是否打开并有

共享 ODP。

• 通常有二个原因使文件不能打开：

—处理的成员必须是 SHARE(*YES)，或用 OVRDBF 或永久改变这个成员

—文件已关闭。要从调用堆栈中最高级的默认活动组中的程序来运行有 OPNSCOPE(*ACTGRPDFN)或 TYPE(*NORMAL)的 OPNQRYF 命令，而不是从得到错误信息的程序来运行，或者简单使用 RCLRSC 命令。因为是从比运行 RCLRSC 命令程序高的调用堆栈的程序打开的文件，因而关闭查询文件。如果打开查询文件已经关闭，你必须再次运行 OPNQRYF 命令。注意，在 V2.3 版本以前中使用带 TYPE（*NORMAL）的 OPNQRYF 命令时，即使在运行 RCLRSC 的同一个程序中打开的文件也会被关闭。

级别检验。通常都使用级别检验，这样可以保证程序与在编译时相同的记录格式运行，如果发生级别检验方面的问题，通常为以下原因：

—在程序生成后记录格式有了改变，重新生成程序可以解决这个问题。

—覆盖使程序指向一个错误文件，用 DSPJOB 中文件覆盖选项来确认规定的覆盖

是否正确。

—需要 FORMAT 参数，但没指定或者指定有错。当用 FORMAT 处理一个文件时，

必须保证：

—使用 TOFILE 参数的 OVRDBF 命令，要说明在 OPNQRYF 中 FILE 参数的

第一个文件。

—FORMAT 指定的文件要包含生成程序的格式。

—FORMAT 参数用来处理不同文件中的格式（如组合处理），但 OVRDBF 中不

需 SHARE(*YES)。

处理的文件已到文件末，正常使用 OPNQRYF 是顺序处理文件。这样文件就处理一次，如果要再处理一次，就会因文件已到末尾而得不到任何记录。要重新从头处理文件，必须或者再用 OPNQRYF，或者在处理前重定位文件。

可以用 POSDBF 命令或高级语言来重定位数据文件。

记录不存在。可能由于使用了 FORMAT 键字，但又没在 OVRDBF 命令中指定。

语法错误。系统在 OPNQRYF 中发现规定项错误。

非法操作。在查询定义中不包含 KEYFLD 参数，但高级语言程序却试图用键字段读查询文件。

非法选项。高级语言程序试图在当前记录前读一个记录或逻辑文件位置，而查询文件或者用组合功能，或者用唯一键字选项或者用 SQL 语句的区分选项。

第七章　基本数据库文件操作

在这章中，讨论基本数据库操作，包括：在文件中定位，从文件中读记录，更新文件记录，向文件中加记录，从文件中删除记录。

2.3.1 文件中定位

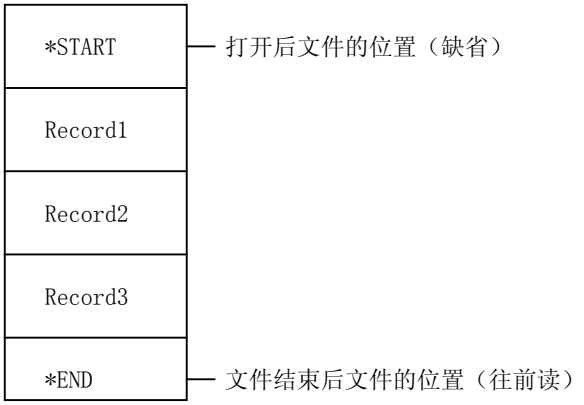
作业中的文件打开后，系统维护一个作业中的文件位置，用它来处理文件。例如，如果程序要依次读下一条记录，系统就用文件位置找到返给程序的记录，然后把文件位置定在刚读过的记录上，以便另外的读操作能返回正确的记录。系统对每个作业保存所有文件位置的轨迹。另外，每个作业在同一文件中可以有多个位置。

文件的第一个位置是由 OVRDBF 命令中的 POSITION 参数设定。如果没有使用 OVRDBF 或使用 POSITION 的默认位置，那么文件位置就定在成员访问路径中第一个记录之前。

程序可以用相关操作改变文件当前位置（如 RPG/400 中 SETLL 或 COBOL/400 中 START），也可以通过 CL 命令（POSDBF）来改变。

注：只有在文件被重新打开后，OVRDBF 的定位才有效，由于用 CL 程序只能打开一个文件，这个命令不能用在同一个通过 RCVF 命令读的 CL 程序中。

在文件末，当读完最后一条记录时，根据程序是向前或是向后读，成员的位置被定为 *START 或*END 位置。



只有读操作，强制结束操作，高级语言定位操作，或用 CL 的修改定位命令才能改变文件位置。增加、更新、删除操作不能改变文件位置。在读操作后，文件定位在一个新的记录上，记录返给程序。在读操作完成后，文件就定位在刚返给程序的记录上。如果用输入打开成员，强制数据结束操作就会把文件定位在文件最后一条记录后(*END)，并发出文件结束信息。

对顺序读操作，当前文件位置用来查找访问路径上的下一个或上一个记录，对键字读或相关记录号读操作中，不使用文件位置。如果在打开时规定 POSITION(*NONE)，不设置文件起始位置，这时，如果要顺序读，就必须在程序中建立文件位置。

如在 OVRDBF 中设置文件末延迟时，在程序读最后一个记录后，文件不定位在*START 或*END 上，仍定位在最后一个记录，带有文件末延迟的文件只在强制结束（FEOD）或控制作业结束时，才定位在*START 或*END 上，详细内容请看 2.3.2.3。

可以对那些用 OPNQRYF 或 OVRDBF 命令打开的文件。使用 POSDBF 命令来定位或修

改当前文件位置。

2.3.2 读数据库记录

AS/400 系统提供了几种读记录的方法。下面详细介绍这些方法（某些高级语言不支持所有系统做的读操作，请看有关的高级语言手册）。

2.3.2.1 用到达顺序访问路径读记录

系统根据在程序中规定的操作读记录。如果在程序中文件定义用到达顺序或是文件在 OPNDBF 或 OPNQRYF 命令中指定了忽略键字顺序访问路径时，允许进行这样的操作，请看 2.1.2.4 的内容。

注：高级语言程序可能不支持下面讲的所有的读操作，请参考有关语言的手册。

读下一个：定位并读到达顺序访问路径上的下一个未删除记录，要跳过在当前位置和下一个活动记录之间的删除记录。（RPG/400 中的 READ 和 COROL/400 中的 READ NEXT 语句是这种操作的例子）。

读上一个：定位并读到达顺序访问路径中的前一个活动记录，跳过在当前位置和上一个活动记录间的删除记录（RPG/400 中的 READP 和 COBOL/400 中的 READ PRIOR 是此操作的例子）。

读第一个：定位并读到达顺序访问路径的第一个活动记录。

读最后一个：定位并读到达顺序访问路径的最后一个活动记录。

读同一记录：读文件当前位置的记录，并且位置不变。

由相关记录号读：定位文件并读由相关记录号指定的到达顺序访问路径上的记录。相关记录号必须指向一个活动记录并小于或等于成员中的最大活动记录号，这个操作也读由当前位置加上或减去一个数的到达顺序路径上的记录。（RPG/400 的 CHAIN 和 COBOL/400 中的 READ），如果文件由相关记录来处理，那么在生成或修改文件时要考虑重用已删除记录的问题，请看 2.1.2.3 的内容。

2.3.2.2 用键字顺序访问路径读记录

系统根据程序中指定的语句进行下列读操作，这些操作能用键字顺序访问得到数据库中的记录。

当使用键字路径时，读操作不能指向已删除记录占用的空间。

注：有的高级语言可能不支持所有的下面操作，参看各自的语言手册。

- 读下一个：读键字访问路径中的下一个记录。如果指定了记录格式名，这个操作就读符合格式的键字路径上的下一个记录，文件当前位置用来定位下一个记录（RPG/400 中的 READ 和 COBOL/400 中的 READ NEXT 就是这种操作）。
- 读上一个：读键字访问路径中的上一个记录。如果指定了格式名，就从符合的格式的键字路径上读前一个记录，当前文件位置用来定位上一个记录（RPG/400 中的 READP 和 COBOL/400 中 READ PRIOR 为这种操作）。
- 读第一个：读键字访问路径上的第一个记录。如果指定了记录格式名，就从符合的格式中读访问路径上的第一个记录。
- 读最后一个：读键字访问路径上的最后一个记录。如果指定了记录格式名，就从符合的格式中读访问路径上的最后一个记录。
- 读同一个：读当前位置上的记录，文件位置不变。
- 用键字读：用键字值读记录，可以指定键字操作为相等、相等或之后、相等或之前、读前一个等键字、读下一个等键字、之后、之前。如果指定了记录格式名，系统就

查找这个格式名和键字值，如果没指定记录格式名，就用键字值查找整个键字路径。如果文件的键字定义包含多个键字段，可以指定部分键字（用键字段数或键字长度规定），这允许做通配键字查找，如果程序没指定键字段数，系统使用默认数。这个默认数取决于是否由程序传递了记录格式名。如果传递了记录格式名，键字段的默认数在是格式中定义的键字段总数。如果没有传递记录格式名，就用路径中所有格式中最大键字段数，程序必须有足够键字数据来满足系统设定的值（RPG/400 中的 CHAIN 和 COBOL/400 中的 READ 就是这种操作）。

- 用相对记录号读：对一个键字顺序访问路径，也能用相对记录号读，即使成员以键字路径打开的，相对目录号也是用到达顺序。如果成员有多个记录格式，必须指定记录格式名，这时，可以请求与记录格式名相符的物理文件成员中的记录。如果成员有选择/忽略语句并且由相对记录号指定的记录被键字忽略，就会给程序送一个错误信息，并且不允许这个操作。操作完成后，文件位置固定在由相对记录号指定的在键字路径中键字值的物理记录上。这个操作也能读由当前文件位置加或减去一个记录数所指的记录。（RPG/400 中的 CHAIN 和 COBOL/400 中的 READ 就是这种操作）。
- 当逻辑文件共享多键字访问路径的读操作：
  当在逻辑文件的 DDS 没有指定 FIFO 和 LIFO 或 FCFO 时，逻辑文件隐含共享比文件生成时多的键字访问路径。这种共享已存在的路径中某些键字的情况可能导致能察觉的数据库读错误，问题表现为：
  要读的记录没有返给程序
  记录被多次返给程序

实际情况是你的程序或另外活动的程序更新了某些共享路径中的物理文件键字段，但它不是程序实际上正在使用的逻辑文件的键字（被更新的键字数超过了程序中使用的逻辑文件所知的键字数），由你的或另外的程序修改的实际键字段经常出现上面的结果。部分共享键字顺序路径能造成更新的物理文件字段键字超出逻辑文件所知的键字数就可以产生上面类似的后果。如果想避免这种情况，就要在逻辑文件的 DDS 中加上 FIFO、LIFO、FCFO 键字，并重新组成逻辑文件。

2.3.2.3 当文件结束时等待更多记录

文件结束延迟是一种在文件结束时仍能顺序读文件（逻辑、物理文件）的方法。当顺序读文件发生文件结束时，（下一个/前一个记录），但已指定了文件结束延迟，（用 OVRDBF 的 EOFDLY 参数），系统就会等待规定的时间。延迟时间结束，会作另一个读操作来制定是否有新记录加到文件中。如果有新记录，会正常处理记录直到又发生文件结束。如果没有新记录加到文件中，系统会再等待指定的时间。当在一个带有选择/忽略的逻辑文件中指定文件结束延迟时应特别注意，这时，一旦文件结束，系统只会取回那些符合逻辑文件选择/忽略条件的加到基于的物理文件中的那些记录。

同样，在一个键字顺序访问路径的文件中使用文件结束延迟时，也要特别注意，它是用键字顺序路径打开的，这时，一旦到了文件末，系统仅取回那些加到物理文件中并符合键字要求的记录。

例如，在一个按升序排列的数字键字段文件中使用文件结束延迟，应用程序用键字路径读记录。程序做读下一个操作，并读到键字值为 99 的记录，程序进行另一个读下一个操作，但没有记录可读，这样，系统会在规定的延迟时间后再读文件，如果增加或更新了一个记录，且记录的键字值小于 99，系统就不取它。如果增加或更新了一个记录，记录的键字值大于或等于 99，系统才取出记录。

如果延迟时间大于或等于 10 秒，作业就会适时从主存中移出。如果不想移出作业，在作业使用的 CRTCLS 命令的 CLASS 指定 PURGE(*NO)。

要显示哪个作业有结束延迟，可以用 WRKACTJOB 命令显示那些等待记录作业的文件结束等待或文件结束活动的级别。

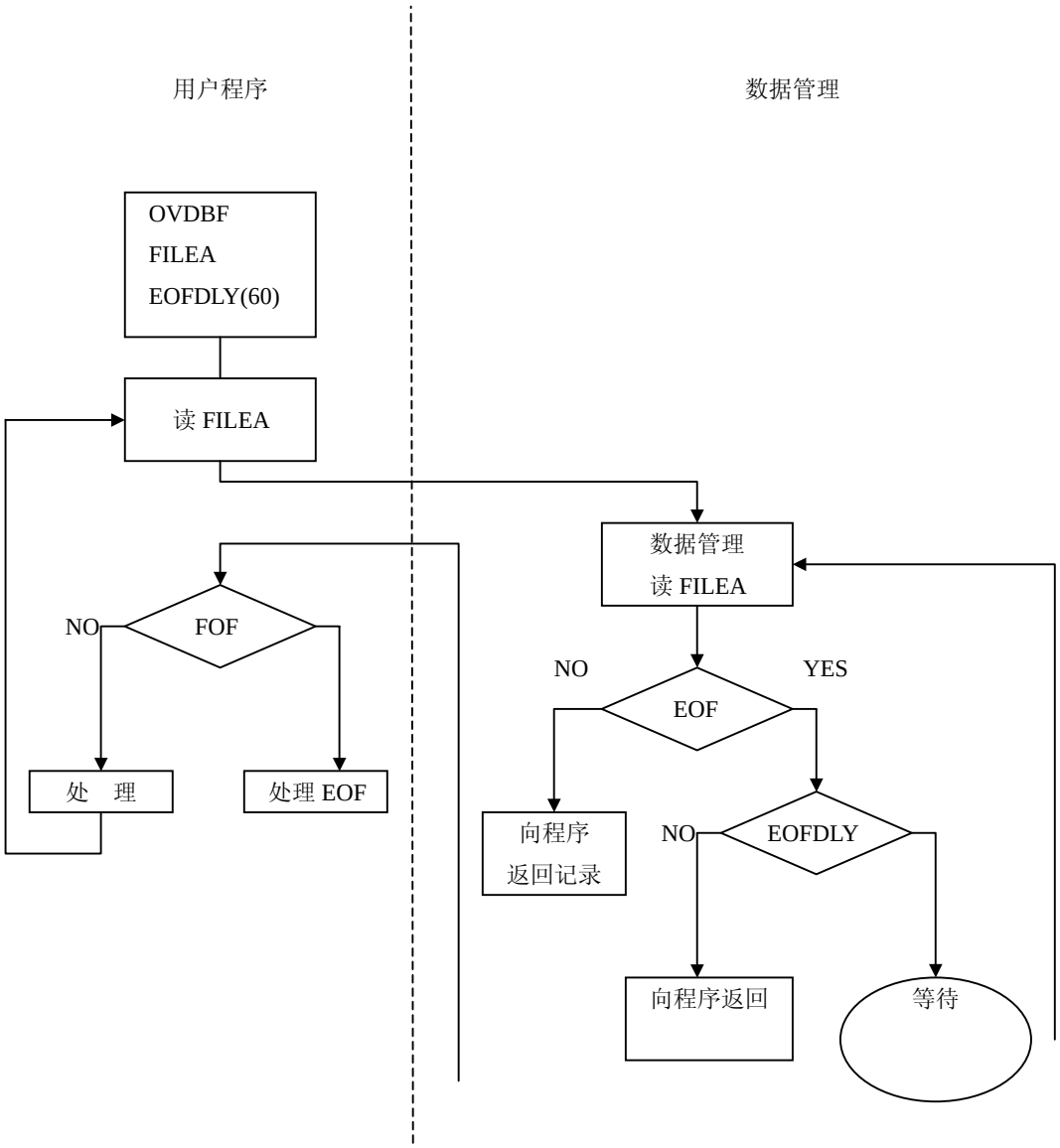
如果作业使用文件结束延迟和落实控制，能对记录加锁的时间长一些，这增加了另外作业要访问同一记录但被拒绝的机会。因此在同一作业中使用落实控制和文件结束延迟的时候要小心。

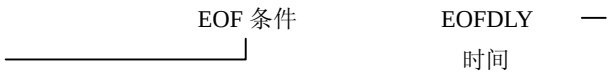
如果文件共享，指定文件结束延迟的 OVRDBF 要在文件第一次打开前使用，在共享文件打开后，就忽略这种复盖。

有几种方法可以结束因在 OVRDBF 中规定结束延迟而等待多个记录的作业：

- 可以由应用程序向有延迟结束的文件写入一个记录作为最后一条记录，这时程序可以指定 FEOD 操作，FEOD 使程序完成正常的结束文件操作。
- 用 ENDJOB 中的 OPTION(*CNTRLD)来控制结束作业，其中 DELAY 参数的值要大于 EOFDLY 的时间。规定的 DELAY 参数时间必须允许 EOFDLY 的时间执行完，把加到文件的新记录处理完和程序把文件结束处理完。在新记录处理完后，系统标识文件结束，做正常的文件结束工作。
- 在 ENDJOB 中规定 OPTION(*IMMED)，不做文件结束处理。
- 如果作业为交互的，用系统请求功能来结束作业。

下面是文件结束延迟的例子：





实际的 EOFDLY 参数的操作比上图给出的更复杂，因为 ENDJOB 中 OPTION(*CNTRLD) 参数会强制做一个真正的文件结束。

不论何时，有新记录加入文件时作业都不活动，作业是在延迟时间结后才活动。在作业活动后，系统检查是否有新记录，如果有，程序就得控制并处理所有新记录，然后再写。因此，作业在处理时，采取批处理的特性。当批处理完成时，作业变为不活动，如果延迟时间短，就会因为启动作业并且检查新记录而导致过多的系统消耗，正常情况下在等待延迟时间时，只有很少的消耗。

注：当一个作业处于长时间等待状态不活动时，这意味着它已从活动级释放出来，在等待时间到后，系统会重调度到活动级。（详细内容请看工作管理的书）。

2.3.2.4 释放加锁记录

系统会在记录更新、删除或读文件中另一个记录时释放加锁记录。有时不想做这些操作但释放加锁记录。某些高级语言支持释放加锁记录的操作，请参阅相关的语言手册。

注：如果作业在落实控制下运行，加锁的规则是不同的。详细内容请看备份与恢复手册。

2.3.3 更新数据库记录

更新操作允许你修改物理文件或逻辑文件中的记录。（RPG/400 中的 UPDAT 和 COBOL/400 中的 REWRITE 是这种操作）。

在更新一条记录前，必须先读再加锁。在 2.3.2.1 和 2.3.2.2 中所列出的读操作规定更新选项，就会加锁。

如果在几种读操作中使用更新选项，每个操作在定位加锁新记录之前要释放前一个加锁记录的锁。当做更新时，系统假定要更新当前加锁的记录。因此在更新操作时，不必指定要更新的记录，在完成更新后，系统释放锁。

注：在落实控制下运行，加锁的规则不同，请看备份与恢复手册。如果更新操作修改的是规定了立即维持路径中的键字段，在程序允许时，访问路径也更新。（某些语言不允许在更新操作中修改键字段）。

如果对一个已加锁记录请求读操作，且作业在*ALL 或*CS 的落实控制级下运行，就必须要等释放锁或是在生成或覆盖命令中指定 WAITRCD 的时间。如果 WAITRCD 等待时间到，而记录锁没有释放就会发生程序异常，并向作业发送有关的文件、成员、相对记录号的信息。如果读操作的作业没有在*ALL 或*CS 的落实控制下运行，作业可以读为更新加锁的记录。

如果要更新的文件中有一个更新触发程序，就会在更新记录前或后调用触发程序，请看第 1-7 章的内容。

如果更新的文件有引用约束，更新操作也会受影响，请看第 16 章的内容。

2.3.4 增加数据库记录

写操作用来向物理文件成员中添加新记录（RPG/400 中的 WRITE 和 COBOL/400 中 WRITE 都是这种操作），可以往物理文件或基于物理文件的逻辑文件中添加记录，当使用多格式逻辑文件时，必须给出要添加记录的物理文件成员。

新记录通常加到物理文件成员的末尾，并分配给它下一个可用相关记录号（包括已删除

记录号），有些高级语言允许你在已删除记录的位置上写记录。请看相应的高级语言手册。

如果要加记录的物理文件重用已删除记录空间，系统会向被删除记录的位置上插入记录，当生成或修改可重用记录空间的文件之前，要清楚有哪些限制及使用的关键，来决定不可以重用已删除记录的空间，请看 2.1.2.3 的内容。

如果往键字访问路径的文件中加记录，新记录立即会加到记录键定义的位置上。如往有选择/忽略选项的逻辑文件成员中加记录，忽略值可以不让记录出现在成员的访问路径中。

如果要加记录的文件带有插入触发器，要在插入记录前或后调用触发程序，请看第 17 章的内容。

如果要加记录的文件与引用约束有关，插入记录要受影响，请看第 16 章的内容。

CRTPF 和 CRTSRCPF 命令中的 SIZE 参数决定可以向物理文件成员中加入多少记录。

2.3.4.1　在多格式文件中指出要加记录的格式

如果程序在往数据库中加记录时不用格式名而用文件名，并且使用的文件是一个多格式逻辑文件，那就要写一个格式选择程序来指定要把记录放在数据库的什么地方，格式选择程序可以是 CL 程序也可是高级语言程序。

如果下面条件都成立，必须用格式选择程序：

文件不是连接逻辑文件或窗口

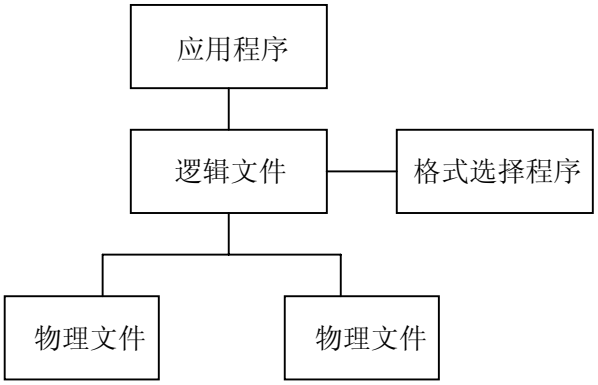
逻辑文件基于多个物理文件

程序在加操作中用文件名而不是用记录格式名

这时如果不用格式选择，程序在向数据库加记录时会错误的结束。

注：格式选择程序只能来选择格式，而不能选择成员。

当程序向数据库中加记录时，系统调用格式选择程序，它来检查记录并确定要使用的格式，系统就用这个格式做加记录操作。



下面给出用 RPG/400 写的格式选择程序

```
C           *ENTRY    PLIST
C                     PARM           RECORD 80
C* The length of field RECORD must equal the length of
C* the longest record expected.
C                     PARM           FORMAT 10
C                     MOVELRECORD    BYTE    1
C           BYTE      IFEQ 'A'
C                     MOVEL'HDR'     FORMAT
C                     ELSE
```

```
C                   MOVEL'DTL'      FORMAT
C                   END
```

格式选择程序用第一个参数获得记录，因此这个字段必须定义成这个格式允许的最大的记录长度。它能存取记录的任何部分来决定记录格式名。在此例中，它检查记录的第一个字母是否为 A，如果为 A，就把记录格式名 HDR 移到第二个参数 FORMAT 中。不为 A，就把 DTL 移到 FORMAT 中。

格式选择程序用 FORMAT（10 个字符长），系统传送记录格式名，当系统知道格式名后，就向数据库中加记录。

如果是下面这些情况，可以不用格式选择程序：

- 仅做更新：对于更新程序已获得记录，系统就知道记录来自哪个物理文件。
- 对增加或删除操作，程序规定了记录格式名。
- 程序所用的所有记录都在一个物理文件中。

要生成格式选择程序，用相应语言的生成命令，在生成命令中不能指定USRPRF(*OWNER)格式选择程序必须在主人用户配置文件下运行。

另外，安全和完整性，因性能会受影响，不要在格式选择程序中有调用或输入/输出操作。

格式选择程序名在 CRTLF、CHGLF 或 OVRDBF 中的 FMTSLR 参数给出。在生成文件时，格式选择程序不一定已经存在，但在程序运行时，一定要存在。

2.3.4.2 使用强制数据结束操作

强制数据结束（FEOD）可以使程序对文件的修改强加到辅存中。通常是由系统决定何时强写辅存，然而自己可用 FEOD 操作来保证辅存的强行修改。

如果此时文件是用输入打开的，FEOD 还可以允许把文件定位到文件头或尾。*START 把当前打开的文件的起始位置定在成员的第一个记录之前。（第一个顺序读操作读当前成员的第一个记录）。如果 OVRDBF 命令的 MBR(*ALL)起作用，读前一个的操作就读出上个成员的最后一条记录。如果此时不存在上一个成员，那么就会发文件结束信息（CPF5001）。*END 把位置定在最后一条记录之后，(读前一个的操作读出当前成员的最后一条记录。)

如果 OVRDBF 命令中*MBR(ALL)起作用，读下一个操作会读到下个成员的第一条记录。此时如果没有下一个成员，会发出文件结束结束信息（CPF5001）。

如果文件有删除触发器，就不允许强制数据结束操作，请看第 17 章的内容。如果文件是引用约束的一部分，也不允许 FEOD 操作，请看第 16 章内容。

FEOD 操作的详细内容，请看相应的高级语言指南。

2.3.5 删除数据库记录

删除操作允许删除已存在的记录（RPG/400 的 DELET 和 COBOL/400 中的 DELETE 为此操作）。要删除一个记录，要先读并加锁，要加记录锁，用 2.3.2.1 或 2.3.2.2 中列出的任何读操作的更新选项，删除加锁记录和标识哪个记录要删的规则与更新操作是一样的。

注：某些高级语言并不要求先读记录，只须在删除语句中指明要删除的记录，RPG/400 可以做到这点。

当一个记录删除后，物理记录做删除标记，通过逻辑文件删除时也同样，删除后记录不能再读，并从所有包括这个记录的访问路径中移出，已删除记录的相对记录号保持不变，文件中所有的记录号都不变。

删除记录的空间保存在文件中，直到发生下面情况，才可以用：

• 运行 RGZPFM 命令用来压缩并释放成员中的空间，请看 3.1.2.3 的内容。

• 程序用相对记录号向文件写记录，所用的记录号与删除记录号相同。

注：如果文件规定了重用记录的属性，系统会自动重用删除记录空间，请看 2.1.2.3 的内容。

系统不许从已删除的记录中获取数据，但可以往这个位置上写记录（用相对记录号），写操作用新记录代替已删除记录。详细内容请看高级语言指南。

用已删除记录的相对记录号写记录时，记录号必须在成员文件中存在。在高级程序中可以用删除操作删除记录，也可以用初始化物理文件成员命令（INZPFM）来删。INZPFM 命令可以初始化整个文件来删除记录，请细内容请看 3.1.2.1 的内容。

如果所用文件有删除触发器，在删除记录前或后调用触发器，请看 17 章内容。

如果文件是引用约束的一部分，删除记录可能受影响，请看 16 章内容。

第八章　关闭数据库文件

在程序处理完数据库文件成员后，必须关闭它。关闭文件中断程序与文件的联系。关闭操作释放所有记录锁和所有文件成员锁。通过 ODP 强行修改辅存，然后去掉 ODP。（当共享文件关闭，ODP 仍保持打开，这个功能是不同的。关于共享文件，请看 2.1.5 的内容）。

在程序中关闭文件的方法有：

高级语言中关闭语句

CLOF 命令

RCLRSC 命令

大部分高级语言允许你指定要关闭文件，用语言关闭文件，参考相应的语言指南。

可以用 CLOF 关闭由 OPNDBF 或 OPNQRYF 命令打开的文件。

也可运行 RCLRSC 命令来关闭文件。RCLRSC 释放所有的锁（除了在落实控制下的加锁记录）。强行修改辅存，然后去掉 ODP。可以用 RCLRSC 使调用程序关闭被调用程序的文件。（例如，被调用程序返回时没关闭文件，调用程序可以关闭被调用程序的文件）。通常的关闭操作却通过高级语言关闭操作或通过 CLOF 命令来完成。详细内容请看 ILE 概念。

如果作业正常结束（比如用户注销），而所有与作业相关的文件未关闭，系统会自动关闭相关打开的文件，强写辅存，并且释放所有的记录锁。如果作业异常结束，系统也这样做。

第九章　处理程序中数据库文件的错误

在处理数据文件中发生的错误条件会向处理文件程序的信息队列发送信息，或向系统操作员信息队列中发送查询信息。另外，文件错误和诊断信息或系统操作信息会做为返回码和状态信息出现在文件反馈区内。

如果程序语言允许你监控信息，可以确定要监控哪条信息，下面是可监控信息的例子（详细内容可看语言指南或 CL 参考）。

信息标识	说　明
CPF5001	文件结束
CPF5006	没找到记录
CPF5007	删除记录
CPF5018	达到最大文件尺寸
CPF5025	试图通过*START 或*END 读
CPF5026	重复键字
CPF5027	记录被另外的作业使用
CPF5028	记录键修改
CPF5029	数据映象错误
CPF502B	触发程序错误
CPF502D	违反引用限制
CPF5030	成员部分损坏
CPF5031	超出记录锁的最大数目
CPF5032	记录已经分配给作业
CPF5033	选择/省略错误
CPF5034	在另外的成员访问路径中有重复键字
CPF503A	违反引用限制
CPF5040	没取得省略记录
CPF5072	成员中的连接值被修改
CPF5079	超出落实控制资源的限制
CPF5084	非落实控制键的重复键字
CPF5085	在另外的访问路径中非落实控制键的重复键字
CPF5090	唯一访问路径问题防止访问另外的成员
CPF5097	键字映象错误

注：要显示完整的信息描述，用 DSPMSGD 命令。

如果你不监控信息，系统处理这些错误，同时也设置程序中相关返回码。根据这些错误，系统可以结束作业或向操作员请求下一步操作。

如果在处理数据文件成员时，有信息送给程序，文件位置不丢失，并保持发送信息之前的位置，除非有下列情况发生：

- 到达文件末，给程序发送信息，文件位置在*START 或*END。
- 在读操作中发出转换映象信息，文件定位在包含产生信息数据的记录上。

第三部分　数据库文件管理

这一部分的各章节主要讲述了管理数据库文件的有关内容：

- 增加新成员
- 改变已存在成员的属性
- 对成员重新命名
- 从一个数据库文件中移出成员

只针对物理文件的成员操作：

- 初始化数据
- 清除数据
- 重新组织数据
- 显示数据

此外，还介绍以下内容：

- 修改数据库文件的描述和属性(包括修改文件描述中字段的影响)
- 修改物理文件的描述及属性
- 修改逻辑文件的描述及属性
- 使用数据库属性和交叉引用信息

这一部分还包括有关数据库文件的信息，如：

- 属性
  —约束
  —触发器
- 字段的描述
- 字段间的联系
- 程序使用的文件
- 系统交叉引用文件
- 如何写一个可直接输出到数据库文件的命令

这一部分还会帮助你如何在系统故障时恢复数据库文件：

- 保存和重存
- 日志
- 使用辅助存贮器
- 使用落实控制

此外，还将介绍访问路径的恢复，包括：

- 重建访问路径
- 做访问路径的日志

在关于源文件的部分将讨论源文件的概念及使用源文件的原因，并介绍如何建立源文件、如何向源文件中输入数据以及使用源文件所在系统中生成目标的方法。

第十章　数据库成员管理

在你对一个文件执行任何输入或输出操作之前，必须保证该文件中至少有一个成员。一般说来，数据库文件中只有一个成员，该成员是在此文件生成时(同时)生成的。该成员与这个文件同名，除非你对它重新命名。由于绝大多数数据库的操作都假设要使用的成员是文件中的第一个成员，又由于绝大多数的文件只有一个成员，因此一般不考虑说明成员名。如果一个文件中有多个成员，则每个成员作为文件中数据的一个子集。这样更容易对数据分类。例如，定义一个接收支票文件，你决定在文件中将数据保存一年，但却要每个月处理一次数据。例如，生成一个有 12 个成员的物理文件，每个月份一个名字，那么，可以分别处理每个月的数据，也可以处理几个或所有的成员。

3.1.1 对所有数据库文件成员的一般操作

系统提供了如下方法：

- 对已存在文件增加新成员
- 修改现存成员的属性(例如，成员的说明)而不重生成该成员
- 对一个成员重新命名
- 从一个文件中移出成员

下面讨论这些操作。

3.1.1.1　往文件中增加新成员

可用以下方法中任何一种往文件中增加成员：

- 自动增加。当使用 CRTPF(生成物理文件)命令或 CRTLF(生成逻辑文件)命令生成一个文件时，其缺省值是自动地在此文件中加一个成员(成员名与文件名相同)。CRTSRCPF 生成源物理文件时，其缺省值是不生成成员，可以在生成数据库文件命令时通过设置 MBR 参数来指定一个不同的成员名。如果你不想在文件生成时加入一个成员，那么就在 MBR 参数中给出*NONE。
- 明确的增加。在文件生成后，可以使用 ADDPFM 命令(增加物理文件成员)或 ADDLFM 命令(增加逻辑文件成员)来增加成员。
- 使用文件拷贝命令 CPYF。如果往文件中拷贝一个成员，而这个文件中原来没有该成员，那么用 CPYF 把成员加到这个文件中。

3.1.1.2　修改成员属性

可以使用 CHGPFM(修改物理文件成员)命令或 CHGLFM(修改逻辑文件成员)命令来修改物理文件或逻辑文件成员的某些属性。对物理文件成员，可以修改以下参数：SRCTYPE(成员的源类型)、EXPDATE(有效期)、SHARE(是否能在作业中共享此成员)以及 TEXT(成员说明)。对于逻辑文件的成员，可以修改其 SHARE 和 TEXT 参数。

注：可以使用 CHGPF 和 CHGLF 命令来修改许多其它的文件属性。例如，在 CHGPF 中用 SIZE 参数来修改文件中每个成员所允许的最大尺寸。

3.1.1.3　成员重新命名

RNMM(重新命名成员)命令修改一个物理或逻辑文件中已存在的成员的名字，而文件名

不变。

3.1.1.4 从文件中移出成员

RMVM(移出成员)命令用来移出成员及其内容。成员中的数据和成员本身都被移出。成员被移出后将不再被系统使用。这不同于仅仅从成员中清除或删除数据，成员仍然存在，程序可以继续使用它(如向其中增加数据)。

3.1.2物理文件成员操作

以下各部分讨论只针对物理文件成员的操作。这些操作包括初始化数据、清除数据、重组数据和显示物理文件成员中的数据。

如果正在操作的成员与引用的某些限制有关，那么该操作将受其影响。细节描述请参看第 16 章”引用的完整性”。

3.1.2.1 初始化物理文件中成员的数据

在程序中使用相关记录号进行处理，该数据库文件必须包含与程序中所用的最高相关记录号相等的记录位置。程序使用相关记录号进行处理时需要初始化这些记录。

可用 INZPFM(初始化物理文件)命令初始化下面两类记录：

- 缺省记录
- 删除记录

在 INZPFM 命令中用 RECORDS 参数指定要处理的记录类型。

如果用缺省记录来初始化，那么每一个新记录中的那些字段都用文件生成时定义的缺省字段值来初始化。如果未定义缺省字段值，那么数字字段被填充为 0，而字符字段被填充为空格。

变长的字符字段有一个长度为 0 的缺省值。空值字段的缺省值为空。日期、时间和时间标记的缺省值分别为当前的日期、时间或时间标记。程序描述文件有一个全部为空格的缺省值。

注：如果物理文件的成员或与其相关的逻辑文件成员的 DDS 中指定了 UNIQUE 键字，那么可以初始化一个缺省记录；否则，会生成一些有重复键值的记录。

如果记录被初始化为缺省记录，可以由相关记录号来读一个记录，并修改数据。

如果记录被初始化为删除记录，则可以通过使用一个被删除记录的相关记录号来加一个记录，然后修改数据(不能使用未被删除的相关记录号去增加一个记录)。

不能读被删除的记录，它们仅在成员中占有一个位置。一个删除的记录可以通过在其上写一个新的记录来做修改。有关处理被删除记录的详细信息，请参阅 2.3.5”删除数据库记录”。

3.1.2.2 从物理文件成员中清除数据

CLRPFM 命令用来从一个物理文件成员中清除数据。在清除操作完成后，成员的描述仍存在，但数据已经不在了。

3.1.2.3 重新组织物理文件成员中的数据

可以使用 RGZPFM 命令来做下面工作：

移出被删掉的记录以便重用被它们占用的空间。

按通常访问记录的次序重新组织文件中记录顺序，以使检索记录的时间最少。这可通过使用 KEYFILE 参数来完成。对于那些按主键字访问的文件而不是按到达顺序访问的文件来说，这样做更具优越性。可用下列方法重新组织一个成员：

—物理文件的键字字段

—依赖于物理文件的逻辑文件的键字字段

可用 RGZPFM 中的 SRCOPT 和 SRCSEQ 参数来重新组织一个源文件成员，插入一个新的源成员，并重新设置源日期数据字段。

可以释放某些文件占用过的空间，这些空间是物理文件格式中的可变长字段，且现在已成为碎片的那部分。

例如，下面的 RGZPFM 命令使用逻辑文件中的访问路径重新组织物理文件的第一个成员：

```
RGZPFM   FILE(DSTPRODLB/ORDHDRP)
         KEYFILE(DSTPRODLB/ORDFILL    ORDFILL)
```

物理文件 ORDHDRP 用到达顺序的访问路径，它用逻辑文件 ORDFILL 中访问路径被重新组织。假设键字字段是 Order 字段，下面说明了如何安排记录：

下面是 ORDHDRP 文件原来的数据，注意在用 RGZPFM 命令前记录 3 被删除了。

相关记录号	Cust	Order	Ordate. . .
1	41394	41882	072480. . .
2	28674	32133	060280. . .
3	deleted	record	
4	56325	38694	062780. . .

下面的例子显示用 Order 作键字并以升序排列来重新组织的 ORDHDRP 文件。

相关记录号	Cust	Order	Ordate. . .
1	28674	32133	060280. . .
2	56325	38694	062780. . .
3	41349	41882	072480. . .

注：

1.如果一个有到达顺序访问路径的文件使用键字顺序访问路径来重新组织，那么这个到达顺序访问路径要被修改。就是说，文件中的记录按照所使用的键字顺序访问路径的顺序被物理地存放。把数据按与所用键字访问路径最接近的物理顺序存放，可以改善顺序处理数据的性能。

2.重组一个压缩过删除记录的文件，可以改变随后的相关记录号。

3.DDS 键字中指定为 FIFO 还是 LIFO 的访问路径依赖于物理文件中记录的物理顺序。因此，在使用键字顺序访问路径重新组织一个物理文件之后，有重复键值字段的记录其顺序可能被改变。另外，由 FCFO 规定的访问路径是由 FIFO 排序的，因此，当文件被重新组织后，有重复键字值字段的记录顺序也可能改变。

4.如果取消了 RGZPFM 命令，那么所有物理文件成员访问顺序都要重建。

如果在运行 RGZPFM 命令时发生下列情况，那么记录可能不被重新组织：

- 系统异常结束
- 包含 RGZPFM 命令的作业*IMMED 结束
- 运行 RGZPFM 命令的子系统*IMMED 结束
- 系统*IMMED 结束

重组的成员状态依赖于在重组结束前系统做的多少以及 SRCOPT 参数的值。如果没规定 SRCOPT 参数，那么这个成员既可能被完全重组，也可能根本未被重组。可显示一下文件的内容，以确定该成员是否被重组。如果用 DSPPFM(显示物理文件成员)命令，该成员未

被重组，就再用一次 RGZPFM 命令。

如果使用了 SRCOPT 参数，成员可能发生以下几种情况：

- 成员被完全重组。一个完成信息会送到你的作业日志中，说明重组操作完全成功了。
- 成员根本未被重组。一个信息会送到你的作业日志，说明重新组织操作没有成功。如果这种情况发生，请再发一次 RGZPFM 命令。
- 成员被重组，但只有一部分序号被改变了。一个完成信息会送至你的作业日志中，说明成员被重组了，但并非所有的序号都被改变了。如果这种情况发生，那么就要再发一次 RGZPFM 命令，并且要把 KEYFILE 参数指定为*NONE。

为了减少一个物理文件成员中被删除的记录的数量，文件可以被生成或修改，以便重新利用被删除记录的空间。有关这方面的更多信息，请参看 2.1.2.3“重用已删除的记录”。

3.1.2.4 显示物理文件成员的记录

DSPPFM(显示物理文件成员)命令按到达顺序显示物理数据文件成员中的数据。这个命令可被用于：

- 问题分析
- 调试
- 记录查询

可以显示源文件或数据文件而不管它们是键字顺序还是到达顺序。即使文件是键字顺序记录也是按到达顺序显示。通过文件编页码，记录号定位一个特殊的记录，或用换档键显示屏幕右边或左边，以看到记录的其它部分。可以按一个功能键来看显示的是字符型数据还是十六进制数据。

如果系统中有 Query(查询)，那么可以用 STRQRY(启动查询)命令来选择记录并显示它。

如果系统中有 SQL，那么也可以用 SRTSQL 命令来交互地选择并显示记录。

第十一章　修改数据库文件的描述及属性

这一章讨论考虑修改数据库文件的描述或属性时要考虑的问题。

3.2.1 在文件描述中修改字段

当编译一个使用了外部描述数据的程序时，编译器将文件的描述拷贝到被编译的程序中。当运行这个程序时，系统会核实被编译程序的记录格式与这个文件当前定义的记录格式是否一致。缺省情况将做级别检测。

当一文件生成时，系统会为每一个记录格式分配一个唯一的级别标识。系统使用记录格式描述中的信息来确定其级别标识。这个信息包括该记录格式的总长度，记录格式名，其中定义的字段数及字段顺序、数据类型、字段大小、字段名以及字段中小数点的位置。修改一个记录格式中的这些信息会引起级别标识符的改变。

下面的 DDS 信息对级别标识没有影响，也正因如此，修改它们而无需重新编译程序：

- TEXT 键字
- COLHDG 键字
- CHECK 键字
- EDTCDE 键字
- EDTWRD 键字
- REF 键字
- REFFLD 键字
- CMP、RANGE 和 VALUES 键字
- TRNTBL 键字
- REFSHIFT 键字
- DFT 键字
- CCSID 键字
- ALWNULL 键字
- 连接说明和连接键字
- 键字字段
- 访问路径键字
- 选择/省略键字

请记住即使修改键字字段或选择/省略字段都不会引起一个级别检查，但使用新的访问路径会产生不可预测的变化。例如，将键字字段从用户号改为用户名会改变记录被检索的顺序，而且在处理文件的程序中引起不可预测的问题。

如果要做(或缺省)层次检查，当文件打开时，文件的级别标识要与程序中文件的顺次标识做比较。如果标识不同，有一个信息被送到该程序，以标识修改条件以及这一修改会影响程序。你需再次编译程序，这样修改就可以包括进去了。

另一种方法是显示一下该文件的描述以决定这些修改是否会影响程序。可以使用 DSPFFD(显示文件字段描述)命令来显示描述，如果用 SEU，也可以显示文件的 DDS 描述。

文件中定义的格式级别描述可以用 DSPFD(显示文件描述)命令来显示。在显示级别标识时，记住要比较的是记录格式标识，而不是文件标识。并非文件的每个修改都必然影响到程序。例如，如果在文件末增加一个字段，而程序并不使用这个新字段，那么就无需重编译程序。如果修改确实不影响程序，可以使用 CHGPF 命令或 CHGLF 命令，并将参数 LVLCHK

规定为*NO，以关闭对文件的层次检查，或者可以输入一个 OVRDBF(Override with Database File)命令并指定 LVLCHK 参数为*NO，这样不进行级别检查来运行程序。

请记住级别检查是操作的优先方法。使用 LVLCHK(*YES)是处理数据库整体性能的好办法，而 LVLCHK(*NO)就不能预测其结果。

3.2.2修改物理文件描述和属性

有时，当修改了一个物理文件的描述然后重新生成该文件时，级别标识符会改变。例如，如果在文件描述中增加一个字段或改变一个已存在字段的长度，那么级别标识可能会改变。如果级别标识改变了，对使用该物理文件的程序要重新编译。在程序被重新编译之后，这些程序将使用新的级别检测标识。

可以建立一个与物理文件的原始记录格式一样的逻辑文件来为程序提供数据，这样可以不用重新编译。程序使用这种方法，可使该逻辑文件与修改前的物理文件有同样的级别检测标识。

例如，要在一个物理文件记录格式中增加一个字段，可以做下列步骤而避免重新编译程序：

1.修改 DDS 并生成一个包含了该字段的新的物理文件(FILEB 在 LIBA 中)

```
CRTPF    FILE(LIBA/FILEB)     MBR(*NONE).......
```

FILEB 没有成员。(旧文件 FILEA 在 LIBA 库中并有一个成员 MBRA)

2.拷贝旧的物理文件的成员到新的物理文件中：

```
CPYFFOMFILE(LIBA/FILEA)  TOFILE(LIBA/FILEB)
          FORMMBR(*ALL)     TOMBR(*FROMMBR)
          MBROPT(*ADD)FMTOPT(*MAP)
```

由于说明了 FROMMBR(*ALL)和 TOMBR(*FROMMBR)，新的物理文件中的成员被自动命名为旧的物理文件的成员的名字。FMTOPT 参数指定要按照字段名拷贝(*MAP)数据。

3.描述一个象旧物理文件一样的新的逻辑文件 FILEC(该逻辑文件记录格式不包括新的物理文件字段)。指定 PFILE 键字为 FILEB(在做级别检测时，由于 FILEA 和 FILEC 有相同的格式，所以逻辑文件中的级别标识与程序中的级别标识符相匹配)。

4.生成新的逻辑文件：

```
CRTLF    FILE(LIBA/FILEC).......
```

5.现在可以做如下的工作之一：

a.在相应的作业中使用 OVRDBF(覆盖数据库文件)命令，用逻辑文件覆盖程序中引用的旧的物理文件(有关 OVRDBF 命令详细的描述在第五章”运行时的考虑”中)。

```
OVRDBF  FILE(FILEA) TOFILE(LIBA/FILEC)
```

b.删除旧的物理文件并把逻辑文件的名改为旧的物理文件名，程序中的文件名就不用被覆盖。

```
DLTFFILE(LIBA/FILEA)
RNMOBJ  OBJ(LIBA/FILEC) OBJTYPE(*FILE)  NEWOBJ(FILEA)
```

下面说明了在三个文件中所用的记录格式的联系：

FILEA(旧物理文件)

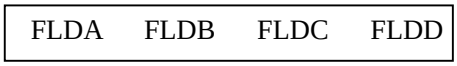
FLDA	FLDB	FLDC	FLDD

FILEB(新物理文件)

FLDB1

FLDB1 被加到记录格式中

FILEC(逻辑文件)



FILEC 用 FILEA 的记录格式，在逻辑文件的记录格式里没有 FLDB1。

当修改一个物理文件并因此而要重新生成文件时，所有引用该物理文件的逻辑文件都要首先删除，才能删除且重生成新的物理文件。在重新生成物理文件之后，可以重新生成或恢复引用它的逻辑文件。下面给出两种方法：

例 1.在不同的库中生成一个同名的新的物理文件

1.在与旧的物理文件所在库不同的库中生成一个具有不同记录格式的新的物理文件。新文件名应与老文件名相同(旧物理文件 FILEPFC 在 LIBB 库中，并且有两个成员，MBRC1 和 MBRC2)。

```
CRTPF    FILE(NEWLIB/FILEPFC) MAXMBRS(2).....
```

2.将旧物理文件中的成员拷贝到新的物理文件中。这个新物理文件中的成员被自动按照旧物理文件中的成员名命名，这是由于说明了 TOMBR(*FROMMBR)和 FROMMBR(*ALL)。

```
CPYFFROMFILE(LIBB/FILEPFC)  TOFILE(NEWLIB/FILEPFC)
          FROMMBR(*ALL)    TOMBR(*FROMMBR)
          FMTOPT(*MAP  *DROP)      MBROPT(*ADD)
```

3.在一个与旧的逻辑文件所在库不同的库中描述并生成一个新的逻辑文件。新逻辑文件名应与旧逻辑文件名相同。如果记录格式无需改变，那么可以使用 FORMAT 键字来使用与当前逻辑文件相同的记录格式。也可以使用 CRTDUPOBJ(生成重复目标)命令从 LIBB 库中的旧逻辑文件 FILELFC 生成另一个逻辑文件：

```
CRTLF    FILE(NEWLIB/FILELFC)
```

4.删除旧的逻辑和物理文件

```
DLTFFILE(LIBB/FILELFC)
DLTFFILE(LIBB/FILEPFC)
```

5.使用以下命令将新生成的文件移入原来的库中：

```
MOVOBJ  OBJ(NEWLIB/FILELFC) OBJTYPE(*FILE)  TOLIB(LIBB)
MOVOBJ  OBJ(NEWLIB/FILEPFC) OBJTYPE(*FILE)  TOLIB(LIBB)
```

例 2.在同一个库中生成新版本的文件

1.在旧的物理文件所在的库中生成一个新的不同记录格式的物理文件。文件名字应该不同(旧物理文件 FILEPF 在 LIBA 库中，并且有两个成员 MBRA1 和 MBRA2)。

```
CRTPF    FILE(LIBA/FILEPFB)  MAXMBRS(2).....
```

2.将旧物理文件中的成员拷贝到新的物理文件中

```
CPYFFROMFILE(LIBA/FILEPFA)  TOFILE(LIBA/FILEPFB)
          FROMMBR(*ALL)      TOMBR(*FROMMBR)
          FROMMBR(*MAP  *DROP)MBROPT(*REPLACE)
```

3.在旧逻辑文件所在的库中生成一个新的逻辑文件。旧文件名和新文件的名应该不同。(如果记录格式无需改变，可以使用 FORMAT 键字以使用与当前逻辑文件相同的记录格式。)PFILE 键字必须引用第一步中生成的新的物理文件。旧的逻辑文件 FILELFA 在 LIBA 库中。

CRTLF　FILE(LIBA/FILELFB)

4.删除旧的逻辑文件和物理文件

DLTFFILE(LIBA/FILELFA)

DLTFFILE(LIBA/FILEPFA)

5.把新逻辑文件改名为旧逻辑文件名(如果决定也重命名物理文件，一定要修改逻辑文件的 DDS 以使 PFILE 键字引用新的物理文件名)：

RNMOBJ(LIBA/FILELFB)OBJTYPE(*FILE)　NEWOBJ(FILELFA)

6.如果重新命名逻辑文件成员，且假设 CRTLF 命令中使用了缺省值，则要给出以下的命令：

RNMMFILE(LIBA/FILELFA)　MBR(FILELFB)NEWMBR(FILELFA)

可以使用 CHGPF 命令修改物理文件及其成员的某些属性。关于这些参数的信息，请参看 CL 参考手册中的 CHGPF 命令。

3.2.3修改逻辑文件描述和属性

一般来说，当修改逻辑文件而引起级别标识符的改变(例如，增加一个新字段、删除一个字段或改变字段的长度)时，建议最好重编译使用该逻辑文件的程序。有时修改文件引起了级别标识的改变但不需重新编译程序(例如，在该文件末增加了一个字段而该字段并未在你的程序中使用)。然而，在那些情况下，将被迫关闭级别检测来运行使用改变后文件的程序。这并非操作的好方法，它增加了将来产生错误数据的机会。

为避免重新编译，可以保持当前逻辑文件(不改变)，并生成一个具有新增加字段的逻辑文件。程序引用旧文件，因为它仍然存在。

可以使用 CHGLF 命令来修改一个逻辑文件及其成员的大多数属性，这些属性是在 CRTLF 命令中规定的。

第十二章　使用数据库属性和交叉引用信息

AS/400 集成数据库提供文件属性及交叉引用信息。交叉引用信息包括：

- 程序中使用的文件
- 数据或访问路径依赖的其它文件
- 文件属性
- 文件定义的字段
- 与文件有关的约束条件
- 文件的键字字段

在以下各节说明的每一个命令都可以在屏幕上、打印输出上提供信息，或者将交叉引用信息写到数据库文件中，也能用于程序或实用工具中做分析。

有关把输出写到数据库文件的更详尽信息，请参阅 3.3.2“把由命令产生的输出直接写到数据库文件中”。

你可以使用 RTVMBRD(Retrieve Member Description)命令来取得应用程序中使用的某个数据库文件成员的信息。有关内容请参阅 CL 程序设计中有关”取得成员描述信息”的章节。

3.3.1显示有关数据库文件的信息

3.3.1.1　显示文件属性

可以使用 DSPFD 命令显示数据库文件和设备文件属性。这些信息可以被显示、打印或写到一个输出文件(OUTFILE)中。这个命令提供的信息包括(参数值在括号中给出)：

- 基本属性(*BASATR)
- 文件属性(*ATR)
- 访问路径说明(*ACCPTH，只对逻辑和物理文件)
- 选择/省略说明(*SELECT，只对逻辑文件)
- 连接逻辑文件说明(*JOIN，只对连接逻辑文件)
- 替换顺序说明(*SEQ，只对物理和逻辑文件)
- 记录格式说明(*RCDFMT)
- 成员属性(*MBR，只对物理和逻辑文件)
- 假脱机属性(*SPOOL，只对打印文件和磁盘文件)
- 成员表(*MBRLIST，只对物理和逻辑文件)
- 文件约束(*CST)
- 触发器(*TRG)

3.3.1.2　显示字段描述

用 DSPFFD 命令显示数据库文件中设备文件的字段信息。这些信息可以显示、打印或写到一个输出文件(OUTFILE)中。

3.3.1.3　显示系统中文件间的相互关系

可以使用 DSPDBR 命令显示有关数据库组织的如下信息：

- 使用某个记录格式的数据库文件(物理文件和逻辑文件)的列表

• 共享某个文件数据的物理和逻辑文件的列表
• 共享某个成员数据或访问路径的成员(物理和逻辑成员列表)
• 依据引用约束关系的物理文件的列表

这些信息可以被显示、打印或写到一个输出文件中。

例如，要显示某些与物理文件 ORDHDRP(记录格式为 ORDHDR)有关的所有数据库文件的列表，要键入以下的 DSPDBR 命令：

DSPDBR FILE(DSTPRODLB/ORDHDRP) RCDFMT(ORDHDR)

注：有关信息请参阅《CL 参考手册》中 DSPDBR 命令的描述。

如果在 RCDFMT 中给出了一个记录格式名，那么先显示一个开头信息，然后再显示哪些文件使用了这个记录格式。

如果在 DSPDBR 命令的 MBR 参数中给出了成员名，则显示有关的成员。

如果 DSPDBR 命令中用缺省的 MBR(*NONE)，则有关的数据文件被显示。要显示共享访问路径，必须给出成员名。

DSPDBR 命令的输出说明了所包括的共享类型。命令的结果如果输出到显示屏上，也包括共享的类型名。如果命令的结果写到一个文件中，则共享类型的编码(如图)放在这个输出文件的 WHTYPE 字段中。

类型	编码	说 明
约束	C	该物理文件依赖于和某个约束有关的另外物理文件的数据
数据	D	该文件或成员依赖于另一个文件中成员的数据
访问路径共享	I	该文件成员共享一个访问路径
访问路径主人	O	如果一个访问路径被共享，该文件成员之一被认为是拥有者。访问路径的拥有者要对存取路径所用的存贮区负责。如果显示的成员被指定为拥有者，则对于共享访问路径一个或多个文件成员就有 I 类型。
SQL 视图	V	SQL 视图或成员依赖于另一个 SQL 视图

3.3.1.4 显示程序使用的文件

可用 DSPPGMREF 命令来确定一个程序使用了哪些文件、数据区和其它程序。这些信息仅对编译程序有用。

这些信息可以显示、打印或写入一个输出文件中。

当生成程序时，程序中所使用的目标信息被保存，然后 DSPPGMREF 命令就用这些信息。

下表显示了高级语言和服务程序所使用的目标的保留信息：

语言或实用程序	文件	程序	数据区	看注
BASIC	Yes	Yes	No	1
C/400 Language	No	No	N/A	
CL	Yes	Yes	Yes	2
COBOL/400 Language	Yes	Yes	No	3
CSP	Yes	Yes	No	4
DFU	Yes	N/A	N/A	
FORTRAN/400* Language	No	No	N/A	

Pascal Pascal	No	No	N/A	
PL/I	Yes	Yes	N/A	3
RPG/400 Language	Yes	Yes	Yes	5
SQL Language	Yes	N/A	N/A	

注：

1.存贮外部描述文件的引用、程序和数据区信息。

2.在编译 CL 程序中的命令时，在命令定义中规定的引用文件、程序和数据的所有系统命令都被保存。如果使用变量，变量名作为目标名(如&FILE)。如果使用表达式，目标名用*EXPR 保存。用户定义命令也可以保存在命令中说明的文件、程序或数据区的信息。请参看《CL 程序设计》中有关内容。

3.当程序名为一文字时才被保存(这是一个静态调用，如 CALL ‘PGM1’)，而用一个 COBOL/400 标识符做程序名时，程序名不被保存(此为动态调用，如 CALL PGM1)。

4.CSP 程序也保存类型为*MSGF、*CSPMAP 和*CSPTBL 目标的信息。

5.不保存使用的本地数据区。

存贮的文件信息包括使用类型(一个数字)。在 DSPPGMREF 命令的结果输出到一个文件中时(用 OUTFILE 参数建立)，使用如下说明：

代码定义

1 输入

2 输出

3 输入和输出

4 修改

8 未被说明

这些代码也可以合并使用。例如，一个文件编码为 7，表示它可被用于输入、输出和修改。

3.3.1.5 显示系统交叉引用文件

系统对所有数据库文件(除在 QTEMP 中的文件)管理两个文件，一个是基本文件属性信息(QSYS/QADBXREF)，另一个是交叉引用信息(QSYS/QADBFDEP)。可用这些文件来确定基本属性和文件依赖关系。要显示这些文件中包含的字段，使用 DSPFFD(Display File Field Description)命令。

注：通常，只有安全管理人员才有权使用这些文件。

3.3.2 把由命令产生的输出直接写入数据库文件中

给出 OUTFILE 参数，可以把许多 CL 命令的输出写到一个物理文件中。

在程序或服务程序(如 Query)中使用输出文件用于数据分析。例如，将 DSPPGMREF 命令的输出送到一个物理文件中，然后再查询这个文件，确定哪个程序使用某个文件。

在命令中指定了 OUTFILE 参数，将生成该物理文件。最初，被生成的文件具有私人特权，只有主人(运行命令的人)可以使用它。但主人可以象其它数据库文件一样授权给其它用户去使用这些生成的文件。

系统为每一个能指定 OUTFILE 参数的命令提供一个指定记录格式的模型文件。如果在 OUTFILE 参数中指定并不存在的文件名，那么系统将使用与模型文件相同的记录格式生成该文件。如果说明了一个已存在输出文件的文件名，则系统会检查该文件的记录格式是否与模型文件的记录格式一致。若不匹配，则系统送给作业一个信息，且命令不能完成。

注：必须使用自己的文件作为输出文件，而不能在 OUTFILE 参数上说明系统提供的模

型文件。请参看《Programming Reference Summary》一书，看哪些命令可以允许使用输出文件及这些命令所用的模型文件名。

注：所有系统提供的模型文件都在 QSYS 库中。

可以使用 DSPFFD 命令显示系统提供模型文件的记录格式中包含的字段。

3.3.2.1 使用命令输出文件的例子

下例使用 DSPPGMREF 命令，收集所有库中编译过的程序的信息，并把输出放在名为 DBROUT 的文件中：

```
DSPPGMREF   PGM(*ALL/*ALL)   OUTPUT(*OUTFILE)   +
                    OUTFILE(DSTPRODLB/DBROUT)
```

可以使用 Query 去处理这个输出文件。处理该文件的另一个方法是生成一个逻辑文件，从文件中选择信息。下面是这个逻辑文件的 DDS 编码。记录是根据文件名选择出来的。

```
|...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
     A* Logical file DBROUTL for query
     A
     A          R DBROUTL                    PFILE(DBROUT)
     A          S WHFNAM                     VALUES('ORDHDRL' 'ORDFILL')
     A
```

3.3.2.2 DSPFD 命令的输出文件

显示文件描述(DSPFD)命令提供唯一的输出文件，它依赖于被说明的参数。请参阅《Programming Refererce Summary》中 DSPFD 命令的模型文件清单。

注：所有系统提供模型文件都在 QSYS 库中。

为收集在 LIBA 库中所有文件的访问路径信息，可用下面命令：

```
 DSPFD   FILE(LIBA/*ALL)  TYPE(*ACCPTH)  OUTPUT(*OUTFILE) +
         OUTFILE(LIBB/ABC)
```

文件 ABC 在库 LIBB 中生成，并且被描述为与系统提供文件 QSYS/QAFDACCP 相同的字段，然后 ABC 文件就有了这样一条记录，它包含了在 LIBA 库中找到的有访问路径的各个文件的每个键字字段。如果 DSPFD 命令为：

```
DSPFD   FILE(LIBX/*ALL)  TYPE(*ATR)  OUTPUT(*OUTFILE) +
         FILEATR(*PF)  OUTFILE(LIBB/DEF)
```

文件 DEF 在 LIBB 库中生成并且其外部描述与 QSYS/QAFDPHY 中的字段相同，于是文件 DEF 包含关于在 LIBX 库中找到的每一个物理文件的一个记录。

可以使用 DSPFFD 命令显示由 IBM 提供的每一个模型文件的字段名。例如，要显示访问路径模型文件的字段描述(在 TYPE 参数中说明*ACCPTH)，用下面命令：

```
      DSPFFD      QSYS/QAFDACCP
```

3.3.2.3 DSPJRN 命令的输出文件

请参看《Programming Refererce Summary》以得到系统提供的可由 DSPJRN(Display Journal)命令显示的模型输出文件的清单。

3.3.2.4 DSPRB 命令的输出文件

请参看《Programming Refererce Summary》以得到系统提供的 DSPRB(Display Problem)命令的模型输出文件的清单。这个命令根据记录类型提供唯一的输出文件：

基本问题数据记录(*BASIC)：它包括问题类型、状态、机器类型/型号/系列号、产品 ID、相关信息以及跟踪数据。

故障点、隔离或应答 FRU 记录(*CAUSE)。应答 FRU 记录在可用的时候才能用。如果应答 FRU 不能用，在隔离 FRU 可用时，就用它。若二者皆不可用，用故障点 FRU。

PTF 修正记录(*FIX)

用户输入文本(短文记录)(*USRTXT)

支持数据标识符记录(*SPTDTA)

在所有以上五种输出文件中的那些记录都有一个问题标识，因此，原因(CAUSE)、修正(FIX)、用户文本信息(USRTXT)以及支持数据(SPTDTA)与基本问题数据(*BASIC)都互相有关。仅有一种类型数据可以写到一个专门的输出文件中。原因、修正、用户文本以及支持数据输出文件对一个专门问题可有多个记录。请参看《CL 参考手册》中有关 DSPRB 命令的信息。

第十三章　数据库恢复的有关问题

这章讨论了由于各类原因引起的系统数据丢失的情况下，能恢复或重存数据库文件所做的一般考虑以及 AS/400 提供的有关工具。关于 AS/400 备份和恢复的策略、计划及工具请参阅《备份和恢复—基础》及《备份和恢复—高级》。

3.4.1数据库的保存和重存

定期保存数据库文件和相关的目标是很重要的，这可以在必要的时候恢复它们。

数据库文件和相关目标可以用任何支持设备、介质或备份文件来保存或恢复。当信息被保存时，特殊格式的付本写到介质上或写入一个备份文件中。某介质将来可在本系统或另一个 AS/400 系统上移出或存贮。当重存信息时从介质或一个备份文件中读到存贮区，它就可以被系统用户访问了。

备份文件是磁盘常驻文件，它可做保存操作的结果，也可做重存操作的原本。备份文件允许做非干预的保存操作，就是说，用备份文件做保存操作时，操作员不用装入磁带或软盘，可用 SAVSAVFDTA(Save Save File Data)命令定期地把备份文件中的内容保存到磁带或软盘上。磁带和软盘是可以放在其它地方的。要把磁带和软盘放在与系统不在一起的地方，这样做是为了避免放系统的场地有灾害时，能用这些磁带和软盘做恢复用。

关于保存和重存的一般计划、设计考虑以及关于使用保存和重存命令的信息，请参看《备份和恢复—基础》；使用活动时保存功能有关的细节设计和编程考虑，请参看《备份和恢复—高级》。

3.4.1.1　保存和重存的考虑

把数据库文件或相关的目标保存到磁带或软盘上时，系统会根据保存操作的日期和时间修改目标的描述。把一个目标保存到一个备份文件时，可以通过说明保存命令(SAVE)中的 UPDHST(*NO)来避免系统修改原有日期和时间。重存一个目标时，系统总是根据重存操作的日期和时间修改目标的描述。可以通过使用 DSPOBJD(Display Object Description)命令规定 DETAIL(*FULL)来显示保存/重存操作的有关信息。可用 DSPSAVF(Display Save File)命令来显示一个备份文件中的目标。

可在 DSPDKT 命令或 DSPTAP 命令中指定 DATA(*SAVRST)来显示介质上的目标。

数据库成员的最后的保存/重存日期可用下列命令显示：

```
DSPFD     FILE(file-name)     TYPE(*MBR)
```

3.4.2数据库数据的恢复

AS/400 系统有整体恢复功能帮助恢复数据库文件中数据。本章所讨论的重要功能有：

- 日志管理，记录文件中数据的修改
- 落实控制，同步事务恢复
- 往辅存中强制写数据的修改
- 系统异常结束的恢复(请参看 3.4.4”系统异常结束后的数据库恢复”)

3.4.2.1　日志管理

日志管理可以记录一个或更多数据库文件所做的数据修改，然后可以用该日志去恢复。

在很多方面可以考虑使用日志管理。一个数据库文件被破坏， 变得无法使用，如果使

用了日志，就可以重新构造文件的大多数内容(细节请参看《备份和恢复—高级》一书中关于日志的讨论)。另外，日志允许你取消对文件所做的修改。

日志功能很容易启动和结束。它不需要附加的程序设计或改变现存程序。

当文件发生修改而你在使用日志时，系统在日志接收器中记录下该变化并且在它被写入文件前将它写入辅存。因此，日志接收器总是有最近的数据库信息。对文件日志不考虑这些修改的程序、用户作业或逻辑文件的类型。

日志项记录了某些记录（增加、修改或删除）或整个文件（打开或是保存）的活动情况。每一项包括标识活动源的控制信息（包括用户、作业、程序、时间和日期）。对于影响一个单独记录的变化，记录映象包括在控制信息之后。变化之后的记录映象总是被包括在内。也可选择将变化之前的记录映象包括进来。可以通过在 SRTJRNPF 命令中指定 IMAGES 参数，来控制是对修改前后的映象都记录，还是只记录修改后的映象。

注意：如果这些文件或记录的修改是由一个触发器程序或一个公用完整的约束引起的，那么相关的日志项指出这一点。请参看第 16 章“引用的完整性”中有关引用完整性的详细信息， 以及第 17 章“触发器”中有关触发器的详细信息。

当系统启动时（IPL 时间），所有日志管理的数据库文件自动地与其日志同步。如果系统异常结束，一些数据库的变化可能已反映在日志中，但是数据库本身仍未受影响。如果是这样的话，那么系统在 IPL 时自动地从日志中更新数据库文件下列其为最近状态。

日志管理可以更容易、更快地保存数据库文件。例如，不用每天做完整的文件备份，只需保存包含该文件变化的日志接收器，然后每周保存一次整个文件。这种方法可以减少用于日常保存操作的时间。

可以使用 APYJRNCHG 命令和 RMVJRNCHG 命令用日志的修改来恢复损坏的或无法使用的数据库文件成员。APYJRNCHG 命令根据日志接收器中记录的变化来修改某个物理文件成员。根据该物理文件损坏的类型和自从文件最后一次保存以来发生的记录下来的修改数量，使用 RMVJRNCHG 命令从文件中移出修改可能更容易。WRKJRN（Work With Journal）命令给出了提供修改的提示。

DSPJRN 命令可把日志项转换到数据库文件中，这个文件可用来做活动报告、审查跟踪、保密和程序调试。

由于日志提供了许多有用的功能，它不仅限于最基本的恢复数据功能，日志应该被认为是恢复策略的关键部分。有关日志管理和命令的详细信息，请参阅《备份和恢复－高级》。

3.4.2.2 用落实控制做交易恢复

落实控制功能，把对数据库文件一定数量的修改当作一个单元（交易）来处理定义它。它是系统中日志功能的扩充，有助于恢复数据和重新启动程序。落实控制能保证一个复杂的应用交易即使在作业或系统结束时也能做到逻辑上的同步。

一个交易（transaetion）是把一组修改做为一个修改来对待的，例如资金从储蓄帐到结算帐的传送，就看做一个交易。可按如下分类：

- 文件在查询中没有修改。
- 简单交易：按一次 Enter 键都有一个文件发生变化。
- 复杂交易：每按一次 Enter 键有两个或更多的文件发生变化。
- 复杂交易：每当你按一次 Enter 键有一个或更多的文件发生变化，但这些修改只是交易逻辑组的一部分。

当使用落实控制时，在交易处理过程中对文件所做的修改被写在日志中。

如果系统或作业异常结束，日志管理本身可以保证至多丢失最后一次记录的修改。然而，如果系统或作业在一笔复杂交易（修改多个文件）期间异常结束，那么这些文件会反映一个

不完整的逻辑。例如，作业可能已经在文件 A 中修改了一个记录，但在作业未来得及修改文件 B 中的相关记录时，作业就异常结束了。在这种情况下，逻辑交易由两个修改组成，但是只有一个修改在作业异常结束之前得以完成。

恢复好一个复杂的应用需要多方面的知识。不是说简单地再启动程序就可以。例如，应用程序或数据文件服务程序对记录所做的修改已经退回到最后一个复杂交易开始之前的状态，如果多个用户在同时访问这些文件，这事就变得更复杂。

落实控制能帮助解决这些问题。在落实控制下，复杂交易期间使用的记录 对于其它用户是锁定的。这保证了其它用户到该交易落实结束才能使用这些记录。在交易结束时，程序发出一个操作，释放记录，然而，如果在落实操作被执行之前系统异常结束，那么从上一次落实操作以来作业对记录发生的所有变化都被退回（Roll Back）。与之有关的仍被锁定的记录被释放。换言之，数据库的修改要返回到一个明显的交易边界。

也可以控制退回的操作。假设一个定单录入，应用程序在每个定单结束后做落实操作。在一个定单的中间，操作员可以通知程序做退回操作。对文件做所的所有修改被退回至这个定单开始。

落实和退回操作可用于几种 AS/400 编程语言，包括 RPG/400， PL/I，SQL 以及 AS/400 控制语言（CL）。

落实控制的一个可选特性是使用一个通报目标。通报目标是一个文件、数据区或信息队列。当作业或系统在交易没完成时结束，那么由程序规定的信息被自动送至通报目标。操作员或应用程序可用这些信息从上次成功的交易边界再启动。

落实控制也可用于批处理环境，正如它对交互式交易恢复提供帮助一样，它也可以有助于批处理作业的恢复。详情请参阅《备份和恢复指南》。

3.4.2.3 往辅存中强写数据

在生成文件和覆盖数据库文件命令中的强写数据参数（FRCRATIO）可用来强制地把数据物理地写入辅存中，一个强写系数参数可以使每个增加、更新或删除请求立即写入文件的辅存中。然而，选择这一参数会降低系统的性能。因些，为了保护数据库文件首选方法应是保存文件和对文件做日志。

3.4.3 访问路径的恢复

在访问路径使用之前，系统保证其完整性。如果系统发现该访问路径不能用，系统会试图恢复它。你可以控制何时恢复访问路径。详细信息参阅 3.4.3.2.2“重建访问路径时的控制”。

访问路径的恢复可能很费时间，尤其是在要重建大的或很多访问路径的时候，可以通过几种方法减少这一恢复时间，包括：

- 保存访问路径
- 使用访问路径日志
- 控制何时重建访问路径
- 设计文件以减少重建时间
- 使用系统管理访问路径保护

3.4.3.1 保存访问路径

可以通过保存访问路径来减少恢复访问路径所需的时间.SAVCHGOBJ/SAVLIB 和 SAVOBJ 命令中的访问路径参数（ACCPTH）允许保存访问路径。通常只有逻辑文件描述被保存，但在以下条件下也可保存访问路径：

- 说明 ACCPTH(*YES)。

• 所有逻辑文件所用的物理文件被保存并且保存在同一个库中。
• 逻辑文件是 MAINT(*IMMED)或 MAINT(*DLY)

注：带有 ACCPTH(*YES)参数的逻辑文件本身不被保存。你必须明确地保存该逻辑文件。

详情请参阅《备份和恢复－基础》和《备份和恢复－高级》。

3.4.3.2 重存访问路径

如果访问路径被保存并且它所依赖的所有的物理文件也被同时保存，那么访问路径可以被重存。请参考《备份和恢复－基础》、《备份和恢复－高级》。

恢复访问路径要比重建它要快。例如，要对一个包含 500,000 个记录的物理文件建立一个逻辑文件而且你已检查了（通过显示目标描述命令（DSPOBJD））该逻辑文件的大小是 15 兆。为该逻辑文件重建访问路径需 50 分钟而从磁带上重存同一访问路径只需一分钟左右。（这里假设系统每分钟可以建立大约 10,000 个索引）。

在重存访问路径后，文件可能需要用最近的一次日志修改（这取决于日志是否活动）来恢复到最新状态。例如，系统每小时可能提供 80,000 到 100,000 个日志项，假设每一个物理文件只有一条访问路径，要使用日志项，这个比率将由于物理文件维护访问路径的参数 *IMMED 时要低。即使算上这一额外的恢复时间，重存访问路径也要比重建它更快。

3.4.3.2.1 重访问路径

重建一条数据库访问路径每 10,000 个记录大约要用一分钟。

注：你的系统如果算不出实际上用多少时间，就用这个值来估计。

以下因素会影响这一时间估算（以影响程序的顺序列出）：

• 存贮池大小。重建访问路径所用的存贮池大小是一个非常重要的因素。可以通过在一个较大存贮池中运行作业来提高重建时间。
• 系统型号。对于重建一个访问路径所需时间，系统的处理器速度是一个关键因素。
• 键字长度。一个长键字将会减慢重建访问路径的速度，因为在访问路径中要连接和存贮较多的键字信息。
• 选择/省略值。选择/省略处理将减慢访问路径的重建速度，这是因为对每条记录都要判断它是否满足选择/省略条件。
• 记录长度。一个较长记录会减慢重建访问路径的速度，因为有更多的数据要做检查。
• 数据的存贮设备。存放实际数据的存贮设备和访问路径存放设备的相对速度影响重建访问路径以使它在使用该路径时能尽快找到信息。文件中的记录顺序对于系统维护一条有效的访问路径时，能否尽快地建立它有一些影响。

以上诸因素在估计重建一条访问路径所需时间时必须加以考虑。

3.4.3.2.2 控制何时重建访问路径

如果系统异常结束，则在下一次 IPL 期间，系统自动地列出那些需要恢复访问路径的文件。可以决定是否重建这些访问路径：

• 在 IPL 期间
• 在 IPL 之后
• 在文件初次被使用时

还可以：

• 改变重建访问路径中调度的顺序

- 无限期的挂起一条访问路径的重建
- 在重建一个小于或等于*IPL 重建阈值的访问路径时继续做 IPL
- 在使用编辑重建访问路径命令（EDTRBDAP），系统结束 IPL 处理之后，控制访问路径的重建。

IPL 值用于确定哪些访问路径要在 IPL 期间重建。所有序列值小于或等于 IPL 阈值改为 99，意味着所有序列值为 1 到 99 的访问路径都在 IPL 期间重建。将 IPL 阈值改为 0，则意味着除了那些做了日志的访问路径和系统文件的访问路径之外的所有访问路径，都在系统做好 IPL 之后重建。

一个文件的访问路径恢复值是在文件的生成和修改命令的 RECOVER 参数中规定的。恢复值若为*IPL（在 IPL 期间重建），它的缺省值是 25，*AFTIPL（在 IPL 之后重建）的缺省值是 75；因此，RECOVER(*IPL)将被显示为 25。初始 IPL 阈值是 50；这使得这些参数能影响何时重建访问路径。你可以在编辑重建访问路径的显示中修改这些值。

如果一个文件在 IPL 后立即不需要了，要指定它稍后一些时候重建该文件。这有助于减少在 IPL 期间重建的文件数量，使系统更快地结束其 IPL。

例如，可以指定所有必须重建访问路径的文件在第一次被使用时重建访问路径，在这种情况下，在 IPL 期间不用重建访问路径，只需运行使用第一个重建文件的程序便可控制访问路径重建的顺序。这一方法可减少 IPL 时间（因为在 IPL 期间没有访问路径要重建），并且使可以更快地运行第一个应用程序。但重建所有访问路径所用的时间可能更长（因为在重建访问路径时可能有其它作业在运行，也可能没有足够的主存来支持重建路径）。

3.4.3.2.3 减少访问路径重建时间的文件设计

文件设计也可有助于减少访问路径恢复时间。例如，可以把一个大的主文件分成一个历史文件和一个交易文件。用交易文件增加新数据，历史文件仅用于查询。可以以天为依据，把交易数据合并到历史文件中，然后清除该交易文件，准备开始下一天的数据。使用这种设计会减少访问路径的重建时间。这就是说，如果系统在某一天异常结束，则只有较小的交易文件的访问路径要重建。而大的历史文件的访问路径由于大多数时间它是只读，将很少与其数据同步，因此大大减少了它被重建的机会。

可以考虑使用一个文件设计和使用系统功能（如建立访问路径日志）二者折中的办法。以上所述的文件设计需要一个更复杂的应用设计。根据你的实际情况，可以选用系统提供的功能（如访问路径日志）而不是设计更复杂的应用程序。

3.4.3.3 访问路径的日志

做访问路径的日志可使在系统异常结束之后减少重建访问路径的数量来有效地缩短恢复时间。

注：AS/400 2.2 版及以后的版本极力推荐用户做访问路径的日志,是因为访问路径越大,重建所用的时间越长。

当做一个数据库文件的日志时，文件中记录被修改的映象都被记录在日志中。这些记录映象被用于系统异常结束时恢复文件。然而，在一个异常结束后，系统可能会发现建立在文件上的访问路径与文件中数据不同步，系统必须重建该访问路径以保证二者同步可用。

当对访问路径做日志时，系统在日志中记录访问路径的映象提供已知的访问路径与数据之间的同步点。掌握了日志中的这一信息，系统就可以一并恢复数据文件和访问路径，并且保证二者同步。在这种情况下，可避免花费较长时间重建访问路径。

做访问路径的日志，在系统中与其它恢复功能一起进行。例如，系统有许多选项助于减少从故障恢复的时间，也可替换磁盘单元。这些选项包括用户辅助存贮池及检验和维护。这

些选项可以减少由于磁盘失败造成的必须重装整个系统的机会，但换下坏磁盘后，启动系统也需重建访问路径。通过成功地使用访问路径日志和前面所讲的一些恢复选项，可以减少必须重装整个系统以及必须重建访问路径的机会。

做访问路径的日志用于引用一致关系中，有由于避免发生检查未决的约束。请参阅第十六章”引用的一致性“以获得更多信息。

访问路径日志的启动较容易。可以使用系统管理的访问路径保护（SMAPP）工具（参阅 3.4.3.3.1 中的”系统管理的访问路径保护（SMAPP)”）或使用启动日志访问路径（STRJRNAP）命令自已管理日志环境。

STRJRNAP 命令用于启动建立某一指定文件的访问路径日志。可以为具有立即维护属性（*IMMED）或延迟维护属性（*DLY）的访问路径做日志。一旦日志被启动，系统就持续保护这些访问路径直至该访问路径被删除，或对其执行结束做访问路径日志（ENDJRNAP）命令。

在做一个访问路径日志之前，必须为那些与该访问路径相联系的物理文件做日志。而且，必须为该访问路径和它的相关物理文件使用同一日志。

访问路径的日志被设计了最小的附加输出操作。例如，把修改了的记录的日志数据与访问路径日志数据写在同一输出操作中。但当你启动访问路径日志时，必须认真考虑在用户辅存池中分开放日志接收器。将日志接收器放入它们自已的辅存池中会提高日志性能，同时保护它们免受磁盘故障的影响。有关访问路径日志的详细信息，请参阅《备份和恢复－高级》。

3.4.3.3.1 系统管理的访问路径保护（SMAPP）

系统管理的访问路径保护（SMAPP）为访问路径提供了自动的保护。使用 SMAPP 支持，就无需使用日志命令（如 STRJRNAP）去获得访问路径日志所带来的益处。SMAPP 支持可在一个系统异常结束之后恢复访问路径而不是在 IPL 期间重建它。SMAPP 支持是随系统一起运行的。

系统根据用户所提供的访问路径恢复时间指标或系统提供的缺省时间来决定保护哪一条访问路径。访问路径恢复时间可以是一个系统范围值或 ASP 基值。记录到一个用户定义日志的访问路径不做 SMAPP 保护，因为它们已经被保护了。有关 SMAPP 的详细信息参见《备份和恢复－高级》。

3.4.3.4 避免重建访问路径的其它方法

如果没做访问路径日志，或未利用 SMAPP，那么可能要考虑使用一些其它可以帮助你减少重建访问路径的系统功能。

由系统确定是否重建一个访问路径的方法是一个文件同步指示器。通常该同步指示器处于开状态（on），以表明访问路径及其相关数据同步。当一个作业改变了一个影响访问路径的文件时，系统关闭文件中的同步指示器。如果系统异常结束，它必须重建所有文件同步指示器为关闭状态的访问路径。

为了减少必须被重建的访问路径的数量，需要一个方法定期地使数据与其访问路径同步。有几种可使一个文件与其访问路径同步的方法：

- 全文件关闭。最后一个全（而非共享）系统范围关闭将使访问路径及其数据同步。
- 强制访问路径。在文件的生成和修改命令中规定强制访问路径（FRCACCPTH）参数。
- 强制写系数为 2 或更大。可在数据库文件的生成、修改或覆盖命令中规定强制写系数（FRCRATIO）参数。
- 强制结束数据。在程序中运行强制结束数据操作使文件的数据和访问路径同步。（一些高级语言没有此操作。操作详细信息请参阅你的高级语言指南）。

请记住当数据及其访问路径在用前面提到的方法取得同步时，下一个数据会引起同步指示器再次关闭。每一种方示在性能上是否可取也是很重要的；因此，要小心使用。考虑把日志访问路径及保存访问路径作为保护访问路径的首选方法。

3.4.4 系统异常结束后数据库的恢复

系统异常结束后，系统通过几个自动的恢复步骤去处理。这包括如下的一些事情：重建系统目录、使日志与被做日志的文件同步。系统在 IPL 期间和 IPL 之后恢复操作。

3.4.4.1 IPL 期间数据库文件的恢复

在 IPL 期间，系统上只有恢复功能是活动的。

在 IPL 期间，数据库文件的恢复由以下几步组成：

- 当系统结束完成时，处理下列功能：

－删除文件

－移出成员

－重命名成员

－移出目标

－重命名目标

－改变目标主人

－修改文件

－授权

－取消授权

－启动物理文件日志

－启动访问路径日志

－结束物理文件日志

－结束访问路径日志

－修改日志

－删除日志

－恢复 SQL 视窗

－移出物理文件约束

- 当系统结束退回时，运行以下功能（你必须再次运行它们）：

－生成文件

－增加成员

－生成日志

－恢复日志

－增加物理文件约束

- 如果操作员做 IPL（干予的 RPL），“编辑重建访问路径”的显示就出现在操作员的屏幕上。该显示允许操作员为立即延迟维护的文件编辑 RECOVER 选项。（有关 IPL 阈值的更多信息，参阅 3.4.3.2.2）。如果所有的访问路径都有效，或者做非干予的 IPL，就不出现这个显示。
- 有立即或延迟维护的访问路径及规定在 IPL 期间做恢复的访问路径，都要重建，并且有一个信息送到历史日志中。正用的做了日志的访问路径的文件和无效访问路径的系统文件，不在“编辑重建访问路径”显示屏上显示。它们被自动恢复并且有一信息送至历史日志。
- 对非干予的 IPL，如果系统值 QDBRCVYWT 是 1（等待），那些被指定在 IPL 之后

恢复的文件就作为在 IPL 期间恢复的文件对待。有关系统值 QDBRCVYWT 的更多信息参阅《工作管理指南》。

- 验证处于检查未决中的引用约束，一条信息送往历史日志。
- 下面信息被送到历史日志中：
  - －以上所列的各项成功或失败
  - －在系统异常结束时被打开的物理文件成员和每个成员中最后一个活动的相关记录号
  - －不能与日志同步的物理文件成员
  - －IPL 数据库恢复已经完成。

3.4.4.2　IPL 之后数据库文件的恢复

这一恢复步骤在 IPL 完成之后运行。在做这一数据库恢复步骤的同时，交互用户可以运行批作业也可以运行交互作业。

IPL 之后的恢复由以下各步组成：

- 指定在 IPL 之后恢复的立即或延迟维护的文件，其访问路径被重建(参看 3.4.4.3 中的图 13－1)。
- 信息送至系统历史日志以说明重建操作的成功或失败。
- IPL 结束之后，用编辑重建访问路径命令（EDTRBDAP）可用来为访问路径的重建排序。
- IPL 结束之后，编辑检查未决约束命令（EDTCPCST）显示了在检查未决中的物理文件约束的清单。可以用这一命令去规定未决约束的检验硕序。

注：如果对一个文件未做日志，在 IPL 恢复之后记录可能存在也可能不存以，如下所述：

对于增加记录，如果在 IPL 恢复之后增加的第 N 个记录存在，那么所有 N 之前的记录也存在。

对于更新和删除文件，如果在 IPL 之后对第 N 个记录的更新和删除实现了，并能保证在第 N 个记录前做的更改或删除的记录也实现。

对于 REUSEDLT（*YES），增加的记录被看作是修改记录，因此，不能保证在 IPL 恢复之后记录仍存在。

3.4.4.3　数据库文件恢复选项表

下面是文件恢复选项的总结：

图 13－1

访问路径维护	*NO	*AFTER	*IPL
键字顺序访问路径立即或延迟维护	在 IPL 时不做数据库恢复。文件立即可用，在文件第一次打开时重建访问路径	IPL 之后重建访问路径	IPL 期间重建访问路径
键字顺序访问路径重建维护	在 IPL 时不做数据库恢复。	对重建维护没有恢复要做	对重建维护没有恢复要做
到达顺序访问路径	在 IPL 期间不做数据文件恢复，文件立即可用	对一个到达顺序访问路径没有恢复要做	对一个到达顺序访问路径没有恢复要做

3.4.5　存贮池分页选项对数据库恢复的影响

共享池分页选项控制系统动态将存贮池的分页特性调整到一个最好性能。

- 对*FIXED 的分页选择，系统不动态调整分页特性。
- 对*CALC 分页选择，系统对分页特性做动态调整。
- 也可以通过一个应用编程接口控制分页特性。详细信息参阅《System API Reference》一书。

非 FIXED 的共享池分页选项会在系统故障时对未做日志的物理文件的数据丢失产生影响。在未对物理文件做日志时，由于系统故障，没有保存内存的数据，造成数据丢失，如果规定*CALC 选项或 USRDFN 选项数据丢失会更加严重。这些选项使文件的修改、写入主存的频率很低，所以丢失数据的危险就比较大。有*FIXED 选项的未做日志文件丢失数据的危险比*CALC 或用户定义（USRDFN）选项危险性更大。

有关分页选项的更多信息请参阅《工作管理指南》中“自动系统调书”一节。

第十四章　使用源文件

本章将描述源文件。讨论源文件的概念以及为什么要使用源文件。同时，也讨论如何建立一个源文件，如何输入数据到源文件，如何使用源文件在系统上生成另一个目标（例如，一个文件或程序）。

3.5.1 源文件的概念

当单单的一个命令不能为生成一个目标提供充分的信息时，使用源文件。它包括输入建立某类目标所需的（源）数据。例如，要生成一个控制语言（CL）程序，必须使用一个包含 CL 语句的源文件，它是一些有格式的命令。要生成一个逻辑文件，必须使用一个包含 DDS 语句的源文件。

要生成以下目标，必须使用源文件：

- 高级语言程序
- 控制语言程序
- 逻辑文件
- 系统内部通讯功能（ICF）文件
- 命令
- 转换表

要生成以下目标，可以使用源文件但不是必须的：

- 物理文件
- 显示文件
- 打印文件

一个源文件可以是一个数据库文件、磁盘文件、磁带文件或联机数据文件。（一个联机数据文件是作业的一部分）。一个源数据库文件只不过是数据库文件的另一种类型。可以象使用系统上任何其它数据库文件一样使用源文件。

3.5.2　建立源文件

要建一个源文件，可以使用生成源物理文件（CRTSRCPF）命令，生成物理文件（CRTPF）命令或生成逻辑文件（CRTLF）命令。通常，会使用 CRTSRCPF 命令去生成一个源文件，这里有许多参数的缺省值都是常用的。（如果想用 DDS 去生成一个源文件，用 CRTPF 或 CRTLF 命令。）

CRTSRCPF 命令生成一个物理文件，但其具有与源物理文件相应的属性。例如，一个源文件的缺省记录长度是 92（源数据字段 80，源顺序号字段 6，源日期字段 6）。

下例说明了如何使用 CRTSRCPF 命令生成一个源文件，这里用的是缺省值。

```
CRTSRCPFFILE(QGPL/FRSOURCE) TEXT('source file')。
```

3.5.2.1　IBM 提供的源文件

为使用方便，OS/400 程序和其它特许程序为每个源类型提供一个数据库源文件。这些源文件是：

文件名	库　名	内　容
QBASSRC	QGPL	BASIC programs
QCBLSRC	QGPL	System/38 compatible COBOL

QCSRC	QGPL	C programs
QCLSRC	QGPL	CL programs
QCMDSRC	QGPL	Command definition statements
QFTNSRC	QGPL	FORTRAN programs
QDDSSRC	QGPL	Files
QFMTSRC	QGPL	Sort source
QLBLSRC	QGPL	COBOL/400 programs
QS36SRC	#LIBRARY	System/36 compatible COBOL programs
QAPLISRC	QPLI	PL/I programs
QPLISRC	QGPL	PL/I programs
QREXSRC	QGPL	Procedures Language 400/REXX programs
QRPGSRC	QRPG	RPG/400 programs
QARPGSRC	QRPG38	System/38 environment RPG
QRPG2SRC	#RPGLIB	System/36 compatible RPG II
QS36SRC	#LIBRARY	System/36 compatible RPG II (after install)
QPASSRC	QPAS	Pascal programs
QTBLSRC	QGPL	Translation tables
QTXTSRC	QPDA	Text

可以把你的源成员加到这些文件中，也可以生成自己的源文件。通常，可以建立与 IBM 提供文件同名的文件，但你的文件与系统提供文件要在不同的库中（当装入一个系统的新版本时，IBM 提供的文件可能会被覆盖）。IBM 提供的源文件按照相应的生成命令所使用的文件名被生成（例如，CRTCLPGM 命令使用 QCLSRC 文件名作为缺省值）。同时，IBM 提供的程序员菜单使用同一缺省名。如果生成自己的源文件，不要将它们与 IBM 提供的源文件放在一个库中。（如果使用了与 IBM 提供名相同的文件名，那么保证在库列表中你的源文件的库放在 IBM 源文件所在库之前。）

3.5.2.2　源文件属性

源文件通常有以下属性：

- 记录长为 92 个字符（这包括 6 字节的序列号，6 字节的日期和 80 字节的源数据）。
- 键字（序号）唯一，不管访问路径是否指定唯一。不必为一个源文件指定键字。源文件被生成为无键字文件（到达顺序访问路径）。具有到达顺序访问路径的源文件需要较少的存贮空间并减少了比较指定键字顺序访问路径的源文件的保存/重存时间。
- 多个成员。
- 成员名与使用该成员生成的目标名一致。
- 所有记录有相同的记录格式。
- 与大多数数据文件相比，每个成员有相对少的记录。

某些限制是：

- 如果规定键字，那么顺序号就做为键字。
- 如果规定键字，它必须是升序的。
- 访问路径不能规定唯一键字。
- 源文件的 DDS 中不允许有 ALTSEQ 键字。
- 第一个字段必须是一个包含区位十进制两位小数的 6 位数字顺序号。
- 第二个字段必须是一个包含区位十进制无小数位的 6 位日期字段。
- 第二个字段以后的所有字段必须是区位十进制或字符。

3.5.2.3　不用 DDS 生成源文件

不用 DDS 而通过说明记录长度（RCDLEN 参数）来建立一个源物理文件时，生成的源文件包括三个字段：SRCSEQ、SRCDAT 和 SRCDTA。（记录长度必须包括 12 个字符的顺序号和最后修改日期字段，这样数据部分的长度就等于记录长减 12）。记录的数据部分可定义为包含多个字段（每个字段必须是字符或区位十进制）。如果你想定义记录的数据部分包括多个字段，必须使用 DDS 来定义字段。

使用生成源物理文件（CRTSRCPF）命令生成的源物理文件自动使用。由以下三个字段组成的记录格式：

字 段	名　字	数据类型和长度	说　　明
1	SRCSEQ	区位十进制 6 位数字二位小数	记录的顺序号
2	SRCDAT	区位十进制 6 位数字	记录最后修改日期
3	SRCDTA	字符型、任意长	记录的数据部分(正文)

注：对于所有 IBM 提供的数据库源文件，数据区长度都为 80 字节。对于 IBM 提供的设备源文件，数据区的长度是相关设备允许的最大记录长度。

3.5.2.4　用 DDS 生成源文件

如果想生成源文件并需为它定义记录格式，就使用生成物理文件（CRTPF）或生成逻辑文件（CRTLF）命令。如果生成一个逻辑文件，该逻辑文件成员只引用一个物理文件或成员以避免重复键字。

3.5.3　处理源文件

以下各节描述如何处理源文件输入和维护数据。

3.5.3.1　使用源语句录入实用程序（SEU）

可以使用源语句录入实用程序（SEU）来输入和修改一个源文件中的语句。如果使用 SEU 在数据库中输入数据，SEU 会为每个记录加上一个顺序号和日期字段。

如果使用 SEU 去更新一个源文件，可以在两个现存记录中间增加记录。例如，如果在 0003.00 和 004.00 两个记录之间加一个记录，加的记录号就为 0003.01，SEU 将按照这种方法自动地安排新加入的语句。

当记录是首次被放入源文件中时，该数据字段全部是　位数零（除非你在 DDS 中说明了 DFT 键字）。如果使用 SEU，则当修改记录时，其日期字段也会改变。

3.5.3.2　使用设备源文件

磁带和磁盘文件可象源文件一样生成。当设备文件作为源文件使用时，其记录长必须包括顺序号和日期字段。最大记录长必须考虑到附加的 12 个字符。例如，一个磁带记录的最大记录长为 32,766，如果数据要被当作源输入处理，则实际磁带数据记录的最大长度为 32,754（即 32,776 减去 12）。

打开一个源设备文件做输入，系统增加顺序号和日期字段，但在日期字段中是零。

打开一个源设备文件做输出，并且被定义为一个源文件，则系统在往设备写数据之前删除顺序号和日期。

3.5.3.3　复制源文件数据

拷贝源文件（CPYSRCF）和拷贝文件（CPYF）命令可用来向源文件写入或输出数据。

从数据库源文件向另一个有触发器的数据库文件中做拷贝时，在每个记录被拷贝时都会调用这个触发器程序。

3.5.3.3.1　使用 CPYSRCF 命令拷贝源文件

CPYSRCF 命令用来对数据库源文件的操作。尽管功能上与拷贝文件（CPYF）命令相似，但 CPYSRCF 命令可提供一些在拷贝源文件时通常要用到的缺省值。例如，它有一个假定 TOMBR 与 FROM MBR 参数相同的缺省值，同时假定任一 TOMBR 记录都被替换。当规定了 TO FILE(*PRINT)，CPYSRCF 命令也支持一个唯一的打印格式。因此，在拷贝数据库源文件时，愿意使用 CPYSRCF 命令。

CPYSRCF 命令自动地将数据从原来文件（from-file）的 CCSID 置换至被拷入文件（to-file）的 CCSID。

3.5.3.3.2　使用 CPYF 命令拷贝文件

CPYF 命令较 CPYSRCF 更多的功能，如：

- 从数据库文件拷贝到设备文件
- 从设备文件拷贝到数据库文件
- 在非源数据库文件中，源数据库文件之间拷贝
- 以十六进制格式打印源成员
- 用选择值来拷贝源文件

3.5.3.3.3　拷贝中使用的源顺序号

向一个数据库源文件做拷贝，可使用 SRCOPT 参数去更新顺序号，并将日期初始为 0。系统使用缺省值给第一个记录安排一个顺序号 1.00 并以 1.00 为步长对后面的记录按升序赋予顺序号。可使用 SRCSEQ 参数设置一个小数增量，并且从哪个顺序号开始重新编号。例如，如果用 SRCSEQ 参数设置增量为.10 并从顺序号 100.00 开始，则被拷贝的记录将会有 100.00，100.10，100.20 等等的顺序号。

如果初值为.01 增量为 01，则它能具有最大记录数的顺序号为 999,999。当达到最大顺序号（9999.99）时，其余记录的顺序号为 9999.99。

下面是一个在同一个文件中拷贝成员的例子。如果 MBRB 不存在，就加进它；如果存在，则所有记录被替换。

```
CPYSRCF  FROMFILE(QCLSRC)  TOFILE(QCLSRC)  FROMFILE(MBRA)+
                           TOMBR(MBRB)
```

下面是一个按成员类属名拷贝文件成员的例子。所有以 PAY 开头的成员都被拷贝。如果相应的成员不存在，则要增加；若存在，则所有记录被替换。

```
CPYSRCF  FROMFILE(LIB1/QCLSRC)  TOFILE(LIB2/QCLSRC)+
                                FROMMBR(PAY*)
```

下面是一个将成员 PAY1 拷贝到打印文件 QSYSPRT（*PRINT 的缺省值）的例子。格式与打印 SEU 原语句的格式相似。

```
CPYSRCF  FROMFILE(QCLSRC)  TOFILE(*PRINT)  FROMMBR(PAY1)
```

把设备文件拷贝到一个数据库文件时，要加顺序号，日期被初始化为 0。顺序号从 1.00 开始以 1.00 为步长增加。如果被拷贝的文件有多于 9999 个记录，除非说明了 SRCOPT 和 SRCSEQ 参数，则顺序号被卷回至 1.00，然后继续增加。

把一个数据库源文件拷贝到一个设备文件时，日期和顺序号字段被移走。

3.5.3.4　在程序中使用源文件

可以在程序中处理源文件。使用源文件的外部定义并对该文件使用任何输入输出操作，就象使用任何数据库文件一样。

源文件是外部描述的数据库文件。因此，当程序中命名一个源文件并编译它时，源文件描述会被自动地包括在程序打印输出中。例如，对源文件 QDDSSRC 中的一个成员 FILEA 做读和修改记录操作。写程序处理该文件时，系统会从源文件中包括进 SRCSEQ，SRCDAT 以及 SRCDTA 字段。

注：可以使用显示文件字段描述（DSPFFD）命令去显示一个文件中定义的字段。关于此命令的详细信息请参阅 3.3.1.2。

处理 QDDSSRC 文件中的 FILEA 成员的程序能够：

- 打开文件成员（象任何其它数据库文件成员一样）
- 从源文件中读或修改记录（可能会修改存贮了源数据的 SRCDTA 字段）。
- 关闭源文件成员（象任何其它数据库文件成员一样）。

3.5.4　用源文件生成目标

可以使用生成命令用源文件去生成一个目标。如果使用源文件生成目标，可以在生成命令中说明源文件名。

例如，要生成一个 CL 程序，使用生成控制语言程序（CRTCLPGM）命令。一个生成命令通过一个 SRCFILE 参数说明源文件存贮在哪里。

生成命令允许你不用说明源文件名和成员名，只要做到：

1、使用生成目标种类缺省的源文件名。（要查找该命令的缺省文件名，请参阅 3.5.2.1。）

2、生成的目标名与源成员同名。

例如，要使用命令缺省值生成 CL 程序 PGMA，你只需敲入：

```
CRTCLPGM  PGM(PGMA)
```

系统将会认为 PGMA 的源语句在源文件 QCLSRC 的成员 PGMA 中。QCLSRC 包含文件的库由库列表决定。

另一个例子，生成物理文件（CRTPF）命令用源文件 FRSOURCE 生成文件 DSTREF。这个源成员名为 DSTREF。因为未说明 SRCMBR 参数，系统认为成员名。DSTREF 就是生成的目标名。

```
CRTPF  FILE(QGPL/DSTREF  SRCFILE(QGPL/FRSOURCE)
```

3.5.4.1　　用批作业中的源语句生成目标

如果在一个批作业中用生成命令，可使用一个联机数据文件作为命令的源文件。这个联机数据文件不应超过 10,000 个记录。联机数据文件可命名也可不命名。命名的联机数据文件有唯一文件名，是在//DATA 命令中说明的。关于联机文件更多信息参阅《数据管理》。

未命名联机数据文件没有唯一文件名，它们都被称为 QINLINE。下面是联机文件用作源文件的例子：

```
//BCHJOB
CRTPF FILE(DSTPRODLB/ORD199) SRCFILE(QINLINE)
//DATA FILETYPE(*SRC)
```

```
  .
  .   (source statements)
  .
//
//ENDBCHJOB
```

此例中，在//DATA 命令没有说明文件名。当作业被假脱机读者处理时，生成一个未命名的假脱机文件。CRTPF 命令必须指定 QINLINE 作为源文件名访问未命名文件。//DATA 命令也说明该联机文件是一个源文件（FILETYPE 参数规定为*SRC）。

如果在//DATA 命令上说明了一个文件名，必须在 CRTPF 命令中为 SRCFILE 参数说明相同的名字。例如：

```
//BCHJOB
CRTPF FILE(DSTPRODLB/ORD199) SRCFILE(ORD199)
//DATA FILE(ORD199) FILETYPE(*SRC)
  .
  .   (source statements)
  .
//
//ENDBCHJOB
```

如果程序使用了联机文件，系统按指定名检索第一个联机文件。如果未找到这样的文件，程序使用第一个未命名的文件（QINLINE）。

如果未在生成命令中说明源文件名，则认为使用一个 IBM 提供的源文件。例如，要生成一个 CL 程序，但并未说明一个源文件名，则会用 IBM 提供的源文件 QCLSRC。在 QCLSRC 中应该已经放置了源数据。

如果源文件是一个数据库文件，可以说明一个包含所需源数据的源成员。如果未说明源成员，则这些源数据必须在一个与生成的目标同名的成员中。

3.5.4.2　确定用哪个源文件成员生成目标

用源语句生成一个目标时，它将获取有关源文件、库及成员的信息。在该目标生成之前，源成员最后一次改动的日期/时间也被保存在目标中。

目标中的信息有助于决定使用了哪一个源成员以及目标生成以来，源成员是否被改动过。

可以用以下命令来保证用于生成目标的源语句就是当前在源成员中的源语句：

- 显示文件描述命令（DSPFD），并使用 TYPE(*MBR)。这显示了源成员的日期/时间。最后一次源语句更新日期/时间值用来与 DSPOBJD 命令的源文件日期/时间值比较。
- 显示目标描述命令（DSPOBJD），并用 DETAIL(*SERVICE)。这显示用于生成目标的源成员的日期/时间。

注：如果用写到输出文件的数据去确定源成员和目标的日期是否一致，那么可将 DSPOBJD　DETAIL(*SERVICE)命令的输出文件中的 ODSRCD(源日期)、ODSRCT(源时间)

字段与 DSPFD　TYPE(*MBR)命令以输出文件中的 MBUPDD(成员更新日期)、MBUPDT(成员更新时间)字段做一个比较。

3.5.5　管理源文件

这节描述管理源文件应考虑的几个问题。

3.5.5.1　修改源文件属性

如果用 SEU 维护数据库源文件，参看《AS/400－源语句录入程序》一书中关于如何修改数据库源文件的信息。如果未使用 SEU 维护数据库文件，必须全部替换已存在的成员。

如果源文件在软盘上，可将它拷贝至一个数据库文件中，用 SEU 修改它，再写回软盘。如果不用 SEU，那不得不删除旧的源文件并生成一个新的源文件。

如果修改了一个源文件， 那么以前用没改过的源文件生成的那个目标，不再与当前源文件匹配，要删掉旧的目标， 然后用修改后的源文件重新生成。例如，如果修改变了 3.5.4 中生成的那个源文件 FRSOURCE，必须删除文件 DSTREF，并使用新的源文件再次生成它，以使 DSTREF 与修改了的 FRSOURCE 源文件匹配。

3.5.5.2　重新组织源文件成员数据

如果使用到达顺序源文件，那么通常无需重组织一个源文件成员。

要给所有记录分配唯一的顺序号，在重组物理文件成员命令（RGZPFM）中说明如下参数：

- KEYFILE(*NONE)以使记录不被重组
- SRCOPT(*SEQNBR)使顺序号改变
- SRCSEQ 带有一个小数值，如.10 或.01，使所有的顺序号唯一。

注：被删除的记录如果确实存在，这些记录将被清除出去。

如果用物理文件生成了一个键字顺序访问路径的逻辑文件，那么要用顺序号重新组织到达顺序访问路径的源文件。

3.5.5.3　确定何时修改了源语句

每个源记录包括一个日期字段，如果语句发生变化，该字段会被 SEU 自动更新。这可用于确定源语句被最后修改的时间。大多数高级语言编译程序都在编译程序清单上打印这些日期。拷贝文件（CPYF）和拷贝源文件（CPYSRCF）命令也打印这些日期。

每个源成员描述字段包括两个日期和时间字段。第一个日期/时间字段反映成员修改结束后关闭的时间。第二个日期/时间字段反映对成员的修改。这包括由 SEU、命令（如 CPYF 和 CPYSRCF）、授权变化和文件状态变化所做的修改。例如，修改物理文件（CHGPF）命令中的 FRCRATIO 参数改变成员属性。这个日期/时间字段由保存修改过目标（SAVCHGOBJ）命令来确定成员是否要保存。日期/时间字段都可用在显示文件描述（DSPFD）命令中规定 TYPE（*MBR）来显示。源成员显示中有两个日期/时间字段：

- Last source update date/time（最后源语句更新日期/时间）。这个值反映了成员中源数据记录的任何变化。当修改一个源成员时，最后修改日期/时间值也被更新，尽管日期/时间值可能会有 1 或 2 秒的不同。
- Last change date/time（最后修改日期/时间）。这个值反映了成员的任何变化。这包括所有由 SEU、命令（如 CPYF 和 CPYSRCF）、授权变化或文件状态发生改变引起的变化。例如，CHGPF 命令中的 FRCRATIO 参数改变成员状态，因此，被反映在最后改变日期/时间值中。

3.5.5.4　文档使用的源文件

可使用 IBM 提供的源文件 QTXTSRC 帮助你生成和更新联机文档。

可象任何其他应用程序（如 QRPGSRC 或 QCLSRC）一样用 SEU 生成和更新 QTXTSRC 成员。QTXTSRC 文件对叙述文档文件非常有用，可被恢复联机或打印。源成员中的文件可很容易地用 SEU 的增加、修改、移动、拷贝操作去更新。可在结束 SEU 的提示上规定打印当前源示文件成员为‘Yes’来打印整个成员。还可以写一个程序来打印所有或部分源成员。

第十五章　物理文件约束

DB2/400 数据库支持以下的物理文件约束条件：

- 引用约束
- 主关键字约束
- 唯一性约束

引用约束说明在第十六章“引用的一致性”中描述。物理文件约束的各方面内容主要在这一章中描述。

主键字和唯一性约束都有用，而且当使用引用约束时也要用到它们。唯一性约束让你在文件访问路径之外为一个文件生成附加的强制的唯一键字。唯一和主键字约束在文件插入、更改和删除期间实施。主键字和唯一约束与逻辑文件中键字访问路径相类似。

可以用 SQL　CREATE　TABLE(SQL 生成表)和 ALTER　TABLE(替换表)语句增加和去掉约束。这一章介绍处理物理文件约束的常用的操作系统方法，也即 CL 命令。关于在 DB2/400 SQL 上处理约束条件的方法，请参阅《DB2/400 SQL 编程》和《DB2/400 SQL 参考手册》。

3.6.1　唯一性约束

一个唯一性约束标识数据库文件中符合下述所有条件的一个或一系列字段：

- 在文件中唯一
- 是升序的
- 可为空值

一个文件可有多个唯一性约束，但不能重复。同一个键字字段，无论其顺序如何，都当一个重复约束对待。

3.6.2　主键字约束

一个主键字约束标识数据库中所有符合以下条件的一个或一系列字段：

- 在文件中唯一
- 升序
- 不可为空值

对一个文件只能定义一个主键字约束。一个主键字约束是一个具有特殊属性的唯一键字约束。

3.6.3　增加唯一和主键字约束

增加物理文件约束（ADDPFCST）命令增加所有类型的物理文件约束。要增加唯一约

束，将类型参数规定*UNQCST，要增加主键字约束，类型规定为*PRIKEY。当增加一个主键字约束时，说明的键字变为文件的主访问路径。如果文件没有可被共享的键字访问路径，系统将会生成一个。

KEY 参数在增加唯一和主键字约束时是必需的。

有关增加引用约束的信息，请参阅第十六章“引用一致性”。

3.6.4 去掉约束

除去物理文件约束命令（RMVPFCST）去掉一个约束。RMVPFCST 命令的影响依赖于要除去什么样的约束以及该约束所处环境的某些条件。

TYPE 参数说明除去的约束类型：

- 唯一约束类型 TYPE(*UNQCST)
  - －当规定 CST(*ALL)时，除了主键字约束之外的所有唯一约束被除掉
  - －CST(约束名称)规定的约束名
- 主键字约束类型 TYPE(*PRIKEY)
  - －CST(*ALL)时，主约束被去掉
  - －CST(约束名)时，规定的约束名被去掉

如果去掉了一个主键字约束或唯一约束但相关访问路径与一个逻辑文件共享，则共享路径的所有权传送给逻辑文件。如果访问路径未被共享，则被除去。

有关去掉引用约束的信息、详参第十六章“引用一致性”。

3.6.5 处理物理文件的约束

对物理文件约束的操作命令（WRKPFCST）显示一个约束列表。其中包括约束名、文件和库名、类型、状态以及它是否在检查未决状态中。从这个显示中可以修改或除去某个约束。也可显示已引起文件被放入检查未决中的记录清单。

Work with Physical File Constraints

Type options, press Enter.
2=Change 4=Remove 6=Display records in check pending

Opt	Constraint	File	Library	Type	State	Check Pending
_	DEPTCST	EMPDIV7	EPPROD	*REFCST	EST/ENB	No
_	ACCTCST	EMPDIV7	EPPROD	*REFCST	EST/ENB	Yes
_	STAT84	EMPDIV7	EPPROD	*REFCST	DEF/ENB	No
_	FENSTER	REVSCHED	EPPROD	*REFCST	EST/DSB	No
_	IRSSTAT3	REVSCHED	EPPROD	*UNQCST		
_	IFRNUMBERO >	REVSCHED	EPPROD	*UNQCST		
_	EVALDATE	QUOTOSCHEM	EPPROD	*REFCST	EST/ENB	No
_	STKOPT	CANSCRONN9	EPPROD	*PRIKEY		

```
Parameters for options 2, 4, 6 or command
===>_____________________________________________________________
F3=Exit  F4=Prompt  F5=Refresh  F12=Cancel  F15=Sort by
F16=Repeat position to  F17=Position to  F22=Display constraint name
```

图 15-1　处理物理文件约束的显示

上面显示了在 WRKPFCST 命令中说明的文件定义的所有约束。列出了约束名、文件名和库名。Type 列给出约束是引用、唯一还是主键字的。State 列说明了该约束是否定义或建立了，以及它能用还是不能用。最后一列包括了检查未决状态的约束。唯一和主键字没有状态（state），因为它们总是已建立的并且是能用的。

使用这一显示屏，可有如下选择：

• 修改（选项 2）到任一允许状态。例如，可以使一个目前不能用的约束变为能用的。这一选项实现与 CHGPFCST 命令相同的功能。

• 去掉（选项 4）一个约束。这一选项实现与 RMVPFCST 命令相同的功能。

• 显示（选项 6）在检查未决中的记录。这一选项实现了与 DSPCPCST 命令相同的功能。关于检查未决和引用的讨论请参阅 3.7.5“未决检查”。

3.6.6 显示检查未决约束

显示检查未决约束命令（DSPCPCST）显示或打印引起约束被标记为检查未决的那些记录。用这一命令之前，用修改物理文件约束命令（CHGPFCST）使约束不能用。当相关文件有大量记录时这一命令可能要运行较长时间。图 15-2 显示了 DSPCPCST 命令是如何显示检查未决中的记录的。

```
                          Display Report
Width . . .:        71
Column  . .:         1
Control  . . . .   _________
Line    ....+....1....+....2....+....3....+....4....+....5....+....
        PARTNUM   PARTID      QUANTITY
        -------   -------     --------
00001   25RFA1    WIDGET          3000
00002   32XGW3    GIZMO             32
*****   * * * * * E N D  O F  D A T A * * * * *
```

```
                                                                    Bottom
F3=Exit  F12=Cancel  F19=Left  F20=Right  F21=Split
```

图 15-2　显示 DSPCPCST 命令

3.6.6.1 处理检查未决约束

编辑检查未决约束命令（EDTCPCST）显示了如图 15-3 所示的编辑检查未决约束的显示屏。

这一显示有助于管理和调度放在检查未决中的物理文件约束。

当检查一个物理文件约束时，数据库必须保证文件中的每一个都符合约束定义。例如，一个检查引用约束会引起数据库去检查每一个外来键字的值是否有一个相匹配的键值。有关引用约束的更多信息参见第十六章“引用的一致性”。

```
                    Edit Check Pending Constraints

Type sequence, press Enter.
Sequence:   1-99, *HLD

                 ----------Constraints----------  Verify    Elapsed
Seq  Status     Cst          File        Library    Time      Time
1    RUN        EMP1         DEP         EPPROD     00:01:00  00:00:50
1    READY      CONST    >   DEP         EPPROD     00:02:00  00:00:00
*HLD CHKPND     FORTH    >   STYBAK      EPPROD     00:03:00  00:00:00
*HLD CHKPND     CST88        STYBAK      EPPROD     00:10:00  00:00:00
*HLD CHKPND     CS317        STYBAK      EPPROD     00:20:00  00:00:00
*HLD CHKPND     KSTAN        STYBAK      EPPROD     02:30:00  00:00:00

                                                                  Bottom
F3=Exit  F5=Refresh  F12=Cancel  F13=Repeat all  F15=Sort by
F16=Repeat position to  F17=Position to  F22=Display constraint name
```

图 15-3　编辑码检测未决限制显示

状态字段有以下值：

• RUN 表明约束正被检查。

• READY 表明约束已准备好被检查。

• INVAP 表明与约束相关的访问路径无效。一旦该访问路径被重建，约束会由系统自动检查。

• HELD 表明约束未被检查。必须改变顺序值为 1 到 99 来改变这个状态。

• CHKPND 表明已试图去检查该约束，但约束仍在检查未决中。必须改变顺序值为 1 到 99 来修改这个状态。

约束列包括约束名的前五个字符。如果这个名字后面接着一个>表示名字超过五个字符，可将光标移到那一行下面并按 F22 键来显示整个名字。

检查时间列，显示系统中无其它作业检查约束需花费的时间。过去的时间列显示不用的检查约束时间。

有关检查未决和引用约束的讨论详见 3.7.5“检查未决”。

3.6.7 物理文件约束考虑和限制因素

文件必须是一个物理文件。

文件有最大成员数 1 即 MAXMBR（1）。

当文件无成员时可定义一个约束。只有等到文件有且仅有一个成员时约束才可建立。

文件可有一个主键字的最大值但可以有许多父代键字。

每个文件最多可有 300 个约束关系。这个最大值是下述各项的总和：

--唯一约束

--主键字约束

--引用约束，即可是源约束的一部分，也可以是依赖源约束的，既可是定义的，也可是后建立的约束。

约束名在一个库中必须唯一。

在 QTEMP 库中不能加约束。

第十六章　引用的一致性

这一章讲述了在 DB2/400 数据库中引用一致性的实施方法。

3.7.1 引用一致和引用约束简介

引用一致性是一个广泛的术语，它包括所有保证关系数据库中引用数据有效性的技术和技巧。引用限制是一个强制数据库达到某个有效级别的技术。从 3.1 版本开始这一能力成为 OS/400 操作系统的一部分。

由于以下几个原因用户想在自己的数据库管理系统中实现引用一致性：

- 保证文件之间的数据值保持在一种适合他们业务要求的状态。例如，如果某业务有些顾客清单在一个文件中而他们的帐目在另一个文件中，如果顾客不存在的话，相关帐目文件就没有什么意义了。与此类似，如果一个用户的帐目记录没有删除，就没有理由删除此用户记录。
- 能够定义数据值之间的联系。
- 不论应用程序怎么修改，系统强制数据关系不发生问题。
- 为了提高整体检查的性能，把在高级语言（HLL）和 SQL 中做的检查都放在数据库中来做。

3.7.1.1 引用一致性的有关术语

引用一致性的讨论需要理解几个术语。这些术语是有一定顺序的有助于理解它们之间联系。

- 主键字。数据库中的一个或一系列字段,必须是唯一的、升序的,并且不包含空值(null)。主键字是主文件访问路径。主键字在父代文件中可变为父代键字。
- 唯一键字。数据库文件中的一个或一系列字段,必须是唯一的、升序的可为空值(null)。
- 父代键字。数据库文件中的一个或一系列字段,必须是唯一的、升序的,可为空值(null)。父代键字可以与主键字或唯一键字一样，也可以是主键字的子集。
- 外来键字。一个或一系列字段，其中每个非空值必须与相关父代文件的一个键值匹配。但属性（数据类型、长度等等）必须与主键字或父代键字相同。
- 引用一致性外围键字的数据库状态是有效的。引用一致性的实施能防止违反“非空外围键字必须有一个匹配父代源键字的规则”。
- 引用约束。这个约束定义了在父代键字标识的记录和由外键字标识记录之间的关系联系。因为一个从属文件总是依赖父代文件，所以引用约束是由从属文件的观点来来定义的。它不能从父代文件观点定义。
- 父代文件。包括父代键字或主键字关系的文件。
- 从属文件。包含外来键字关系的文件。
- 检查未决。当数据库不确定地知道某一个从属文件中是否包含与相关父代键字有关的有效数据时发生的状态。
- 删除规则。在试图删除一个父代记录时数据库应采取动作的规定。
- 更新规则。在试图更新一个父代记录时数据库应采取动作的规定。

3.7.1.2 一个简单的引用一致性的例子

数据库包含一个雇员文件和一个部门文件。两个文件都有一个名为 DEPTNO 的部门号

字段。两个数据库文件的相关记录是在雇员文件中 DEPTNO 与部门文件中 DEPTNO 相等的记录。

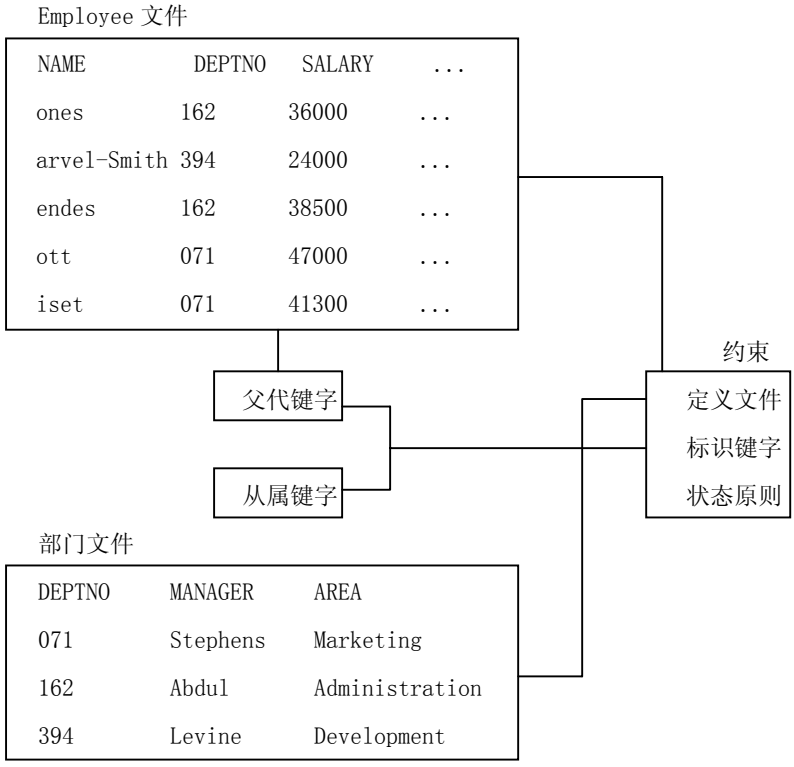


图 16-1　雇员和部门文件

部门文件是父代文件而 department.DEPTNO 是父代键字。它也是文件的主键字。雇员文件是从属文件，而 employee.DEPTNO 是外来键字。DEPTNO 为非空值的雇员文件中的每一个记录与一个且仅与一个部门文件中的记录相关。

在雇员文件的 DEPTNO 值上的约束 是引用约束。这个约束规定了文件、键字，以及用户或应用程序要修改（删除和更新）父代文件时必须遵循的规则。

3.7.2 生成一个引用约束

引用约束关系中涉及的文件必须是最多有一个成员的物理文件。引用约束是一个文件层属性。可以在成员存在之前为一个文件生成一个约束定义。然后可以为每个文件生成成员。

在生成一个引用约束之前必须满足一定的条件：

- 必须有一个带有可做父代键字的键字文件。
- 必须有一个带有某一属性与父代文件相匹配的从属文件：
  --分类序列（SRTSEQ）必须与数据类型 CHAR、OPEN、EITHER 和 HEX 匹配。
  --除非一个或两个 CCSID 是 65535，CCSID 必须与每个 SRTSEQ 表匹配。
  --每一个分类序列表必须严格地匹配。
- 从属文件必须有一个与以下父代键字匹配的外来键字：
  --数据类型
  --长度

--精度（若为压缩、装配或二进制）
--CCSID（除非其中有一个是 65535）
--REFSHIFT（如果数据类型是 OPEN、EITHER 或 ONLY）

3.7.2.1 约束的规则

引用约束允许为系统的实施说明一定的规则。引用约束规则适于父代键字的删除和更新。

约束规则由增加物理文件约束命令（ADDPFCST）设置。用 DLTRULE 参数说明删除规则，用 UPDRULE 参数说明更新规则。

3.7.2.1.1 删除规则

DLTRULE 参数有五个可选值。删除规则定义了当一个父代键字被删除时采取的动作。空的父代键值不受删除规则的影响。

*NOACTION（这是缺省值）
—如果父代键值有一个匹配的外来键字值，不删除父代文件的记灵。

*CASCADE
—父代键字值与外来键字值匹配时，删除父代文件中的记录引起从属文件具有匹配外来键了值的记录被删除。

*SETNULL
—删除父代文件记录会更新其从属文件中的父代非空键值与外来键值相匹配的记录。对于那些符合前面标准的从属记录，外来键字中的所有空属性字段被置为空（null)。具有非空属性的外来键字字段则不修改。

*SETDFT
—父代文件中记录的删除会更新那些从属文件中父代非空键值与外来键值相匹配的记录。对那些符合前面标准的记录，其外来键字字段（一个或多个）被设置为相应的缺省值。

*RESTRICT
—如果父代键字值有一个匹配的外来键字值，则不会发生父代文件中记录的删除。

注：当试图做删除时，系统立即实施一个删除规则（*RESTRICT）。系统在操作的逻辑结束点实施其它约束。这个操作，在其它约束的情况下，包括删除前或后运行的触发器程序。对一个触发器来说可能要去改正违犯潜在引用一致性的操作。例如，一个触发器程序可以加一个不存在的父代记录，有关触发器内容请参看第十七章“触发器”。用*RESTRICT 规则不能防止违反规则的行为。

3.7.2.1.2 更新规则

UPDRULE 参数有两个可选的值，它为父代文件和从属文件之间的约束关系规定更新规则。更新规则说明当试图更新父代文件时应采取的动作。

*NOACTION（这是缺省值）
—如果在从属文件中有一个匹配的外来键字值则不发生父代文件中记录更新。

*RESTRICT
—如果父代键值与外来键值匹配，则不会发生父代文件中的记录更新。

注：当试图更新时系统立即实施一个更新*RESTRICT 规则。系统在操作的逻辑末实施其它约束。例如，一个触发器程序可以增加一个不存在的父代记录。（参见第 17 章“触发器”）用*RESTRICT 规则不能防止违反规则的行为。

如果对一个使用引用约束和删除规则、更新规则但不是*RESTRICT 的文件进行插入、删除、更新操作，必须做日志。父代文件和从属文件都必须用同一个日志接收器。

如果对一个使用引用约束和删除规则、更新规则但不是*RESTRICT 的文件进行插入、删除、更新操作，必须要做落实控制。如果还没有启动落实控制，它将自动地启动和结束。

3.7.2.2 定义父代文件

一个父代文件必须是一个最多有一个成员的物理文件。可以在一个约束中用生成物理文件（CRTPF）、指定唯一、升序并且非空的主键字来生成父代文件。也可以使用一个已存在文件作为一个父代文件。该文件的主访问路径，就是一个潜在的父代键字。“潜在地”意味着用户必须明确地使用 SQL 或增加物理文件约束（ADDPFCST）命令把访问路径变成一个父代键字。

另外，一个唯一键字访问路径可用作父代键字。一个主键字是一个有特别属性的唯一键字。

如果文件生成后，主键字和唯一键字都不足以作父代键字，可选用下面的作法：

- 删除文件并用适当的键字重新生成它。
- 对生成的文件增加一个唯一或主键字约束。

有几种把已有文件作为父代文件的方法，它依赖于已存在文件的环境和与文件相关的约束。

- 可用一个规定 TYPE 参数为*PRIKEY 的增加物理文件约束命令（ADDPFCST），往文件增加一个主键字，也可用 KEY 参数说明键字字段（一个或多个）。
- 如果文件中已有主键字,那么规定了 TYPE(*PRIKEY)的 ADDPFCST 命令就会失败,因为一个文件只能有一个主键字。必须首先用移去物理文件约束命令(RMVPFCST)移去已存在的主键字，然后再增加一个新主键字。
- 用增加物理文件约束命令(ADDPFCST)并指定 TYPE 参数为*UNQCST,将一个唯一约束加到一个文件中。还必须用 KEY 参数说明多个键字字段。也可用 SQL ALTER TABLE（SQL 替换表）增加一个唯一约束。
- 如果父代文件没有一个可为主键字或唯一键字约束共享的已有的键字访问路径，那么系统会生成一个。

3.7.2.3 定义从属文件

一个从属文件必须是一个最多有一个成员的物理文件。可以如同生成任一物理文件一样生成从属文件。

当生成实际的约束时，从属文件并不必需一个键字访问路径。如果没有存在的访问路径符合外来键字标准，系统会为文件加上一个访问路径。

正如 3.7.2 中所列出的，外来键字字段的属性与父代文件中的那些父代键字匹配。

3.7.2.4 检查引用约束

在用 ADDPFCST 命令生成一个约束期间，系统检查证实每一个非空外来键字值是否有一个匹配的父代键字值。一般不会向一个包含大量数据的文件加一个引用约束的。当要引用非常大量的记录时，ADDPFCST 会花费好几个小时来完成。在加约束期间文件全部要锁定。应在生成一个引用约束之前，将时间因素和可用的文件数都计算在内。

如果检查成功，则用户或应用程序把约束规则用于其后的存取中。检查不成功，会使该约束被标记为检查未决，参看 3.7.5.3。

3.7.3 引用一致性的实施

对已建立和可用约束的文件进行的 I/O 访问的不同，依赖于文件在约束关系中是否包括父代键字或外来键字。引用一致性实施是由系统根据所有父代和从属文件的 I/O 请求执行的。数据库为来自应用程序和系统命令（如 INZPFM 命令）的 I/O 请求实施约束规则。

3.7.3.1 外来键字的实施

生成约束中规定的删除和更新规则，提供父代键字的修改。为维护引用一致性，数据库为外来键字更新和插入实施了一个无动作规则。这一规则必须用于外来键字更新和插入，以保证每一个非空外来键字值有一个匹配的父代键字值。

如果对于新的外来键值不存在匹配的父代键字值，返回一个引用约束犯规，从属记录未能插入或更新。详见第九章“处理程序中数据库文件的错误”。

3.7.3.2 父代键字的实施

这一节解释数据库如何根据规则处理父代键字更新和删除。父代键字唯一属性是在所有的父代文件 I/O 上实现的。

3.7.3.2.1 删除规则

从父代文件中删除一个记录时，会执行一个检查以确定在从属文件中是有从属记录（与非空外键字值匹配）。如果发现有从属记录，则由删除规则控制所发生动作：

- no Action（无动作）--如果找到从属记录，返回一个约束犯规，无记录被删除。
- Cascade(级联）--找到的从属记录从文件中删除。
- Set null（置空）--在每一个找到的从属记录中，外来键字的可空字段被置为空。
- Set Default（置缺省值）--当相匹配的父代键字被删除时，所有外来键字字段被置为它们的缺省值。
- Restrict（限制）--除了要立即实施外，与 no action 相同。

如果删除操作的某部分失败，则整个删除操作失败，且所有相关变化被退回。例如，一个 cascade 规则让用户删除十个从属记录，但在删除最后一个记录时系统发生故障。数据库将不允许父代键字被删除，而已删除的从属记录被重新插入。

如果一个引用约束实施引起一个记录改变，则相关的日志项指出这个原因。例如，一个由删除 cascade 规则删除的从属记录将会有一个日志项指示器说明这一原因。关于日志项和指示器的详细信息参见《备份和恢复——高级》。

3.7.3.2.2 更新规则

当更新一个父代文件的父代键字时，实施一个检查以确定从属文件中是否有从属记录（匹配的非空外来键字）。如果有，则由约束关系规定的更新规则控制采取的动作：

- no Action（无动作）--若发现从属记录，返回一个约束犯规，无记录被更新。
- Restrict（限制）--除了要立即实施外，与 no action 相同。

参看第九章“处理程序中数据库文件的错误”。

3.7.4 约束状态

一个文件可处理三种约束状态。其中的两种，约束可改为可用或不可用。图 16-2 显示了这些状态之间的关系。

- 无约束关系状态。在这一状态下不存在对文件的引用约束。如果一个约束关系一度对此文件存在，所有关于它的信息已被移走。

- 定义状态。在从属文件和父代文件之间，已-定义了一个约束关系。没有必要在任一文件中生成一个成员去定义一个约束关系。在此状态下，约束可为：
  —定义的/可用的。这个约束关系只适于做定义用。约束的规则未被实施。此状态下约束在它变为建立状态时保持可用。
  —定义的/不可用的。一个定义为不可用的约束关系做定义用。对约束的各种规则未被实施。此状态的约束在它变为建立状态时仍保持不可用。
- 建立状态。从属文件与父代文件有一个约束关系。仅在外来键字和父代键字之间属性匹配时才建立一个约束。在两个文件中都有成员。在此状态下，约束可为：
  —建立的/可用的。一个建立且可用的约束关系引起数据库实施引用一致性。
  —建立的/不可用的。一个建立而不可用的约束关系导致数据库不去实施引用一致性。

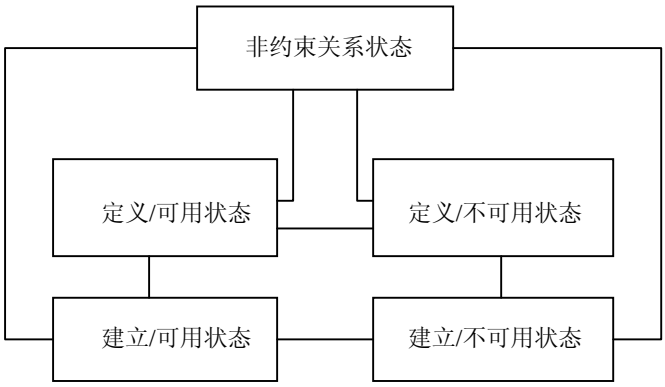


图 16-2　引用一致性状态图

3.7.5 检查未决

检查未决是在父代键字和外来键字之间存在潜在的不匹配时约束关系的条件。当系统确定引用一致性约束可能违约时，约束关系被标记为检查未决。例如：

- 一个恢复操作，其中只有从属文件中的数据被恢复而且数据不与父代键字文件同步（一个外来键字没有父代键字）。
- 当存在一个匹配的外来键字时，系统允许删除父代键字值。这只有当从属文件和父代文件未做日志时才会发生。
- 一个外来键字值没有相应的父代键值。当想往一个从未做过约束关系的文件中增另一个引用约束时，会发生此种情况（参阅 3.7.5.3）。

检查未决状态可为*NO 或*YES。当本书说一个关系是检查未决时，总是意味着检查未决是*YES。

检查未决仅在约束是建立状态才出现。一个建立/可用的约束可有一个*YES 或*NO 的检查未决状态。

要使约束关系脱离检查未决，必须先使关系不可用，更改键字（外来、父代或两者）数据，然后再使约束成为可用的，那么数据库将再检查该约束关系。

当一个约束关系处于检查未决状态时，父代文件和从属文件处在限制它们的使用的情况。父代文件的 I/O 限制与从属文件的限制不同。检查未决限制并不出现在建立/不可用状态的约束（它们总是处于检查未决状态）。

3.7.5.1 检查未决中的从属文件限制

约束关系中被标记为检查未决的从属文件不能有任何关于它的文件 I/O 操作 。从属文件和父代文件的不匹配必须被修正，同时允许在 I/O 操作之前将关系从检查未决中取出。从这样一个文件中读记录是不允许的，因为用户和应用程序可能不知道检查未决状态和约束犯规。

3.7.5.2 检查未决中的父代文件限制

被标记为检查未决的约束关系的父代键字文件可以被打开，但是允许的 I/O 类型有限制：

- 允许读
- 允许插入
- 不允许删除
- 不允许更新

3.7.5.3 检查未决和 ADDPFCST 命令

ADDPFCST 命令让系统验证每个非空外来键字是否有匹配父代键字（参阅 3.7.2.4）。如果数据库将新生成的约束关系标识为检查未决时，约束变为不可用，因此数据可以被更改。一旦数据被更改约束就变为可用的，这样数据库可以再检查从属文件中符合约束定义的每个记录。

可以用显示检查未决约束命令（DSPCPCST）标识违反约束的记录。在这种情况下若有大量记录，DSPCPCST 命令要用相当长时间去处理。

3.7.5.4 检测检查未决约束

显示检查未决约束命令（DSPCPCST）显示或打印引起约束关系被标记为检查未决的记录（或外来键字值）。在使用这一命令前，先用修改物理文件约束（CHAPFCST）命令使约束变为不可用。当相关文件有大量记录时这一命令会用很长时间。3.6.6 中图 15-2 是一个 DSPCPCST 如何显示检查未决中的记录的例子。

3.7.6 允许和禁止一个约束

修改物理文件约束命令（CHGPFCST）使一个或多个引用约束关系成为可用和不可用。在改变一个约束时总是说明从属文件，不能通过说明父代文件使一个约束可用或不可用。把一个约束变为可用或不可用，必须对从属文件有目标管理授权（或 ALTER 即改变权）。

当一个约束是可用或不可用时，父代文件和从属文件被锁定，双方的成员也被锁定，双方的访问路径也被锁定。当做好可用或不可用时锁定被解除。

试图把可用的约束变为可用、不可用的变为不可用，不做什么事，但发布一个通知信息。

一个建立/不可用或检查未决的约束关系可变为可用。可用操作会引起对约束的重检查。如果检查发现父代键字和外来键字不匹配，该约束被标记为检查未决。

不可用约束关系允许对所有父代文件和从属文件 I/O 操作，前提是用户有相应权利。约束仍保持整个基础结构，仍然维护父代键字和外来键字访问路径。但对在不可用关系中的两个文件不能完成引用操作。所有余下的可用约束仍可操作。

让约束不可用，能使文件 I/O 操作运行得更快。在这种情况下总是考虑要权衡一下利弊。文件数据会变成引用无效，而且数据库最终将在约束重新可用后花费大量时间去重新检查约束关系。

注：

- 当修改一个带有处理建立/不可用状态约束关系的文件时，用户和应用程序必须很小

心。

• 只有在约束重变成可用之后，关系才能违约，才能不检查某些关系。

当约束关系是不可用才能用 ALCOBJ 命令定位（锁住）一个文件。这种定位防止文件在引用约束关系是不可用期间做其他改动。

一个锁定的独立使用是必要的，这样使其它用户只能读这些文件。一旦该约束被重新可用，重新定位目标（DLCOBJ）命令解除文件的锁定。

对多个文件处理可用及不可用，它们被依次处理。如果一个约束不能被修改时，会收到一个诊断信息并且处理表中的下一个约束，当所有的约束处理完后，会收到一个完整的约束修改的列表。

3.7.7 去掉一个约束

去掉物理文件约束命令（RMVPFCST）用来去掉一个约束。

RMVPFCST 命令的作用，取决于要解除的是何种约束以及约束所处环境的条件。

使用 CST 参数，可以规定去掉的约束；

• 与一个文件有关的所有约束 CST(*ALL)
• 某一个引用约束 CST(约束名)
• 检查未决中的引用约束（在 3.7.4 中说明）CST(*CHKPND)

使用 TYPE 参数，可规定要去掉的约束类型：

• 所有类型 TYPE(*ALL)
  －所有约束 CST(*ALL)
  －所有检查未决的约束 CST(*CHKPND)
  －CST(约束名)中指定的约束
• 引用约束 TYPE(*REFCST)
  －所有引用约束 CST(*ALL)
  —所有检查未决中的引用约束 CST(*CHKPND）
  －CST(约束名)中规定名字的引用约束
• 唯一约束 TYPE(*UNQCST)
  －除了主键字约束外的所有唯一约束 CST(*ALL)
  －对 CST(*CHKPND)不适的约束，唯一约束不能在检查未决中
  －对 CST(约束名)中规定的唯一约束
• 主键字约束 TYPE(*PRIKEY)
  －对 CST(*ALL），主约束
  －对 CST(*CHKPND)不适用,主约束不可能在检查未决中
  －对 CST(约束名)中规定的主约束

当解除一个引用约束时，相关外来键字和访问路径从文件中移走。外来键字访问路径当其被系统上的任一用户或约束共享时不被去掉。

如果去掉一个引用约束，主键字约束或唯一约束和相应的访问路径被逻辑文件共享，则共享路径的所有权被移交给该逻辑文件。

3.7.8 与引用一致性有关的 AS/400 其他功能

3.7.8.1 SQL 生成表（SQL CREATE TABLE）

SQL 生成表语句允许在生成表时定义列和表约束。更多信息参见《DB2/400 SQL 参考手册》。

3.7.8.2　SQL 修改表(SQL　ALTER　TABLE)

SQL 替换表语句允许对一个已存在 SQL 表增加或除去一个约束。更多信息，详参《DB2/400 SQL 参考手册》。

3.7.8.3　增加物理文件成员（ADDPFM）

在不含有成员的从属文件和父代文件之间定义一个约束关系时：

- 如果一个成员是第一个加到父代文件中，则约束关系保持其定义时的状态。
- 如果接着在从属文件中加一个成员，外来键字访问路径被建立，并且建立与父代文件的约束关系。

3.7.8.4　修改物理文件（CHGPF）

当一个文件存在一个约束关系，在 CHGPF 中就有某些参数不能修改。这些受限制的参数是：

- MAXMBRS

对一个有约束关系的文件，其最多成员数是 1：MAXMBRS（1）。

- CCSID

没有约束关系的文件，其 CCSID 可被修改。如果文件与一个约束有关系，CCSID 只能改为 65535。

3.7.8.5　清除物理文件成员（CLRPFM）

对一个包含记录且与可用的引用约束有关的父代文件使用 CLRPFM 命令时，会导致失败。

3.7.8.6　FORTRAN 中强行结束数据(FEOD)

当对一个有记录的与可用引用有关父代文件使用 FEOD 命令时会失败.

3.7.8.7　生成重复目标(CRTDUPOBJ)

用 CRTDUPOBJ 命令生成一个文件时,源文件(FROM)中的约束被包括进来.

如果重复生成父代文件,无论重复目标放在同一库中还是在不同库中，都会用一个系统交叉引用文件来定位一个定义了引用约束的从属文件，并且会试图建立约束关系。

如果重复生成从属文件，则 TOLIB 会被用来检验约束关系：

- 如果父代文件和从属文件都在同一库中，引用约束关系要建立在 TOLIB 中的从属文件。
- 如果父代文件和从属文件在不同库中，则重复的从属文件引用约束关系将会用初始的父代文件。

3.7.8.8　拷贝文件（CPYF）

拷贝生成一个新文件而原来的文件有约束时，不把约束传给新文件。

有关引用一致性对 CPYF 命令影响，请参考《数据管理》。

3.7.8.9　移出目标(MOVOBJ)

MOVOBJ 命令将文件从一个库移至另一个库，并试图在新库的文件上建立已存在了的引用约束。

3.7.8.10　重命名目标（RNMOBJ）

RNMOBJ 在同一个库中时文件重命名或对库进行重命名。

此命令试图对重命名文件或库建立已定义的引用约束。

3.7.8.11　删除文件（DLTF）

DLTF 命令有一个可选的键字，说明如何控制引用约束关系。RMVCST 键字应用于一个约束关系中的从属文件。该键字说明在父代文件被删除时，有多少文件的约束关系被移走：

*RESTRICT

如果在一个父代文件和从属文件之间定义成建立了一个约束关系,则该父代文件不被删除而关系也不能移走。这是缺省值。

*REMOVE

父代文件被删除，约束关系和定义被移走。父代文件和从属文件之间的约束关系也被移走。从属文件的相应外来键字访问路径（一条或多条）及约束定义被移走。

*KEEP

父代文件被删除，引用约束关系定义被留在定义状态。从属文件的相应外来键字访问路径及约束定义不被移走。

3.7.8.12　移出物理文件成员（RMVM）

当一个约束关系中父代文件的成员被移走时，约束关系置于定义状态。外来键字访问路径和引用约束定义不被移走。因为父代文件成员被移走，父代键字访问路径被移走，但是父代约束定义仍保持在文件级别。

当移走一个约束关系中从属文件的成员，约束关系被置于定义状态中。父代键字访问路径和约束定义不被移走。由于从属成员被移走，所以外来键字访问路径被移走，但是，引用约束定义不被移走。

3.7.8.13　保存/重存

如果重存一个父代文件，无论是重存在同一个库中还是不同库中，会用一个系统交叉引用文件来定义一个引用约束的从属文件来建立约束关系。

如果重存从属文件，TOLIB 被用于决定约束关系：

- 如果父代文件和从属文件在同一库中，则在 TOLIB 中用父代文件建立引用约束关系。
- 如果父代文件和从属文件在不同库中，则用原来的父代文件建立重复的从属文件的引用约束关系。

约束关系中父代文件和从属文件的重存顺序并不重要（父代在从属之前或之后恢复），约束关系被建立。

关于保存或重存功能的详细信息参阅《备份和恢复－基础》及《备份和恢复－高级》。

3.7.9　引用约束的考虑和限制因素

父代文件必须是一个物理文件。

父代文件最多可有一个成员，MAXMBRS(1)。

从属文件必须是物理文件。

从属文件最多可有一个成员，MAXMBRS(1)。

当从属文件和父代文件二者之一或二者都没有成员时可定义一个约束。只有当两个文件都各有一个成员时才可建立一个约束。

一个文件最多可有一个主键字，但可有多个父代键字。

每个文件最多可有 300 个约束。这个最大值是以下各项的总和：

－父代及从属文件的引用约束，定义的或建立的引用约束。

－唯一约束。

只有外部描述文件允许使用引用约束，源文件不允许，程序描述文件也不允许。

在*RESTRICT 关系中不允许文件有插入、更新或删除操作。

约束名在库中必须唯一。

约束不能被加到 QTEMP 库的文件中去。

3.7.9.1 约束周期

约束周期是一个约束关系序列，周期中父代文件的一个后代成为原父代文件的父代。

在 DB/400 数据库中不禁止约束循环。但最好避免使用它们。

第十七章　触发器

触发器是一系列动作，对某个物理数据库文件进行修改时能自动运行。可以由应用程序中的高级语言语句做插入，更新或删除操作。

数据库用户可以使用触发器来：

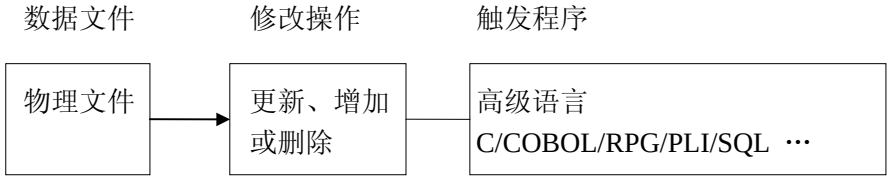
- 实施商务规则；
- 验证输入数据；
- 给在不同文件中的新插入行产生一个唯一键字值（代理功能）；
- 写入其它文件，做审查跟踪；
- 为交叉引用目的从其它文件查询；
- 存取系统功能（如当违犯一个规则时打印例外信息）；
- 把数据复制到不同文件中以使数据一致。

在客户的商务环境中可以得到以下好处：

- 更快的应用开发速度：因为触发器存于数据库中，由触发器执行的动作，不用在每个数据应用中编码。
- 商务的规则的轮回实施：一个触发器要一次定义后由应用程序来使用。
- 维护简单：如果一个商务策略改变，只需改变相应的触发器程序而不用改每一个应用程序。
- 改善客户/服务器环境中性能：所有规则在返回结果之前在服务器运行。

在 AS/400 系统上，可在任一支持的高级语言上定义一系列触发器动作。要使用 AS/400 触发器支持，必须生成一个触发器程序并将其加入到一个物理文件中。为将触发器加入文件，必须：

- 标识被管理的物理文件；
- 标识在该文件中被管理的操作种类；
- 生成一个高级语言或 CL 程序来执行所需动作；



3.8.1　给文件加一个触发器

增加物理文件触发器命令（ADDPFTRG）把一个触发器程序与一个物理文件联系起来。一旦建立了联系，在修改物理文件、它的一个成员和任何由该物理文件生成的逻辑文件时，系统在操作开始时就调用触发器程序。

一个物理文件最多可与 6 个触发器相联，触发器可放在：

- 插入前
- 插入后
- 删除前
- 删除后
- 更新前
- 更新后

在修改前后都可调用触发器。如，图 17－2 中更新 EMP 文件一个记录，更新前和更新后都调用一个触发器程序。

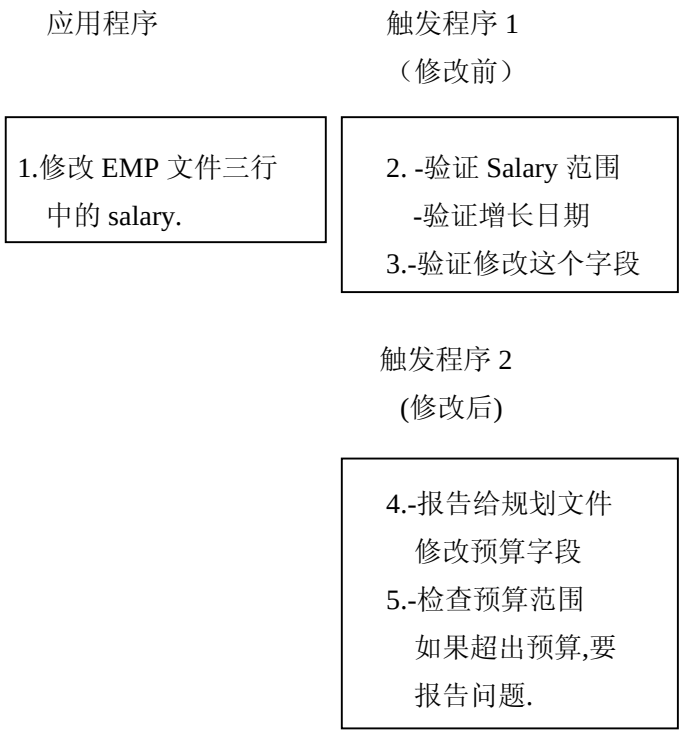


图 17-2　修改操作前后的触发器

在图 17-2 中,假定在下面条件中：

所有文件在同一落实定义下被打开。

EMP 文件有一个更新前触发器（触发程序 1）和一个更新后触发器（触发程序 2）。

图 17－2 注释：

1.这个应用程序试图更新 EMP 文件中一个雇员的薪水字段。

2.系统在记录被修改前调用更新前触发器。更新前触发器验证所有规则。

3.如果验证失败，触发器发出一个例外信息通知系统，该触发器程序发生了错误。系统随后通知应用程序更新操作失败，并退回更新前触发器所做的所有修改。在这种情况下，不调用更新后触发器。

4.如果所有规则检查合格，触发器程序正常返回，系统随后做更新操作。若更新操作成功，系统调用更新后触发器来执行传递的更新动作。

5.更新后触发器执行所有必要动作返回。如果在更新后触发器程序发生错误，程序向系统发出一个例外信息。系统通知应用程序更新操作失败，由两个触发器所做的所有修改和更新操作被退回。

3.8.2　移出一个触发器

去掉物理文件触发器命令（RMVPFTRG）移去文件和触发器程序之间的联系。如果物理文件已经修改完，移去该联系不会有任何动作，而触发器程序仍保存在系统中。

3.8.3 显示触发器

显示文件描述命令（DSPFD）提供了一个与文件相联的触发器的清单。规定 TYPE(*ALL) 可得到这一清单。提供的信息有：

- 触发器程序的数量
- 触发器程序名和库
- 触发器事件
- 触发器时间
- 触发器更新条件

3.8.4 生成一个触发程序

要把触发器加到一个物理文件中，必须提供一个触发程序。这一程序在增加物理文件触发器（ADDPFTRG）命令中用（PGM）参数规定。当用户或应用程序对一个具有触发器的物理文件做修改操作时，修改操作调用相应的触发程序。

修改操作将两个参数传递给触发程序。从这些输入中触发器程序可引用原来记录和新记录的拷贝。要设计触发器程序来接受这些参数。

3.8.4.1 触发程序的输入参数

这些参数是：

顺序	说明
1	触发器缓冲区，包括调用此触发器的修改操作的有关信息。
2	触发器缓冲区长度。

3.8.4.2 触发器缓冲区

触发器缓冲区，参看图 17－3，有两个逻辑部分，一个是静态的，一个是可变的：

- 静态部分包括：
  －一个触发器样板，包括物理文件名、成员名。触发器事件。触发器时间、落实锁定级和当前修改记录的 CCSID。
  －记录区的偏移量和长度及空字节映象。
  －这一部分占用的（十进制）偏移量 0 到 95。
- 可变部分包括：
  －旧记录、旧空字节映象、新记录和新空字节映象的区域。

偏移 十进制	十六进制	类型	字段
0	0	字符（10）	物理文件名
10	A	字符（10）	物理文件库名
20	14	字符（10）	物理文件成员名
30	1E	字符（1）	触发事件
31	1F	字符（1）	触发时间
32	20	字符（1）	落实锁定级别

33	21	字符（3）	保留
36	24	二十制（4）	数据 CCSID
40	28	字符（8）	保留
48	30	二进制（4）	原记录偏移量
52	34	二进制（4）	原记录长度
56	38	二进制（4）	原记录空字节映象偏移量
60	3C	二进制（4）	原记录空字节映象长度
64	40	二进制（4）	新记录偏移量
68	44	二进制（4）	新记录长度
72	48	二进制（4）	新记录空字节映象偏移量
76	4C	二进制（4）	新记录空字节映象长度
80	50	字符（16）	保留
*	*	字符（*）	原记录
*	*	字符（*）	原记录空字节映象
*	*	字符（*）	新记录
*	*	字符（*）	新记录空字节映象

图 17－3　触发器缓冲区

3.8.4.2.1　触发器缓冲区字段描述

下表是按字段的字母顺序排列的。

- 数据的 CCSID，在新记录和原来记录中的数据 CCSID。数据被数据库转换为作业 CCSID。
- 落实锁定级别。当前应用程序的落实锁定级别，可能的值是：

  ‘0’ *NONE

  ‘1’ *CHG

  ‘2’ *CS

  ‘3’ *ALL
- 新记录。在物理文件中作为修改操作结果的插入、更新记录拷贝。新记录只用于插入或更新操作。
- 新记录长。最大长度为 32766 个字节。
- 新记录空字节映象。这一结构包括新记录每个字段的空值（NULL）信息。一个字节代表一个字段。每个字节的可能值为：

  ‘0’ 非空

  ‘1’ 空（NULL）
- 新记录偏移量。新记录的位置。偏移量从触发器缓冲区的开始处算起。如果记录的新值未用于修改操作（如删除），则这个字段没用。
- 新记录空字节映象长度。该长度等于物理文件中的字段数。
- 新记录空字节映象偏移量。新记录空字节映象的位置。该偏移量从触发器缓冲区的开始处算起。如果记录的新值不用于修改操作（如删除），则该字段没用。
- 原记录。更新或删除之前的原物理记录拷贝。原记录只用于更新和删除操作。
- 原记录长度。最大值为 32766 字节。
- 原记录空字节映象。这个结构为原记录的每个字段空值（NULL）信息。一个字节代表一个字段。每个字节的可能值是：

‘0’非空

‘1’空（NULL）

- 原记录空字节映象长度。该长度等于物理文件中字段数。
- 原记录空字节映象偏移量。原记录空字节映象的位置。偏移量从触发器缓冲区的开始算起。如果原记录值未用于修改（如插入），则该字段没用。
- 原记录偏移量。原记录的位置。偏移量值从触发器缓冲区的开始处算起。若记录的原值未用于修改操作（如插入），则该字段不用。
- 物理文件库名。物理文件所在库的库名。
- 物理文件成员名。物理文件成员的名字。
- 物理文件名。被修改的物理文件名。
- 触发器事件。引起调用触发器程序的事件：

  可能的值有：

  ‘1’插入操作

  ‘2’删除操作

  ‘3’更新操作
- 触发器时间。当调用触发器程序时，相对于物理文件修改操作的时间，可能值为：

  ‘1’在修改操作之后

  ‘2’在修改操作之前

3.8.4.3 触发程序的编码原则和用法

一个触发程序可以是一个高级语言、SQL 或 CL 程序。

一个触发程序不能包括以下命令、语句及操作，如果包括了，就会返回一个例外信息。

—对与调用触发器的插入、更新或删除操作有关的落实定义，不允许 COMMIT 操作，作业中其它落实定义允许 COMMIT 操作。

—对与调用触发器的插入、更新或删除操作有关的落实定义，不允许 ROLLBACK 操作。作业中的其它落实定义允许 ROLLBACK 操作。

—不允许 SQL CONNECT，DISCONNECT，SET CONNECTION 和 RELEASE 语句。

—对于调用触发器的插入、更新或删除操作的落实定义，不允许 ENDCMTCTL 这个 CL 命令，作业中其它落实定义允许 ENDCMTCTL 命令。

—试图将一个本地 API 落实资源（QTNADDCR）加到与调用触发器的插入、更新或删除操作同一个落实定义中。

—试图对一个已经被用*SHARE 参数调用触发程序的文件做任何 I/O 操作。

被激活的触发程序可使用与调用触发器的插入、更新或删除操作且有远程资源的落实定义。然而，如果触发程序失败并发出一个逃逸信息，而且所有远程资源在非主落实周期时被更新，那么整个交易被置于一种退回－需求状态。

触发程序可将一个远程资源加到与调用了触发器的插入、更新或删除操作相联系的落实定义上。这允许 LU62 远程资源（保护对话）和 DFM 远程资源（DDM 文件打开），但是不允许 DRDA 远程资源。

如果一个触发程序修改远程资源发生故障，那么触发程序必须结束，且发出一个逃逸信息。这样，系统能保证整个交易——对所有的远程资源－—能正确地退回。如果触发程没有用一个逃逸信息终止，在不同的远程位置上的数据库可能变得不一致。

应用程序的落实锁定级传至触发程序。我们建议触发程序在与应用程序相同的锁定级下运行。

触发程序和应用程序可在相同或不同的活动组中运行。建议用 ACTGRP（*CALLER）

来编译触发程序，以保证触发程序与应用程序之间的一致。

一个触发程序可以调用其它程序或被嵌套进触发程序的一个语句调用另一个触发程序。此外，触发程序本身可循环调用。触发最大嵌套数，对插入和更新来说是 200。不能让触发程序导致下列情况，如果程序在落实控制下运行，这些情况要引发错误。

－修改同一文件中的同一个记录。这个文件已经修改或由触发器来操作。

－在一个修改操作中对同一记录做矛盾的操作。

例如，一个记录修改由操作插入，然后又被触发程序删除。

注：如果修改操作未在落实控制下运行，则修改操作总是被保护的。然而，在触发程序中更新同一记录，就不被监控。

对物理文件中每一行的修改，都调用触发程序。

如果物理文件或相关的逻辑文件，用 SEQONLY（*YES）打开，而且物理文件有一个与之相关的触发程序，系统修改 SEQONLY 为(*NO)，以使它可对每一行的修改都调用触发程序。

3.8.4.4 触发程序和落实控制

以下各节中有关 COMMIT 的每项也都适用于 ROLLBACK 操作。

3.8.4.4.1 在落实控制下运行触发和应用程序

当触发程序和应用程序在同一落实定义下运行时，触发程序的失败引起与触发程序有关的所有语句的退回操作。这包括一个嵌套的触发程序中所有语句。原修改操作也被退回。这要求触发程序在其遇到错误时给出例外信息。

当触发程序和应用程序在不同落实定义下运行时，应用程序中的 COMMIT 语句只影响它自己的落实定义。程序员必须用其它方法落实触发程序的修改。参见《备份和恢复－高级》中有关落实定义的细节描述和《ILE 概念》中有关活动组的详述。

3.8.4.4.2 不在落实控制下运行的触发或应用程序

若两个程序均未在落实控制下运行，触发程序中的错误将使文件留在错误发生时的状态。不发生退回操作。

若触发程序未在落实控制下运行，而应用程序却在落实控制下运行，则应用程序中 COMMIT 语句只做由应用程序所做的修改。

若应用程序没有触发程序却在落实控制下运行，所有从触发程序来的修改在下面任一种情况下被落实：

- 活动组结束。在通常情况下，在活动组结束时，执行一个内含的落实。然而，如果发生系统失败异常，将执行退回。
- 在触发程序中执行落实操作。

一个未在落实控制运行下的程序，都有数据一致性问题。如果程序中有一个修改语句，则用户有责任维护数据的一致性。

3.8.4.5 触发程序错误信息

如果在触发器程序运行期间有错误，它必须在退出前发出一个适当的逃逸信息。这一信息可以是从系统发出的原始信息或由触发程序设计者生成的信息。

3.8.5 触发程序样本

这一节包含四个触发程序，分别由 ATMTRANS 文件的写、更新和删除操作触发。这些

触发程序由 C、COBOL 和 RPG 编写。

这一应用中所用的数据结构如下说明：

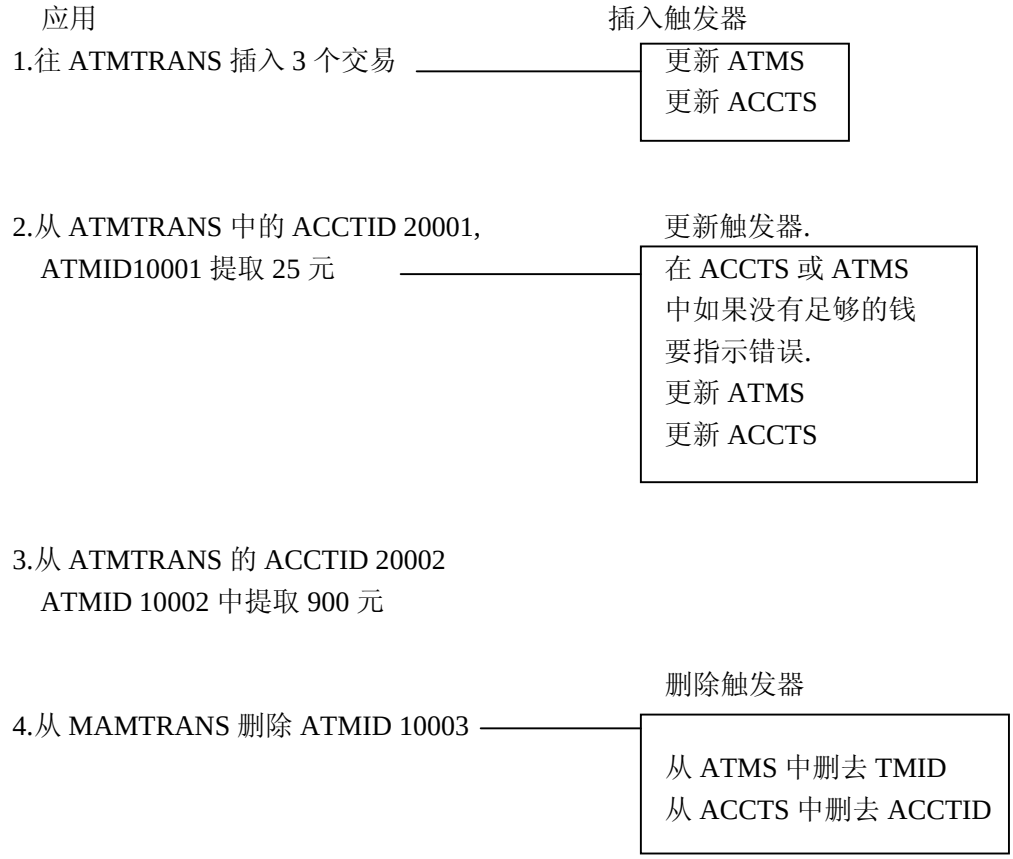
```
ATMTRANS :                  /* 交易记录                    */

      ATMID       CHAR(5) (KEY)  /* ATM 机器标志码      */
      ACCTID      CHAR(5)        /* 数量                 */
      TCODE       CHAR(1)        /* 交易码               */
      AMOUNT      ZONED          /* 存取款数额           */

ATMS :                      /* ATM 机器记录                */

      ATMN        CHAR(5)  (KEY) /* ATM 机器标志码      */
      LOCAT       CHAR(2)        /* ATM 位置             */
      ATMAMT      ZONED          /* ATM 中的总数         */
```

这一应用包括四种类型的交易：



注:

1、这个应用将三个记录插入到 ATMTRANS 文件中，激活插入触发器。插入触发器将正确数目插入到 ATMS 文件和 ACCTS 文件以反映这个变化。

2、该应用激活更新触发程序从帐号为 20001 和 ATM 为 10001 中提取了＄25.00，更新触发器从 ACCTS 和 ATMS 文件中减去了＄25.00。

3、该应用从账号为 20002 和 ATM 为 10002 提取了＄900.00，激活一个更新触发器。更新触发器对应用发出一个例外信息以说明交易失败。

4、应用程序从 ATMTRANS 文件中删除 ATM 号，激活了一个删除触发器。删除触发器从 ACCTS 文件中删除了 ACCTID，从 ATMS 文件中删除了 ATMID。

3.8.5.1 用 RPG 写的插入触发程序

```
     * 程序名 : INSTRG
     * 这是应用文件的插入触发器，应用程序往 ATMTRANS 文件中
     * 插入下面三个记录。
     *
     * ATMID    ACCTID    TCODE     AMOUNT
     * ---------------------------------
     * 10001    20001       D       100.00
     * 10002    20002       D       250.00
     * 10003    20003       D       500.00
     *
     * 当记录插入 ATMTRANS，系统调用此程序。它用当前存、取款数
     * 修改 ATMS 和 ACCTS 文件。触发程序的输入参数为：
     *  - TRGBUF : 包括触发信息及最新插入的 ATMTRANS 的
     *             记录映象。
     *  - TRGBUF : TRGBUF 长度。
     *
    H         1
     *
     * 打开 ATMS 和 ACCTS 文件。
     *
    FATMS    UF  E                    DISK         KCOMIT
    FACCTS   UF  E                    DISK         KCOMIT
     *
     * 定义传给此程序的结构。
     *
    IPARM1        DS
     * 物理文件名
    I                                       1   10 FNAME
     * 物理文件库名
    I                                      11   20 LNAME
     * 成员名
    I                                      21   30 MNAME
     * 触发事件
    I                                      31   31 TEVEN
```

```
 * 触发时间
I                                     32  32 TTIME
 * 落实控制级
I                                     33  33 CMTLCK
 * 保留
I                                     34  36 FILL1
 * CCSID
I                                  B  37  40OCCSID
 * 保留
I                                     41  48 FILL2
 * 对源记录的位移
I                                  B  49  52OLDOFF
 * 源记录长度
I                                  B  53  56OLDLEN
 * 对源记录空字节映象的位移
I                                  B  57  60ONOFF
 * 空字节映象长度
I                                  B  61  64ONLEN
 * 对新记录的位移
I                                  B  65  68NOFF
 * 新记录长度
I                                  B  69  72NEWLEN
 * 对新记录空字节映象的位移
I                                  B  73  76NNOFF
 * 空字节映象的长度
I                                  B  77  80NNLEN
 * 保留
I                                     81  96 RESV3
 * 老记录
I                                     97 112 OREC
 * 老记录的空字节映象
I                                    113 116 OOMAP
 * 最新插入的 ATMTRANS 记录
I                                    117 132 RECORD
 * 新记录的空字节映象
I                                    133 136 NNMAP
IPARM2       DS
I                                  B   1   4OLENG
```

```
*****************************************************************
* 建立参数列表项
*****************************************************************
C           *ENTRY    PLIST
C           PARM1     PARM           PARM1
C           PARM2     PARM           PARM2
*****************************************************************
* 用 NOFF，从传到触发程序的第一个参数得到新记录的位置。
*   - 由于传到程序的位移是从 0 开始，所以对 NOFF 加 1。
*   - 对 CHARACTER 字段取子串。如必要，把它传送给 NUMERIC 字段。
*****************************************************************
C                     Z-ADDNOFF      0        50
C                     ADD  1         0
*****************************************************************
*         - 取出 ATM 号码
*****************************************************************
C           5         SUBSTPARM1:0   CATM     5
*****************************************************************
*         - 加 0，它是参数串中的位移，取出 ACCOUNT 号码。
*****************************************************************
C                     ADD  5         0
C           5         SUBSTPARM1:0   CACC     5
*****************************************************************
*         - 加 0，它是参数串中的位移，取出 CODE 号码。
*****************************************************************
C                     ADD  5         0
C           1         SUBSTPARM1:0   TCODE    1
*****************************************************************
*         - 加 0，它是参数串中的位移，取出 AMOUNT 号码。
*****************************************************************
C                     ADD  1         0
C           5         SUBSTPARM1:0   CAMT     5
C                     MOVELCAMT      TAMT     52
*****************************************************************
*  处理  ATM 文件
*****************************************************************
* 读文件找到当前记录
C           ATMN      DOUEQCATM
```

```
C                   READ ATMS                     61EOF
C                   END
C   61              GOTO EOF
 * 修改 ATM 余额
C           TCODE   IFEQ 'D'
C                   ADD  TAMT     ATMAMT
C                   ELSE
C           TCODE   IFEQ 'W'
C                   SUB  TAMT     ATMAMT
C                   ELSE
C                   ENDIF
C                   ENDIF
 * 更新 ATM 文件
C           EOF     TAG
C                   UPDATATMFILE
C                   CLOSEATMS
 ************************************************************
 *  处理 ACCOUNT 文件
 ************************************************************
 * 读文件找到当前记录
C           ACCTN   DOUEQCACC
C                   READ ACCTS                    62  EOF2
C                   END
C   62              GOTO EOF2
 * 修改 ACCOUNTS 余额
C           TCODE   IFEQ 'D'
C                   ADD  TAMT     BAL
C                   ELSE
C           TCODE   IFEQ 'W'
C                   SUB  TAMT     BAL
C                   ELSE
C                   ENDIF
C                   ENDIF
 * 更新 ACCT 文件
C           EOF2    TAG
C                   UPDATACCFILE
C                   CLOSEACCTS
 *
```

C SETON LR

在应用程序做插入之后，ATMTRANS 文件包含以下数据：

图 17-6 ATMTRANS 记录

ATMID	ACCTID	TCODE	AMOUNT
10001	20001	D	100.00
10002	20002	D	250.00
10003	20003	D	500.00

插入触发器程序更新 ATMTRANS 之后，ATMS 文件和 ACCTS 文件包括以下数据：

图 17-7 由插入触发器修改后的 ATMS 文件

ATMN	LOCAT	ATMAMT
10001	MN	300.00
10002	MN	750.00
10003	CA	750.00

图 17-8 由插入触发器修改后的 ACCTS 文件

ACCTN	BAL	ACTACC
20001	200.00	A
20002	350.00	A
20003	500.00	C

3.8.5.2 用 COBOL 写的更新触发程序

```
100         IDENTIFICATION DIVISION.
200         PROGRAM-ID. UPDTRG.
700 ************************************************************************
800 **** 程序名 : UPDTRG                                                    *
800 *****                                                                   *
1900 ***** 当修改 ATMTRANS 文件中的记录时，调用此触发程序。                  *
1900 ***** 此程序检查 ATMS 中的 ACCTS 的余额和总数，如果其中有一个不满足取款 *
1900 ***** 额，则产生例外信息。如果 ACCTS 和 ATMS 都有足够的钱，程序修改这两 *
1900 ***** 个文件，以反映修改情况。在 ATMRTANS 文件中 ATMIDs 为 10001 和    *
1900 ***** 10002 的记录，用下列数据修改：                                   *
1900 ***** ATMID   ACCTID   TCODE    AMOUNT                                 *
1900 ***** -------------------------------                                  *
```

```
1900 ***** 10001    20001       W        25.00                                  *
1900 ***** 10002    20002       W       900.00                                  *
1900 ***** 10003    20003       D       500.00                                  *
1900 *****                                                                       *
2000 ******************************************************************
2100       ******************************************************************
2200        ENVIRONMENT DIVISION.
2300        CONFIGURATION SECTION.
2400        SOURCE-COMPUTER. IBM-AS400.
2500        OBJECT-COMPUTER. IBM-AS400.
2700        SPECIAL-NAMES. I-O-FEEDBACK IS FEEDBACK-JUNK.
2700        INPUT-OUTPUT SECTION.
2800        FILE-CONTROL.
3300            SELECT ACC-FILE ASSIGN TO DATABASE-ACCTS
3400                   ORGANIZATION IS INDEXED
3500                   ACCESS IS RANDOM
3600                   RECORD KEY IS ACCTN
3700                   FILE STATUS IS STATUS-ERR1.
3800
3900            SELECT ATM-FILE ASSIGN TO DATABASE-ATMS
4000                   ORGANIZATION IS INDEXED
4100                   ACCESS IS RANDOM
4200                   RECORD KEY IS ATMN
4300                   FILE STATUS IS STATUS-ERR2.
4400
4500       ******************************************************************
4600       *                    落实控制区                                  *
4700       ******************************************************************
4800        I-O-CONTROL.
4900            COMMITMENT CONTROL FOR ATM-FILE, ACC-FILE.
5000
5100       ******************************************************************
5200       *                    数据部                                      *
5300       ******************************************************************
5400
5500        DATA DIVISION.
5600        FILE SECTION.
5700        FD  ATM-FILE
```

```
5800            LABEL RECORDS ARE STANDARD.
5900         01  ATM-REC.
6000         COPY DDS-ATMFILE OF ATMS.
6100
6200         FD  ACC-FILE
6300             LABEL RECORDS ARE STANDARD.
6400         01  ACC-REC.
6500         COPY DDS-ACCFILE OF ACCTS.
6600
7000
7100        ************************************************************
7200        *                          工作区节                          *
7300        ************************************************************
7400         WORKING-STORAGE SECTION.
7500         01  STATUS-ERR1         PIC XX.
7600         01  STATUS-ERR2         PIC XX.
7700
7800         01 INPUT-RECORD.
7900            COPY DDS-TRANS OF ATMTRANS.
8000
8100         05 OFFSET-NEW-REC   PIC 9(8)  BINARY.
8200
8300         01 NUMBERS-1.
8400           03 NUM1          PIC 9(10).
8500           03 NUM2          PIC 9(10).
8600           03 NUM3          PIC 9(10).
8700
8800         01  FEEDBACK-STUFF      PIC X(500) VALUE SPACES.
8900
7100        ************************************************************
7200        * 标识触发错误的信息                                        *
7200        * - 用下列数据定义信息标识和信息文件                        *
7300        ************************************************************
9000         01 SNDPGMMSG-PARMS.
9100           03 SND-MSG-ID        PIC X(7)    VALUE "TRG9999".
9200           03 SND-MSG-FILE      PIC X(20)   VALUE "MSGF      LIB1    ".
9300           03 SND-MSG-DATA      PIC X(25)   VALUE "Trigger Error".
9400           03 SND-MSG-LEN       PIC 9(8) BINARY  VALUE 25.
```

```
 9500          03 SND-MSG-TYPE      PIC X(10)   VALUE "*ESCAPE   ".
 9600          03 SND-PGM-QUEUE     PIC X(10)   VALUE "*     ".
 9700          03 SND-PGM-STACK-CNT PIC 9(8) BINARY  VALUE 1.
 9800          03 SND-MSG-KEY       PIC X(4)    VALUE "    ".
 9900          03 SND-ERROR-CODE.
10000            05 PROVIDED          PIC 9(8) BINARY VALUE 66.
10100            05 AVAILABLE         PIC 9(8) BINARY VALUE 0.
10200            05 RTN-MSG-ID        PIC X(7) VALUE " ".
10300            05 FILLER            PIC X(1) VALUE " ".
10400            05 RTN-DATA          PIC X(50) VALUE " ".
10500
10600       ************************************************************
10700       *                          连接节                          *
10700       *  PARM 1 是触发缓冲区        PARM 2 是触发缓冲区的长度      *
10800       ************************************************************
10900        LINKAGE SECTION.
11000        01 PARM-1-AREA.
11100          03 FILE-NAME        PIC X(10).
11200          03 LIB-NAME         PIC X(10).
11300          03 MEM-NAME         PIC X(10).
11400          03 TRG-EVENT        PIC X.
11500          03 TRG-TIME         PIC X.
11600          03 CMT-LCK-LVL      PIC X.
11700          03 FILLER           PIC X(3).
11800          03 DATA-AREA-CCSID  PIC 9(8)  BINARY.
11900          03 FILLER           PIC X(8).
12000          03 DATA-OFFSET.
12100            05 OLD-REC-OFF      PIC 9(8)  BINARY.
12200            05 OLD-REC-LEN      PIC 9(8)  BINARY.
12300            05 OLD-REC-NULL-MAP PIC 9(8)  BINARY.
12400            05 OLD-REC-NULL-LEN PIC 9(8)  BINARY.
12500            05 NEW-REC-OFF      PIC 9(8)  BINARY.
12600            05 NEW-REC-LEN      PIC 9(8)  BINARY.
12700            05 NEW-REC-NULL-MAP PIC 9(8)  BINARY.
12800            05 NEW-REC-NULL-LEN PIC 9(8)  BINARY.
12900            05 FILLER           PIC X(15).
12000          03 RECORD-JUNK.
12900            05 OLD-RECORD       PIC X(16).
```

```
12900            05 OLD-NULL-MAP     PIC X(4).
12900            05 NEW-RECORD       PIC X(16).
12900            05 NEW-NULL-MAP     PIC X(4).
13000
13100        01 PARM-2-AREA.
13200          03  TRGBUFL              PIC X(2).
13300 ****************************************************************
13400 ******                          过程部                          *
13500 ****************************************************************
13600        PROCEDURE DIVISION USING PARM-1-AREA, PARM-2-AREA.
13700        MAIN-PROGRAM SECTION.
13800        000-MAIN-PROGRAM.
14000            OPEN I-O ATM-FILE.
14100            OPEN I-O ACC-FILE.
14200
14300            MOVE 0 TO BAL.
14400
14500       ************************************************************
14600       * 设置位移点，复制新记录。                                   *
14600       * 由于输入的位移是从 0 开始，所以要给位移加 1。              *
14700       ************************************************************
14800            ADD 1, NEW-REC-OFF GIVING OFFSET-NEW-REC.
14900
15000            UNSTRING PARM-1-AREA
15100              INTO INPUT-RECORD
15200              WITH POINTER OFFSET-NEW-REC.
15300
15400       ************************************************************
15500       * 从 ACCTS 文件读记录                                        *
15600       ************************************************************
15700            MOVE ACCTID TO ACCTN.
15800            READ ACC-FILE
15900                 INVALID KEY PERFORM 900-OOPS
16000                 NOT INVALID KEY PERFORM 500-ADJUST-ACCOUNT.
16100
16200       ************************************************************
16300       * 从 ATMS 文件读记录                                         *
16400       ************************************************************
```

```
16500            MOVE ATMID TO ATMN.
16600            READ ATM-FILE
16700                 INVALID KEY PERFORM 950-OOPS
16800                 NOT INVALID KEY PERFORM 550-ADJUST-ATM-BAL.
17100            CLOSE ATM-FILE.
17200            CLOSE ACC-FILE.
17300            STOP RUN.
17400
17500 ******************************************************************
17600 ******************************************************************
17700 ******************************************************************
17800 ******************************************************************
17900 ****** 如果在 ACCTS 中没有足够的钱来取款，用这个过程。              ****
18100 ******************************************************************
18200        200-NOT-ENOUGH-IN-ACC.
18300            DISPLAY "NOT ENOUGH MONEY IN ACCOUNT.".
18400            ROLLBACK.
18600            CLOSE ATM-FILE.
18700            CLOSE ACC-FILE.
18800            PERFORM 999-SIGNAL-ESCAPE.
18900            STOP RUN.
19000       delete ROLLBACK after DISPLAY
19000
19100 ******************************************************************
19200 ****** 如果在 ATMS 中没有足够的钱来取款，用这个过程。
19400 ******************************************************************
19500        250-NOT-ENOUGH-IN-ATM.
19600            DISPLAY "NOT ENOUGH MONEY IN ATM.".
19700            ROLLBACK.
19900            CLOSE ATM-FILE.
20000            CLOSE ACC-FILE.
20100            PERFORM 999-SIGNAL-ESCAPE.
20200            STOP RUN.
20300
19000       delete ROLLBACK after DISPLAY
20300
20400 ******************************************************************
20500 ****** 此例程用来为用此交易的用户修正余额总数。
```

```
20700 ****************************************************************
20800        500-ADJUST-ACCOUNT.
20900            IF TCODE = "W" THEN
21000               IF (BAL < AMOUNT) THEN
21100                  PERFORM 200-NOT-ENOUGH-IN-ACC
21200               ELSE
21300                  SUBTRACT AMOUNT FROM BAL
21400                  REWRITE  ACC-REC
21500            ELSE IF TCODE = "D" THEN
21600               ADD AMOUNT TO BAL
21700               REWRITE  ACC-REC
21800            ELSE DISPLAY "TRANSACTION CODE ERROR, CODE IS: ", TCODE.
21900
22000 ****************************************************************
22100 ****** 此例程用在交易后 ATM 文件的余额修正                          ***
22300 ****************************************************************
22400        550-ADJUST-ATM-BAL.
22500            IF TCODE = "W" THEN
22600               IF (ATMAMT < AMOUNT) THEN
22700                  PERFORM 250-NOT-ENOUGH-IN-ATM
22800               ELSE
22900                  SUBTRACT AMOUNT FROM ATMAMT
23000                  REWRITE  ATM-REC
23100            ELSE IF TCODE = "D" THEN
23200               ADD AMOUNT TO ATMAMT
23300               REWRITE  ATM-REC
23400            ELSE DISPLAY "TRANSACTION CODE ERROR, CODE IS: ", TCODE.
23500
23600 ****************************************************************
23700 ****** 此例程用于在 ACCTS 文件中没找到所用键字值。                    **
23900 ****************************************************************
24000        900-OOPS.
24100            DISPLAY "INVALID KEY: ", ACCTN, "  ACCOUNT FILE STATUS: ",
24200                     STATUS-ERR1.
24500            CLOSE ATM-FILE.
24600            CLOSE ACC-FILE.
24700            PERFORM 999-SIGNAL-ESCAPE.
24800            STOP RUN.
```

```
24900
25000 ****************************************************************
25100 ****** 此例程用于在 ATM 文件中没找到所用键字值。                          **
25300 ****************************************************************
25400         950-OOPS.
25500             DISPLAY "INVALID KEY: ", ATMN, "  ATM FILE STATUS: ",
25600                     STATUS-ERR2.
25900             CLOSE ATM-FILE.
26000             CLOSE ACC-FILE.
26100             PERFORM 999-SIGNAL-ESCAPE.
26200             STOP RUN.
26300
26400 ****************************************************************
26500 ****** 标识例外信息                                              ********
26600 ****************************************************************
26700         999-SIGNAL-ESCAPE.
26800
26900             CALL "QMHSNDPM" USING SND-MSG-ID,
27000                                   SND-MSG-FILE,
27100                                   SND-MSG-DATA,
27200                                   SND-MSG-LEN,
27300                                   SND-MSG-TYPE,
27400                                   SND-PGM-QUEUE,
27500                                   SND-PGM-STACK-CNT,
27600                                   SND-MSG-KEY,
27700                                   SND-ERROR-CODE.
27800        *DISPLAY RTN-MSG-ID.
27900        *DISPLAY RTN-DATA.
28000
```

更新触发程序在 ATMTRANS 文件更新后，ATMS 和 ACCTS 文件包含以下数据。由于帐目中无足够数量，对 ATM10002 的更新失败。

图 17-9　由修改触发器修改后的 ATMS 文件

ATMN	LOCAT	ATMAMT
10001	MN	275.00
10002	MN	750.00
10003	CA	750.00

图 17-10　由修改触发器修改后的 ACCTS 文件

ACCTN	BAL	ACTACC
20001	175.00	A
20002	350.00	A
20003	500.00	C

3.8.5.3　用 ILE C 写的删除触发程序

```
/*************************************************************/
/* 程序名 - DELTRG                                           */
/* 当在 ATMTRANS 文件中有删除操作时，调用此程序。             */
/* 这个程序根据从触发缓冲区传来的 ATM ID 和 ACCT ID 来从 ATMS  */
/* 和 ACCTS 文件中删除记录。                                  */
/* 用户要从 ATMTRANS 文件中删除 ATM ID 为 10003 的记录。      */
/*************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <recio.h>
#include "applib/csrc/msghandler" /* message handler include       */
#include "qsysinc/h/trgbuf"       /* trigger buffer include without*/
                                  /* old and new records          */
Qdb_Trigger_Buffer *hstruct;      /* pointer to the trigger buffer */
char *datapt;

#define KEYLEN 5

/*************************************************************/
/* 由于在各文件中有非字符字段，所以要定义文件结构。          */
/* 对每个非字符字段，C 语言需要对齐边界。                    */
/* 这样，要用一个_PACKED 结构来存取传给触发程序的数据。      */
/*************************************************************/

/** record area for ATMTRANS                **/
_Packed struct rec {
             char atmid[5];
             char acctid[5];
             char tcode[1];
             char amount[5];
```

```
                } oldbuf, newbuf;

/** record area for ATMS                    **/
_Packed struct rec1{
            char atmn[5];
            char locat[2];
            char atmamt[9];
                } atmfile;

/** record area for ACCTS                   **/
_Packed struct rec2{
            char acctn[5];
            char bal[9];
            char actacc[1];
                } accfile;

/*************************************************************************/
/* 主程序开始                        ***********************************/
/*************************************************************************/
main(int argc, char **argv)
{
_RFILE  *out1;              /* ATMS 的文件指针              */
_RFILE  *out2;              /* ACCTS 的文件指针             */
_RIOFB_T *fb;               /* 文件反馈指针                 */
char record[16];           /* 记录缓冲区                   */
_FEEDBACK fc;               /* 信息处理的反馈               */
_HDLR_ENTRY hdlr = main_handler;
                            /*******************************/
                            /* 激活例外处理                */
                            /*******************************/
CEEHDLR(&hdlr, NULL, &fc);
                            /*******************************/
                            /* 确保例外处理就绪            */
                            /*******************************/
if (fc.MsgNo != CEE0000)
 {
   printf("Failed to register exception handler.\n");
   exit(99);
```

```
 }

/* 对输入参数设置指针                          */
hstruct = (Qdb_Trigger_Buffer *)argv[1];
datapt  = (char *) hstruct;

/* 对输入参数复制新老记录                      */

 if ((strncmp(hstruct ->trigger_event,"2",1)== 0)|| /* 删除事件     */
     (strncmp(hstruct -> trigger_event,"3",1)== 0)) /* 更新事件     */
   { obufoff = hstruct ->old_record_offset;
     memcpy(&oldbuf,datapt+obufoff, hstruct->old_record_len);
   }
 if ((strncmp(hstruct -> trigger_event,"1",1)== 0)  |  /* 插入事件   */
     (strncmp(hstruct -> trigger_event,"3",1)== 0))    /* 更新事件   */
   { nbufoff = hstruct ->new_record_offset;
     memcpy(&newbuf,datapt+nbufoff, hstruct->new_record_len);
   }

/*****************************************************/
/* 打开 ATM 和 ACCTS 文件                             */
/* 检查应用程序的落实锁级别。                         */
/* 如果它是在落实控制下运行，就用落实控制打开这两个    */
/* 文件，否则，不用落实控制打开它们。                 */
/*****************************************************/
if(strcmp(hstruct->commit_lock_level,"0") == 0)    /* 未用落实控制    */
 {
   if ((out1=_Ropen("APPLIB/ATMS","rr+")) == NULL)

     {
       printf("Error opening ATM file");
       exit(1);
     }
   if ((out2=_Ropen("APPLIB/ACCTS","rr+")) == NULL)
     {
       printf("Error opening ACCTS file");
       exit(1);
     }
```

```
 }
else                    /* 用落实控制   */
{
  if ((out1=_Ropen("APPLIB/ATMS","rr+,commit=Y")) == NULL)
    {
      printf("Error opening ATMS file");
      exit(1);
    }
  if ((out2=_Ropen("APPLIB/ACCTS","rr+,commit=Y")) == NULL)
    {
      printf("Error opening ACCTS file");
      exit(1);
    }
 }

/* 根据输入参数删除记录  */
fb =_Rlocate(out1,&oldbuf.atmid,KEYLEN,__DFT);
if (fb->num_bytes != 1)
 {
  printf("record not found in ATMS\n");
  _Rclose(out1);
  exit(1);
 }
 _Rdelete(out1);          /* 从 ATMS 中删除记录*/
 _Rclose(out1);

fb =_Rlocate(out2,&oldbuf.acctid,KEYLEN,__DFT);
if (fb->num_bytes != 1)
 {
  printf("record not found in ACCOUNTS\n");
  _Rclose(out2);
  exit(1);
 }
 _Rdelete(out2);          /* 从 ACCOUNTS 中删除记录 */
 _Rclose(out2);

 } /* end of main */
```

在该应用程序的删除操作后，ATMTRANS 包括以下数据：

图 17-11　ATMTRANS 记录

ATMID	ACCTID	TCODE	AMOUNT
10001	20001	W	25.00
10002	20002	W	900.00

删除触发器程序从 ATMTRANS 文件做删除之后，ATMS 文件和 ACCTS 文件包含以下数据：

图 17-12　由删除触发器修改后的 ATMS 文件

ATMN	LOCAT	ATMAMT
10001	MN	275.00
10002	MN	750.00

图 17-13　由删除触发器修改后的 ACCTS 文件

ACCTN	BAL	ACTACC
20001	175.00	A
20002	350.00	A

```
/*********************************************************************/
/*   INCLUDE 名    : MSGHANDLER                                        */
/*   说  明        : 信息处理标识一个例外信息给触发程序的调用者。   */
/*********************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <recio.h>
#include <leawi.h>

#pragma linkage (QMHSNDPM, OS)
void QMHSNDPM(char *,              /* 信息标识                 */
              void *,              /* 限定的信息文件名         */
              void *,              /* 信息数据或正文           */
              int,                 /* 信息数据或正文的长度     */
              char *,              /* 信息类型                 */
              char *,              /* 调用信息队列             */
              int,                 /* 调用堆栈记录             */
              void *,              /* 信息键                   */
              void *,              /* 错误码                   */
```

```
            ...);               /* 选项:
                                       调用信息队列名的长度
                                       调用堆栈入口限定
                                       显示外部信息
                                       屏幕等待时间             */
/*********************************************************************/
/********  例外管理功能开始                                          */
/*********************************************************************/
  void main_handler(_FEEDBACK *cond, _POINTER *token, _INT4 *rc,
                    _FEEDBACK *new)
   {
                           /*************************************/
                           /* 初始化变量来调用 QMHSNDPM。              */
                           /* 用户对下列数据定义信息标识和信息文件。 */
                           /*************************************/
  char      message_id[7] = "TRG9999";
  char      message_file[20] = "MSGF      LIB1      ";
  char      message_data[50] = "Trigger error               ";
  int       message_len = 30;
  char      message_type[10] = "*ESCAPE   ";
  char      message_q[10] = "_C_pep    ";
  int       pgm_stack_cnt = 1;
  char      message_key[4];
                           /*************************************/
                           /* 对 QMHSNDPM 说明错误码结构。             */
                           /*************************************/
  struct error_code {
    int bytes_provided;
    int bytes_available;
    char message_id[7];
  } error_code;

  error_code.bytes_provided = 15;
                           /*************************************/
                           /* 设置错误处理以保证和标志                 */
                           /* 最后处理的逃逸信息。                     */
                           /*************************************/
  *rc = CEE_HDLR_RESUME;
```

```
                        /****************************************/
                        /* 送给本人的*ESCAPE 信息。              */
                        /****************************************/
  QMHSNDPM(message_id,
           &message_file,
           &message_data,
           message_len,
           message_type,
           message_q,
           pgm_stack_cnt,
           &message_key,
           &error_code );
                        /****************************************/
                        /* 检查调用 QMHSNDPM 是否正确完成。      */
                        /****************************************/
  if (error_code.bytes_available != 0)
      {
        printf("Error in QMHOVPM : %s\n", error_code.message_id);
      }
  }

/*****************************************************************/
/*   INCLUDE 名 : TRGBUF                                         */
/*   说    明: 用户触发程序的输入，触发缓冲区结构，存在 QSYSINC/H 中。*/
/*****************************************************************/
/*   注  : 下面的类型定义只定义了格式的固定部分。                 */
/*         源记录的数据区、源记录空字节映象、新记录、             */
/*         新记录的空字节映象都是变长的，它紧接在下面             */
/*         定义的后边。                                          */
/*****************************************************************/
 typedef _Packed struct Qdb_Trigger_Buffer {
           char  file_name[10];
           char  library_name[10];
           char  member_name[10];
           char  trigger_event[1];
           char  trigger_time[1];
           char  commit_lock_level[1];
           char  resevered[3];
```

```
int   data_area_ccsid;
char  resevered]8];
int   old_record_offset;
int   old_record_len;
int   old_record_null_byte_map;
int   old_record_null_byte_map_len;
int   new_record_offset;
int   new_record_len;
int   new_record_null_byte_map;
int   new_record_null_byte_map_len;
      } Qdb_Trigger_Buffer;
```

3.8.6 与触发器有关的其他 AS/400 功能

3.8.6.1 保存/重存基本文件（SAVOBJ/RSTOBJ）

保存/重存功能不在保存/重存期间检索触发器程序。管理程序是用户的责任。在运行期间，若没有重存触发器程序，则会返回一个带有触发器程序、物理文件名和触发器事件的严重错误。

如果保存过整个库，且物理文件和所有触发器程序在同一库中，但却被重存在不同库中，则物理文件中所有触发器程序名被修改以反映出新库。

3.8.6.2 保存/重存触发程序（SAVOBJ/RSTOBJ）

若触发程序重存在不同库中，则由于在原来库中找不到触发程序，修改操作失败。返回一个带有触发程序名、物理文件名和触发事件信息的严重错误。

这种情况下有两种恢复方法：

—把触发程序恢复到同一库中；

—在新库中用相同名生成一个新的触发程序。

3.8.6.3 删除文件（DLTF）

可以移出在触发程序和被删除文件之间的联系。触发程序仍保存在系统上。

3.8.6.4 拷贝文件（CPYF）

如果一个 TOFILE 与一个插入触发器相关，则每个插入的记录引起触发程序的调用。

如果一个 TOFILE 与一个删除触发程序相关并且在 CPYF 命令上说明了 MBROPT(*REPLACE)，则拷贝操作失败。

说明了 CREATE(*YES)的 COPY 不留下触发器信息。

3.8.6.5 生成重复目标（CRTDUPOBJ）

当把一个触发程序拷贝到另一个库时：

—如果触发程序和基本物理文件最初在同一库中，则新的触发程序与新的物理文件相关。

—如果触发程序和物理文件最初在不同库中，则重复触发器与和最初触发程序的同一物理文件相关。即使物理文件被在同一新库中重复，重复的触发程序仍与最初的物

理文件有关。

3.8.6.6　清除物理文件成员（CLRPFM）

若物理文件与一个删除触发器相关，CLRPFM 操作失败。

3.8.6.7　初始化物理文件成员（INZPFM）

若物理文件与一个插入触发器相关，INZPFM 失败。

3.8.6.8　FORTRAN 中的强制数据结束语句（FEOD）

若物理文件与一个删除触发器相关，FEOD 操作失败。

3.8.6.9　提供日志修改或消除日志修改（APYJRNCHG/RMVJRNCHG）

若物理文件与任何一类型触发器相关，APYJRNCHG 和 RMVJRNCHG 操作不会引起调用触发程序。因此，应保证对所有触发程序中的文件做日志。然后，在使用 APYJRNCHG 或 RMVJRNCHG 命令时，一定要说明所有这些文件，这保证了所有应用程序对物理文件的修改和触发程序一致。

3.8.7　对触发程序的建议

以下是对触发程序的建议：

- 用 USRPRF(*OWNER)生成程序，而不要将触发程序授权给 USER(*PUBLIC)。避免触发程序被其它用户修改或替代。不管调用触发器运行的用户对触发程序有没有特权，数据库都激活这个触发程序。
- 如果程序是在一个 ILE 环境中运行，生成程序要用 ACTGRP(*CALLER)，这样使触发程序与应用在相同的落实定义下运行。
- 打开文件使用与应用程序相同的落实锁定级别。这使触发程序在与应用程序相同的落实锁级别下运行。
- 在被修改文件同一库中生成程序。
- 若触发程序与应用程序在不同的活动组下运行，在触发程序中使用落实或退回控制。
- 触发程序发生错误，则给出一个例外信息。若无错误信息从触发器发出，则数据库认为触发器成功运行。这会引起用户数据在不一致状态下结束。

触发程序是非常强有力的。在设计存取一个系统资源（如磁带设备）的触发程序时要小心。例如，把记录修改拷贝到磁带上的触发程序可能会很有用，但程序本身不能检查磁带是否准备好或磁带是不是正确。你必须在设计触发程序时将这些类问题考虑进来。

应认真考虑以下功能，建议在触发程序中不使用它们：

- STRCMTCTL
- RCLSPSTG
- RCLSRC
- CHGSYSLIBL
- DLTLICPGM，RSTLICPGM 和 SAVLICPGM
- SAVLIB　SAVACT(*YES)
- 任何带有 DKT 或 TAP 的命令
- 任何迁移命令
- 调试程序（安全方面）
- 任何与运程作业录入（RJE）相关的命令

- 调用另一个 CL 或交互录入—可能达到锁定资源的极限

3.8.8 触发器和引用一致性的相互关系

一个物理文件可同时有与其相关的触发器和引用约束。触发器动作和引用约束之中的运行顺序依赖于与文件相关的约束和触发器。

在一些情况下，引用约束是在一个其后的触发程序被调用之前赋值的。这种情况下，是说明了 RESTRICT 规则的约束。

在一些情况下，触发程序中的所有语句——包括嵌套触发程序——都在运用约束之前运行。对于 NO ACTION，CASCADE，SET NULL 和 SET DEFAULT 引用约束规则来说都是这样的。当说明了这些规则时，系统根据触发程序嵌套的结果来检查文件的约束。例如，一个应用程序把一个雇员记录插入到一个 EMP 文件中，有一个约束和触发器：

- 引用约束规定对一个插入到 EMP 文件的雇员记录，其部门号必须存在于 DEPT 文件中。
- 触发程序——无论何时向 EMP 文件中插入——都要检查 DEPT 文件中是否存在该部门号，若不存在，则加上该号。

当向 EMP 文件做插入时，系统首先调用触发程序。如果部门号不在 DEPT 文件中，触发程序将新的部门号插入 DEPT 文件，然后系统检查引用约束。在这种情况下，插入是成功的，因为部门号在 DEPT 文件中。

当触发和引用约束都对同一物理文件定义时有一些限制：

- 若一个删除触发器与一个物理文件相关，则那个文件一定不能是带有 CASCADE 删除规则的引用约束中的从属文件。
- 若一个更新触发器与一个物理文件相关,该物理文件中没有字段为带有删除规则 SET NULL 或 SET DEFAULT 的引用约束中的外来键字。

如果在一个触发程序或引用约束生效期间发生错误，假如所有文件在同一落实定义之下运行，则所有与修改操作相关的触发程序被退回，而在同一落实定义下运行的所有用触发程序和引用集成网中的文件都得到保证。若文件在没有落实控制打开或文件处于一种混合状态（即一些文件落实控制打开，另一些没有），那么会产生意外结果。有关引用约束和触发器之间相互作用的详细信息和例子，参见红皮书《DB2/400 高级数据库功能》，书号 GG24－4249。

可以用触发器实施引用约束和交易规则。例如，可用触发器去模仿对物理文件的更新约束。在系统引用一致性功能中定义的约束就没有提供同样功能。若约束与触发器一起定义则以下引用一致性的优点可能会丢失：

- 从属文件可能有违反一个或多个引用约束的行，而使约束被放入检查未决中，但仍允许对文件操作。
- 当一个约束已被放入检查未决时通知用户的能力。
- 当应用程序在 COMMIT(*NONE)之下运行而在一个删除期间发生了错误，所有变化被数据库退回。
- 当保存一个与某约束相关的文件时，所有存储在一个数据库网络中同一个从属文件被保存。